

52. ročník matematické olympiády na středních školách

52. ročník matematické olympiády na středních školách. Zpráva o řešení úloh ze soutěže konané ve školním roce 2002/2003. 44. mezinárodní matematická olympiáda. 15. mezinárodní olympiáda v informatice. (Czech). Praha: Jednota českých matematiků a fyziků, 2004.

Persistent URL: <http://dml.cz/dmlcz/405053>

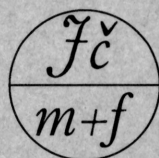
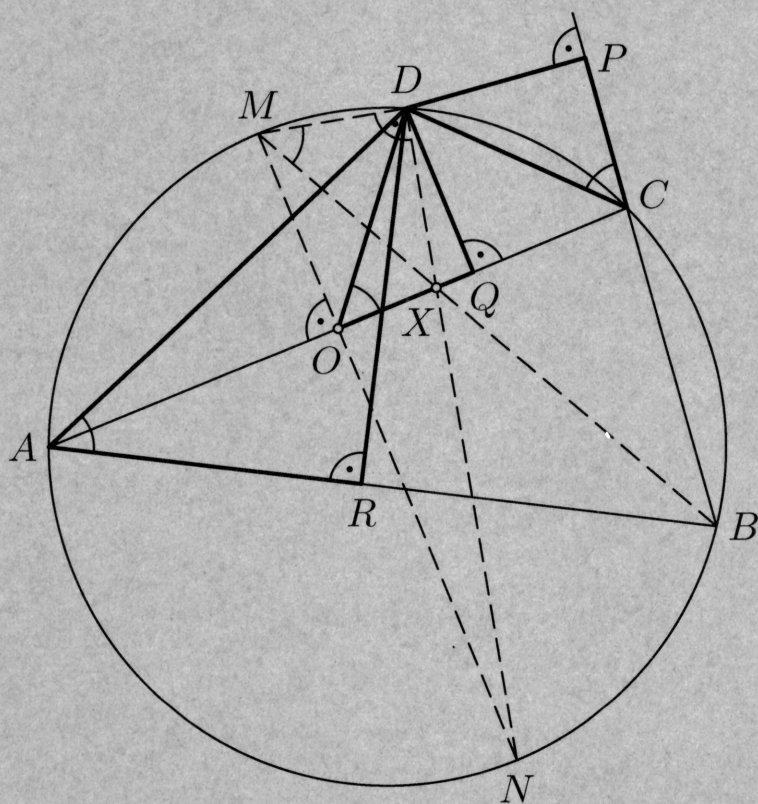
Terms of use:

Institute of Mathematics of the Czech Academy of Sciences provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.



This document has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ: The Czech Digital Mathematics Library* <http://dml.cz>

52. ročník matematické olympiády na středních školách



PADESÁTÝ DRUHÝ ROČNÍK MATEMATICKÉ OLYMPIÁDY NA STŘEDNÍCH ŠKOLÁCH

Zpráva o řešení úloh ze soutěže konané ve školním roce 2002/2003

44. MEZINÁRODNÍ MATEMATICKÁ OLYMPIÁDA
PATNÁCTÁ MEZINÁRODNÍ OLYMPIÁDA V INFORMATICE

JEDNOTA ČESKÝCH MATEMATIKŮ A FYZIKŮ PRAHA

Za přispění spolupracovníků zpracovali
doc. RNDr. Leo Boček, CSc., RNDr. Karel Horák, CSc.,
RNDr. Tomáš Pitner, Dr., doc. RNDr. Jaromír Šimša, CSc.,
RNDr. Jaroslav Švrček, CSc., doc. RNDr. Pavel Töpfer, CSc.
RNDr. Jaroslav Zhouf, PhD.

© Karel Horák, editor, 2004

ISBN 80-7015-960-X

Obsah

O průběhu 52. ročníku matematické olympiády	5
Nejúspěšnější řešitelé II. kola MO v kategoriích A, B, C a P	9
Výsledky celostátního kola 52. ročníku MO kategorie A	26
Výsledky celostátního kola 52. ročníku MO kategorie P	27
Kategorie C	28
Texty úloh	28
Řešení úloh	31
Kategorie B	45
Texty úloh	45
Řešení úloh	48
Kategorie A	66
Texty úloh	66
Řešení úloh	70
Kategorie P	97
Texty úloh	97
Řešení úloh	114
Přípravná soustředění před 44. MMO	146
Úlohy zadané na přípravném soustředění	147
Mezinárodní střetnutí česko-polsko-slovenské	150
Texty soutěžních úloh	151
Řešení úloh	152
44. mezinárodní matematická olympiáda	157
Texty soutěžních úloh	160
Řešení úloh	161
Desátý ročník Středoevropské olympiády v informatice	171
15. mezinárodní olympiáda v informatice	173
Texty soutěžních úloh	175

O průběhu 52. ročníku matematické olympiády

Stejně jako v předešlých letech byly hlavními pořadateli 52. ročníku matematické olympiády Ministerstvo školství, mládeže a tělovýchovy ČR, Jednota českých matematiků a fyziků a Matematický ústav akademie věd ČR. Organizací soutěže je pak pověřen Ústřední výbor MO, jehož předsedou je již druhým rokem doc. dr. *Jaromír Šimša*, CSc. O organizaci kategorií A, B, C, Z5–Z9 a P se starají místopředsedové ÚV MO — po řadě dr. *Jaroslav Švrček*, CSc., doc. dr. *Pavel Tlustý*, CSc., a doc. dr. *Pavel Töpfer*, CSc. Funkci tajemníka ÚV MO vykonával i v tomto ročníku soutěže dr. *Karel Horák*, CSc.

Výběrem úloh pro jednotlivá kola každého ročníku soutěže jsou pověřeny dvě úlohové komise; jedna pro úlohy v kategoriích A, B, C a druhá pro úlohy v kategoriích Z. Obě komise se scházejí pravidelně dvakrát během školního roku (vždy v listopadu a v květnu) tak, aby včas připravily všechny podklady pro zdárný průběh následujícího ročníku matematické olympiády. Úlohy kategorie P připravili pro tento ročník soutěže doktorskí MFF UK v Praze Zdeněk Dvořák, Martin Mareš, Jan Kára a Daniel Král.

Na tomto místě dlužno podotknout, že 52. ročník matematické olympiády se rozbíhal poněkud těžkopádněji než ročníky předešlé. Bylo to způsobeno především pozdním rozesláním letáků se zadáními úloh a také komentářů k úlohám I. kola soutěže. Učitelé matematiky tak měli první dva měsíce nového školního roku k dispozici pouze zadání úloh zveřejněná v časopise *Matematika-fyzika-informatika*.

Ústřední výbor matematické olympiády se sešel během tohoto ročníku na dvou jednodenních zasedáních, a to v Praze (13. prosince 2002) a podruhé u příležitosti celostátního kola kategorií A a P v Liberci (31. března 2003). Na obou zasedáních byly kromě pravidelných bodů programu řešeny i některé výhledové otázky týkající se příštích ročníků matematické olympiády.

III. (celostátní) kolo 52. ročníku matematické olympiády (MO) v kategoriích A a P se uskutečnilo v termínu 30. března – 5. dubna 2003 v Liberci. V tomto ročníku se jeho organizace ujala liberecká pobočka JČMF

ve spolupráci s Gymnáziem F. X. Šaldy a Technickou univerzitou v Liberci, v jejichž objektech se také konala vlastní soutěž. Všichni soutěžící (44 v kategorii A a 30 v kategorii P) a další pozvaní hosté byli ubytováni hotelovém domě *Aréna* poblíž zimního stadionu.

Oficiální zahájení soutěže proběhlo v neděli 30. března v prostorách TU v Liberci za přítomnosti dr. *Jaroslava Müllnera*, náměstka ministryně školství mládeže a tělovýchovy ČR, doc. dr. *Jaroslava Vilda*, CSc., prorektora TU v Liberci, doc. dr. *Karla Segetha*, CSc., ředitele MÚ AV ČR, a dalších významných představitelů společenského života v Libereckém kraji. (Na závěr této zprávy si můžete připomenout brilantní proslov předsedy ÚV MO.)

Pořadatelé zajistili i letos pro soutěžící v obou kategoriích zajímavý doprovodný program. Vždy druhý soutěžní den navštívili soutěžící každé z kategorií radnici a vystoupili na její věž, poté si mohli vybrat mezi návštěvou liberecké zoo a blízké botanické zahrady. Soutěžící kategorie A měli možnost zhlédnout představení opery G. Rossiniho *Lazebník sevillský* v Divadle F. X. Šaldy.

Hned následující týden, tj. 7.–11. dubna, proběhlo v Kostelci nad Černými lesy výběrové soustředění před 44. MMO. Na toto soustředění bylo pozváno deset nejlepších soutěžících III. kola kategorie A. Na základě jejich výsledků na tomto soustředění a dále na základě výsledků ve II. a ve III. kole bylo šest nejlepších vybráno do českého reprezentačního družstva pro 44. MMO. O výsledcích našich soutěžících na 44. mezinárodní matematické olympiádě a na 15. mezinárodní olympiádě v informatice se dočtete v příslušných kapitolách této ročenky.

Pro nejlepší řešitele krajských kol v kategoriích B a C uspořádal ÚV MO v termínu 3.–10. června 2003 tradiční soustředění v Jevíčku. Zúčastnilo se ho 40 žáků z celé republiky. Podobně pro nejlepší řešitele kategorie A (všichni pak byli mezi účastníky celostátního kola) uspořádal ÚV MO 14.–20. září 2003 v Janských Lázních výběrové soustředění před 45. MMO.

Ústřední výbor matematické olympiády v České republice si dovoluje na tomto místě co nejupřímněji poděkovat všem těm učitelům matematiky, kteří jsou zapojeni do náročné a neocenitelné práce s matematicky talentovanými žáky našich středních a základních škol. Bez jejich pomoci si chod této nejstarší předmětové olympiády v České republice nelze vůbec představit.

Projev předsedy Ústředního výboru MO při slavnostním zahájení ústředního kola 52. ročníku v Liberci

Vážení hosté, milí soutěžící,

je mi ctí, že vás mohu všechny pozdravit jménem týmu lidí, kteří připravují pro matematickou olympiádu soutěžní úlohy. Vzletněji vyjádřeno, bohatost světa čísel a rozmanitost prostorových vztahů nám umožňují i po padesáti ročnících soutěže vymýšlet stále relativně nové a zajímavé matematické úkoly. Děláme to se stejným zaujetím, jako spisovatelé píší knihy nebo skladatelé hudební díla, jen místo čtenářů či posluchačů máme vás, mladé řešitele úloh.

Milí účastníci nejvyšší kategorie A, Váš první soutěžní den bude mít letos datum, které je zapsáno třemi prvočísly 31, 3 a 2003. Možná to pro Vás není žádná novinka, že letopočet 2003 je prvním prvočíslem nového století i tisíciletí. Druhým prvočíselným rokem bude rok 2011, v něm ale naše soutěž patrně začne v pondělí 28. března, což je datum složené. Prvočíselné datum nebudeme asi mít ani ve třetím prvočíselném roce 2017, kdy pondělí připadne na 27. března. Teprve ve čtvrtém prvočíselném roce 2027 bude pondělí 29. března. Zajímavá je okolnost, že v tomto století budeme mít právě čtrnáct prvočíselných letopočtů a že nejdéle budeme na prvočíselný Nový rok čekat rovněž čtrnáct let; za prvočíslem 2039 totiž bude následovat prvočíslo 2053. Z našeho pohledu budou bohatá devadesátá léta tohoto století, kdy lidstvo prožije teoreticky nejvyšší možný prvočíselných let v jedné dekádě. Ano, čísla 2081 a 2083 jsou prvočíselná dvojčata, stejně jako čísla 2087 a 2089. Poslední březnové pondělí bude mít v těchto čtyřech letech datum 31, 29, 31 a 28, což nám trochu kalí radost. Vy mladší, nezapomeňte prosím, že poslední prvočíselný rok v tomto století bude mít (stejně jako ve století minulém) na konci dvě devítky.

Předjeme od řeči dat a čísel k řeči jazyků. Českému termínu *prvočíslo* odpovídá anglické spojení *prime number*, nebo též samotné *prime*. Toto slovo má v angličtině i několik dalších sympatických významů, které lze vyjádřit českými podstatnými jmény *mládí*, *rozkvět* nebo přídavným jménem *nejlepší*. A proto přeji každému z vás, milí soutěžící, ať už se na výsledkové listině octnete u pořadového čísla, které je prvočíslo, nebo číslo složené, nebo číslo 1, abyste se do svých domovů vraceli s pocitem, že v Liberci to bylo *prima*. Děkuji.

Tabulka 1

Počty žáků středních škol soutěžících v I. kole 52. ročníku MO

Oblast	Kategorie								Celkem	
	A		B		C		P			
	S	U	S	U	S	U	S	U	S	U
Praha	38	32	131	98	161	109	16	16	346	255
Středočeský kraj	78	45	103	47	160	87	19	18	360	197
Jihočeský kraj	71	42	63	41	98	53	20	14	252	150
Plzeňský kraj	35	27	37	28	43	27	4	4	119	86
Karlovarský kraj	24	10	32	17	26	17	3	3	85	47
Ústecký kraj	43	25	42	28	54	33	2	2	141	88
Liberecký kraj	50	35	61	42	64	30	0	0	175	107
Královéhradecký kraj	35	28	38	26	49	34	8	3	130	91
Pardubický kraj	32	20	25	13	51	36	10	5	118	74
kraj Vysočina	56	42	60	47	70	50	15	10	201	149
Jihomoravský kraj	154	120	117	88	160	111	22	22	453	341
Zlínský kraj	78	38	64	43	95	53	5	5	242	139
Olomoucký kraj	51	22	38	16	69	34	2	2	160	74
Moravskoslezský kraj	42	35	80	50	108	79	13	13	243	177
ČR	787	521	891	584	1 208	753	139	117	3 025	1 975

Tabulka 2

Počty žáků středních škol soutěžících v II. kole 52. ročníku MO

Oblast	Kategorie								Celkem	
	A		B		C		P			
	S	U	S	U	S	U	S	U	S	U
Praha	32	9	92	41	101	45	16	10	241	105
Středočeský kraj	44	8	43	10	72	18	17	4	176	40
Jihočeský kraj	33	8	30	15	44	22	12	5	119	50
Plzeňský kraj	27	8	27	10	26	9	4	1	84	28
Karlovarský kraj	10	0	16	8	13	3	2	0	41	11
Ústecký kraj	25	1	28	6	33	8	2	0	88	15
Liberecký kraj	34	8	34	15	19	9	0	0	87	32
Královéhradecký kraj	28	7	26	10	29	10	3	1	86	28
Pardubický kraj	18	2	12	4	33	9	4	1	67	16
kraj Vysočina	36	7	39	15	40	9	8	2	123	33
Jihomoravský kraj	117	23	83	26	104	35	22	9	326	93
Zlínský kraj	37	2	42	17	50	12	0	0	129	31
Olomoucký kraj	22	4	16	8	34	14	2	1	74	27
Moravskoslezský kraj	34	6	48	21	75	23	13	4	170	54
ČR	497	93	536	206	673	226	105	38	1 811	563

S ... počet všech soutěžících

U ... počet úspěšných řešitelů

Nejúspěšnější řešitelé II. kola MO
v kategoriích A, B, C a P

Z každého kraje a z každé kategorie jsou dle dostupných výsledků uvedeni všichni úspěšní řešitelé, kteří skončili do desátého místa. Označení G znamená gymnázium, M, resp. MF zaměření studijního oboru 01 Matematika, resp. 02 Matematika a fyzika.

• • • • • **Kraj Praha** • • • • •

Kategorie A

1. *Václav Potoček*, SPŠST, Praha 1, Panská
- 2.–3. *Ondřej Čertík*, G Ch. Dopplera, Praha 5
Martin Káldy, G Ch. Dopplera, Praha 5
- 4.–5. *Pavel Kocourek*, SPŠST Panská 3, Praha 1
Jaroslav Trnka, G Praha 3, Nad Ohradou
6. *Jan Blažek*, G Ch. Dopplera, Praha 5
7. *Petr Škoda*, G Praha 8, Ústavní

Kategorie B

- 1.–2. *Petr Čermák*, G Praha 5, Mezi školami
Pavel Kocourek, SPŠST Panská 3, Praha 1
3. *Michaela Reiterová*, G Praha 6, Nad Alejí
4. *Jan Drašnar*, GJK, Parlářova, Praha 6
- 5.–8. *Jan Lachnitt*, G Ch. Dopplera, Praha 5
Tomáš Skalický, ArcG, Praha 2, Korunní
Vojtěch Smolka, ArcG, Praha 2, Korunní
Jan Tomaník, G Praha 6, Nad Štolou
9. *Helena Švandelíková*, ArcG, Praha 2, Korunní
- 10.–14. *Václav Kozmík*, G Praha 5, Mezi školami
Jaromír Mareček, G Praha 3, Sladkovského nám.

Ondřej Novák, G Ch. Dopplera, Praha 5
Petr Sýkora, G Ch. Dopplera, Praha 5
Filip Trnka, G Ch. Dopplera, Praha 5

Kategorie C

- 1.–3. *Miroslav Kolář*, G Praha 4, Na Vítězné pláni
Lukáš Malina, ZŠ Praha 5, Kuncova
Dan Marek, G Ch. Dopplera, Praha 5
4. *Jan Papež*, G Praha 8, Ústavní
- 5.–7. *Tomáš Hejda*, G Ch. Dopplera, Praha 5
Vojtěch Horký, G Praha 6, Nad Alejí
Martin Macek, G Praha 4, Písnická
- 8.–14. *Tomáš Blahovec*, G Ch. Dopplera, Praha 5
Ondřej Fremund, GJN, Praha 1, Hellichova
Jan Holinka, G Ch. Dopplera, Praha 5
Jan Kundrát, G Praha 8, Ústavní
Adam Přenosil, G Praha 3, Sladkovského nám.
Martin Ziegler, G Ch. Dopplera, Praha 5
Radek Žlebčík, G Ch. Dopplera, Praha 5

Kategorie P

1. *Jan Kadlec*, G Ch. Dopplera, Praha 5
- 2.–3. *David Matoušek*, G Ch. Dopplera, Praha 5
Petr Škoda, G Ústavní 400 Praha 8
4. *Dan Marek*, G Ch. Dopplera, Praha 5
5. *Pavel Kocourek*, SPŠST Panská 3 Praha 1
- 6.–7. *Jan Havlíček*, G Ch. Dopplera, Praha 5
Dan Lessner, G B. Bolzana, Praha 8
8. *Ondřej Zarevúcky*, G Arabská 14 Praha 6
- 9.–10. *Ján Zahornadský*, G Ch. Dopplera, Praha 5
Lukáš Turek, G Ch. Dopplera, Praha 5

Kategorie A

1. *Pavel Čížek*, GaOA Kralupy nad Vltavou
- 2.-3. *Jan Lamač*, G Mnichovo Hradiště
- 2.-3. *Pavel Brom*, G dr. J. Pekaře, Mladá Boleslav
- 4.-5. *Jiří Paleček*, G Kladno
- 4.-5. *Jan Procházka*, G dr. J. Pekaře, Mladá Boleslav
- 6.-7. *Radka Picková*, G dr. J. Pekaře, Mladá Boleslav
- 6.-7. *Jan Bureš*, G+SPŠ Čáslav
8. *Jan Váňa*, G Kutná Hora

Kategorie B

1. *Ondřej Petera*, G Kolín
2. *Petr Balek*, G Beroun
3. *Zdeněk Kabát*, G Říčany
- 4.-5. *Eva Vyhnalová*, G Benešov
Michal Veselý, G Český Brod
- 6.-7. *Emil Macků*, G Benešov
Nikola Vávra, G Poděbrady
- 8.-10. *Matěj Outlý*, G Kolín
Petr Dušek, G Český Brod
Jakub Mikulka, G Kladno, nám. E. Beneše

Kategorie C

- 1.-2. *Lukáš Hermann*, G dr. J. Pekaře, Mladá Boleslav
Marek Scholz, G Neratovice
- 3.-4. *Václav Gergelits*, G Benešov
Josef Schmidt, G Benešov
- 5.-7. *Jan Korběl*, G Říčany
Marek Sacha, G Říčany
Renata Dolejšová, G Kladno, nám. E. Beneše
8. *Jan Matoušek*, G Kolín
- 9.-10. *Martin Kadlec*, G Kladno, nám. E. Beneše
Petr Šafránek, G Poděbrady

Kategorie P

1. *Pavel Čížek*, GaOA Kralupy nad Vltavou
2. *Martin Kruliš*, G Kolín
3. *Jiří Paleček*, G Kladno, nám. E. Beneše
4. *Jiří Hyldebrant*, G Vlašim

• • • • • Jihočeský kraj • • • • •

Kategorie A

1. *Jan Kouba*, G Český Krumlov
2. *Milan Straka*, G Strakonice
3. *Dana Bartošová*, G Jindřichův Hradec
- 4.–5. *Jiří Danihelka*, SPŠ Písek
Pavel Kubas, G Jindřichův Hradec
6. *Ondřej Šedivý*, G České Budějovice, Jírovcova
- 7.–8. *Zuzana Kožíšková*, G Tábor
Ondřej Prašnička, G České Budějovice, Jírovcova

Kategorie B

1. *Karel Vácha*, G Český Krumlov
2. *Václavík Jiří*, G Strakonice
- 3.–5. *Radek Pustina*, G České Budějovice, Česká
Markéta Zíková, G České Budějovice, Jírovcova
Jiří Koubek, G Týn nad Vltavou
- 6.–7. *Karel Chromý*, G České Budějovice, Jírovcova
Václav Eigner, G Strakonice
- 8.–11. *Štěpánka Králová*, G Týn nad Vltavou
Michal Bizzarri, G Týn nad Vltavou
Hana Nůsková, G Strakonice

Kategorie C

- 1.–2. *Jan Zálaha*, G Písek
Tomáš Chmelík, Biskupské G, České Budějovice

- 3.-4. *Jiří Kníže*, G Strakonice
Josef Orel, G Tábor
- 5.-8. *Jan Klouda*, G České Budějovice, Jírovcova
Eliška Lehečková, G Jindřichův Hradec
Karel Fišer, G Jindřichův Hradec
Tomáš Jindra, G České Budějovice, Česká
9. *Zuzana Bainová*, G České Budějovice, Jírovcova
- 10.-13. *Vojtěch Dušek*, G České Budějovice, Jírovcova
Petr Anděl, G Český Krumlov
Michal Dojčar, G Prachatice
Vojtěch Kulhavý, G Jindřichův Hradec

Kategorie P

1. *Milan Straka*, G Strakonice
2. *Jiří Danihelka*, SPŠ Písek
3. *Jan Kouba*, G Český Krumlov
4. *Martin Pilát*, G České Budějovice, Česká
5. *Josef Pitera*, G Strakonice

• • • • • Plzeňský kraj • • • • •

Kategorie A

1. *Michal Kvíz*, G Plzeň, Mikulášské nám.
2. *Jiří Reitspies*, G Plzeň, Mikulášské nám.
3. *Josef Mládek*, G Plzeň, Mikulášské nám.
- 4.-5. *Jiří Ajgl*, G Plzeň, Mikulášské nám.
Eva Wagnerová, G Plzeň, Mikulášské nám.
- 6.-7. *Jaroslav Svoboda*, G Plzeň, Mikulášské nám.
Václav Varvařovský, G Plzeň, Mikulášské nám.
8. *Petr Cvachovec*, G Plzeň, Mikulášské nám.

Kategorie B

1. *Jakub Bulín*, G Plzeň, Mikulášské nám.
- 2.-3. *Pavel Hudák*, Masarykovo G, Plzeň
Libor Nolč, G Plzeň, Mikulášské nám.

4. *Jindřich Pergler*, G Klatovy
5. *Tomáš Ibehej*, G Plzeň, Mikulášské nám.
6. *Ondřej Hort*, G Plzeň, Mikulášské nám.
7. *Aleš Turnovec*, G Plzeň, Mikulášské nám.
- 8.–9. *Jan Bulánek*, G Klatovy
Jiří Pouska, G Plzeň, Mikulášské nám.
10. *Petr Vaško*, Masarykovo G, Plzeň

Kategorie C

1. *Vojtěch Kolomičenko*, G Plzeň, Mikulášské nám.
2. *Marie Dostálová*, G Stříbro
3. *Vladimír Širotek*, G L. Pika, Plzeň
- 4.–6. *Tomáš Jirotko*, G Klatovy
Miloslav Kepka, G L. Pika, Plzeň
Tomáš Kouřim, G L. Pika, Plzeň
- 7.–9. *Martin Dvořák*, G Plzeň, Mikulášské nám.
Marie Matějovicová, G Plzeň, Mikulášské nám.
Jan Šípek, G L. Pika, Plzeň

Kategorie P

1. *Stanislav Haviar*, G Klatovy

• • • • • **Karlovarský kraj** • • • • •

Kategorie B

1. *Eva Černohorská*, První české G Karlovy Vary
2. *Petr Žáček*, SCHŠ Cheb
3. *Tereza Volná*, G Mariánské Lázně
- 4.–7. *Jiří Pöpperl*, G Sokolov
Radek Pupák, G Cheb
Jan Rieger, G Ostrov
Štěpán Trojáněk, G Cheb
8. *Michaela Bublová*, SCHŠ Cheb

Kategorie C

1. *Eva Černohorská*, První české G Karlovy Vary
2. *Jan Heczko*, G Cheb
3. *Jakub Irber*, G Cheb

• • • • • Ústecký kraj • • • • •

Kategorie A

1. *Vít Šípál*, G Ústí nad Labem, Jateční

Kategorie B

1. *Daniel Petřík*, G Most
2. *Jiří Dvořák*, G Roudnice nad Labem
- 3.-4. *Michal Valach*, SPŠ a VOŠ Chomutov
Jakub Florián, G Podbořany
- 5.-6. *Jan Semerád*, G Teplice, Čs. Dobrovolců
Jan Patera, G Most

Kategorie C

1. *Vojtěch Kaiser*, G Ústí nad Labem, Stavbařů
2. *Michal Pelc*, G Most
3. *Jakub Klener*, G J. Jungmanna, Litoměřice
4. *Zdeněk Fikar*, G Ústí nad Labem, Jateční
5. *Věra Skorkovská*, G Teplice, Čs. Dobrovolců 11
- 6.-7. *Michaela Tichá*, G Chomutov
Jiří Šarkady, G J. Jungmanna, Litoměřice
8. *Jan Krahulík*, G V. Hlavatého, Louny

• • • • • Liberecký kraj • • • • •

Kategorie A

- 1.-2. *Petr Piša*, G Liberec, Jeronýmova

- Marek Sobota*, G Liberec, Jeronýmova
3.–4. *Lukáš Bajer*, GFXŠ, Liberec
Michal Kollert, GFXŠ, Liberec

Kategorie B

1. *Šárka Vejvodová*, GFXŠ, Liberec
2. *Kateřina Kapletová*, GFXŠ, Liberec
3.–4. *Zdeněk Böh*m, GFXŠ, Liberec
Lukáš Ježek, GFXŠ, Liberec
5.–8. *Jaroslav Jirá*k, GFXŠ, Liberec
Marie Mártonová, GFXŠ, Liberec
Martin Otava, GFXŠ, Liberec
Ondřej Soroka, GFXŠ, Liberec
9.–15. *Domín*ik Fišer, G Jablonec, U Balvanu
Jakub Hušek, GFXŠ, Liberec
Tomáš Jakoubek, G Liberec, Jeronýmova
Čeněk Jirsák, GFXŠ, Liberec
Martin Morávek, GFXŠ, Liberec
Vojtěch Růžička, GFXŠ, Liberec
Klára Vitoušová, GFXŠ, Liberec

Kategorie C

1. *Michal Vaner*, G Turnov
2. *Jan Iser*, G U Balvanu Jablonec
3. *Ondřej Kudláček*, G Liberec, Jeronýmova
4. *Pavel Staněk*, G Jablonec, U Balvanu
5. *Kateřina Teplá*, G Liberec, Jeronýmova
6. *Jan Hrnčíř*, GFXŠ, Liberec
7. *Lucie Krtoušová*, GFXŠ, Liberec
8. *Martin Vlček*, G Turnov
9. *Jiří Toman*, G Liberec, Jeronýmova

Kategorie P

1. *Jiří Schejbal*, G Turnov

Kategorie A

1. *Jan Moláček*, GJKT, Hradec Králové
2. *Miroslav Hejna*, GFMP, Rychnov nad Kněžnou
3. *Jan Prachař*, GFMP, Rychnov nad Kněžnou
- 4.–5. *Jan Ondruš*, GFMP, Rychnov nad Kněžnou
Jan Schindler, G B. Němcové, Hradec Králové
- 6.–7. *Ivo Machek*, GJKT, Hradec Králové
Martin Selecký, G B. Němcové, Hradec Králové

Kategorie B

1. *Radek Moravec*, G B. Němcové, Hradec Králové
- 2.–3. *Michal Kopecký*, GJKT, Hradec Králové
Jan Marek, G B. Němcové, Hradec Králové
4. *Josef Beneš*, G B. Němcové, Hradec Králové
5. *Filip Studnička*, G B. Němcové, Hradec Králové
- 6.–7. *Kateřina Fišerová*, LG Jičín
Šárka Pátková, GJKT, Hradec Králové
- 8.–10. *Aleš Balcar*, G B. Němcové, Hradec Králové
Julie Brátová, G Náchod
Ondřej Novák, G B. Němcové, Hradec Králové

Kategorie C

1. *Libor Šimůnek*, GJKT, Hradec Králové
- 2.–3. *Karel Vašata*, G B. Němcové, Hradec Králové
Jan Voltr, GJKT, Hradec Králové
- 4.–5. *Filip Andrš*, G Dobruška
Věra Peterová, G Dobruška
- 6.–9. *Petr Adlaf*, GJKT, Hradec Králové
Martin Doležal, G Nový Bydžov
Adam Janáček, GaSOŠ Úpice
Vendula Uchytlová, GJKT, Hradec Králové
10. *Vojtěch Beneš*, GJKT, Hradec Králové

Kategorie P

1. *Jan Prachař*, GFMP, Rychnov nad Kněžnou

..... Pardubický kraj

Kategorie A

1. *Petr Pošta*, G Pardubice, Dašická
2. *Petr Rezek*, G Litomyšl

Kategorie B

1. *M. Dobroucký*, G Moravská Třebová
2. *B. Scholleová*, G Pardubice, Dašická
- 3.–4. *K. Hauptová*, G Vysoké Mýto
M. Benešová, G Polička

Kategorie C

- 1.–2. *M. Scholle*, G Pardubice, Dašická
K. Stodolová, G Polička
3. *H. Šediváková*, G Pardubice, Dašická
4. *Z. Novák*, G Hlinsko
- 5.–6. *T. Hubík*, G Pardubice, Dašická
T. Klímošová, G Lanškroun
7. *J. Škořepa*, G Pardubice, Dašická
- 8.–9. *M. Ferjenčík*, G Přelouč
L. Šerá, G Jevíčko

Kategorie P

1. *Martin Dobroucký*, 6.AV, G Moravská Třebová

Kategorie A

1. *Radek Mlada*, G Pelhřimov
2. *Dominik Macáš*, G Bystřice nad Pern.
- 3.–4. *Martin Havlena*, G Jihlava
Michal Zavadil, SPŠ Jihlava
5. *Václav Řezníček*, G Třebíč
- 6.–7. *Martin Bartušek*, G Jihlava
Michal Fikrle, G Pelhřimov

Kategorie B

- 1.–3. *Ondřej Křivánek*, G Třebíč
Rostislav Kváš, G Jihlava
Petr Lavička, G Jihlava
4. *Petr Houštěk*, G Pelhřimov
- 5.–6. *Michaela Krpálková*, G Jihlava
Martin Tomec, G Třebíč
- 7.–8. *Jaroslav Kezníkl*, G Žďár nad Sáz.
Jan Novotný, G Havlíčkův Brod
- 9.–10. *Kateřina Kujanová*, G Žďár nad Sáz.
Aleš Řáda, G Pelhřimov

Kategorie C

1. *Karel Lavička*, G Jihlava
2. *Ondřej Hoferek*, G Žďár nad Sáz.
- 3.–4. *Marie Kolářová*, G Žďár nad Sáz.
Filip Rosenkranc, G Třebíč
5. *Jan Šedo*, SPŠ Jihlava
6. *Jiří Svoboda*, G Humpolec
- 7.–9. *Vojtěch Fišar*, G Žďár nad Sáz.
Hana Mádrová, G Jihlava
Iveta Selingerová, G Jihlava

Kategorie P

- 1.–2. *Jakub Sedlák*, G Jihlava
Martin Tomec, G Třebíč

• • • • • Jihomoravský kraj • • • • •

Kategorie A

1. *Marek Krčál*, G Brno, tř. Kpt. Jaroše
2.–4. *Vítězslav Kala*, G Brno, tř. Kpt. Jaroše
Veronika Trnková, G Brno, tř. Kpt. Jaroše
Pavel Troubil, G Brno, tř. Kpt. Jaroše
5. *Jaroslav Zůda*, G Brno, tř. Kpt. Jaroše
6. *Milan Werl*, G Brno, tř. Kpt. Jaroše
7.–9. *Václav Brožek*, G Brno, Barvičova
Jaromír Kuben, G Brno, tř. Kpt. Jaroše
Jan Novotný, G Brno, tř. Kpt. Jaroše
10. *Jiří Zelinka*, G Brno, tř. Kpt. Jaroše

Kategorie B

1. *Michal Rychnovský*, G Brno, tř. Kpt. Jaroše
2.–3. *Vojtěch Kubáň*, G Brno, tř. Kpt. Jaroše
Michaela Neumannová, G Hustopeče
4. *Tomáš Hošek*, G Brno, tř. Kpt. Jaroše
5.–8. *Kateřina Fialová*, G Brno, Elgartova
Michala Henzlová, G Brno, Barvičova
Pavel Turčinek, G Brno, Vídeňská
Ondřej Zapletal, G Brno, Křenová
9.–11. *Ondřej Hotový*, G Brno, tř. Kpt. Jaroše
Ondřej Maňák, G Brno, Táborská
Jiří Zelinka, G Brno, tř. Kpt. Jaroše

Kategorie C

1. *Jaromír Kuben*, G Brno, tř. Kpt. Jaroše
2.–5. *Jakub Opršal*, G Brno, tř. Kpt. Jaroše

- Alexandr Pícha*, G Brno, tř. Kpt. Jaroše
Jakub Pracný, G Brno, tř. Kpt. Jaroše
Vojtěch Říha, G Brno, tř. Kpt. Jaroše
6.–7. *Jiří Řihák*, G Brno, tř. Kpt. Jaroše
Jiří Zelinka, G Brno, tř. Kpt. Jaroše
8.–9. *Petr Kadlec*, G Brno, tř. Kpt. Jaroše
Zbyněk Konečný, G Brno, tř. Kpt. Jaroše
10.–15. *Ondřej Budík*, G Brno, tř. Kpt. Jaroše
Lucie Fabriková, G Brno, tř. Kpt. Jaroše
Marek Filakovský, G Brno, tř. Kpt. Jaroše
Martin Kočí, G Brno, tř. Kpt. Jaroše
Martin Křivánek, G Brno, tř. Kpt. Jaroše
Jakub Skalický, G Brno, Žižkova

Kategorie P

1. *Marek Krčál*, G Brno, tř. Kpt. Jaroše
2. *Jiří Štěpánek*, G Brno, tř. Kpt. Jaroše
3.–4. *Martin Lopatář*, G Brno, tř. Kpt. Jaroše
Jan Hladký, G Brno, tř. Kpt. Jaroše
5. *Martin Křivánek*, G Brno, tř. Kpt. Jaroše
6. *Martin Vejnár*, G Brno, tř. Kpt. Jaroše
7.–8. *Kryštof Hoder*, G Brno, tř. Kpt. Jaroše
Sven Dražan, G Brno, tř. Kpt. Jaroše
9. *Jana Fabriková*, G Brno, tř. Kpt. Jaroše

• • • • • Zlínský kraj • • • • •

Kategorie A

1. *František Konopecký*, G L. Jaroše, Holešov
2. *Marek Pechal*, G Zlín, Lesní čtvrť
3.–4. *Martin Cetkovský*, G Zlín, Lesní čtvrť
Martin Dungal, G Kroměříž
5.–6. *Radovan Polanský*, GJAK, Uherský Brod
Martin Horáček, G Zlín, Lesní čtvrť

Kategorie B

- 1.–2. *Stanislav Basovník*, G Kroměříž
František Konopecký, G L. Jaroše, Holešov
3. *Aleš Holub*, G Uherské Hradiště
4. *Michal Humpula*, GJAK Uherský Brod
- 5.–6. *Miroslav Blažek*, GFP, Valašské Meziříčí
Petra Klimentová, GFP, Valašské Meziříčí
7. *Martin Koníček*, GJAK, Uherský Brod
- 8.–10. *Ondřej Bílka*, G Zlín, Lesní čtvrť
Zbyněk Savara, GJAK, Uherský Brod
Václav Slimáček, GFP, Valašské Meziříčí

Kategorie C

1. *Marek Pechal*, G Zlín, Lesní čtvrť
2. *Jan Váňa*, G Zlín, Lesní čtvrť
3. *Jiří Machálek*, G L. Jaroše, Holešov
4. *Kateřina Běhmová*, G Rožnov pod Radhoštěm
- 5.–6. *Jan Pelc*, GJAK, Uherský Brod
Václav Výmola, GJAK, Uherský Brod
- 7.–10. *Michal Čudrnák*, G L. Jaroše, Holešov
Marek Chytil, G Zlín, Lesní čtvrť
Michal Studený, G Zlín, nám. TGM
Pavel Šalom, G Rožnov pod Radhoštěm

• • • • • Olomoucký kraj • • • • •

Kategorie A

1. *Helena Kubátová*, G Olomouc-Hejčín
2. *Pavel Kašpar*, VOŠ a SPŠ, Šumperk
3. *Veronika Chromčíková*, GJŠ, Přerov
4. *Lukáš Perůtka*, G Hranice

Kategorie B

1. *Jan Průša*, GJW, Prostějov

2. *Jaroslav Fibichr*, G Šternberk
3. *Alena Sejkorová*, G Jeseník
- 4.–8. *Michal Garlík*, G Olomouc-Hejčín
Jana Sedláčková, GJŠ, Přerov
Jiří Horký, GJŠ, Přerov
Zuzana Dřízgová, G Lipník nad Bečvou
Martin Rumlík, G Kojetín

Kategorie C

1. *Jiří Zajíček*, RG Prostějov
- 2.–3. *Zdeněk Černohouz*, SGO, Olomouc
Jan Šrámek, GJŠ, Přerov
- 4.–6. *Martina Miková*, G Olomouc-Hejčín
Martina Brijarová, GJW, Prostějov
Martin Přemyslovský, GJŠ, Přerov
7. *Jiřina Mertová*, G Šumperk
- 8.–12. *Rostislav Halaš*, GJW, Prostějov
Michal Hapala, G Hranice
Zdeněk Mlčoch, G Zábřeh
Jana Dvořáková, G Zábřeh
Tomáš Jílek, VOŠ a SPŠ, Šumperk

Kategorie P

1. *Jan Matoušek*, GJW, Prostějov

• • • • • Moravskoslezský kraj • • • • •

Kategorie A

1. *Pavel Ludvík*, GMK, Bílovec
2. *Václav Cviček*, G P. Bezruče, Frýdek-Místek
3. *Tomáš Gavenčíak*, GMK, Bílovec
4. *Tomáš Staněk*, G Ostrava-Zábřeh, Volgogradská
- 5.–6. *Aleš Havel*, G Frenštát pod Radhoštěm
Tomáš Andrášek, MOG, Bruntál
7. *Vojtěch Skubanič*, GMK, Bílovec

- 8.–10. *Ondřej Májek*, GMK, Bílovec
Pavel Jež, G P. Bezruče, Frýdek-Místek
Jakub Gemrot, GMK, Bílovec

Kategorie B

1. *Michal Sikora*, G Třinec
2. *Zuzana Safernová*, GMK, Bílovec
3. *Pavel Žižka*, G Třinec
4.–5. *Kateřina Váňová*, G Ostrava-Poruba, Čs. exilu
Alexandr Žák, G Ostrava-Poruba, Čs. exilu
6.–9. *Jan Richter*, Masarykovo G Příbor
Jakub Dvorský, GMK, Bílovec
Silvie Kafková, G P. Bezruče, Frýdek-Místek
Pavel Motloch, G P. Bezruče, Frýdek-Místek
10.–14. *Eva Sobolová*, G Třinec
Kamil Babula, G Třinec
Jakub Rajman, SPŠ Moravská Ostrava
Lukáš Kuna, SPŠ Moravská Ostrava
Štěpánka Mohylová, G Ostrava-Poruba, Čs. exilu

Kategorie C

- 1.–2. *Jaroslav Hančl*, GMK, Bílovec
Jan Uhlík, GMK, Bílovec
3. *Rostislav Měch*, G Frenštát p. R.
4. *Miloslav Holík*, GMK, Bílovec
5.–12. *Lenka Borišová*, G Opava
Jakub Křístek, G Ostrava-Poruba, Čs. exilu
Michael Kučera, GMK, Bílovec
Pavel Motloch, G P. Bezruče, Frýdek-Místek
Dita Mušálková, G Třinec
Petr Sittek, G P. Bezruče, Frýdek-Místek
Roman Vybíral, G Františka Hajdy
Josef Žabenský, GMK, Bílovec

Kategorie P

1. *Václav Cviček*, G P. Bezruč, Frýdek-Místek
2. *Josef Toman*, GMK, Bílovec
3. *Pavel Motloch*, G P. Bezruč, Frýdek-Místek
4. *Tomáš Gavenčiak*, GMK, Bílovec

Výsledky celostátního kola 52. ročníku MO
kategorie A

Vítězové

1. <i>Pavel Čížek</i> , 8/8, GaOA Kralupy nad Vltavou	37 b.
2. <i>Marek Krčál</i> , 4/4, G Brno, tř. Kpt. Jaroše	36 b.
3. <i>Petr Pošta</i> , 6/6, G Pardubice, Dašická	29 b.
4. <i>Jaromír Kuben</i> , 1/4, G Brno, tř. Kpt. Jaroše	28 b.
5.–6. <i>Václav Cviček</i> , 6/6, G P. Bezruče, Frýdek-Místek	27 b.
<i>Pavel Kocourek</i> , 2/4, SPŠST, Praha 1, Panská	27 b.
7.–8. <i>Tomáš Gavenčiak</i> , 3/4, GMK, Bílovec	26 b.
<i>Martin Káldy</i> , 4/4, GChD, Praha 5	26 b.
9. <i>Vítězslav Kala</i> , 3/4, G Brno, tř. Kpt. Jaroše	25 b.
10. <i>Miroslav Hejna</i> , 8/8, GFMP Rychnov nad Kněžnou	24 b.
11. <i>Jan Moláček</i> , 3/4, GJKT, Hradec Králové	23 b.

Další úspěšní řešitelé

12.–13. <i>František Konopecký</i> , 6/8, G Holešov	22 b.
<i>Jan Prachař</i> , 8/8, GFMP, Rychnov nad Kněžnou	22 b.
14.–16. <i>Marek Pechal</i> , 5/8, G Zlín, Lesní čtvrť	21 b.
<i>Milan Straka</i> , 4/4, G Strakonice	21 b.
<i>Veronika Trnková</i> , 4/4, G Brno, tř. Kpt. Jaroše	21 b.
17. <i>Pavel Kubas</i> , 8/8, G Jindřichův Hradec	19 b.
18. <i>Jiří Ajgl</i> , 8/8, G Plzeň, Mikulášské nám.	18 b.
19.–20. <i>Pavel Ludvík</i> , 4/4, GMK, Bílovec	17 b.
<i>Václav Potoček</i> , 3/4, SPŠST, Praha 1, Panská	17 b.
21. <i>Jiří Danihelka</i> , 4/4, SPŠ a VOŠ Písek	16 b.
22. <i>Pavel Dvořák</i> , 4/4, G Brno, tř. Kpt. Jaroše	15 b.

Výsledky celostátního kola 52. ročníku MO
kategorie P

Vítězové

1. <i>Jan Kadlec</i> , 4/4, GChD, Praha 5	37 b.
2. <i>Milan Straka</i> , 4/4, G Strakonice	36 b.
3. <i>Tomáš Gavenčiak</i> , 3/4, GMK, Bílovec	34 b.
4. <i>Marek Krčál</i> , 8/8, G Brno, tř. Kpt. Jaroše	31 b.
5. <i>Martin Kruliš</i> , 8/8, G Kolín	30 b.
6. <i>Kryštof Hoder</i> , 7/8, G Brno, tř. Kpt. Jaroše	29 b.
7. <i>Pavel Čížek</i> , 8/8, GaOA Kralupy nad Vltavou	28 b.
8. <i>Jiří Danihelka</i> , 4/4, SPŠ Písek	27 b.

Další úspěšní řešitelé

9.–10. <i>Václav Cviček</i> , 6/6, G P. Bezruče, Frýdek-Místek	26 b.
9.–10. <i>Martin Vejnár</i> , 6/8, G Brno, tř. Kpt. Jaroše	26 b.
11.–12. <i>Sven Dražan</i> , 7/8, G Brno, tř. Kpt. Jaroše	24 b.
11.–12. <i>Jiří Schejbal</i> , 7/8, G Turnov	24 b.
13.–14. <i>Dan Marek</i> , 1/4, GChD, Praha 5	23 b.
13.–14. <i>Jiří Paleček</i> , 4/4, G Kladno, nám. E. Beneše	23 b.

Kategorie C

Texty úloh

C – I – 1

Z pěti jedniček, pěti dvojek, pěti trojek, pěti čtyřek a pěti pětetek sestavte pět navzájem různých pětimístných čísel tak, aby jejich součet byl co největší.

(J. Šimša)

C – I – 2

Je dán trojúhelník ABC s ostrými vnitřními úhly při vrcholech A a B . Označme Q průsečík těžnice AD s výškou CP a E patu kolmice z bodu D na stranu AB . Dále nechť R je bod na polopřímce opačné k PC takový, že $|PR| = |CQ|$. Dokažte, že přímky AD a RE jsou různoběžné a že jejich průsečík leží na kolmici k přímce AB procházející bodem B .

(J. Švrček)

C – I – 3

Předpokládejme, že každá ze dvou bank A a B bude mít po následující dva roky stálou roční úrokovou míru. Kdybychom uložili $5/6$ našich úspor u banky A a zbytek u banky B , vzrostly by naše úspory po jednom roce na 67 000 Kč a po dvou letech na 74 900 Kč. Kdybychom však uložili $5/6$ našich úspor u banky B a zbytek u banky A , vzrostly by naše úspory po jednom roce na 71 000 Kč. Na jakou částku by se v takovém případě naše úspory zvýšily po dvou letech?

(J. Šimša)

C – I – 4

Sestrojte lichoběžník $ABCD$ s výškou 3 cm a shodnými stranami BC , CD a DA , pro který platí: Na základně AB existuje takový bod E , že úsečka DE má délku 5 cm a dělí lichoběžník na dvě části se stejnými obsahy.

(E. Kováč)

C – I – 5

K přirozenému číslu m zapsanému stejnými číslicemi jsme přičetli čtyřmístné přirozené číslo n . Získali jsme čtyřmístné číslo s opačným pořadím číslic, než má číslo n . Určete všechny takové dvojice čísel m a n .

(J. Zhouf)

C – I – 6

V rovině je dána přímka p a kružnice k . Sestrojte takový trojúhelník ABC , aby k byla kružnicí jemu vepsanou, její střed ležel ve čtvrtině jeho těžnice na stranu AB a aby vrchol C ležel na přímce p . Proveďte diskusi o počtu řešení v závislosti na vzájemné poloze přímky p a kružnice k .

(P. Černek)

C – S – 1

Odtrhneme-li od libovolného alespoň dvojmístného přirozeného čísla číslici na místě jednotek, dostaneme číslo o jednu číslici „kratší“. Najděte všechna původní čísla, která se rovnají absolutní hodnotě rozdílu druhé mocniny „kratšího“ čísla a druhé mocniny odtržené číslice. (J. Zhouf)

C – S – 2

Na straně CD čtverce $ABCD$ je zvolen bod E tak, že úhel DAE má velikost 30° . Bod P je patou kolmice vedené bodem B na přímkou AE , bod Q patou kolmice vedené bodem C na přímkou BP . Rozhodněte, zda je obsah lichoběžníku $PQCE$ menší než třetina obsahu čtverce $ABCD$.

(L. Boček)

C – S – 3

Z pěti jedniček, pěti dvojek, pěti trojek, pěti čtyřek a pěti pěték sestavíme pět pětimístných čísel, která se čtou zepředu stejně jako zezadu (např. 32 223), a pak tato čísla sečteme. Jakou nejmenší a jakou největší hodnotu může mít výsledný součet?

(J. Šimša)

C – II – 1

Najděte nejmenší přirozené číslo n , pro které je součin

$$2\,003 \cdot 2\,004 \cdot 2\,005 \cdot \dots \cdot (2\,003 + n)$$

dělitelný všemi dvojmístnými prvočísly.

(*J. Šimša*)

C – II – 2

V rovině je dána úsečka AP . Sestrojte pravidelný šestiúhelník $ABCDEF$ tak, aby bod P byl středem jeho strany DE .

(*J. Švrček*)

C – II – 3

Kdyby Karel půjčil jednomu známému p tisíc Kč s úrokem $p\%$ a druhému známému q tisíc Kč s úrokem $q\%$, kde p a q jsou celá čísla, přinesly by obě půjčky Karlovi stejný zisk, jako kdyby jedné osobě půjčil celkovou částku s úrokem $(p + 2,4)\%$. Kdyby půjčil jednomu známému p tisíc Kč s úrokem $2p\%$ a druhému známému q tisíc Kč s úrokem $2q\%$, přinesly by mu tyto půjčky stejný zisk, jako kdyby jedné osobě půjčil celkovou částku s úrokem $(p + 5,8)\%$. Určete čísla p a q . (*J. Šimša, J. Zhouf*)

C – II – 4

Určete délku ramen rovnoramenného lichoběžníku se základnami délek 10 a 12 tak, aby délky všech jeho stran i úhlopříček byly vyjádřeny celými čísly.

(*P. Černek*)

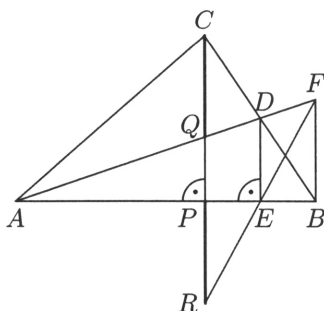
Řešení úloh

C – I – 1

Největší možný součet by vytvořila pětice čísel 54321, 54321, 54321, 54321, 54321. Jelikož mají být ale čísla navzájem různá, pokusíme se změnit tuto pětici tak, aby se nenarušilo trojčíslí 543, tj. aby změna součtu byla co nejmenší. Tak ale budou ještě dvě z pěti čísel stejná, neboť z číslic 1, 2 je možné sestavit pouze čtyři různá dvojčíslí 11, 12, 21, 22. Změníme proto jedno trojčíslí 543 na 542 tak, že zaměníme číslici 2 číslici 3 na místě desítek. Stejně tak na místě jednotek nemůže být všech pět jedniček, protože by poslední trojčíslí nejméně tři pětimístních čísel bylo 321. Vyměníme proto číslici 1 z místa jednotek s číslici 2 z místa stovek a to proto, aby změna součtu pětice čísel byla co nejmenší. Po těchto výměnách mohou být poslední dvojčíslí pěti čísel tato: 31, 22, 21, 21, 11, nebo 31, 21, 21, 21, 12, nebo 32, 21, 21, 21, 11. Snažíme se nyní rozmístit tato dvojčíslí za trojčíslí 543, 543, 543, 543, 542. Zjistíme, že vyhovuje pouze první pětice dvojčíslí. Hledaná pětice pětimístních čísel s největším možným součtem je 54331, 54322, 54321, 54311, 54221.

C – I – 2

Ze zadání víme, že $|PR| = |CQ|$, proto je i $|QR| = |CP|$ (obr. 1). Úsečka DE je střední příčkou trojúhelníku CPB , proto je $|DE| = \frac{1}{2}|CP|$.



Obr. 1

Je tedy také $|DE| = \frac{1}{2}|QR|$. Protože je $DE \parallel QR$, nemůžou být úsečky RE a QD rovnoběžné (jinak by byl $REDQ$ rovnoběžník a platilo by $|DE| = |QR|$). Proto se přímky RE a QD protínají v bodě, který je na

obrázku označen jako F , a úsečka DE je střední příčkou trojúhelníků CPB a QRF , jejichž strany CP a QR leží na stejné přímce. Proto je vzdálenost bodů F a B od přímky CR stejná, neboli přímky CR a FB jsou rovnoběžné, a tudíž přímka FB je (stejně jako přímka CR) kolmá k přímce AB .

C – I – 3

Nechť naše výchozí úspory činí x Kč a necht' roční úroková míra u banky A (banky B) je $p\%$ ($q\%$), tj. vklad u banky A (banky B) vzroste po jednom roce a -krát (b -krát), kde $a = 1 + p/100$ ($b = 1 + q/100$). Podle zadání platí

$$\begin{aligned}\left(\frac{5}{6} \cdot x\right) \cdot a + \left(\frac{1}{6} \cdot x\right) \cdot b &= 67\,000, \\ \left[\left(\frac{5}{6} \cdot x\right) \cdot a\right] \cdot a + \left[\left(\frac{1}{6} \cdot x\right) \cdot b\right] \cdot b &= 74\,900, \\ \left(\frac{1}{6} \cdot x\right) \cdot a + \left(\frac{5}{6} \cdot x\right) \cdot b &= 71\,000,\end{aligned}$$

a po úpravě

$$\begin{aligned}5 \cdot \frac{xa}{6} + \frac{xb}{6} &= 67\,000, \\ 5 \cdot \frac{xa}{6} \cdot a + \frac{xb}{6} \cdot b &= 74\,900, \\ \frac{xa}{6} + 5 \cdot \frac{xb}{6} &= 71\,000.\end{aligned}$$

Označíme-li $u = \frac{1}{6}xa$ a $v = \frac{1}{6}xb$, přejdou první a třetí rovnice v soustavu

$$\begin{aligned}5u + v &= 67\,000, \\ u + 5v &= 71\,000,\end{aligned}$$

ze které vychází $u = 11\,000$ a $v = 12\,000$. Protože $a = 6u/x$ a $b = 6v/x$, lze druhou rovnici soustavy zapsat jako

$$\frac{5}{6} \cdot x \cdot \frac{36u^2}{x^2} + \frac{1}{6} \cdot x \cdot \frac{36v^2}{x^2} = 74\,900,$$

neboli

$$\frac{30u^2 + 6v^2}{x} = 74\,900,$$

odkud pro $u = 11\,000$ a $v = 12\,000$ vychází $x = 60\,000$, tudíž

$$a = \frac{6u}{x} = \frac{66\,000}{60\,000} = 1,1,$$

$$b = \frac{6v}{x} = \frac{72\,000}{60\,000} = 1,2.$$

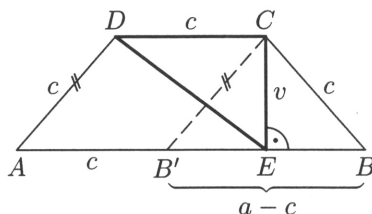
Hledaná částka je proto rovna

$$\left(\frac{1}{6} \cdot x \cdot a^2 + \frac{5}{6} \cdot x \cdot b^2\right) \text{ Kč} = (10\,000 \cdot 1,1^2 + 50\,000 \cdot 1,2^2) \text{ Kč} =$$

$$= 84\,100 \text{ Kč}.$$

C – I – 4

Rozbor: Označíme-li $|AB| = a$, $|CD| = c$ a výšku lichoběžníku v (obr. 2),



Obr. 2

můžeme pro jeho obsah S psát

$$S = \frac{1}{2}(a + c)v.$$

Obsah trojúhelníku AED je podle zadání roven

$$\frac{|AE| \cdot v}{2} = \frac{1}{2} \cdot S = \frac{1}{2} \cdot \frac{1}{2}(a + c)v,$$

odkud plyne, že $|AE| = \frac{1}{2}(a + c)$ (tj. úsečka AE má délku stejnou jako střední příčka lichoběžníku $ABCD$). Protože bod E leží na úsečce AB , platí

$$|EB| = |AB| - |AE| = a - \frac{1}{2}(a + c) = \frac{1}{2}(a - c),$$

takže je $a > c$. Označíme-li B' bod úsečky AB , pro který je $|AB'| = c$, bude $|B'B| = a - c$, a protože hledaný lichoběžník $ABCD$ je rovnoramenný, je rovnoramenný i trojúhelník $B'BC$, takže střed E úsečky $B'B$ je zároveň patou výšky z z vrcholu C na základnu AB (obr. 2). Pomocí Pythagorovy věty vypočteme, že

$$c = \sqrt{|DE|^2 - v^2} = \sqrt{5^2 - 3^2} \text{ cm} = 4 \text{ cm}.$$

Popis konstrukce:

1. $\triangle DEC$; $|DC| = 4 \text{ cm}$, $|CE| = 3 \text{ cm}$, $|\sphericalangle ECD| = 90^\circ$;
2. p ; $p \parallel CD$, $E \in p$;
3. $k(D, 4 \text{ cm})$, $l(C, 4 \text{ cm})$;
4. A ; $A \in p \cap k$, úhel ADC je tupý;
5. B ; $B \in p \cap l$, úhel BCD je tupý.

Úloha má jediné řešení.

C – I – 5

Nechť je číslo $n = \overline{abcd} = 1000a + 100b + 10c + d$, kde $a, b, c, d \in \{0, 1, \dots, 9\}$, $a \neq 0$. Číslo $m + n$ je čtyřmístné, proto je číslo m nejvýše čtyřmístné. Rozebereme jednotlivé případy podle počtu číslic m :

1. Číslo m je jednomístné, tj. $m = \overline{x} = x$, kde $x \in \{1, 2, \dots, 9\}$. Podle zadání úlohy je jednak

$$m + n = 1000a + 100b + 10c + d + x,$$

jednak

$$m + n = 1000d + 100c + 10b + a.$$

Odtud postupně dostaneme

$$1000a + 100b + 10c + d + x = 1000d + 100c + 10b + a,$$

$$x = 999(d - a) + 90(c - b).$$

Pravá strana poslední rovnosti je dělitelná devíti, proto může být jediné $x = 9$. Po dosazení této hodnoty do rovnosti a vykrácení devíti vychází

$$1 = 111(d - a) + 10(c - b),$$

$$10(b - c) + 1 = 111(d - a).$$

Z nerovností $-9 \leq b - c \leq 9$ plyne $-89 \leq 10(b - c) + 1 \leq 91$. Mezi čísla -89 a 91 je jediný násobek 111 , a to číslo 0 . Rovnost $10(b - c) + 1 = 0$ však není splněna. Žádné jednomístné číslo m tedy není řešením dané úlohy.

2. Číslo m je dvojmístné, tj. $m = \overline{xx} = 10x + x = 11x$, kde $x \in \{1, 2, \dots, 9\}$. Analogicky jako v předchozím případě můžeme postupně psát

$$\begin{aligned} 1\,000a + 100b + 10c + d + 11x &= 1\,000d + 100c + 10b + a, \\ 11x &= 999(d - a) + 90(c - b). \end{aligned}$$

Pravá strana poslední rovnosti je dělitelná devíti, proto může být jediné $x = 9$. Potom je

$$\begin{aligned} 11 &= 111(d - a) + 10(c - b), \\ 10(b - c) + 11 &= 111(d - a). \end{aligned}$$

Zde je $-79 \leq 10(b - c) + 11 \leq 101$, odkud plyne jediná možnost $10(b - c) + 11 = 0$, která však neplatí pro žádné číslice b, c . Žádné dvojmístné číslo m tedy není řešením dané úlohy.

3. Číslo m je trojmístné, tj. $m = \overline{xxx} = 100x + 10x + x = 111x$, kde $x \in \{1, 2, \dots, 9\}$. Opět můžeme psát

$$\begin{aligned} 1\,000a + 100b + 10c + d + 111x &= 1\,000d + 100c + 10b + a, \\ 111x &= 999(d - a) + 90(c - b), \\ 37x &= 333(d - a) + 30(c - b). \end{aligned}$$

Pravá strana poslední rovnosti je dělitelná třemi a číslo 37 není dělitelné třemi, proto musí být $x = 3$, nebo $x = 6$, nebo $x = 9$.

Nechť $x = 3$. Potom je

$$\begin{aligned} 37 &= 111(d - a) + 10(c - b), \\ 10(b - c) + 37 &= 111(d - a). \end{aligned}$$

Zde je $-53 \leq 10(b - c) + 37 \leq 127$, odkud je buď $10(b - c) + 37 = 0$, nebo $10(b - c) + 37 = 111$. Ani jedna z posledních dvou rovností však není splněna pro žádné číslice b, c .

Nechť $x = 6$. Potom je

$$\begin{aligned} 74 &= 111(d - a) + 10(c - b), \\ 10(b - c) + 74 &= 111(d - a). \end{aligned}$$

Zde je $-16 \leq 10(b - c) + 74 \leq 164$, odkud je buď $10(b - c) + 74 = 0$, nebo $10(b - c) + 74 = 111$. Ani jedna z posledních dvou rovností však není splněna pro žádné číslice b, c .

Nechť $x = 9$. Potom je:

$$111 = 111(d - a) + 10(c - b),$$

$$10(b - c) = 111(d - a - 1).$$

Zde je $-90 \leq 10(b - c) \leq 90$, odkud je jediné $10(b - c) = 0$ a $111(d - a - 1) = 0$, tj. jediné $c - b = 0$ a $d - a = 1$. Řešením dané úlohy jsou tedy čísla $n \in \{\overline{1bb2}, \overline{2bb3}, \overline{3bb4}, \overline{4bb5}, \overline{5bb6}, \overline{6bb7}, \overline{7bb8}, \overline{8bb9}\}$ pro $b \in \{0, 1, \dots, 9\}$, tj. celkem 80 čísel. Číslo m je rovno 999.

4. Číslo m je čtyřmístné, tj. $m = \overline{xxxx} = 1111x$, kde $x \in \{1, 2, \dots, 9\}$. Opět můžeme psát

$$1111x = 999(d - a) + 90(c - b).$$

Opět může být jediné $x = 9$, což dává rovnost

$$10(b - c) + 1111 = 111(d - a).$$

Platí jednak $10(b - c) + 1111 \geq 1111 - 90 = 1021$, jednak $111(d - a) \leq 999$. Proto žádné čtyřmístné číslo m není řešením dané úlohy.

Závěr: Úloha má 80 řešení, a to čísla $m = 999$ a

$$n \in \{\overline{1bb2}, \overline{2bb3}, \overline{3bb4}, \overline{4bb5}, \overline{5bb6}, \overline{6bb7}, \overline{7bb8}, \overline{8bb9}\}$$

pro $b \in \{0, 1, \dots, 9\}$.

C – I – 6

Rozbor: Předpokládejme, že požadovaný trojúhelník ABC je sestrojen. Střed kružnice vepsané libovolnému trojúhelníku leží na osách jeho vnitřních úhlů. Podle zadání leží střed kružnice k na těžnici t_c trojúhelníku ABC , proto osa vnitřního úhlu při vrcholu C splývá s těžnicí t_c . Trojúhelník ABC je tedy rovnoramenný se základnou AB (obr. 3). Leží-li střed S kružnice k s poloměrem r ve čtvrtině těžnice t_c , leží tedy ve vzdálenosti r od strany AB a ve vzdálenosti $3r$ od vrcholu C . (Bod S nemůže mít od vrcholu C vzdálenost $\frac{1}{3}r$, neboť by bod C ležel ve vnitřní oblasti kružnice k , která je však trojúhelníku ABC vepsána, tudíž body

The diagram shows a large circle labeled l . Inside it, there are two triangles, $A_1B_1C_1$ and $A_2B_2C_2$, which are circumscribed about a smaller circle k . The vertices of both triangles lie on the circumference of circle l : A_1, B_1, C_1 for the first triangle and A_2, B_2, C_2 for the second. The two triangles share a common vertex $C_1 = C_2$ at the top. Their bases A_1B_1 and A_2B_2 are positioned such that they are tangent to a small circle k at points S and X_1 respectively. Dashed lines connect S to C_1 and S to the center of circle k . A solid line segment connects X_1 to the center of circle k . An arc indicates an angle k at vertex X_1 .

Popis konstrukce:

1. dáno: $k(S, r)$, p ;
2. $l(S, 3r)$;
3. C ; $C \in p \cap l$;
4. X ; $X \in \rightarrow CS$, $|XC| = 4r$;
5. x ; $x \perp XC$, $X \in x$;
6. tečny a , b z bodu C ke k (např. pomocí Thaletovy kružnice nad průměrem CS);
7. A , B ; $A \in x \cap b$, $B \in x \cap a$.

Diskuse pro případ, že pořadí vrcholů A, B, C je proti směru pohybu hodinových ručiček:

úloha má dvě řešení $\iff |Sp| < 3r$,

úloha má jedno řešení $\iff |Sp| = 3r$,

úloha nemá žádné řešení $\iff |Sp| > 3r$.

C – S – 1

Označme hledané číslo $10a + b$, kde a, b jsou celá čísla, $a \geq 1, 0 \leq b \leq 9$. Podle zadání má platit

$$10a + b = |a^2 - b^2|.$$

Předpokládejme nejprve, že $a \geq b$. V tom případě jednoduchými úpravami dostáváme

$$\begin{aligned} 10a + b &= a^2 - b^2, \\ a^2 - 10a + 25 &= b^2 + b + 25, \\ (a - 5)^2 &= b^2 + b + 25. \end{aligned}$$

Do poslední rovnosti pak postupně dosazujeme $b = 0, b = 1, \dots, b = 9$ a zjišťujeme, zda výraz $b^2 + b + 25$ je druhou mocninou nějakého nezáporného celého čísla. Rovnici vyhovují dvojice $b = 0, a = 0$; $b = 0, a = 10$; $b = 7, a = 14$.

V případě, kdy $a < b$, obdobnými úpravami dostaneme

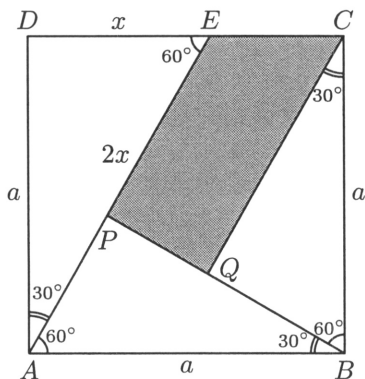
$$\begin{aligned} 10a + b &= b^2 - a^2, \\ a^2 + 10a + 25 &= b^2 - b + 25, \\ (a + 5)^2 &= b^2 - b + 25 \end{aligned}$$

a podobně jako v prvním případě získáme dvojice $b = 0, a = 0$; $b = 1, a = 0$; $b = 8, a = 4$.

Závěr: S přihlédnutím k podmínkám zadání jsou řešením úlohy tři čísla 48, 100, 147.

C – S – 2

Označme a délku strany čtverce $ABCD$. Trojúhelníky AED , BAP a CBQ jsou podobné podle věty uu , přičemž trojúhelníky BAP a CBQ jsou dokonce shodné (obr. 4). Trojúhelník AED je polovinou rovnostranného trojúhelníku o straně AE . Označíme-li $|ED| = x$, je $|AE| = 2x$.



Obr. 4

V pravoúhlém trojúhelníku AED platí

$$a = |AD| = \sqrt{|AE|^2 - |ED|^2} = \sqrt{4x^2 - x^2} = x\sqrt{3},$$

odkud $x = \frac{\sqrt{3}}{3}a$. (Velikost x můžeme také spočítat užitím goniometrického vzorce $x : a = |ED| : |AD| = \operatorname{tg} 30^\circ = \frac{\sqrt{3}}{3}$.)

Trojúhelníky BAP a CBQ jsou polovinami rovnostranného trojúhelníku o straně a . Rovnostranný trojúhelník o straně délky a má výšku $\frac{\sqrt{3}}{2}a$ a jeho obsah je $\frac{\sqrt{3}}{4}a^2$. Součet obsahů trojúhelníků AED , BAP a CBQ je tudíž

$$\frac{1}{2} \cdot \frac{\sqrt{3}}{3}a \cdot a + \frac{\sqrt{3}}{4}a^2 = \frac{5\sqrt{3}}{12}a^2.$$

Jelikož obsah čtverce $ABCD$ je a^2 , je poměr obsahů lichoběžníku $PQCE$ a čtverce $ABCD$ roven

$$\frac{a^2 - \frac{5\sqrt{3}}{12}a^2}{a^2} = \frac{12 - 5\sqrt{3}}{12},$$

což je číslo menší než 0,29.

Závěr: Obsah lichoběžníku $PQCE$ je menší než třetina obsahu čtverce $ABCD$.

Pro zajímavost uvedeme ještě jedno řešení, ve kterém ukážeme, že zkoumaný obsah lze odhadnout pomocí úvah o vzájemné poloze vhodných bodů (bez výpočtu délek a obsahu).

Jiné řešení. Protože nás zajímají jen poměry obsahů, můžeme předpokládat, že $ABCD$ je čtverec o straně 1. Ve středové souměrnosti podle středu O čtverce přejdou body E , P a Q v body, které označíme G , R a S (obr. 5). Z pravoúhlého trojúhelníku AED s úhlem 60° při vrcholu E plyne

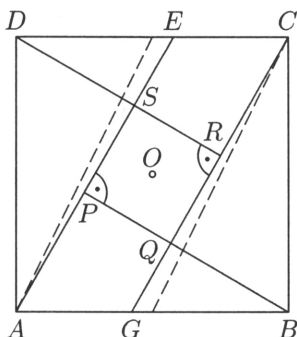
$$|DE| = \frac{1}{2}|AE| > \frac{1}{2}|AD| = \frac{1}{2},$$

takže pro obsah rovnoběžníku $AGCE$ platí nerovnost

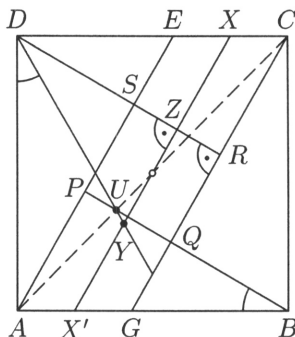
$$S(AGCE) < \frac{1}{2}.$$

Zároveň se zdá, že shodné lichoběžníky $RCES$ a $AGQP$ mají větší obsah než čtverec $PQRS$. Pokud tomu tak opravdu je, musí být $S(RCES) > \frac{1}{3}S(AGCE)$, takže nutně platí

$$\begin{aligned} S(PQCE) &= S(AGCE) - S(AGQP) = S(AGCE) - S(RCES) < \\ &< \frac{2}{3}S(AGCE) < \frac{2}{3} \cdot \frac{1}{2} = \frac{1}{3}. \end{aligned}$$



Obr. 5



Obr. 6

Tím bude úloha vyřešena.

Strana SR čtverce $PQRS$ je současně výškou lichoběžníku $RCES$. Proto bude nerovnost $S(PQRS) < S(RCES)$ dokázána, když ověříme, že strana čtverce je kratší než střední příčka lichoběžníku. Tou je úsečka XZ , kde Z označuje střed úsečky SR , což je zároveň pata výšky rovnostranného trojúhelníku XYD (obr. 6). Označme U průsečík úhlopříčky AC daného čtverce s úsečkou PQ . Tímto bodem prochází i přímka DY , která je souměrně sručená s přímkou BP právě podle osy AC , neboť $|\sphericalangle YDA| = |\sphericalangle ABP| = 30^\circ$. To ovšem znamená, že bod Y , který je průsečíkem DU a XZ , leží vně čtverce $PQRS$! Proto je opravdu $|XZ| = |ZY| > |QR|$.

Obsah lichoběžníku $PQCE$ je tudíž menší než třetina obsahu čtverce $ABCD$.

C – S – 3

Označme a zapišme v desítkové soustavě pět pětimístných čísel, která se čtou zepředu stejně jako zezadu a jsou sestavena z daných číslic:

$$\overline{a_1 b_1 c_1 b_1 a_1} = a_1 \cdot 10^4 + b_1 \cdot 10^3 + c_1 \cdot 10^2 + b_1 \cdot 10 + a_1,$$

$$\overline{a_2 b_2 c_2 b_2 a_2} = a_2 \cdot 10^4 + b_2 \cdot 10^3 + c_2 \cdot 10^2 + b_2 \cdot 10 + a_2,$$

$$\overline{a_3 b_3 c_3 b_3 a_3} = a_3 \cdot 10^4 + b_3 \cdot 10^3 + c_3 \cdot 10^2 + b_3 \cdot 10 + a_3,$$

$$\overline{a_4 b_4 c_4 b_4 a_4} = a_4 \cdot 10^4 + b_4 \cdot 10^3 + c_4 \cdot 10^2 + b_4 \cdot 10 + a_4,$$

$$\overline{a_5 b_5 c_5 b_5 a_5} = a_5 \cdot 10^4 + b_5 \cdot 10^3 + c_5 \cdot 10^2 + b_5 \cdot 10 + a_5.$$

Mezi číslicemi c_1, c_2, c_3, c_4, c_5 je právě jedna jednička, právě jedna dvojka, právě jedna trojka, právě jedna čtyřka a právě jedna pětka. Kdyby

totiž na místě stovek uvažovaných pěti čísel chyběla např. jednička, musela by se na místech ostatních řádů vyskytovat v lichém počtu (pětkrát), což vzhledem k symetrii uvažovaných čísel není možné.

Pro součet S uvažovaných čísel tedy platí

$$\begin{aligned}
 S &= \overline{a_1 b_1 c_1 b_1 a_1} + \overline{a_2 b_2 c_2 b_2 a_2} + \overline{a_3 b_3 c_3 b_3 a_3} + \overline{a_4 b_4 c_4 b_4 a_4} + \overline{a_5 b_5 c_5 b_5 a_5} = \\
 &= (a_1 + a_2 + a_3 + a_4 + a_5) \cdot (10^4 + 1) + \\
 &\quad + (b_1 + b_2 + b_3 + b_4 + b_5) \cdot (10^3 + 10) + \\
 &\quad + (c_1 + c_2 + c_3 + c_4 + c_5) \cdot 10^2 = \\
 &= 10\,001 \cdot (a_1 + a_2 + a_3 + a_4 + a_5) + 1\,010 \cdot (b_1 + b_2 + b_3 + b_4 + b_5) + \\
 &\quad + 100 \cdot (1 + 2 + 3 + 4 + 5) = \\
 &= 10\,001 \cdot (a_1 + a_2 + a_3 + a_4 + a_5) + \\
 &\quad + 1\,010 \cdot (b_1 + b_2 + b_3 + b_4 + b_5) + 1\,500.
 \end{aligned}$$

S ohledem na číslice, jež máme k dispozici, bude součet S nejmenší, jestliže bude

$$\begin{aligned}
 a_1 + a_2 + a_3 + a_4 + a_5 &= 1 + 1 + 2 + 2 + 3 = 9, \\
 b_1 + b_2 + b_3 + b_4 + b_5 &= 5 + 5 + 4 + 4 + 3 = 21.
 \end{aligned}$$

Nejmenší možný součet má tudíž hodnotu

$$S_{\min} = 10\,001 \cdot 9 + 1\,010 \cdot 21 + 1\,500 = 112\,719$$

a vznikne např. jako součet

$$S_{\min} = 13\,131 + 14\,241 + 24\,342 + 25\,452 + 35\,553.$$

Podobně bude součet S největší, pokud bude

$$\begin{aligned}
 a_1 + a_2 + a_3 + a_4 + a_5 &= 5 + 5 + 4 + 4 + 3 = 21, \\
 b_1 + b_2 + b_3 + b_4 + b_5 &= 1 + 1 + 2 + 2 + 3 = 9.
 \end{aligned}$$

Největší možný součet má tudíž hodnotu

$$S_{\max} = 10\,001 \cdot 21 + 1\,010 \cdot 9 + 1\,500 = 220\,611$$

a vznikne např. jako součet

$$S_{\max} = 53\,535 + 52\,425 + 42\,324 + 41\,214 + 31\,113.$$

C – II – 1

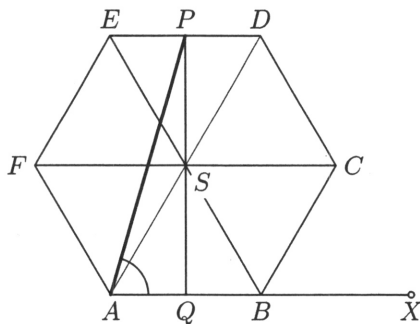
Pro každé z dvojčíselných prvočísel 97, 89, 83, 79, 73, ... hledáme jeho nejmenší násobek, který převyšuje číslo 2003. Vzhledem k tomu, že mezi k po sobě jdoucími celými čísly je právě jedno dělitelné k , a protože je $97 \cdot 21 = 2037$, $89 \cdot 23 = 2047$, $83 \cdot 25 = 2075$, $79 \cdot 26 = 2054$, musí být $2003 + n \geq 2075$, tedy $n \geq 72$. Pro takové n máme zaručeno, že pro každé z prvočísel 97, 89, 83, 79 je mezi čísly 2003, 2004, 2005, ..., $2003 + n$ aspoň jedno dělitelné.

Mezi uvedenými 73 čísly 2003 až 2075 je vždy aspoň jedno dělitelné prvočíslem 73, aspoň jedno dělitelné prvočíslem 71 atd.

Hledané číslo n je tedy 72.

C – II – 2

V pravidelném šestiúhelníku $ABCDEF$ se středem S , v němž Q je střed strany AB a P je střed strany DE , známe velikost úhlu PAQ (obr. 7),



Obr. 7

neboť všechny pravidelné šestiúhelníky jsou navzájem podobné. V pravouhlém trojúhelníku APQ tedy známe délku přepony AP a velikosti dvou úhlů (AQP je pravý úhel). Odtud vyplývá postup *konstrukce*:

1. úsečka AP ,
2. Thaletova kružnice k nad průměrem AP ,
3. polopřímka AX , jež svírá s úsečkou AP úhel velikosti PAQ (ten sestrojíme pomocí libovolného pravidelného šestiúhelníku),
4. bod Q jako průsečík kružnice k s polopřímkou AX ,
5. střed S úsečky PQ ,

6. kružnice se středem S a poloměrem $|SQ|$,

7. pravidelný šestiúhelník $ABCDEF$.

Úloha má dvě řešení souměrně sdružená podle osy AP podle toho, v které polorovině s hraniční přímkou AP sestrojíme polopřímku AX (bod 3 konstrukce).

C – II – 3

V prvním případě platí

$$1\,000p \cdot \frac{p}{100} + 1\,000q \cdot \frac{q}{100} = 1\,000(p+q) \cdot \frac{p+2,4}{100},$$

v druhém případě platí

$$1\,000p \cdot \frac{2p}{100} + 1\,000q \cdot \frac{2q}{100} = 1\,000(p+q) \cdot \frac{p+5,8}{100}.$$

Úpravou obou rovnic získáme soustavu

$$\begin{aligned} p^2 + q^2 &= (p+2,4)(p+q), \\ 2p^2 + 2q^2 &= (p+5,8)(p+q). \end{aligned} \tag{1}$$

Protože levá strana druhé rovnice je dvojnásobkem levé strany první rovnice, musí platit

$$2(p+2,4)(p+q) = (p+5,8)(p+q).$$

Odtud po vykrácení nenulovým výrazem $p+q$ vychází $p=1$. Dosazením této hodnoty např. do rovnice (1) a po úpravě získáme kvadratickou rovnici

$$q^2 - 3,4q - 2,4 = 0.$$

Protože hledáme celočíselné kořeny, přepíšeme rovnici do tvaru

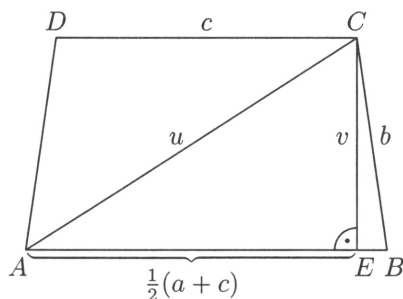
$$q(5q - 17) = 12$$

a snadno zjistíme, že mezi děliteli čísla 12 rovnici vyhovuje jedině $q=4$.

C – II – 4

Označme E patu kolmice spuštěné z vrcholu C na základnu AB rovnoramenného lichoběžníku $ABCD$ a jednotlivé délky úseček označme takto

(obr. 8): $|AB| = a = 12$, $|BC| = b$, $|CD| = c = 10$, $|AC| = u$, $|CE| = v$.
Potom je $|BE| = \frac{1}{2}(a - c) = 1$, $|AE| = \frac{1}{2}(a + c) = 11$.



Obr. 8

Podle Pythagorovy věty pro trojúhelníky AEC a EBC můžeme tedy psát

$$v^2 = u^2 - 11^2 = b^2 - 1^2, \quad (1)$$

neboli

$$u^2 - b^2 = 11^2 - 1^2 = 120.$$

Odtud je vidět, že čísla u a b jsou zároveň obě sudá, nebo obě lichá, proto v rozkladu

$$(u - b)(u + b) = 120 = 2 \cdot 60 = 4 \cdot 30 = 6 \cdot 20 = 10 \cdot 12$$

přicházejí v úvahu jen uvedené rozklady čísla 120 na sudé činitele.

Uvedeným rozkladům pak odpovídají čtyři soustavy rovnic pro neznámé u a b :

$$\begin{array}{llll} u - b = 2, & u - b = 4, & u - b = 6, & u - b = 10, \\ u + b = 60; & u + b = 30; & u + b = 20; & u + b = 12. \end{array}$$

Jejich řešením (nejlépe tak, že vždy odečteme druhou rovnici od první) dostaneme pro délku ramene b lichoběžníku $ABCD$ čtyři možnosti, $b \in \{29, 13, 7, 1\}$. Z rovnosti (1) ovšem vidíme, že musí být $b > 1$, úloze tedy vyhovují jen první tři hodnoty.

Odpověď. Možná délka ramene lichoběžníku je buď 7, nebo 13, nebo 29.

Kategorie B

Texty úloh

B – I – 1

Palindromem rozumíme přirozené číslo, které se čte zepředu i zezadu stejně, např. 16 261. Najděte největší čtyřmístný palindrom, jehož druhá mocnina je také palindromem. (E. Kováč)

B – I – 2

Najděte všechny trojice reálných čísel (x, y, z) vyhovující soustavě rovnic

$$\begin{aligned}x^3 + y^3 &= 9z^3, \\ x^2y + y^2x &= 6z^3.\end{aligned}$$

(J. Zhouf)

B – I – 3

Je dán trojúhelník se stranami délek a, b, c a obsahem S . Dokažte, že rovnost $2c^2 = |a^2 - b^2|$ platí, právě když existuje trojúhelník se stranami délek $a, b, 2c$ a obsahem $2S$. (P. Černek)

B – I – 4

Krokem budeme rozumět nahrazení uspořádané trojice celých čísel (p, q, r) trojicí $(r + 5q, 3r - 5p, 2q - 3p)$. Rozhodněte, zda existuje celé číslo k taskové, že z trojice $(1, 3, 7)$ vznikne po konečném počtu kroků trojice $(k, k + 1, k + 2)$. (P. Černek)

B – I – 5

V rovině je dán pravoúhlý lichoběžník $ABCD$ s delší základnou AB a pravým úhlem při vrcholu A . Kružnice k_1 sestrojená nad stranou AD

jako průměrem a kružnice k_2 , která prochází vrcholy B , C a dotýká se přímky AB , mají vnější dotyk v bodě P . Dokažte, že úhly CPD a ABC jsou shodné. (J. Švrček)

B – I – 6

V kartézské soustavě souřadnic Ouv znázorněte množinu všech bodů $[u, v]$, kde $u > 0$, pro něž má rovnice

$$|x^2 - ux| + vx - 1 = 0$$

s neznámou x právě tři různá reálná řešení. (J. Šimša)

B – S – 1

Najděte největší pětimístné přirozené číslo, které je dělitelné číslem 101 a které se čte zepředu stejně jako zezadu. (J. Šimša)

B – S – 2

Je dán konvexní čtyřúhelník $ABCD$. Označme P průsečík jeho úhlopříček a Q průsečík spojnic středů jeho protějších stran. Leží-li bod Q na úhlopříčce BD , je bod P středem úhlopříčky AC . Dokažte. (E. Kováč)

B – S – 3

Kolik různých výsledků můžeme dostat, sečteme-li každá dvě z daných pěti různých přirozených čísel? Pro každý možný počet uveďte příklad takové pětičky čísel. (P. Černek)

B – II – 1

Určete největší počet po sobě jdoucích pětimístných přirozených čísel, mezi nimiž není žádný palindrom, tj. číslo, které se čte zepředu stejně jako zezadu. (J. Šimša)

B – II – 2

V rovině je dán pravoúhlý trojúhelník ABC , na jehož přeponě AB uvažujeme libovolný bod K . Kružnice sestrojená nad úsečkou CK jako nad průměrem protne odvěsny BC a CA ve vnitřních bodech, které označíme

po řadě L a M . Rozhodněte, pro který bod K má čtyřúhelník $ABLM$ nejmenší možný obsah. (*J. Švrček*)

B – II – 3

Určete všechna reálná čísla p , pro něž má rovnice

$$(x - 1)^2 = 3|x| - px$$

právě tři různá řešení v oboru reálných čísel. (*J. Šimša*)

B – II – 4

V rovině je dán pravoúhlý lichoběžník $ABCD$ s delší základnou AB a pravým úhlem při vrcholu A . Označme k_1 kružnici sestrojenou nad stranou AD jako nad průměrem a k_2 kružnici procházející vrcholy B, C a dotýkající se přímky AB . Mají-li kružnice k_1, k_2 vnější dotyk v bodě P , je přímka BC tečnou kružnice opsané trojúhelníku CDP . Dokažte.

(*J. Švrček*)

Řešení úloh

B – I – 1

Každý čtyřmístný palindrom $p = \overline{abba}$ lze zapsat ve tvaru

$$p = a \cdot 1001 + b \cdot 110,$$

kde $a \in \{1, 2, \dots, 9\}$ a $b \in \{0, 1, 2, \dots, 9\}$. Potom druhá mocnina čísla $\overline{abba}$ má tvar

$$\begin{aligned} p^2 &= a^2 \cdot 1\,002\,001 + 2ab \cdot 110\,110 + b^2 \cdot 12\,100 = \\ &= a^2 \cdot 10^6 + 2ab \cdot 10^5 + (b^2 + 2ab) \cdot 10^4 + \\ &\quad + (2a^2 + 2b^2) \cdot 10^3 + (b^2 + 2ab) \cdot 10^2 + 2ab \cdot 10^1 + a^2. \end{aligned}$$

Poslední číslice čísla p^2 je tedy stejná jako poslední číslice čísla a^2 .

Pro $a \geq 4$ je číslo p^2 nutně osmimístné. Jeho první číslice je rovna jedné z hodnot c , $c+1$, $c+2$, kde c je první číslice dvojmístného čísla a^2 . (Maximální přenos z nižšího řádu je roven číslu 2.) Je-li však dané číslo opět palindromem, je jeho první i poslední číslice stejná. Porovnáním první a poslední číslice u čísel 16, 25, 36, 49, 64, 81 vidíme, že žádné z nich není tvaru $\overline{c(c+2)}$, $\overline{c(c+1)}$ nebo $\overline{cc}$.

Je-li $a = 3$ a $b \geq 2$, je číslo p^2 opět osmimístné, jeho poslední číslice je 9 a první je 1, nejedná se tedy o palindrom.

Ve všech ostatních případech je číslo p^2 sedmimístné. Protože a^2 je pouze jednomístné a zápis čísla p^2 je symetrický, musí být nutně všechny tři hodnoty $2ab$, $2ab + b^2$, $2a^2 + 2b^2$ menší než 10, aby nedošlo k přenosu do vyššího řádu. Diskutujeme tři případy:

- $a = 3$: nerovnici $2 \cdot 3^2 + 2b^2 < 10$ nevyhovuje žádné b ,
- $a = 2$: nerovnici $2 \cdot 2^2 + 2b^2 < 10$ vyhovuje pouze $b = 0$,
- $a = 1$: nerovnici $2 \cdot 1^2 + 2b^2 < 10$ vyhovuje pouze $b = 0$, $b = 1$,

Závěr: Největším čtyřmístným palindromem splňujícím podmínky úlohy je číslo 2002.

B – I – 2

Přičteme-li k první rovnici trojnásobek rovnice druhé, získáme rovnici

$$x^3 + 3x^2y + 3xy^2 + y^3 = 27z^3.$$

Její úpravou dostaneme

$$(x + y)^3 = (3z)^3, \quad \text{tj.} \quad x + y = 3z.$$

Dosadíme-li tento výraz do levé strany druhé rovnice soustavy, dostaneme

$$x^2y + xy^2 = xy(x + y) = 3xyz, \quad \text{tj.} \quad 3xyz = 6z^3.$$

Rozlišíme dva případy.

Je-li $z = 0$, je poslední rovnice splněna pro všechna $x, y \in \mathbb{R}$. Z první rovnice soustavy získáme $x^3 + y^3 = 0$, tj. $y = -x$. Řešením je každá trojice $(t, -t, 0)$, kde t je libovolné reálné číslo.

Je-li $z \neq 0$, pak $xy = 2z^2$. Společně s rovnicí $x + y = 3z$ dostáváme soustavu

$$\begin{aligned}x + y &= 3z, \\xy &= 2z^2\end{aligned}$$

dvou rovnic o dvou neznámých x, y s parametrem z . Eliminací např. neznámé y dostaneme kvadratickou rovnici

$$x^2 - 3zx + 2z^2 = 0.$$

Ze vztahů mezi kořeny a koeficienty kvadratické rovnice získáme řešení ve tvaru $x = z, y = 2z$ nebo $x = 2z, y = z$. Řešením je tedy každá trojice $(t, 2t, t)$ a $(2t, t, t)$, kde t je libovolné reálné číslo (různé od nuly).

Závěr: Soustava má řešení $(t, 2t, t)$ a $(2t, t, t)$ pro každé $t \neq 0$, $(t, -t, 0)$ pro každé t a žádné jiné řešení nemá.

Jiné řešení. První rovnici vynásobíme dvěma a odečteme od ní trojnásobek rovnice druhé (vyloučíme tak neznámou z). Získáme rovnici

$$2x^3 + 2y^3 - 3x^2y - 3xy^2 = 0.$$

Levou stranu rovnice postupně upravíme na tvar:

$$\begin{aligned}2(x + y)(x^2 - xy + y^2) - 3(x + y)xy &= 0, \\(x + y)(2x^2 - 5xy + 2y^2) &= 0, \\(x + y)(2x - y)(x - 2y) &= 0.\end{aligned}$$

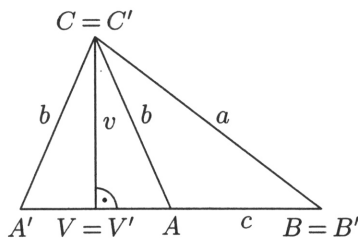
Mohou tedy nastat tři případy:

- $x + y = 0$, potom $y = -x$. Dosazením do první rovnice soustavy dostaneme $9z^3 = x^3 + (-x)^3 = 0$, tj. $z = 0$.
- $2x - y = 0$, potom $y = 2x$. Dosazením do první rovnice soustavy dostaneme $9z^3 = x^3 + (2x)^3 = 9x^3$, tj. $z = x$.
- $x - 2y = 0$, potom $x = 2y$. Dosazením do první rovnice soustavy dostaneme $9z^3 = (2y)^3 + y^3 = 9y^3$, tj. $z = y$.

Závěr: Řešením jsou všechny trojice $(t, -t, 0)$, $(t, 2t, t)$ a $(2t, t, t)$, kde t je libovolné reálné číslo.

B – I – 3

Bez újmy na obecnosti předpokládejme, že platí $a \geq b$. Jestliže je obsah trojúhelníku $A'B'C'$ se stranami délek $a, b, 2c$ roven dvojnásobnému obsahu trojúhelníku ABC se stranami délek a, b, c , jsou výšky CV a $C'V'$ těchto trojúhelníků shodné. Trojúhelníky ACV a $A'C'V'$ jsou tedy shodné podle věty *Ssu*, proto můžeme oba trojúhelníky ABC a $A'B'C'$ přemístit tak, aby platilo $B = B', C = C'$ a $V = V'$; pak už ovšem nemůže platit $A = A'$. Jaká je poloha bodů A a A' na přímce BV ? Protože $b = |AC| = |A'C'|$, je trojúhelník $AA'C$ je rovnoramenný a jeho základna AA' má střed v bodě V (obr. 9). Předpoklad $a \geq b$ znamená,



Obr. 9

že $|AC| = |A'C'| \leq |BC|$, takže bod B neleží na úsečce AA' ; protože $|AB| = c$ a $|A'B| = 2c$, leží bod B na polopřímce opačné k AA' tak, že bod A je středem úsečky $A'B$.

Z pravoúhlých trojúhelníků AVC a BVC vyplývá

$$v^2 = a^2 - \left(\frac{3}{2}c\right)^2,$$

$$v^2 = b^2 - \left(\frac{1}{2}c\right)^2.$$

Porovnáním pravých stran dostaneme po úpravě

$$a^2 - b^2 = 2c^2.$$

Ukázali jsme tak, že pokud k danému trojúhelníku ABC existuje trojúhelník se stranami $a, b, 2c$ a obsahem $2S$, pak pro délky a, b, c musí být splněna rovnost $|a^2 - b^2| = 2c^2$.

Předpokládejme naopak, že pro velikosti stran a, b, c trojúhelníku ABC platí $|a^2 - b^2| = 2c^2$. Nejprve ukážeme, že trojúhelník se stranami $a, b, 2c$ existuje, tj. že platí trojúhelníková nerovnost

$$a + b > 2c > |a - b|.$$

Pro trojúhelník ABC platí trojúhelníková nerovnost $a + b > c > |a - b|$. Proto platí $2c > c > |a - b|$. Vynásobíme-li dále obě strany nerovnosti $c > |a - b|$ kladným výrazem $a + b$, obdržíme nerovnost

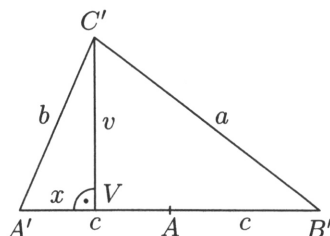
$$c(a + b) > |a^2 - b^2| = 2c^2,$$

z níž po dělení c vyplývá nerovnost

$$a + b > 2c.$$

Předpokládejme nyní, že v trojúhelníku $A'B'C'$ o stranách $a, b, 2c$ platí rovnost $2c^2 = a^2 - b^2$ (opět bez újmy na obecnosti předpokládáme, že $a > b$ — zde nemůže být $a = b$, protože by bylo $c = 0$).

Vysvětlíme, proč pata V výšky z vrcholu C' na stranu $A'B'$ padne dovnitř této strany (a ne na její prodloužení). K tomu stačí ukázat, že trojúhelník $A'B'C'$ má ostré vnitřní úhly u vrcholů A' i B' (obr. 10). Úhel $A'B'C'$ je menší než úhel $B'A'C'$, neboť předpokládáme, že $a > b$. Úhel



Obr. 10

$B'A'C'$ je ostrý, právě když platí nerovnost $|B'C'|^2 < |A'B'|^2 + |A'C'|^2$, neboli $a^2 < 4c^2 + b^2$. Poslední nerovnost je ale zaručena rovností $a^2 = b^2 + 2c^2$.

Z pravoúhlých trojúhelníků $A'VC'$ a $B'VC'$ plyne, že pro délky $x = |A'V|$ a $v = |C'V|$ platí

$$\begin{aligned}v^2 &= b^2 - x^2, \\v^2 &= a^2 - (2c - x)^2.\end{aligned}$$

Porovnáním pravých stran dostaneme po úpravě

$$4cx = 4c^2 - (a^2 - b^2)$$

a dosazením za $a^2 - b^2$ vyjde

$$4cx = 4c^2 - 2c^2 = 2c^2, \quad \text{tj. } x = \frac{1}{2}c.$$

Označíme-li A (ve shodě s první částí) střed strany $A'B'$, platí

$$|AC'| = |A'C'| = b,$$

tudíž trojúhelník $AB'C'$ má strany délek a, b, c a obsah rovný polovině obsahu trojúhelníku $A'B'C'$. Tím jsme dokázali opačnou implikaci.

Jiné řešení. Z Heronova vzorce pro obsah S_1 trojúhelníku ABC a pro obsah S_2 trojúhelníku $A'B'C'$ máme

$$\begin{aligned}S_1 &= \frac{1}{4}\sqrt{((a+b)^2 - c^2)(c^2 - (a-b)^2)}, \\S_2 &= \frac{1}{4}\sqrt{((a+b)^2 - 4c^2)(4c^2 - (a-b)^2)}.\end{aligned}$$

Z podmínky $S_2 = 2S_1$ plyne

$$((a+b)^2 - 4c^2)(4c^2 - (a-b)^2) = 4((a+b)^2 - c^2)(c^2 - (a-b)^2).$$

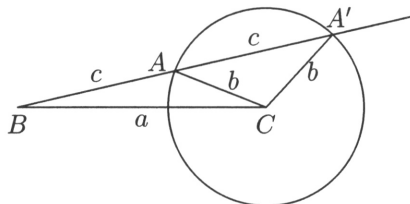
Z této podmínky po úpravě dostaneme

$$(a^2 - b^2)^2 = 4c^4, \quad \text{tj. } |a^2 - b^2| = 2c^2.$$

Provedené úpravy jsou ekvivalentní, proto je možno celý postup obrátit. Z rovnosti $|a^2 - b^2| = 2c^2$ vyplývá, že trojúhelník $A'B'C'$ má dvakrát

větší obsah než trojúhelník ABC . Existenci trojúhelníků lze dokázat stejným postupem jako v prvním řešení.

Jiné řešení. Uvažujme úsečku BC délky a ($a > b$) a kružnici k se středem v bodě C a poloměrem b (obr. 11).



Obr. 11

Ve stejné polorovině (s hraniční přímkou BC) uvažujme body A a A' , pro něž platí $|AB| = c$, $|A'B| = 2c$. Leží-li body B , A a A' na téže přímce, potom obsah trojúhelníku $A'BC$ je dvojnásobkem obsahu trojúhelníku ABC . Z mocnosti bodu B ke kružnici k plyne

$$|BA| \cdot |BA'| = 2c^2 = a^2 - b^2.$$

Je-li naopak splněna poslední rovnost, protne polopřímka opačná k AB kružnici k v bodě, jehož vzdálenost od bodu B je rovna $2c$, tímto bodem je však A' . Odtud již plyne tvrzení pro obsahy trojúhelníků. Existenci trojúhelníků lze dokázat stejným postupem jako v prvním řešení.

B – I – 4

Sečteme-li všechna tři čísla nově vzniklé trojice, dostaneme

$$(r + 5q) + (3r - 5p) + (2q - 3p) = 4r + 7q - 8p = 3(r + 2q - 3p) + (p + q + r).$$

Toto číslo dává při dělení třemi stejný zbytek jako číslo $(p + q + r)$, tj. zbytek při dělení třemi součtu čísel v trojici zůstává zachován. Pro trojici $(1, 3, 7)$ je zbytek roven dvěma ($1 + 3 + 7 = 11 = 3 \cdot 3 + 2$).

Součet tří po sobě jdoucích celých čísel je však dělitelný třemi, takže dává zbytek nula. Plyne to z rovnosti $k + (k + 1) + (k + 2) = 3(k + 1)$.

Závěr: Po konečném počtu *kroků* nemůžeme z trojice $(1, 3, 7)$ dospět k trojici po sobě jdoucích celých čísel.

Jiné řešení. Předpokládejme, že z nějaké trojice (a, b, c) vznikne v následujícím *kroku* trojice po sobě jdoucích čísel (a_1, b_1, c_1) . Tato tři čísla jsou tedy nutně členy aritmetické posloupnosti s diferencí 1. Musí proto platit

$$c_1 - b_1 = b_1 - a_1.$$

Dosadíme-li sem $a_1 = c + 5b$, $b_1 = 3c - 5a$, $c_1 = 2b - 3a$, dostaneme po úpravě

$$7(a + b) = 5c.$$

Odtud nutně platí $c = 7k$, $a + b = 5k$ pro nějaké celé číslo k . Potom ale $a_1 = 32k - 5a$, $b_1 = 21k - 5a$, $c_1 = 10k - 5a$. Aby tato trojice tvořila aritmetickou posloupnost s diferencí jedna, muselo by být $11k = -1$, tj. $k = -\frac{1}{11}$. To je spor s předpokladem, že k je celé číslo.

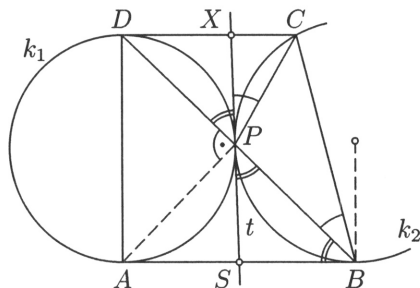
Jiné řešení. Zkoumejme, jak se mění parita trojice čísel v následujících *krocích*. Na začátku jsou všechna tři čísla lichá. Postupně dostáváme:

$$(l, l, l) \rightarrow (s, s, l) \rightarrow (l, l, s) \rightarrow (l, l, l) \rightarrow \dots$$

Protože se parita čísel pravidelně mění dle daného schématu, nemůžeme z trojice lichých čísel dospět k trojici (s, l, s) , resp. (l, s, l) , které reprezentují všechny trojice po sobě jdoucích čísel (za sudým číslem následuje liché a naopak).

B – I – 5

Protože úsečka AD je průměrem kružnice k_1 , je úhel APD pravý (obr. 12).



Obr. 12

Uvažujme společnou tečnu t obou kružnic procházející bodem P . Označme po řadě S a X průsečíky tečny t s úsečkami AB a CD . Přímka AB je však také společnou tečnou obou kružnic. Platí proto $|SA| = |SP| = |SB|$. Bod S je proto středem Thaletovy kružnice sestavené na stranou AB jako průměrem. Úhel APB je tudíž stejně jako úhel APD pravý a bod P je tedy vnitřním bodem úsečky BD .

Trojúhelník BPS je rovnoramenný se základnou BP , pro jeho úhly tedy platí $|\sphericalangle SBP| = |\sphericalangle SPB|$. Úhel SPB má navíc stejnou velikost jako úhel DPX (dvojice vrcholových úhlů). Platí proto $|\sphericalangle ABP| = |\sphericalangle DPX|$. Současně však je úhel XPC úhlem úsekovým pro tětivu CP kružnice k_2 . Z rovnosti obvodového a úsekového úhlu máme $|\sphericalangle PBC| = |\sphericalangle XPC|$.

Celkově dostáváme

$$|\sphericalangle ABC| = |\sphericalangle ABP| + |\sphericalangle PBC| = |\sphericalangle DPX| + |\sphericalangle XPC| = |\sphericalangle DPC|,$$

což jsme chtěli dokázat.

B – I – 6

Nulové body výrazu $x^2 - ux$ jsou $x = 0$ a $x = u$. Protože dle zadání platí $u > 0$, rozdělíme reálnou osu na tři vzájemně disjunktní intervaly $I_1 = (-\infty, 0)$, $I_2 = \langle 0, u \rangle$ a $I_3 = (u, \infty)$.

Na intervalech I_1 a I_3 řešíme kvadratickou rovnici

$$x^2 - (u - v)x - 1 = 0.$$

Tato rovnice má kladný diskriminant $(u - v)^2 + 4$, a tudíž dva různé reálné kořeny

$$x_1 = \frac{u - v - \sqrt{(u - v)^2 + 4}}{2},$$

$$x_2 = \frac{u - v + \sqrt{(u - v)^2 + 4}}{2}.$$

Protože $\sqrt{(u - v)^2 + 4} > |u - v|$, platí $x_1 < 0$ a $x_2 > 0$. Znamená to, že číslo x_1 je vždy řešením rovnice (1), neboť $I_1 = (-\infty, 0)$, zatímco číslo x_2 je řešením rovnice (1), právě když platí $x_2 \in I_3$, neboli $x_2 > u$.

Na intervalu I_2 řešíme kvadratickou rovnici

$$x^2 - (u + v)x + 1 = 0.$$

Tato rovnice má diskriminant $D = (u + v)^2 - 4$ a případné reálné kořeny

$$x_3 = \frac{u + v - \sqrt{(u + v)^2 - 4}}{2},$$

$$x_4 = \frac{u + v + \sqrt{(u + v)^2 - 4}}{2}.$$

Ze zadání vyplývá, že aspoň jeden z kořenů x_3, x_4 musí být řešením rovnice (1) (ležícím na intervalu I_2). Proto předně musí být diskriminant D nezáporný, z čehož plyne podmínka $|u + v| \geq 2$. Protože navíc $\sqrt{(u + v)^2 - 4} < |u + v|$, mají oba kořeny x_3, x_4 stejné znaménko jako součet $u + v$. Dohromady to znamená, že musí platit $u + v \geq 2$ (v případě $u + v \leq -2$ by totiž žádné z čísel x_3, x_4 neleželo na I_2). Za podmínky $u + v \geq 2$ ovšem platí $0 < x_3 \leq x_4$, takže ze zadání plyne, že na intervalu $I_2 = \langle 0, u \rangle$ leží číslo x_3 (a případně i číslo x_4).

Z dosavadních úvah plyne, že naší úlohou je posoudit otázku, kdy za podmínek

$$u > 0 \quad \text{a} \quad u + v \geq 2 \tag{2}$$

nastane některý z těchto případů:

- a) $x_2 \notin I_3, \{x_3, x_4\} \subset I_2, x_3 \neq x_4$;
- b) $x_2 \in I_3, x_3 = x_4 \in I_2$;
- c) $x_2 \in I_3, x_3 \in I_2, x_4 \notin I_2$.

Ad a. Zjistíme, kdy jsou splněny jednotlivé podmínky, které tento případ vymezují (pro lepší přehled je v textu uvádíme černými puntíky).

- $x_2 \notin I_3$, neboli $x_2 \leq u$. Po úpravě získáme nerovnost

$$\sqrt{(u - v)^2 + 4} \leq u + v,$$

jejíž pravá strana je podle (2) kladná, takže obě strany můžeme umocnit na druhou. Po další snadné úpravě dostaneme podmínku $uv \geq 1$. Proto platí:

$$x_2 \notin I_3 \iff uv \geq 1.$$

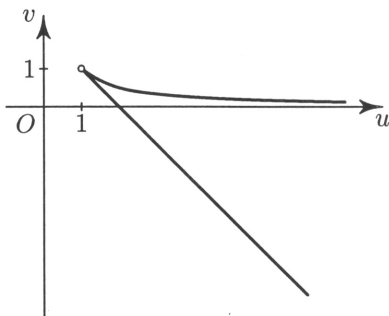
- $\{x_3, x_4\} \subset I_2$. Jak víme, za podmínek (2) platí $0 < x_3 \leq x_4$, stačí proto pouze zkoumat nerovnost $x_4 \leq u$, neboli $\sqrt{(u + v)^2 - 4} \leq u - v$. Poslední nerovnost může platit jedině tehdy, když $u \geq v$. Pak po umocnění stran zkoumané nerovnosti a následné úpravě dostaneme podmínku $uv \leq 1$. Proto platí:

$$\{x_3, x_4\} \subset I_2 \iff u \geq v \wedge uv \leq 1.$$

• $x_3 \neq x_4$. Z dřívějšího odvození podmínky $u + v \geq 2$ je jasné, že rovnost $x_3 = x_4$ nastane, právě když $u + v = 2$. Za podmínek (2) tedy platí

$$x_3 \neq x_4 \iff u + v > 2.$$

Shrňme nyní všechny podmínky pro zkoumaný případ A. Z nerovností $uv \geq 1$ a $uv \leq 1$ plyne $uv = 1$, neboli $v = 1/u$. Zbývající podmínky jsou pak tvaru $u \geq 1/u$ a $u + 1/u > 2$ a jsou zřejmě obě splněny, právě když $u > 1$. Hledané body $[u, v]$ v případě A tedy tvoří část hyperboly $v = 1/u$ určenou omezením $u > 1$ (obr. 13).



Obr. 13

Ad b. Z předchozího rozboru případu A plyne, že za podmínek (2) platí tyto ekvivalence:

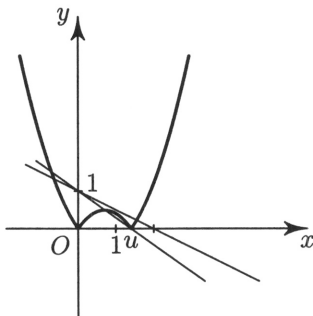
$$\begin{aligned} x_2 \in I_3 &\iff uv < 1, & x_4 \in I_2 &\iff u \geq v \wedge uv \leq 1, \\ x_3 = x_4 &\iff u + v = 2. \end{aligned}$$

Vidíme, že v případě B musí platit $v = 2 - u$. Tehdy jsou zbývající podmínky tvaru $(2 - u)u < 1$ a $u \geq 2 - u$ a jsou zřejmě obě splněny, právě když $u > 1$. Hledané body $[u, v]$ v případě B tedy tvoří polopřímku určenou rovnicí $v = 2 - u$ a omezením $u > 1$.

Ad c. Podmínku $x_3 \in I_2$ lze vyjádřit nerovností $x_3 \leq u$, která je ekvivalentní s nerovností $\sqrt{(u + v)^2 - 4} \geq v - u$, jež je splněna triviálně, pokud $u \geq v$. Jak jsme ale ukázali dříve, v případě $u \geq v$ platí nejen $x_3 \in I_2$, ale také $x_4 \in I_2$, což případ C vylučuje. V případě C tedy nutně platí $u < v$ a z nerovnosti $\sqrt{(u + v)^2 - 4} \geq v - u$ po umocnění a úpravě dostaneme podmínku $uv \geq 1$. Jak ale víme, z poslední nerovnosti plyne $x_2 \notin I_3$, takže případ C nemůže nikdy nastat.

Závěr: Množinou všech bodů vyhovujícím zadání je část hyperboly $v = 1/u$ a část přímky $v = 2 - u$, v obou případech části určené podmínkou $u > 1$.

Jiné řešení. Rovnici lze řešit také graficky. Zkoumáme, kdy budou mít grafy funkcí $f(x) = |x^2 - ux|$ a $g(x) = 1 - vx$ právě tři společné body (obr. 14).



Obr. 14

Graf funkce f je složen z částí paraboly, grafem funkce g je přímka procházející bodem $[0, 1]$. Aby tato přímka měla s grafem $f(x)$ společné právě tři body, musí být buď tečnou paraboly na intervalu $(0, u)$ (potom $u + v = 2$, odvození je analogické jako v předchozím řešení — pomocí diskriminantu), nebo musí procházet bodem $[u, 0]$ a současně protínat graf funkce f ve vnitřním bodě intervalu $(0, u)$. Dosadíme-li souřadnice bodu $[u, 0]$ do rovnice přímky g , dostaneme $0 = 1 - vu$, tj. $uv = 1$. Stejně jako v předchozím řešení musí platit $u > 1$, což můžeme ověřit nalezením druhého průsečíku přímky s parabolou.

B – S – 1

Libovolné z uvažovaných pětimístných čísel má desítkové soustavě zápis tvaru $\overline{abcba}$. Jeho rozvinutím a úpravou získáme rovnost

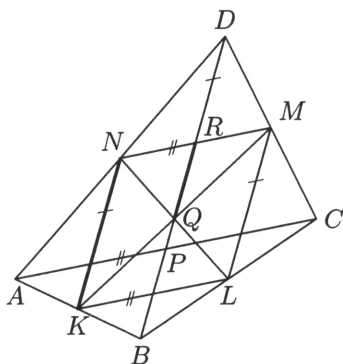
$$\overline{abcba} = 10\,001a + 1\,010b + 100c = 101(99a + 10b + c) + 2a - c.$$

Odtud plyne, že zkoumané číslo je dělitelné 101, právě když $2a - c = 0$ (pro libovolné číslice a, c totiž jistě platí $|2a - c| < 101$). Z rovnosti $2a = c$ plyne $a \leq 4$, a protože hledáme co největší takové číslo, zvolíme

jeho první číslici $a = 4$, které odpovídá číslice $c = 8$. Protože číslice b nemá na dělitelnost číslem 101 vliv, zvolíme ji co největší: $b = 9$. Hledané číslo je tudíž 49 894.

B – S – 2

Středy stran čtyřúhelníku $ABCD$ označme K, L, M, N podle obr. 15. Protože úsečky KL a MN jsou střední příčky trojúhelníků ABC



Obr. 15

resp. ACD , platí $KL \parallel AC \parallel MN$. Obdobně platí $LM \parallel BD \parallel KN$, tudíž $KLMN$ je rovnoběžník a bod Q půlí úsečku KM . Všimněme si nyní trojúhelníku KMN . Středem Q jeho strany KM prochází podle předpokladu úlohy úhlopříčka BD , která je, jak víme, rovnoběžná s druhou stranou KN . Proto i střed R třetí strany MN leží na úhlopříčce BD . Protože úsečka MN je stejnohlá s úsečkou CA podle středu D , půlí úhlopříčka BD nejen úsečku MN (v bodě R), ale i úsečku AC (v odpovídajícím bodě P).

B – S – 3

Daná přirozená čísla označme podle velikosti $x_1 < x_2 < x_3 < x_4 < x_5$. Protože platí

$$x_1 + x_2 < x_1 + x_3 < x_1 + x_4 < x_1 + x_5 < x_2 + x_5 < x_3 + x_5 < x_4 + x_5,$$

je mezi všemi součty $x_i + x_j$ aspoň sedm různých hodnot. Nevypsány zůstaly pouze tři z možných součtů, a to součty $x_2 + x_3$, $x_2 + x_4$ a $x_3 + x_4$. Proto pro počet p možných hodnot uvažovaných součtů platí $7 \leq p \leq 10$.

Pro každou z hodnot $p \in \{7, 8, 9, 10\}$ uvedeme příklad pětiprvkové množiny M_p přirozených čísel, pro kterou uvažované součty nabývají právě p různých hodnot (jejich množinu označíme S_p):

$$\begin{aligned} M_7 &= \{1, 2, 3, 4, 5\}, & S_7 &= \{3, 4, 5, 6, 7, 8, 9\}; \\ M_8 &= \{1, 2, 3, 4, 6\}, & S_8 &= \{3, 4, 5, 6, 7, 8, 9, 10\}; \\ M_9 &= \{1, 2, 3, 4, 7\}, & S_9 &= \{3, 4, 5, 6, 7, 8, 9, 10, 11\}; \\ M_{10} &= \{1, 2, 3, 5, 8\}, & S_{10} &= \{3, 4, 5, 6, 7, 8, 9, 10, 11, 13\}. \end{aligned}$$

B – II – 1

Mezi 109 po sobě jdoucími pětímístnými čísly

$$10\,902, 10\,903, \dots, 10\,999, 11\,000, \dots, 11\,009, 11\,010$$

není žádný palindrom (je možné uvést i jiné vyhovující příklady 109 pětímístných čísel, my jsme vypsali skupinu nejmenších z nich).

Nejmenší a největší pětímístné palindromy jsou čísla 10 001 a 99 999; před číslem 10 001 je jen jedno pětímístné číslo, za číslem 99 999 už dokonce žádné takové číslo není. Ukážeme nyní, že za každým pětímístným palindromem x , $x \neq 99\,999$, následuje pětímístný palindrom $x + 100$ nebo $x + 110$ nebo $x + 11$. Skutečně, je-li $x = \overline{abcba}$, pak v případě $c \neq 9$ je palindromem číslo $x + 100 = \overline{ab(c+1)ba}$, v případě $c = 9 \neq b$ je palindromem číslo $x + 110 = \overline{a(b+1)0(b+1)a}$, konečně v případě $c = b = 9$ (kdy nutně $a \neq 9$) je palindromem číslo $x + 11 = \overline{(a+1)000(a+1)}$.

Odpověď. Hledaný největší počet čísel je roven 109.

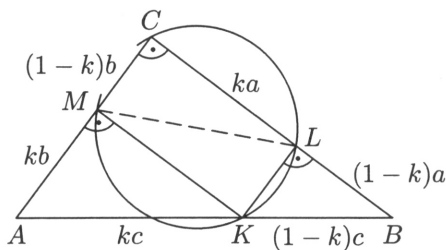
B – II – 2

Protože úhly KLC , KMC a LCM jsou pravé (obr. 16), je čtyřúhelník $KLCM$ pravoúhelník a trojúhelníky AKM a KBL jsou podobné trojúhelníku ABC . Označme jako obvykle $a = |BC|$, $b = |AC|$, $c = |AB|$ a položme $|AK| = kc$, kde $0 < k < 1$. Pak ovšem $|KB| = (1 - k)c$ a ze zmíněné podobnosti trojúhelníků dostáváme vyjádření $|AM| = kb$, $|LC| = |KM| = ka$, $|BL| = (1 - k)a$ a $|MC| = |KL| = (1 - k)b$. Proto

platí

$$\begin{aligned}
 S_{ABLM} &= S_{ABC} - S_{LMC} = \frac{1}{2}ab - \frac{1}{2} \cdot ka \cdot (1-k)b = \\
 &= \frac{1}{2}ab(1-k+k^2) = \frac{1}{2}ab \left(\left(k - \frac{1}{2}\right)^2 + \frac{3}{4} \right) \geq \\
 &\geq \frac{1}{2}ab \cdot \frac{3}{4} = \frac{3}{4}S_{ABC},
 \end{aligned}$$

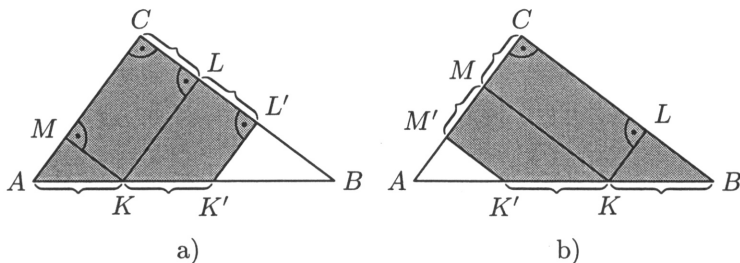
přičemž rovnost $S_{ABLM} = \frac{3}{4}S_{ABC}$ nastane, právě když $k = \frac{1}{2}$, tedy právě když je bod K středem přepony AB .



Obr. 16

Jiné řešení. Čtyřúhelník $ABLM$ má minimální obsah, právě když má maximální obsah trojúhelník LMC , který je „polovinou“ pravoúhelníku $KLCM$. Stačí proto ukázat, že obsah S_{KLCM} je maximální, právě když je bod K středem přepony AB (kdy zřejmě $S_{KLCM} = \frac{1}{2}S_{ABC}$). Je-li bod K vybrán tak, že $|AK| < \frac{1}{2}|AB|$, je úsečka KL střední příčkou lichoběžníku $AK'L'C$, který má o $S_{K'L'B}$ menší obsah než trojúhelník ABC (obr. 17a), takže platí

$$S_{KLCM} = \frac{1}{2}S_{AK'L'C} < \frac{1}{2}S_{ABC}.$$



Obr. 17

Je-li naopak $|AK| > \frac{1}{2}|AB|$, využijeme obdobný lichoběžník $BK'M'C$ (obr. 17b) a usoudíme, že platí

$$S_{KLCM} = \frac{1}{2}S_{BK'M'C} < \frac{1}{2}S_{ABC}.$$

Tím je tvrzení o maximálním obsahu S_{KLCM} dokázáno.

Odpověď. Čtyřúhelník $ABLM$ má nejmenší možný obsah, právě když bod K leží uprostřed přepony AB .

B – II – 3

I když lze danou úlohu řešit názorně geometrickou úvahou o vzájemné poloze paraboly $y = (x-1)^2$ a lomené čáry $y = 3|x| - px$, dáme nejprve přednost čistě algebraickému postupu.

Daná rovnice zřejmě nemá řešení $x = 0$. Po odstranění absolutní hodnoty a snadné úpravě dostaneme rovnice

$$x^2 + (p+1)x + 1 = 0 \quad \text{pro } x < 0, \quad (1)$$

$$x^2 + (p-5)x + 1 = 0 \quad \text{pro } x > 0. \quad (2)$$

Protože každá kvadratická rovnice má nejvýše dva různé kořeny, hledáme všechna ta čísla p , pro která má jedna z rovnic (1), (2) jeden kořen a druhá dva různé kořeny (a to vždy předepsaných znamének). Všimněme si, že pro každé $q \in \mathbb{R}$ mají reálné kořeny $x_{1,2}$ rovnice $x^2 + qx + 1 = 0$ (pokud vůbec existují) stejné znaménko, které je opačné než znaménko čísla q ; platí totiž $x_1x_2 = 1$ a $x_1 + x_2 = -q$. Pro rovnice (1), (2) tak předně dostáváme podmínky

$$p+1 > 0 \quad \text{a} \quad p-5 < 0, \quad \text{neboli} \quad p \in (-1, 5).$$

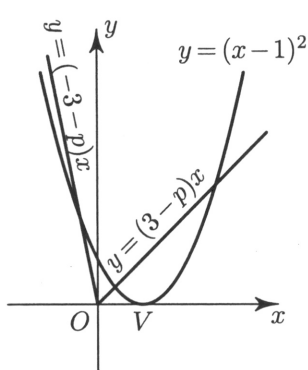
Kromě toho už jen požadujeme, aby pro diskriminanty obou rovnic

$$D_1 = (p+1)^2 - 4, \quad D_2 = (p-5)^2 - 4$$

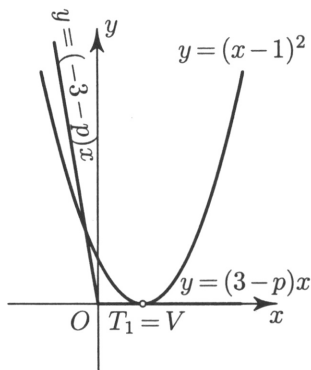
platilo buď $D_1 = 0$ a $D_2 > 0$, nebo $D_1 > 0$ a $D_2 = 0$. Rovnost $D_1 = 0$ platí pouze pro $p \in \{-3, 1\}$, rovnost $D_2 = 0$ pouze pro $p \in \{3, 7\}$. Z těchto čtyř hodnot leží v intervalu $(-1, 5)$ pouze čísla $p = 1$ a $p = 3$, přičemž pro $p = 1$ vychází $D_2 = 12 > 0$, pro $p = 3$ zase $D_1 = 12 > 0$.

Odpověď. Hledané hodnoty jsou $p = 1$ a $p = 3$.

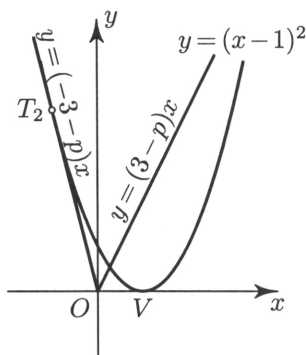
Jiné řešení. Grafem funkce $y = (x-1)^2$ je parabola s vrcholem $V[1, 0]$, grafem funkce $y = 3|x| - px$ je lomená čára tvořená rameny některého úhlu s vrcholem $O[0, 0]$ (obrázky 18a pro $p = 2$). Oba grafy mají společné



a) $p = 2$



b) $p = 3$



c) $p = 1$

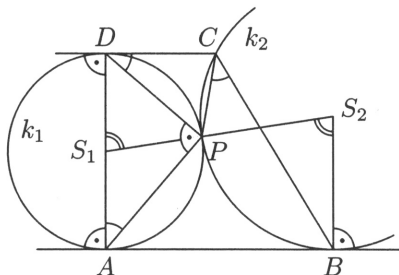
Obr. 18

tři body, právě když jedno z ramen zmíněného úhlu je tečnou paraboly a druhé je její „sečnou“. Protože zkoumaná parabola nemá tečnu rovnoběžnou s osou y , můžeme rovnice obou tečen procházejících bodem $[0, 0]$ hledat ve tvaru $y = kx$. Jak je známo, směrnice k se určí z podmínky, že rovnice $kx = (x-1)^2$ má dvojnásobný kořen, tedy nulový diskriminant. Ten má vyjádření $(k+2)^2 - 4$, takže hledané hodnoty jsou $k_1 = 0$, $k_2 = -4$ a odpovídající body dotyku $T_1 = V[1, 0]$ a $T_2[-1, 4]$. Z rovnic pro směrnice tečných ramen zkoumaných úhlů $3-p = 0$ a $-3-p = -4$ najdeme řešení $p_1 = 3$ a $p_2 = 1$ a snadno se přesvědčíme, že druhé ra-

meno je v obou případech skutečně sečnou paraboly (obr. 18b pro $p = 3$ a obr. 18c pro $p = 1$).

B – II – 4

Označme S_1 a S_2 středy uvažovaných kružnic (obr. 19). Obě úsečky S_1A a S_2B jsou kolmé na přímkou AB , jsou tudíž rovnoběžné a střídavé úhly



Obr. 19

$\angle PS_2B$ a $\angle PS_1D$ shodné. Podle věty o obvodových a střídavých úhlech proto platí

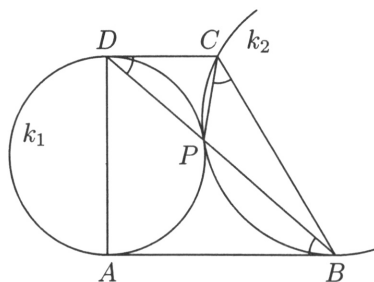
$$|\sphericalangle PCB| = \frac{1}{2} |\sphericalangle PS_2B| = \frac{1}{2} |\sphericalangle PS_1D| = |\sphericalangle PAD|.$$

Oba úhly $\angle APD$ a $\angle ADC$ jsou však pravé, tudíž

$$|\sphericalangle PAD| = 90^\circ - |\sphericalangle ADP| = |\sphericalangle CDP|.$$

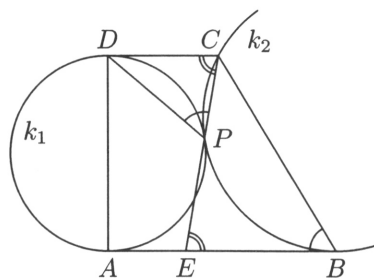
Dohromady dostáváme, že úhly $\angle PCB$ a $\angle CDP$ jsou shodné, což podle věty o obvodovém a úsekovém úhlu znamená, že přímka BC je tečnou ke kružnici opané trojúhelníku CDP .

Jiné řešení. Ve stejnolehlosti se středem P , při které kružnice k_1 přejde v kružnici k_2 , musí tečna CD kružnice k_1 přejít v rovnoběžnou tečnu AB kružnice k_2 , přitom se bod dotyku D zobrazí do bodu dotyku B . Bod P tudíž leží na úhlopříčce BD (obr. 20). Odtud plyne shodnost střídavých úhlů $\angle CDP$ a $\angle PBA$ (mezi rovnoběžkami AB a CD). Úhel $\angle PBA$ je ale úsekový úhel mezi tětivou BP a tečnou AB kružnice k_2 , je tedy shodný s příslušným obvodovým úhlem $\angle PCB$. Úhly $\angle CDP$ a $\angle PCB$ jsou proto shodné, což jsme potřebovali dokázat (viz závěr předchozího řešení).



Obr. 20

Poznámka. Podle úlohy B-I-5 jsou shodné úhly ABC a CPD (obr. 21). Protože jsou shodné i střídavé úhly PEB a PCD , kde E je



Obr. 21

průsečík polopřímky CP se stranou AB , lze kýženou shodnost úhlů CDP a PCB odvodit z trojúhelníků BCE a PDC .

Kategorie A

Texty úloh

A – I – 1

Posloupnost $(x_n)_{n=1}^{\infty}$ celých čísel s prvním členem $x_1 = 1$ splňuje podmínku

$$x_n = \pm x_{n-1} \pm \dots \pm x_1$$

s vhodnou volbou znamének „+“ a „-“ pro libovolné $n > 1$, například $x_2 = -x_1$, $x_3 = -x_2 + x_1$, $x_4 = x_3 - x_2 - x_1$, ... Pro dané n určete všechny možné hodnoty x_n .
(J. Földes)

A – I – 2

Na přímce p jsou dány různé body A, B, C v tomto pořadí, kde $|AB| = 1$ a $|BC| = h$. Uvažujme kružnice k_A, k_B, k_C , které se dotýkají přímky p po řadě v bodech A, B, C . Kružnice k_A, k_B mají přitom vnější dotyk v bodě P a kružnice k_B, k_C vnější dotyk v bodě Q . Určete všechny hodnoty poloměru kružnice k_B , pro něž je trojúhelník BPQ rovnoramenný.

(J. Zhouf)

A – I – 3

Určete všechny možné hodnoty výrazu

$$\frac{a^4 + b^4 + c^4}{a^2b^2 + a^2c^2 + b^2c^2},$$

kde a, b, c jsou délky stran trojúhelníku.

(P. Kaňovský)

A – I – 4

Určete všechna přirozená čísla $n > 1$ taková, že v některé číselné soustavě o základu $z \geq 5$ platí následující kritérium dělitelnosti: trojmístné

číslo $(abc)_z$ je dělitelné číslem n , právě když je číslem n dělitelné číslo $c + 3b - 4a$. (P. Černek)

A – I – 5

V rovině jsou dány tři různé body K, L, M , které v tomto pořadí leží na přímce. V této rovině najdete množinu všech vrcholů C čtverců $ABCD$ takových, že bod K leží na straně AB , bod L na úhlopříčce BD a bod M na straně CD . (J. Šimša)

A – I – 6

Hráči A a B hrají na desce složené ze šesti polí očíslovaných $1, 2, \dots, 6$ následující hru. Na začátku je umístěna na pole s číslem 2 figurka a pak se hází běžnou hrací kostkou. Padne-li číslo dělitelné třemi, posune se figurka na pole s číslem o 1 menším, jinak na pole s číslem o 1 větším. Hra končí vítězstvím hráče A resp. B , dostane-li se figurka na pole s číslem 1 resp. 6. S jakou pravděpodobností zvítězí hráč A ? (P. Černek)

A – S – 1

Řekneme, že tři navzájem různá přirozená čísla tvoří součtovou trojici, je-li součet prvních dvou z nich roven číslu třetímu. Určete, jaký největší počet součtových trojic se může nacházet v množině dvaceti přirozených čísel. (P. Černek)

A – S – 2

V rovině jsou dány kružnice $k_1(S_1, r_1)$ a $k_2(S_2, r_2)$ tak, že $S_2 \in k_1$ a $r_1 > r_2$. Společné tečny obou kružnic se dotýkají kružnice k_1 v bodech P a Q . Dokažte, že přímka PQ se dotýká kružnice k_2 . (J. Földes)

A – S – 3

Zjistěte, pro které reálné číslo p mají rovnice

$$\begin{aligned}x^3 + x^2 - 36x - p &= 0, \\x^3 - 2x^2 - px + 2p &= 0\end{aligned}$$

společný kořen.

(P. Černek)

A – II – 1

Najděte základy z všech číselných soustav, ve kterých je čtyřmístné číslo $(1001)_z$ dělitelné dvojmístným číslem $(41)_z$. (P. Černek)

A – II – 2

Uvnitř strany AB daného ostroúhlého trojúhelníku ABC zvolte bod S tak, aby trojúhelník SXY , kde X a Y jsou po řadě středy kružnic opsaných trojúhelníkům ASC a BSC , měl nejmenší možný obsah. (P. Černek)

A – II – 3

V oboru reálných čísel řešte soustavu rovnic

$$\log_x(y + z) = p,$$

$$\log_y(z + x) = p,$$

$$\log_z(x + y) = p$$

s neznámými x, y, z a nezáporným celočíselným parametrem p .

(J. Švrček)

A – II – 4

Posloupnost $(x_n)_{n=1}^{\infty}$ s prvním členem $x_1 = 1$ splňuje pro každé $n > 1$ podmínku

$$x_n = x_{n-1}^{\pm 1} + x_{n-2}^{\pm 1} + \dots + x_1^{\pm 1}$$

s vhodnou volbou znamének „+“ a „–“ v exponentech mocnin.

- a) Rozhodněte, zda některý člen takové posloupnosti musí být větší než 1 000.
- b) Zjistěte nejmenší možnou hodnotu členu $x_{1\,000\,000}$.
- c) Dokažte, že nerovnost $x_n < 4$ nemůže platit pro devět členů x_n takové posloupnosti. (J. Földes)

A – III – 1

V oboru reálných čísel řešte soustavu rovnic

$$x^2 - xy + y^2 = 7,$$

$$x^2y + xy^2 = -2.$$

(J. Földes)

A – III – 2

Uvnitř stran BC , CA , AB daného trojúhelníku ABC zvolíme po řadě body D , E , F tak, aby se úsečky AD , BE , CF protály v jednom bodě, který označíme G . Pokud lze čtyřúhelníkům $AFGE$, $BDGF$, $CEGD$ vepsat kružnice, z nichž každé dvě mají vnější dotyk, pak je trojúhelník ABC rovnostranný. Dokažte. (M. Tancer)

A – III – 3

Posloupnost $(x_n)_{n=1}^{\infty}$ s prvním členem $x_1 = 1$ splňuje pro každé $n > 1$ podmínku

$$x_n = \pm(n-1)x_{n-1} \pm (n-2)x_{n-2} \pm \dots \pm 2x_2 \pm x_1$$

s vhodnou volbou znamének „+“ a „–“. Rozhodněte, zda je možné, aby nerovnost $x_n \neq 12$ platila pouze pro konečně mnoho indexů n .

(P. Černek)

A – III – 4

V rovině je dán tupý úhel AKS . Sestrojte trojúhelník ABC tak, aby jeho strana BC ležela na přímce KS , bod S byl jejím středem a bod K jejím průsečíkem s osou protilehlého úhlu BAC . (P. Leischner)

A – III – 5

Ukažte, že v číselné soustavě s libovolným základem $z \geq 3$ existují dvojmístná čísla A a B , která se liší jen pořadím svých číslic a mají tuto vlastnost: kvadratická rovnice $x^2 - Ax + B = 0$ má v oboru reálných čísel dvojnásobný kořen. Dokažte rovněž, že pro daný základ z je taková dvojice A , B jediná. Například v desítkové soustavě ($z = 10$) to jsou jediné čísla $A = 18$ a $B = 81$. (J. Šimša)

A – III – 6

Je-li součin kladných čísel a , b , c roven 1, pak platí

$$\frac{a}{b} + \frac{b}{c} + \frac{c}{a} \geq a + b + c.$$

Dokažte.

(P. Kaňovský)

Řešení úloh

A – I – 1

Vypišme, jaké hodnoty mohou nabývat první členy uvedené posloupnosti. Dostaneme

$$x_1 = 1, \quad x_2 \in \{-1, 1\}, \quad x_3 \in \{-2, 0, 2\}, \quad x_4 \in \{-4, -2, 0, 2, 4\}, \\ x_5 \in \{-8, -6, -4, -2, 0, 2, 4, 6, 8\}.$$

Všimněme si, že všechny členy, které jsme vypsalí, jsou celá čísla. Dále je zřejmé, že pro $i > 2$ je každý člen x_i sudé číslo. (Další pozorování je, že pokud najdeme posloupnost, pro kterou $x_i = a$ pro nějaké číslo a a dané $i > 1$, tak existuje i posloupnost, pro kterou $x_i = -a$.)

Zjistíme, jakou největší a jakou nejmenší hodnotu může nabývat číslo x_n (v závislosti na n). Označme a_i největší hodnotu, kterou může nabývat člen x_i . Protože posloupností délky i splňujících dané vlastnosti je jen konečný počet, maximum a_i existuje a je zřejmě kladné. K číslu a_i musí pro každé $i > 1$ existovat posloupnost $x_1, \dots, x_{i-1}$, pro kterou platí

$$a_i = \pm x_{i-1} \pm \dots \pm x_1 \leq |x_{i-1}| + \dots + |x_1| \leq a_{i-1} + \dots + a_1. \quad (1)$$

Víme, že $a_1 = 1, a_2 = 1, a_3 = 2$. Pomocí předcházejícího vzorce dokažme, že $a_i = 2^{i-2}$ pro každé $i > 1$.

Důkaz provedeme matematickou indukcí vzhledem k i .

1. Tvrzení platí pro $i = 1$ ($a_1 = 1$) a $i = 2$ ($a_2 = 1$).

2. Předpokládejme, že tvrzení platí pro každé $k, 2 \leq k \leq i - 1$, a dokažme, že tvrzení platí i pro $k = i$. Z odhadu (1), indukčního předpokladu a vzorce pro součet geometrické řady dostaneme

$$a_i \leq a_{i-1} + \dots + a_1 = 2^{i-3} + \dots + 2 + 1 + 1 = \frac{2^{i-2} - 1}{2 - 1} + 1 = 2^{i-2}.$$

Uvažujme posloupnost $x_1 = 1$ a $x_i = x_{i-1} + \dots + x_1$ pro každé $i > 1$. V tomto případě bude podle předchozího platit $x_i = 2^{i-2}$, takže $a_i = 2^{i-2}$ pro každé $i > 1$.

Podobně dokažeme, že nejmenší hodnota, jaké může x_n nabýt, je -2^{n-2} .

Zjistili jsme, že pro každé $n > 1$ leží člen x_n libovolné uvažované posloupnosti v množině $\{-2^{n-2}, -2^{n-2} + 2, -2^{n-2} + 4, \dots, 2^{n-2}\}$, kterou označíme M_n . Dokažme nakonec, že x_n může pro $n > 1$ nabývat libovolnou hodnotu z množiny M_n .

Volme znaménka následujícím způsobem: $x_i = x_{i-1} + \dots + x_1$ pro $i < n$. Pro takovou posloupnost platí $x_i = 2^{i-2}$ pro $1 < i < n$. Dokažme, že v rovnosti $x_n = \pm 2^{n-3} \pm 2^{n-2} \pm \dots \pm 1 \pm 1$ lze znaménka vybrat tak, aby se hodnota x_n rovnala libovolně zvolenému číslu z množiny M_n .

Důkaz provedeme opět matematickou indukcí.

1. Tvrzení platí pro $n = 2$ ($-1 = -x_1$ a $1 = +x_1$, neboť $x_1 = 1$) a $n = 3$ ($-2 = -1 - 1$, $0 = -1 + 1$, $2 = 1 + 1$).

2. Předpokládejme, že tvrzení platí pro $k \leq n-1$, kde $n \geq 4$. Dokažme tvrzení pro $k = n$. Zvolme libovolné číslo a z množiny M_n . Dokážeme, že existuje taková volba znamének $+ a -$, že $a = \pm 2^{n-3} \pm 2^{n-4} \pm \dots \pm 1 \pm 1$. Rozeberme dvě možnosti.

1. $a \geq 0$. Protože $a \in M_n$, je $a - 2^{n-3}$ sudé celé číslo z intervalu $\langle -2^{n-3}, 2^{n-3} \rangle$, a tedy $a - 2^{n-3} \in M_{n-1}$. Z indukčního předpokladu plyne, že existuje volba znamének $+ a -$ taková, že $a - 2^{n-3} = \pm 2^{n-4} \pm 2^{n-5} \pm \dots \pm 1 \pm 1$. Potom $a = 2^{n-3} \pm 2^{n-4} \pm 2^{n-5} \pm \dots \pm 1 \pm 1$, což jsme chtěli dokázat.

2. $a < 0$. Podobně jako v předcházejícím případě dokážeme, že a se dá napsat ve tvaru $a = -2^{n-3} \pm 2^{n-4} \pm 2^{n-5} \pm \dots \pm 1 \pm 1$.

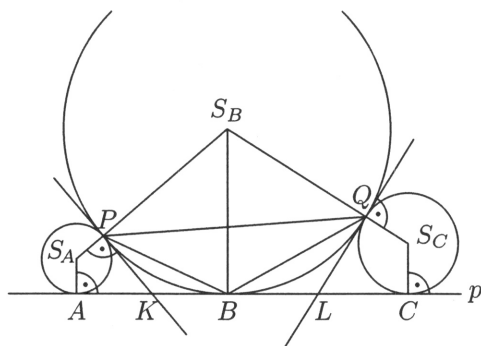
Tím jsme dokázali, že všechny hodnoty x_n tvoří právě množinu M_n .

A - I - 2

Zvolíme-li velikost $r_B > 0$ poloměru kružnice k_B , jsou už tím obě další kružnice k_A, k_C určeny. K jejich sestrojení využijeme základní vlastnosti tečen kružnic.

Předpokládejme, že kružnice k_A, k_B, k_C mají vlastnosti popsané v zadání. Označíme-li např. K průsečík vnitřní společné tečny kružnic k_A a k_B (v bodě P jejich vnějšího dotyku) s přímkou p , která je společnou vnější tečnou všech tří kružnic, musí být $|KA| = |KP|$ a $|KB| = |KP|$ (obr. 22). To znamená, že bod K je středem úsečky AB a zároveň bod P leží na Thaletově kružnici nad průměrem AB . Známe-li bod P , snadno už sestrojíme kružnici k_A , o níž víme, že se dotýká přímky p v bodě A . Analogicky sestrojíme kružnici k_C .

Máme zjistit, pro které hodnoty r_B je trojúhelník BPQ rovnoramenný. Protože body dotyku P, Q kružnice k_B s oběma sousedními kružnicemi leží uvnitř opačných polorovin určených přímkou BS_B , jsou oba úhly BPQ a BQP ostré (příslušné středové úhly jsou menší než 180°). Pokud tedy náhodou vyjde trojúhelník PBQ tupouhlý, může být rovnoramenný, jen když $|BP| = |BQ|$. V takovém případě je ale ze souměrnosti

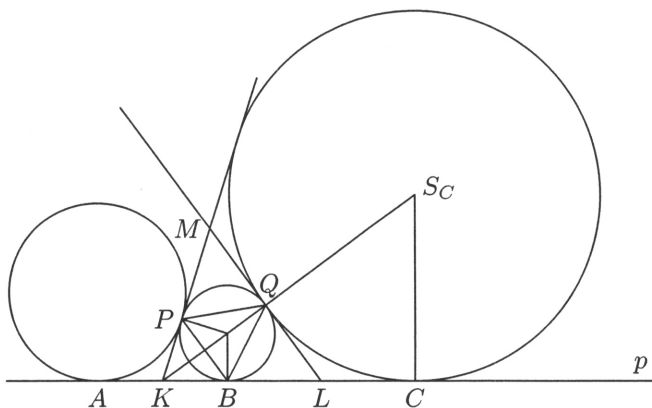


Obr. 22

zřejmé, že $|AB| = |BC|$, tj. $h = 1$. Trojúhelník BPQ je pak rovnoramenný pro každé $r_B > 0$.

Předpokládejme dále, že $h \neq 1$. V takovém případě můžeme předpokládat, že trojúhelník BPQ je ostroúhlý (jinak podle předchozího odstavce nemůže být rovnoramenný). Je-li rovnoramenný, je buď $|PQ| = |BQ|$, anebo $|PQ| = |BP|$. Předpokládejme, že je např. $|PQ| = |BQ|$ (jak ukážeme později, druhý případ lze řešit využitím souměrnosti).

Trojúhelník BPQ je souměrný podle spojnice $S_B S_C$ středů obou kružnic, která prochází bodem dotyku Q obou kružnic a průsečíkem K tečen KB , KP . Označme ještě L průsečík společné vnitřní tečny kružnic k_C a k_B s přímkou p (L je střed úsečky BC , obr. 23) a M průsečík obou tečen KP a LQ (ten je obrazem bodu L v uvedené



Obr. 23

osové souměrnosti). Trojúhelník KLM je tedy rovnoramenný se stranami $|KL| = |KM| = \frac{1}{2}(1+h)$, $|ML| = 2|LQ| = h$, jeho obvod je $1+2h$. Velikost poloměru r_B vepsané kružnice spočteme pomocí obsahu: Pro obsah S trojúhelníku KLM platí

$$S = \frac{1}{2}h\sqrt{\left(\frac{1+h}{2}\right)^2 - \left(\frac{h}{2}\right)^2} = \frac{1}{4}h\sqrt{1+2h}$$

a zároveň

$$S = \frac{1}{2}r_B(1+2h).$$

Odtud vychází

$$r_B = \frac{h}{2\sqrt{2h+1}}. \quad (1)$$

Naopak je-li r_B dáno vztahem (1), můžeme sestrojit rovnoramenný trojúhelník KLM s rameny KL a KM délky $\frac{1}{2}(1+h)$ a základnou ML , $|ML| = h$, přičemž jeho vepsaná kružnice k_B se bude dotýkat ramene KL v bodě B . Označme P , Q po řadě body dotyku kružnice k_B se stranami KM a LM . Protože K je střed úsečky AB , je $|KA| = |KB| = |KP|$. To znamená, že kružnice k_A dotýkající se přímky p v bodě A a procházející bodem P se bude dotýkat kružnice k_B v bodě P . Analogicky sestrojíme i kružnici k_C dotýkající se přímky p v bodě C a procházející bodem Q . Ze souměrnosti trojúhelníku KLM podle přímky KQ plyne, že $|PQ| = |BQ|$. Tím je první případ vyřešen.

V případě rovnosti $|PQ| = |BP|$ můžeme postupovat úplně stejně. Jednodušší však bude, když změníme měřítko původního obrázku v poměru $1 : h$, takže bude $|AB| = h' = 1/h$, $|BC| = 1$. Když navíc prohodíme označení bodů A a C , tak se z rovnosti $|BP| = |PQ|$ stane rovnost $|BQ| = |PQ|$. Podle předchozího pak pro velikost poloměru $r'_B = (1/h)r_B$ dostaneme

$$\frac{1}{h}r_B = r'_B = \frac{h'}{2\sqrt{2h'+1}} = \frac{\frac{1}{h}}{2\sqrt{2\frac{1}{h}+1}},$$

tj.

$$r_B = \frac{h}{2\sqrt{2h+h^2}}. \quad (2)$$

Anebo jsme mohli řešit úlohu poněkud obecněji za předpokladu $|AB| = a$, $|BC| = b$, pak bychom místo vztahu (1) dostali

$$r_B = \frac{b\sqrt{a^2+2ab}}{2(a+2b)}. \quad (1')$$

Pro $a = 1$, $b = h$ vyjde za předpokladu $|PQ| = |BQ|$ původní vztah (1), zatímco pro $|PQ| = |BP|$ prohodíme označení bodů A , C (a tím i bodů P , Q) a do vzorce (1') dosadíme $a = h$, $b = 1$. Dostaneme tak vztah (2).

Závěr: Pro $h = 1$ je trojúhelník BPQ rovnoramenný pro libovolné $r_B > 0$. Pro $h \neq 1$ je trojúhelník BPQ rovnoramenný pro r_B určené vztahem (1) ($|PQ| = |BQ|$) nebo pro r_B určené vztahem (2) ($|PQ| = |BP|$).

A – I – 3

Nejdříve ukážeme, že žádná hodnota zkoumaného výrazu V není menší než 1. Použijeme nerovnosti mezi aritmetickým a geometrickým průměrem ($x + y \geq 2\sqrt{xy}$) pro všechny dvojice kladných čísel x , y z množiny $\{a^4, b^4, c^4\}$:

$$\begin{aligned} V &= \frac{a^4 + b^4 + c^4}{a^2b^2 + a^2c^2 + b^2c^2} = \frac{1}{2} \cdot \frac{(a^4 + b^4) + (b^4 + c^4) + (c^4 + a^4)}{a^2b^2 + a^2c^2 + b^2c^2} \geq \\ &\geq \frac{1}{2} \cdot \frac{2 \cdot (a^2b^2 + b^2c^2 + c^2a^2)}{a^2b^2 + a^2c^2 + b^2c^2} = 1. \end{aligned}$$

Nyní ukážeme, že každá hodnota V je menší než 2. Z *Heronova vzorce* pro obsah S trojúhelníka se stranami a , b , c víme, že

$$S^2 = s(s-a)(s-b)(s-c),$$

kde $s = \frac{1}{2}(a+b+c)$. Po dosazení za s a roznásobení dostaneme

$$0 < 16S^2 = -a^4 - b^4 - c^4 + 2a^2b^2 + 2b^2c^2 + 2c^2a^2.$$

Odtud

$$a^4 + b^4 + c^4 < 2a^2b^2 + 2b^2c^2 + 2c^2a^2, \quad \text{tj.} \quad \frac{a^4 + b^4 + c^4}{a^2b^2 + b^2c^2 + c^2a^2} < 2.$$

Shrňme výsledek úvah prvních dvou odstavců: zjistili jsme, že $V \in \langle 1, 2 \rangle$. Ukažme, že všechny hodnoty V zaplní celý interval $V \in \langle 1, 2 \rangle$. Zvolíme libovolnou hodnotu $k \in \langle 1, 2 \rangle$ a najdeme trojúhelník, pro který má výraz V hodnotu k . Uvažujme trojúhelník se stranami a , 1 , 1 , který podle trojúhelníkové nerovnosti existuje, právě když $0 < a < 2$. Zjistíme, pro které a je $V = k$, proto vyřešíme rovnici

$$\frac{a^4 + 2}{2a^2 + 1} = k \tag{1}$$

s neznámou a . Po substituci $a^2 = b$ dostaneme kvadratickou rovnici $b^2 - 2kb + 2 - k = 0$ s neznámou b . Její diskriminant je roven $D = 4k^2 - 4(2 - k) = 4(k^2 + k - 2)$. Na to, aby měla rovnice řešení, musí být diskriminant nezáporný, tedy musí platit $k^2 + k - 2 \geq 0$. Tato nerovnost je splněna pro $k \in (-\infty, -2) \cup \langle 1, \infty)$, tedy i pro uvažované $k \in \langle 1, 2)$. Potřebujeme ještě dokázat, že zkoumaná rovnice má aspoň jeden kořen b v intervalu $(0, 4)$, neboť $b = a^2$ a $a \in (0, 2)$. Všimněme si, že pro oba kořeny $b_{1,2}$ platí

$$\begin{aligned} b_{1,2} &= \frac{2k \pm 2\sqrt{k^2 + k - 2}}{2} = k \pm \sqrt{k^2 + k - 2} \leq \\ &\leq k + \sqrt{k^2 + k - 2} < k + \sqrt{k^2} = 2k < 4, \end{aligned}$$

přičemž jsme využili nerovnost $k < 2$. Na druhé straně pro kořen b_1 (se znaménkem $+$ před $\sqrt{D}$) platí

$$b_1 = k + \sqrt{k^2 + k - 2} \geq k > 0.$$

Tím jsme ukázali, že $0 < b_1 < 4$. Existuje tedy číslo $a = \sqrt{b_1}$ splňující rovnici (1).

Jiné řešení. Opakovaným dosazováním délek stran konkrétních trojúhelníků dojdeme k hypotéze, že $1 \leq V < 2$. Dokazujme nejprve dolní odhad $1 \leq V$, který je ekvivalentní s nerovností

$$a^2b^2 + a^2c^2 + b^2c^2 \leq a^4 + b^4 + c^4.$$

Je to bikvadratická nerovnice s proměnnou a , takže po substituci $a^2 = t$ dostaneme kvadratickou nerovnici

$$0 \leq t^2 - t(b^2 + c^2) + b^4 + c^4 - b^2c^2. \quad (2)$$

Její diskriminant je $D = (b^2 + c^2)^2 - 4(b^4 + c^4 - b^2c^2) = -3(b^4 + c^4 - 2b^2c^2) = -3(b^2 - c^2)^2 \leq 0$. Protože navíc je koeficient při t^2 na pravé straně (2) kladný, je nerovnice (2) splněna pro všechna reálná čísla b, c a t . Tím je nerovnost $V \geq 1$ dokázána.

Přejdeme k nerovnosti $V < 2$. Danou nerovnici přenásobme kladným jmenovatelem, dostaneme

$$2(a^2b^2 + a^2c^2 + b^2c^2) > a^4 + b^4 + c^4.$$

Je to opět bikvadratická nerovnice s proměnnou a . Po substituci $t = a^2$ přejde nerovnice do tvaru $t^2 - 2t(b^2 + c^2) + b^4 + c^4 - 2b^2c^2 < 0$. Její diskriminant je $D = 4(b^2 + c^2)^2 - 4(b^4 + c^4 - 2b^2c^2) = 16b^2c^2$. Protože koeficient při t^2 je kladný, je řešením této nerovnice interval určený nerovnostmi

$$\frac{2(b^2 + c^2) - \sqrt{D}}{2} < t < \frac{2(b^2 + c^2) + \sqrt{D}}{2},$$

neboli

$$(b - c)^2 < t < (b + c)^2.$$

Tyto nerovnosti platí, protože $t = a^2$ a $|b - c| < a < b + c$ podle trojúhelníkových nerovností. Tím je nerovnost $V < 2$ dokázána.

Že hodnoty V zaplní celý interval $\langle 1, 2 \rangle$, dokážeme stejně jako v prvním řešení.

Jiný důkaz nerovnosti $V < 2$. Vyjděme z trojúhelníkové nerovnosti $|a - b| < c < a + b$. Po umocnění na druhou a následně úpravě dostaneme $-2ab < c^2 - a^2 - b^2 < 2ab$, tj. $|c^2 - a^2 - b^2| < 2ab$. Po dalším umocnění na druhou dostaneme

$$c^4 + b^4 + a^4 - 2c^2a^2 - 2c^2b^2 + 2a^2b^2 < 4a^2b^2,$$

neboli

$$c^4 + b^4 + a^4 < 2c^2a^2 + 2c^2b^2 + 2a^2b^2.$$

Odtud již plyne, že $V < 2$.

Poznámky. 1. Všimněme si, že podobné trojúhelníky mají stejnou hodnotu výrazu V . Skutečně, pokud a, b, c jsou strany trojúhelníku, jsou ka, kb, kc pro každé reálné $k > 0$ stranami podobného trojúhelníku a platí

$$\frac{(ka)^4 + (kb)^4 + (kc)^4}{(ka)^2(kb)^2 + (ka)^2(kc)^2 + (kb)^2(kc)^2} = \frac{a^4 + b^4 + c^4}{a^2b^2 + a^2c^2 + b^2c^2}.$$

To znamená, že bez újmy na obecnosti můžeme předpokládat, že $c = 1$. Máme tedy zkoumat obor hodnot výrazu

$$\frac{a^4 + b^4 + 1}{a^2b^2 + a^2 + b^2}$$

za předpokladu $|a - b| < 1 < a + b$, což zjednodušuje a zpřehledňuje výpočty.

2. V druhé části řešení jsme měli zjistit obor hodnot funkce $f(a) = \frac{a^4 + 2}{2a^2 + 1}$. Zřejmě $f(1) = 1$ a $f(0) = 2$. Ze spojitosti funkce f vyplývá, že na intervalu $(0, 1)$ nabývá všechny hodnoty z intervalu $\langle 1, 2 \rangle$.

3. Pečlivým rozбором uvedených důkazů zjistíme, že nerovnost $V \geq 1$ platí pro všechna reálná čísla a, b, c , z nichž aspoň dvě jsou nenulová.

A – I – 4

Protože $(abc)_z$ je číslo $az^2 + bz + c$, máme zjistit, kdy obecně platí ekvivalence: $n \mid c + 3b - 4a$, právě když $n \mid az^2 + bz + c$. V ní jsou a, b, c libovolné číslice při základu z , tj. čísla z množiny $\{0, 1, \dots, z-1\}$. Všimněme si, že $z-1 \geq 4$, neboť předpokládáme, že $z \geq 5$.

Zvolíme-li $a = b = c = 1$, dostaneme, že $n \mid 0$, právě když $n \mid z^2 + z + 1$. Protože nula je dělitelná každým celým číslem, musí platit $n \mid z^2 + z + 1$. Zvolíme-li $a = 1, b = 0$ a $c = 4$, dostaneme, že $n \mid 0$, právě když $n \mid z^2 + 4$. Podobnou úvahou jako výše zjistíme, že $n \mid z^2 + 4$.

Pokud nějaké číslo dělí dvě čísla, musí dělit i jejich největší společný dělitel, tedy $n \mid \text{nsd}(z^2 + 4, z^2 + z + 1)$. Tento společný dělitel najdeme pomocí *Eukleidova algoritmu*:

$$\begin{aligned} \text{nsd}(z^2 + 4, z^2 + z + 1) &= \\ &= \text{nsd}(z^2 + 4, z^2 + z + 1 - (z^2 + 4)) = \text{nsd}(z^2 + 4, z - 3) = \\ &= \text{nsd}(z^2 + 4 - z(z - 3), z - 3) = \text{nsd}(4 + 3z, z - 3) = \\ &= \text{nsd}(4 + 3z - 3(z - 3), z - 3) = \text{nsd}(13, z - 3). \end{aligned}$$

Zjistili jsme, že $n \mid 13$. Protože $n > 1$, je nutně $n = 13$. Má-li některé n požadovanou vlastnost, je to nutně číslo $n = 13$.

Dokažme, že číslo 13 skutečně danou vlastnost má. Odvozená nutná podmínka $n \mid \text{nsd}(13, z - 3)$ je pro $n = 13$ splněna např. pro $z = 16$. Daná ekvivalence má pak tvar $13 \mid c + 3b - 4a$, právě když $13 \mid a \cdot 16^2 + b \cdot 16 + c$. Dokážeme silnější vlastnost, že totiž čísla $a \cdot 16^2 + b \cdot 16 + c$ a $c + 3b - 4a$ dávají při dělení třinácti stejný zbytek, neboli že jejich rozdíl je dělitelný třináctí:

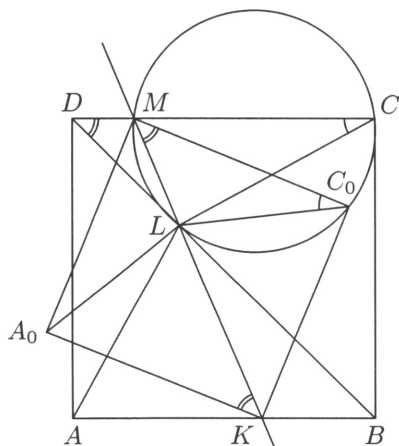
$$(a \cdot 16^2 + b \cdot 16 + c) - (c + 3b - 4a) = 260a + 13b = 13(20 + b).$$

Úloha má jediné řešení $n = 13$.

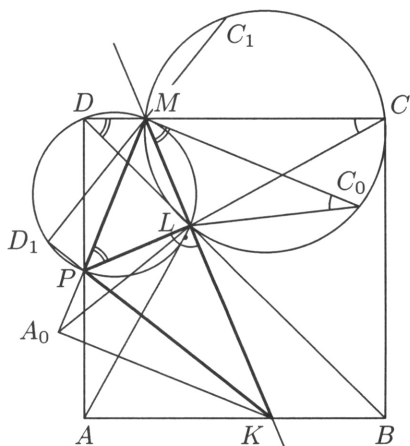
Poznámka. Podobně jako v závěru řešení můžeme dokázat, že uvedené kritérium dělitelnosti pro $n = 13$ platí i v libovolné číselné soustavě se základem $z = 13k + 3$.

Je-li $ABCD$ libovolný čtverec, který splňuje podmínky úlohy, bude stejným podmínkám vyhovovat i čtverec, který dostaneme osovou souměrností podle přímky MK . Hledaná množina bude tedy osově souměrná podle této přímky a nám stačí určit tu její část, která leží v jedné z obou polorovin s hraniční přímkou MK .

Kromě libovolného čtverce $ABCD$, který splňuje podmínky úlohy, uvažujme čtverec $A_0B_0C_0D_0$ s úhlopříčkou $B_0D_0 = KM$ ($B_0 \equiv K$, $D_0 \equiv M$), přičemž vrchol C_0 leží ve stejné polorovině ohraničené přímkou KM jako vrchol C čtverce $ABCD$ (obr. 24). (Vrchol C_0 zřejmě rovněž patří do hledané množiny.)



Obr. 24



Obr. 25

Protože trojúhelníky KLB a MLD jsou podobné podle věty uu , dělí bod L úhlopříčky obou čtverců ve stejném poměru

$$|BL| : |LD| = |KL| : |LM| = \text{konst.}$$

Velikost úhlu LCD ($|\sphericalangle LCD| = |\sphericalangle LC_0M|$) je určena polohou bodu L na úsečce MK , má tedy konstantní velikost, takže bod C leží na stejném oblouku γ kružnice opsané trojúhelníku LC_0M nad tětívou LM jako bod C_0 . Navíc kružnice opsaná trojúhelníku A_0KL je shodná s kružnicí opsanou trojúhelníku C_0ML , protože v jedné z nich je vidět tětivu A_0L

z bodu K pod úhlem 45° a v druhé tětivu C_0L shodné délky pod stejným úhlem z bodu M .

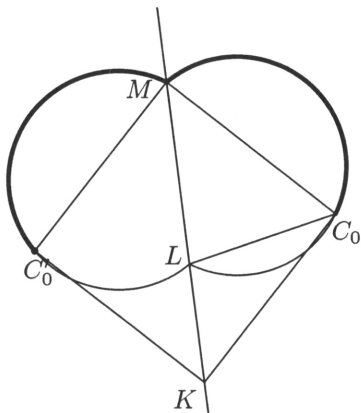
Protože bod M leží na straně CD , je zřejmě $|\sphericalangle LMC| \geq |\sphericalangle LDC| = 45^\circ$ (pokud $M \neq D$, je to vnější úhel trojúhelníku DML , který má při vrcholu D úhel 45°). Protože úhel LMC_0 měří právě 45° , leží bod C na části oblouku γ mezi body C_0 a M .

Dále si všimněme, že vrchol D čtverce $ABCD$ leží na oblouku, ze kterého je vidět úsečku LM pod úhlem 45° v polorovině opačné ke KMC . Sestrojme bod P (obr. 25), který leží na průsečíku přímky AD a kolmice k přímce MK v bodě L . Body M , D , L a P leží na Thaletově kružnici s průměrem MP , a protože $|\sphericalangle MPL| = |\sphericalangle MDL| = 45^\circ$, je trojúhelník MPL rovnoramenný pravoúhlý. To znamená, že bod P je jednoznačně určen polohou bodu L na úsečce MK . (Bod P vznikne otočením bodu M kolem středu L o 90° , protože bod L jako bod úhlopříčky BD má od přímek CD a DA stejnou vzdálenost, je tedy přímka DA ve zmíněném otočení obrazem přímky CD ; odtud rovněž plyne rovnost $|LM| = |LP|$.)

Bod D tudíž musí ležet na oblouku δ Thaletovy polokružnice nad průměrem MP v polorovině opačné k PML , zároveň však polopřímka DP (která obsahuje vrchol A) nesmí protnout úsečku LK . Odtud plyne, že vrchol D může ležet jen v té části zmíněné polokružnice nad průměrem MP , která leží v polorovině PKL . Přitom je zřejmé, že přímka PK tuto polokružnici protne v dalším bodě různém od P , právě když $|KL| > |LM|$ (pro $|KL| = |LM|$ bude KP tečnou kružnice nad průměrem MP). Označíme-li v takovém případě D_1 průsečík KP s polokružnicí δ a C_1 průsečík polopřímky D_1M s obloukem γ , je zřejmé, že vrchol C padne do části C_0C_1 oblouku γ . V opačném případě, tj. pro $|KL| \leq |LM|$, vyplní zřejmě vrcholy C celou část C_0M oblouku γ .

Skutečně. Zvolme libovolný bod C na části C_0C_1 oblouku γ v prvním případě, resp. na C_0M v druhém případě. Přímka CM protne oblouk δ v bodě, který označíme D . Vrchol A pak sestrojíme jako průsečík polopřímky DP s Thaletovou kružnicí nad průměrem PK (v prvním případě máme zaručeno, že bude ležet v polorovině PKA_0 , a ne v opačné). Protože jak už víme, jsou kružnice opsané trojúhelníkům LC_0M a LA_0K shodné, zjistíme snadno z příslušných obvodových úhlů, že $|\sphericalangle DAL| = |\sphericalangle DCL|$, takže trojúhelníky DAL a DCL jsou shodné, tudíž $|DA| = |DC|$. Protože polopřímka DL protíná úsečku MK v bodě L , protne polopřímka AK polopřímku DL v bodě B za bodem K , přičemž trojúhelník DAB je rovnoramenný pravoúhlý. Je tedy $ABCD$ čtverec, který splňuje podmínky úlohy.

to oblouk C_0M stejné kružnice a oblouk s ním osově souměrný podle přímky MK (obr. 27).



80

A – I – 6

Označme p_i pravděpodobnost, že vyhraje hráč A , přičemž figurka stojí na i -tém políčku. Dostaneme soustavu rovnic

$$\begin{aligned} p_2 &= \frac{1}{3} + \frac{2}{3}p_3, \\ p_3 &= \frac{1}{3}p_2 + \frac{2}{3}p_4, \\ p_4 &= \frac{1}{3}p_3 + \frac{2}{3}p_5, \\ p_5 &= \frac{1}{3}p_4. \end{aligned}$$

Postupným dosazováním z jedné rovnice do druhé dostaneme řešení $p_2 = \frac{15}{31}$. Protože na začátku stojí figurka na políčku číslo 2, je pravděpodobnost výhry hráče A rovna $\frac{15}{31}$. Podobným způsobem sestavíme a vyřešíme systém rovnic pro hráče B a dostaneme tak, že hráč B vyhraje s pravděpodobností $\frac{16}{31}$. Pravděpodobnost remízy je tedy 0. Tím je úloha vyřešena.

A – S – 1

Pro libovolně vybraných dvacet přirozených čísel

$$x_1 < x_2 < \dots < x_{20}$$

odhadneme, kolik mezi nimi může být součtových trojic, tedy trojic $\{x_i, x_j, x_k\}$ splňujících podmínky $1 \leq i < j < k \leq 20$ a $x_i + x_j = x_k$, a to nejprve při pevném indexu $k \in \{3, 4, \dots, 20\}$. Nechť jsou to trojice $\{x_{i_1}, x_{j_1}, x_k\}, \{x_{i_2}, x_{j_2}, x_k\}, \dots, \{x_{i_p}, x_{j_p}, x_k\}$. Pak čísla

$$x_{i_1}, x_{j_1}, x_{i_2}, x_{j_2}, \dots, x_{i_p}, x_{j_p}$$

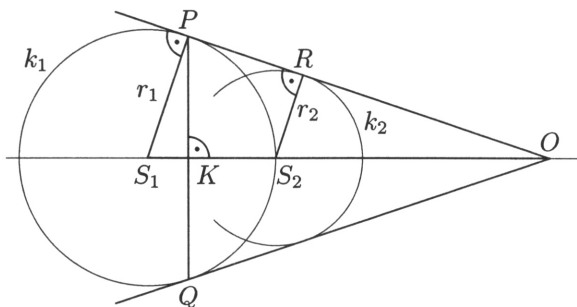
jsou navzájem různá a všechna leží v množině $\{x_1, x_2, \dots, x_{k-1}\}$, takže pro jejich počet $2p$ platí odhad $2p \leq k - 1$, odkud $p \leq [\frac{1}{2}(k - 1)]$ (kde $[a]$ značí celou část čísla a). Proto počet všech součtových trojic nemůže být číslo větší než součet

$$\sum_{k=3}^{20} \left[\frac{k-1}{2} \right] = 1 + 1 + 2 + 2 + 3 + 3 + \dots + 9 + 9 = 90.$$

Příklad množiny $M = \{1, 2, \dots, 20\}$ ukazuje, že počet 90 součtových trojic je dosažitelný, neboť při každém $k \in \{3, 4, \dots, 20\}$ můžeme za číslo i vybrat libovolné číslo z množiny $\{1, 2, \dots, [\frac{1}{2}(k-1)]\}$; odpovídající celé číslo $j = k - i$ pak skutečně splňuje nerovnosti $i < j < k$, takže $\{i, j, k\}$ je součtová trojice ležící v M .

A – S – 2

Ze souměrnosti společných tečen plyne, že body dotyku P a Q jsou souměrně sdružené podle přímky S_1S_2 , takže platí $PQ \perp S_1S_2$. Přímka PQ proto bude tečnou ke kružnici k_2 , když ukážeme, že průsečík K přímek PQ a S_1S_2 leží na kružnici k_2 (obráz. 28). Označme ještě O průsečík obou



Obr. 28

tečen s přímkou S_1S_2 a R bod dotyku tečny PO s kružnicí k_2 . Z podobných pravoúhlých trojúhelníků S_1OP a S_2OR plyne úměra

$$\frac{r_1}{r_2} = \frac{|S_1P|}{|S_2R|} = \frac{|S_1O|}{|S_2O|} = \frac{|S_1O|}{|S_1O| - r_1}, \quad \text{odkud} \quad |S_1O| = \frac{r_1^2}{r_1 - r_2}.$$

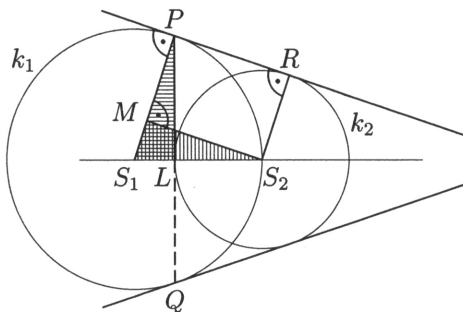
Z Eukleidovy věty o odvěsně S_1P trojúhelníku S_1OP proto vyplývá, že

$$r_1^2 = |S_1P|^2 = |S_1K| \cdot |S_1O| = |S_1K| \frac{r_1^2}{r_1 - r_2},$$

tudíž $|S_1K| = r_1 - r_2$, a proto $|S_2K| = |S_1S_2| - |S_1K| = r_1 - (r_1 - r_2) = r_2$. To znamená, že bod K skutečně leží na kružnici k_2 a důkaz tvrzení je hotov.

Jiné řešení. Označme L průsečík kružnice k_2 s úsečkou S_1S_2 , M patu kolmice spuštěné z bodu S_2 na úsečku S_1P a R bod dotyku

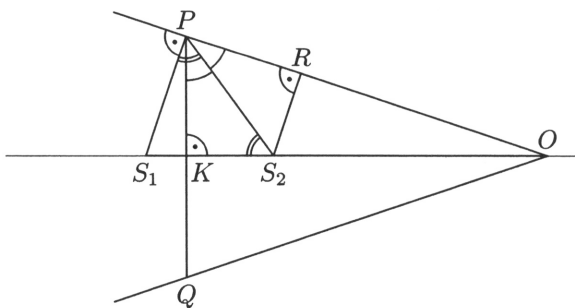
kružnice k_2 s tou společnou tečnou, která prochází bodem P (obr. 29). Protože S_2RPM je pravoúhelník, platí $|MP| = |S_2R| = r_2$, a proto



Obr. 29

$|S_1M| = |S_1P| - |MP| = r_1 - r_2$. Stejnou délku $r_1 - r_2$ má rovněž úsečka S_1L , neboť $r_1 = |S_1S_2|$ a $r_2 = |S_2L|$. Trojúhelníky S_1MS_2 a S_1LP mají tudíž shodné úhly při vrcholu S_1 i přilehlé strany, jsou proto shodné podle věty *sus*. Platí tedy nejen $S_1M \perp S_2M$, ale také $S_1L \perp PL$. Bod L ale leží na kružnici k_2 , takže přímka PL je její tečnou, která s ohledem na souměrnost prochází rovněž bodem Q . Důkaz je ukončen.

Jiné řešení. Označme O průsečík obou tečen, K patu kolmice z bodu P na OS_1 (vzhledem k souměrnosti obou tečen podle spojnice S_1S_2 je to průsečík PQ s OS_1) a R patu kolmice z bodu S_2 na OP (obr. 30).



Obr. 30

Protože $|S_1P| = |S_1S_2| = r_1$, je $|\sphericalangle S_1PS_2| = |\sphericalangle S_1S_2P|$, proto

$$\begin{aligned} |\sphericalangle S_2PK| &= 90^\circ - |\sphericalangle S_1S_2P| = \\ &= 90^\circ - |\sphericalangle S_1PS_2| = |\sphericalangle S_2PR|, \end{aligned}$$

takže pravoúhlé trojúhelníky KS_2P a RS_2P se shodují v přeponě S_2P a přilehlém úhlu u vrcholu P . Je tudíž $|S_2K| = |S_2R|$ a kružnice se středem S_2 a poloměrem $r_2 = |S_2R|$ se dotýká spojnice PQ v bodě K .

A – S – 3

Levou stranu druhé rovnice upravíme na součin:

$$x^3 - 2x^2 - px + 2p = x^2(x - 2) - p(x - 2) = (x - 2)(x^2 - p).$$

Pro společný kořen x obou rovnic tedy platí $x = 2$ nebo $x^2 = p$. V prvním případě po dosazení do první rovnice dostaneme

$$2^3 + 2^2 - 36 \cdot 2 - p = 0, \quad \text{neboli} \quad p = -60;$$

ve druhém případě můžeme první rovnici zjednodušit na tvar $x^3 - 36x = 0$, odkud plyne $x = 0$ nebo $x = \pm 6$, a proto z podmínky $p = x^2$ vychází $p = 0$ respektive $p = 36$.

Dodejme, že po nalezení rozkladu levé strany druhé rovnice jsme mohli vypsát její kořeny $x_1 = 2$, $x_{2,3} = \pm\sqrt{p}$ a po jejich postupném dosazení do první rovnice určit hledané hodnoty $p = -60, p = 0$ a $p = 36$.

Jiné řešení. Z první rovnice snadno vyjádříme $p = x^3 + x^2 - 36x$ a dosazením do druhé rovnice dostaneme rovnici (bez parametru p), kterou musí splňovat společný kořen obou původních rovnic:

$$x^3 - 2x^2 - (x^3 + x^2 - 36x)x + 2(x^3 + x^2 - 36x) = 0.$$

Po úpravě dostaneme rovnici $x^4 - 2x^3 - 36x^2 + 72x = 0$, jejíž kořeny snadno určíme (jsou to totiž celá čísla) například postupným rozkladem:

$$\begin{aligned} x^4 - 2x^3 - 36x^2 + 72x &= x[x^2(x - 2) - 36(x - 2)] = x(x - 2)(x^2 - 36) = \\ &= x(x - 2)(x - 6)(x + 6). \end{aligned}$$

Vidíme, že společným kořenem musí být jedno z čísel $x_1 = 0$, $x_2 = 2$, $x_3 = 6$, $x_4 = -6$. Dosadíme-li je do původních rovnic, ihned zjistíme příslušné hodnoty p ; jsou to čísla 0, -60 a 36 (poslední odpovídá oběma kořenům $x_{3,4} = \pm 6$).

Jiné řešení. Společné kořeny mnohočlenů

$$P_1(x) = x^3 + x^2 - 36x - p, \quad P_2(x) = x^3 - 2x^2 - px + 2p$$

(pokud vůbec existují) jsou kořeny mnohočlenu, který je největším společným dělitelem mnohočlenů P_1 a P_2 . Najdeme ho Eukleidovým algoritmem postupného dělení se zbytkem. V prvních dvou krocích dostaneme jako zbytky mnohočleny

$$P_3(x) = P_1(x) - P_2(x) = 3x^2 + (p - 36)x - 3p,$$

$$P_4(x) = P_2(x) - \left(\frac{x}{3} + \frac{30-p}{9} \right) P_3(x) = (p-36) \left(\frac{(p-30)x}{9} - \frac{p}{3} \right).$$

V případě, kdy $p = 36$, je algoritmus ukončen; největší společný dělitel je tehdy roven $P_3(x) = 3x^2 - 3 \cdot 36 = 3(x-6)(x+6)$, takže mnohočleny P_1 , P_2 mají dva společné kořeny $x = \pm 6$. Dále proto předpokládejme, že $p \neq 36$. Jediný kandidát na společný kořen mnohočlenů P_1 , P_2 je kořen mnohočlenu P_4 , tedy číslo $x = 3p/(p-30)$. Stačí jen zjistit, kdy je toto číslo kořenem mnohočlenu P_3 . Protože

$$P_3 \left(\frac{3p}{p-30} \right) = \frac{9p(p+60)}{(p-30)^2},$$

mají požadovanou vlastnost pouze hodnoty $p = 0$ a $p = -60$ (kterým odpovídá společný kořen $x = 0$ respektive $x = 2$).

A – II – 1

Protože v zápisu dvojmístného čísla vystupuje číslice 4, nutně platí $z \geq 5$. Z rozvinutých zápisů $(1001)_z = z^3 + 1$ a $(41)_z = 4z + 1$ vyplývá, že hledáme právě ta přirozená $z \geq 5$, pro která je číslo $z^3 + 1$ násobkem čísla $4z + 1$. Pomocí Eukleidova algoritmu najdeme jejich největší společný dělitel. Můžeme postupovat tak, že nejprve vydělíme oba výrazy jako mnohočleny a pak se „zbavíme“ zlomků:

$$\begin{aligned} z^3 + 1 &= \left(\frac{1}{4}z^2 - \frac{1}{4^2}z + \frac{1}{4^3} \right) (4z + 1) + \frac{63}{4^3}, & / \cdot 4^3 \\ 4^3(z^3 + 1) &= (16z^2 - 4z + 1)(4z + 1) + 63. \end{aligned} \quad (1)$$

Protože čísla 4 a $4z + 1$ jsou nesoudělná, vidíme odtud, že číslo $4z + 1$ dělí číslo $z^3 + 1$, právě když dělí číslo 63, tedy právě když $4z + 1 \in \{1, 3, 7, 9, 21, 63\}$. Z podmínky $z \geq 5$ ovšem plyne $4z + 1 \geq 21$, takže $4z + 1 = 21$ (rovnice $4z + 1 = 63$ nemá celočíselné řešení) a $z = 5$.

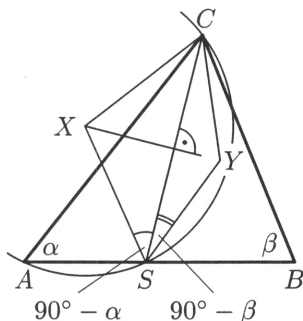
Poznámka. Rozklad (1) také snadno odhalíme, využijeme-li známý vzorec $a^3 + b^3 = (a+b)(a^2 - ab + b^2)$: podle něj můžeme rovnou psát

$$4^3(z^3 + 1) = (4^3z^3 + 1) + 63 = (4z + 1)(16z^2 - 4z + 1) + 63.$$

A – II – 2

Vnitřní úhly trojúhelníku ABC označme jako obvykle α, β, γ .

Podle věty o obvodovém a středovém úhlu v kružnici opsané trojúhelníku ASC platí (obr. 31) $|\sphericalangle XSC| = 2\alpha$, tudíž úhel při základně SC



Obr. 31

rovnoramenného trojúhelníku SCX má velikost $|\sphericalangle XSC| = \frac{1}{2}(180^\circ - 2\alpha) = 90^\circ - \alpha$ (využili jsme předpokladu, že α je ostrý úhel). Analogicky se odvodí rovnost $|\sphericalangle YSC| = 90^\circ - \beta$. Protože úhly při vrcholech A a C trojúhelníku ASC jsou ostré, je střed X vnitřním bodem úhlu ASC ; obdobně je střed Y vnitřním bodem úhlu BSC . Proto lze vyjádřit velikost úhlu $XS Y$ jako součet velikostí úhlů XSC a YSC :

$$|\sphericalangle XS Y| = |\sphericalangle XSC| + |\sphericalangle YSC| = (90^\circ - \alpha) + (90^\circ - \beta) = \gamma.$$

Označíme-li ještě $\omega = |\sphericalangle ASC|$, pak pro poloměry kružnic opsaných trojúhelníkům ASC a BSC platí vzorce

$$|SX| = \frac{|AC|}{2 \sin \omega} \quad \text{a} \quad |SY| = \frac{|BC|}{2 \sin(180^\circ - \omega)} = \frac{|BC|}{2 \sin \omega},$$

které spolu s dříve určenou velikostí úhlu $XS Y$ vedou k následující závislosti mezi obsahy S_{SXY} a S_{ABC} trojúhelníků SXY a ABC :

$$S_{SXY} = \frac{|SX| \cdot |SY| \cdot \sin |\sphericalangle XS Y|}{2} = \frac{|AC| \cdot |BC| \cdot \sin \gamma}{8 \sin^2 \omega} = \frac{S_{ABC}}{4 \sin^2 \omega}.$$

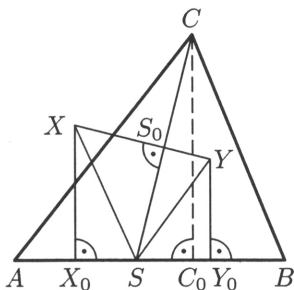
Odtud plyne nerovnost $S_{SXY} \geq \frac{1}{4} S_{ABC}$, přičemž rovnost nastane, právě když $\sin \omega = 1$, neboli $\omega = 90^\circ$. Obsah trojúhelníku SXY je proto

nejmenší, právě když je bod S patou výšky z vrcholu C ke straně AB . (Tato pata je vnitřním bodem strany AB díky podmínce, že trojúhelník ABC je ostroúhlý.)

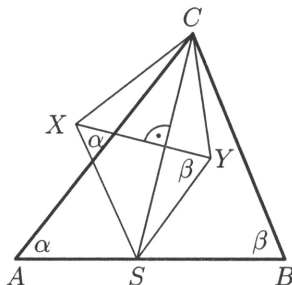
Jiné řešení. Středná XY obou opsaných kružnic protíná společnou tětivu CS v jejím středu S_0 a kolmé průměty X_0, Y_0 bodů X, Y na stranu AB jsou středy úseček AS, SB (obr. 32). Je tedy $|X_0Y_0| = \frac{1}{2}|AB|$ a pro obsah trojúhelníku SXY tudíž platí

$$\begin{aligned} S_{SXY} &= \frac{1}{2}|XY| \cdot |S_0S| \geq \frac{1}{2}|X_0Y_0| \cdot |S_0S| = \\ &= \frac{1}{2} \cdot \frac{1}{2}|AB| \cdot \frac{1}{2}|CS| \geq \frac{1}{4} \cdot \frac{1}{2}|AB| \cdot |CC_0| = \frac{1}{4}S_{ABC}, \end{aligned}$$

kde CC_0 je výška trojúhelníku ABC . Rovnost v první z předchozích dvou nerovností nastane, právě když $XY \parallel AB$, tj. právě když $CS \perp AB$, neboli $S = C_0$. A právě tehdy přejde v rovnost i druhá nerovnost.



Obr. 32



Obr. 33

Jiné řešení. Průsečíky C a S kružnic opsaných trojúhelníkům ASC a BSC jsou souměrně sdružené podle přímky XY , takže pro velikost úhlu SXY platí (obr. 33)

$$|\sphericalangle SXY| = \frac{1}{2}|\sphericalangle SXC| = \frac{1}{2} \cdot 2|\sphericalangle BAC| = |\sphericalangle BAC|,$$

obdobně $|\sphericalangle SYX| = |\sphericalangle ABC|$. Proto jsou trojúhelníky SXY a CAB podobné podle věty *uu*, takže jejich obsahy S_{SXY} a S_{ABC} jsou pomocí koeficientu podobnosti $k = |XS| : |AC|$ svázány rovností $S_{SXY} = k^2 S_{ABC}$. Protože úsečka AC je tětivou kružnice o poloměru $|XS|$, platí nerovnost $|AC| \leq 2|XS|$, neboli $k \geq \frac{1}{2}$; rovnost $k = \frac{1}{2}$ přitom nastane,

jen když je strana AC průměrem kružnice opsané trojúhelníku ASC , což je ekvivalentní s podmínkou $CS \perp AB$. Tím je dokázána nerovnost $S_{SXY} \geq \frac{1}{4}S_{ABC}$ i nalezena podmínka, kdy nastane rovnost.

A – II – 3

Rovnice dané soustavy mají smysl, jen když jsou čísla x, y, z kladná a různá od 1. Pro taková čísla x, y, z (jiná dále neuvažujeme) dostáváme odlogaritmováním ekvivalentní soustavu rovnic

$$y + z = x^p, \quad z + x = y^p, \quad x + y = z^p. \quad (1)$$

Ukážeme nejprve, že v oboru $M = (0, 1) \cup (1, \infty)$ je každé řešení soustavy (1) tvořeno trojicí stejných čísel. Využijeme k tomu známého poznatku, že pro $p \geq 0$ je funkce $f(t) = t^p$ na množině kladných čísel t neklesající (přesněji: rostoucí pro $p > 0$ a konstantní pro $p = 0$). Připusťme naopak, že pro některé řešení (x, y, z) platí například $x < y$.

Odečtením prvních dvou rovnic z (1) dostaneme $y - x = x^p - y^p$. Z předpokladu $x < y$ ale plyne $x^p \leq y^p$, takže $y - x > 0$ a zároveň $x^p - y^p \leq 0$, což je ve sporu s předchozí rovností. Podobně odvodíme spor i v případech, kdy $x > y$, a v případech, kdy $x \neq z$ resp. $y \neq z$ (soustava (1) je totiž v neznámých x, y, z symetrická).

Soustava (1) se proto redukuje na rovnici $x + x = x^p$, kterou máme řešit v oboru $M = (0, 1) \cup (1, \infty)$. Protože $x \neq 0$, dostáváme po dělení číslem x ekvivalentní rovnici $2 = x^{p-1}$. Tato rovnice nemá řešení pro $p = 1$, pro $p = 0$ má jediné řešení $x = \frac{1}{2}$, pro přirozené $p \geq 2$ má jediné řešení $x = 2^{\frac{1}{p-1}}$, které lze pro $p \geq 3$ zapsat jako $x = \sqrt[p-1]{2}$. (Čísla $\frac{1}{2}$ i $2^{\frac{1}{p-1}}$ zřejmě náležejí do M .)

Odpověď. Daná soustava má pro $p = 0$ jediné řešení $x = y = z = \frac{1}{2}$, pro $p = 1$ nemá řešení, pro přirozené $p \geq 2$ má jediné řešení $x = y = z = 2^{\frac{1}{p-1}}$.

A – II – 4

Úvodem si všimneme, že v důsledku rovnosti $x_1 = 1$ platí

$$\begin{aligned} x_2 &= x_1^{\pm 1} = 1^{\pm 1} = 1, \\ x_3 &= x_2^{\pm 1} + x_1^{\pm 1} = 1^{\pm 1} + 1^{\pm 1} = 2. \end{aligned} \quad (1)$$

Protože pro každé $x > 0$ jsou obě čísla x^{+1}, x^{-1} kladná, plyne od tud snadno matematickou indukcí, že nerovnost $x_n \geq 1$ je splněna pro

každé n . Je-li $x \geq 1$, pak ovšem $0 < x^{-1} \leq x^1$, a proto pro každé $n \geq 4$ platí odhady

$$1 + 1 + \frac{1}{2} + \frac{1}{x_4} + \dots + \frac{1}{x_{n-1}} \leq x_n \leq 1 + 1 + 2 + x_4 + \dots + x_{n-1}, \quad (2)$$

které využijeme ve všech třech částech řešení.

a) Dokážeme (sporem), že existuje k , pro něž $x_k > 10^3$. Pripusťme naopak, že pro každé k platí opačná nerovnost $x_k \leq 10^3$. Z levé nerovnosti v (2) pak pro každé $n > 4$ plyne odhad

$$x_n \geq \frac{5}{2} + \underbrace{10^{-3} + 10^{-3} + \dots + 10^{-3}}_{(n-4) \text{ krát}} = \frac{5}{2} + (n-4) \cdot 10^{-3}.$$

Odtud ale vyplývá, že $x_n > 10^3$ pro každé $n > 10^6 + 4$, což je spor.

b) Dokážeme nejprve, že z pravých nerovností v (2) vyplývá odhad $x_n \leq 2^{n-2}$ pro každé $n \geq 2$. Využijeme indukci: pro $n = 2$ i pro $n = 3$ platí podle (1) rovnost $x_n = 2^{n-2}$; necht' $n \geq 4$ a necht' pro všechna $k \in \{2, 3, \dots, n-1\}$ platí $x_k \leq 2^{k-2}$, potom z (2) dostáváme

$$x_n \leq 1 + (1 + 2 + 2^2 + \dots + 2^{n-3}) = 1 + (2^{n-2} - 1) = 2^{n-2}.$$

Tím je důkaz indukci ukončen.

Dosadíme-li odhady $x_n \leq 2^{n-2}$ do levé nerovnosti v (2), vyjde nám pro hodnotu x_{10^6} dolní odhad

$$x_{10^6} \geq 2 + \frac{1}{2} + \frac{1}{2^2} + \dots + \frac{1}{2^{10^6-3}} = 3 - \frac{1}{2^{10^6-3}}.$$

To spolu s příkladem vyhovující posloupnosti

$$\begin{aligned} x_n &= x_{n-1} + x_{n-2} + \dots + x_1 \quad (n \in \{2, 3, \dots, 10^6 - 1\}), \\ x_{10^6} &= x_{10^6-1}^{-1} + x_{10^6-2}^{-1} + \dots + x_2^{-1} + x_1^{-1}, \end{aligned}$$

ve které $x_n = 2^{n-2}$ pro každé $n \in \{2, 3, \dots, 10^6 - 1\}$ a $x_{10^6} = 3 - 2^{3-10^6}$, ukazuje, že nejmenší možná hodnota členu x_{10^6} je rovna $3 - 2^{3-10^6}$.

c) Předpokládejme, že nerovnost $x_n < 4$ platí kromě tří hodnot $n \in \{1, 2, 3\}$ ještě pro některá další n , která označíme $n_4, n_5, n_6, \dots$ tak, že $4 \leq n_4 < n_5 < n_6 < \dots$ (zatím ještě nevíme, zda jde o konečnou či

nekonečnou posloupnost). Ukažme, že pro každé takové n_k jsou ve všech exponentech příslušné rovnosti

$$x_{n_k} = 1 + 1 + 2^{\pm 1} + x_4^{\pm 1} + x_5^{\pm 1} + \dots + x_{n_k-1}^{\pm 1}$$

vybrána znaménka „minus“. Pro mocninu $2^{\pm 1}$ je to zřejmé, neboť $x_{n_k} < 4$; z téže nerovnosti dále plyne, že znaménko v exponentu kterékoli mocniny $x_j^{\pm 1}$ ($4 \leq j \leq n_k - 1$) musí být vybráno tak, aby platilo

$$x_j^{\pm 1} < 4 - \frac{5}{2} = \frac{3}{2}.$$

Tato nerovnost však může být splněna pouze se znaménkem „minus“, neboť podle (2) máme $x_j \geq 5/2$. Tím je tvrzení o výběru znamének dokázáno. Porovnáním dvou za sebou jdoucích rovností

$$\begin{aligned} x_{n_k} &= \frac{5}{2} + \frac{1}{x_4} + \frac{1}{x_5} + \dots + \frac{1}{x_{n_k-1}}, \\ x_{n_{k+1}} &= \frac{5}{2} + \frac{1}{x_4} + \frac{1}{x_5} + \dots + \frac{1}{x_{n_k-1}} + \frac{1}{x_{n_k}} + \dots + \frac{1}{x_{n_{k+1}-1}} \end{aligned}$$

dostaneme pro všechna $k \geq 4$ nerovnosti

$$x_{n_{k+1}} \geq x_{n_k} + \frac{1}{x_{n_k}},$$

které s přihlédnutím k tomu, že funkce $f(t) = t + 1/t$ je na intervalu $t \in \langle 1, \infty \rangle$ rostoucí a že $x_{n_4} \geq 2,5$, vedou postupně k odhadům

$$\begin{aligned} x_{n_5} &\geq f(2,5) = 2,9, \quad x_{n_6} \geq f(2,9) > 3,24, \quad x_{n_7} > f(3,24) > 3,54, \\ x_{n_8} &> f(3,54) > 3,82, \quad x_{n_9} \geq f(3,82) > 4,08. \end{aligned}$$

Poslední nerovnost je ale ve sporu s podmínkou $x_{n_k} < 4$ určující výběr indexů n_k . Proto nerovnost $x_n < 4$ nemůže platit pro devět indexů n .

A – III – 1

Protože druhou rovnici můžeme upravit na tvar $xy(x+y) = -2$, upravme podobně i první rovnici: $(x+y)^2 - 3xy = 7$. Pro čísla $s = x+y$, $p = xy$ tak dostáváme ekvivalentní soustavu

$$\begin{aligned} s^2 - 3p &= 7, \\ sp &= -2, \end{aligned} \tag{1}$$

která po vyjádření $p = -2/s$ (zřejmě nemůže být $s = 0$) z druhé rovnice vede na kubickou rovnici $s^3 - 7s + 6 = 0$. Ta má celočíselné kořeny $s_1 = 1$, $s_2 = 2$ a $s_3 = -3$. Nalezeným hodnotám s odpovídají tyto hodnoty součinu $p = xy$: $p_1 = -2$, $p_2 = -1$, $p_3 = \frac{2}{3}$. Čísla x , y tvoří dvojici kořenů kvadratické rovnice $t^2 - st + p = 0$, takže se jedná o jednu z rovnic

$$t^2 - t - 2 = 0, \quad t^2 - 2t - 1 = 0, \quad t^2 + 3t + \frac{2}{3} = 0.$$

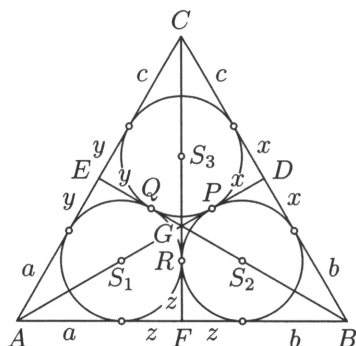
Jejich řešením dostaneme (všech) šest řešení dané soustavy:

$$\{x, y\} = \{-1, 2\}, \quad \{x, y\} = \{1 + \sqrt{2}, 1 - \sqrt{2}\},$$

$$\{x, y\} = \left\{ \frac{-9 + \sqrt{57}}{6}, \frac{-9 - \sqrt{57}}{6} \right\}.$$

A – III – 2

Předpokládejme, že zmíněné čtyřúhelníky mají uvedenou vlastnost. Ze souměrnosti tečen z daného bodu k dané kružnici vyplývá, že strany trojúhelníku ABC jsou rozděleny body D , E , F a body dotyku kružnic vepsaných uvažovaným čtyřúhelníkům na úseky délek, jež označíme podle obr. 34. Jsou na něm rovněž vyznačeny body P , Q , R vzájemného dotyku zmíněných kružnic. Naším cílem je dokázat rovnosti $x = y = z$ a $a = b = c$.



Obr. 34

Pro úseky tečen z bodu A ke kružnicím při straně BC platí rovnosti $a + 2z = |AP| = a + 2y$, odkud ihned plyne $y = z$; z důvodů symetrie

tudíž skutečně platí $x = y = z$. (Všude dále budeme psát x namísto y a z .) Všimněme si nyní trojúhelníků AEG a AFG . Mají společnou stranu AG a shodné strany AF a AE (délky $a + x$). Také jejich třetí strany EG a FG jsou shodné:

$$|EG| = |EQ| + |QG| = x + |RG| = |FR| + |RG| = |FG|.$$

Proto $\triangle AEG \simeq \triangle AFG$ podle věty *sss*, tudíž úhly BAD a CAD jsou shodné a polopřímka AD je osou úhlu BAC . Jak víme, osa úhlu trojúhelníku protíná protější stranu v poměru délek přilehlých stran. V našem případě to znamená, že

$$\frac{a + 2x + b}{a + 2x + c} = \frac{b + x}{c + x}.$$

Snadnou úpravou dostaneme rovnost $(b - c)(a + x) = 0$, ze které vidíme, že $b = c$. Z důvodů symetrie tudíž platí $a = b = c$ a celý důkaz je hotov.

Jiné řešení. Označme S_1, S_2, S_3 středy vepsaných kružnic (obr. 34). Stejně jako v předchozím řešení si nejprve všimneme, že platí $x = y = z$ a že trojúhelníky AEG a AFG jsou shodné. K tomu jsme využili rovnost $|GQ| = |GR|$, ze které plyne, že podle věty *sss* jsou shodné i trojúhelníky S_1QG a S_1RG . Jelikož $R \in S_1S_2$ a $Q \in S_1S_3$, ze souměrnosti podle osy AD nyní plyne, že přímky AB a S_1S_2 svírají stejný úhel jako přímky AC a S_1S_3 , a protože kolmé průměty úseček S_1S_2 a S_1S_3 na odpovídající přímky AB , resp. AC jsou shodné (mají délku $2x$), je $|S_1S_2| = |S_1S_3|$. Analogicky $|S_1S_2| = |S_2S_3|$, takže trojúhelník $S_1S_2S_3$ je rovnostranný. Odtud pro poloměry r_1, r_2 a r_3 vepsaných kružnic plyne $r_1 + r_2 = r_2 + r_3 = r_3 + r_1$, neboli $r_1 = r_2 = r_3$. Kružnice jsou tedy shodné, takže je $AB \parallel S_1S_2, BC \parallel S_2S_3$ a $CA \parallel S_3S_1$ a trojúhelník ABC je rovnostranný.

Poznámka. K dokončení předchozího důkazu můžeme úvahu o délkách úseček S_iS_j nahradit úvahou o tzv. *orientovaných úhlech* mezi přímkami. Orientovaný úhel $\langle p, q \rangle$ přímek p, q (v tomto pořadí) je úhel, o který musíme v kladném směru otočit přímku q , aby byla rovnoběžná s přímkou p . Přitom $\langle p, q \rangle = \langle q, p \rangle$, právě když $\langle p, q \rangle$ je násobek 90° . Ze souměrnosti podle os AD, BE a CF tak postupně dostáváme $\langle S_1S_3, AC \rangle = \langle AB, S_1S_2 \rangle = \langle S_2S_3, BC \rangle = \langle AC, S_1S_3 \rangle$. Protože obě odpovídající kružnice se středy S_1, S_3 mají společnou tečnu AC a leží v téže polorovině určené přímkou AC , znamená to, že S_1S_3 a AC jsou rovnoběžné.

A – III – 3

Pokud se nám podaří sestavit podle daného pravidla $(k + 3)$ -člennou posloupnost

$$x_1 = 1, x_2, x_3, \dots, x_{k-1}, x_k = x_{k+1} = x_{k+2} = x_{k+3} = 12,$$

můžeme všechny následující členy $x_{k+4}, x_{k+5}, x_{k+6}, \dots$ definovat tak, aby se rovněž rovnaly číslu 12. Skutečně, s ohledem na matematickou indukci stačí ukázat, jak s vytčeným cílem vybrat znaménka v rovnosti určující člen x_{k+4} . Položme

$$\begin{aligned} x_{k+4} = & +12(k+3) - 12(k+2) - 12(k+1) + 12k \pm \\ & \pm (k-1)x_{k-1} \pm (k-2)x_{k-2} \pm \dots \pm x_1, \end{aligned}$$

přítom znaménka v druhém řádku vybereme přesně taková, jaká byla v součtu určujícím člen $x_k = 12$. Pak se součet v druhém řádku rovná 12, takže vychází

$$x_{k+4} = 12(k+3) - 12(k+2) - 12(k+1) + 12k + 12 = 12.$$

Vhodný příklad pro $k = 8$ vypadá takto: $x_1 = x_2 = x_3 = 1$,

$$x_4 = 3 - 2 + 1 = 2,$$

$$x_5 = 4 \cdot 2 - 3 - 2 + 1 = 4,$$

$$x_6 = 5 \cdot 4 - 4 \cdot 2 - 3 - 2 - 1 = 6,$$

$$x_7 = 6 \cdot 6 - 5 \cdot 4 - 4 \cdot 2 + 3 - 2 + 1 = 10,$$

$$x_8 = 7 \cdot 10 - 6 \cdot 6 - 5 \cdot 4 - 4 \cdot 2 + 3 + 2 + 1 = 12,$$

$$x_9 = 8 \cdot 12 - 7 \cdot 10 - 6 \cdot 6 + 5 \cdot 4 + 4 \cdot 2 - 3 - 2 - 1 = 12,$$

$$x_{10} = 9 \cdot 12 - 8 \cdot 12 - 7 \cdot 10 + 6 \cdot 6 + 5 \cdot 4 + 4 \cdot 2 + 3 + 2 + 1 = 12,$$

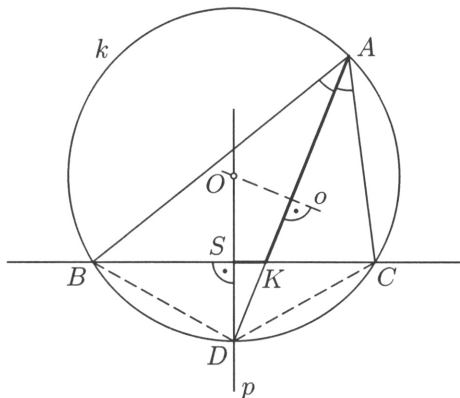
$$x_{11} = 10 \cdot 12 + 9 \cdot 12 - 8 \cdot 12 - 7 \cdot 10 - 6 \cdot 6 - 5 \cdot 4 + 4 \cdot 2 - 3 + 2 - 1 = 12.$$

Dokázali jsme, že jedna z uvažovaných posloupností má pouze prvních sedm členů různých od čísla 12.

A – III – 4

Rozbor. Předpokládejme, že trojúhelník ABC má všechny požadované vlastnosti a označme D průsečík kružnice k opsané trojúhelníku ABC

s polopřímku opačnou k rameni KA daného úhlu AKS (obr. 35). Polopřímka AD půlí úhel BAC , proto jsou úhly BAD a CAD shodné, tudíž jsou shodné i tětivy BD a CD kružnice k . Bod S je proto středem základny BC rovnoramenného trojúhelníku BCD , takže úhel BSD je pravý. To znamená, že bod D leží na přímce p , která prochází bodem S kolmo k danému rameni KS . Střed O kružnice k leží jednak na přímce p (ose tětivy BC), jednak na přímce o , která je osou tětivy AD .



Obr. 35

Konstrukce. Pro daný úhel AKS nejprve proložíme bodem S přímku p kolmo k rameni KS . Pak sestrojíme průsečík D přímky p s polopřímku opačnou k rameni KA . Dále sestrojíme osu o úsečky AD a její průsečík s přímku p označíme O . Konečně sestrojíme kružnici k se středem O a poloměrem $r = |OA|$ ($= |OD|$) a její průsečíky s přímku KS označíme B a C .

Důkaz konstrukce. Ukážeme, že sestrojený trojúhelník ABC má všechny požadované vlastnosti. Z posledního kroku konstrukce plyne, že body B, C leží na přímce KS a že bod S je středem úsečky BC . Protože na přímce p , ose úsečky BC , leží i bod D , platí $|BD| = |CD|$, a proto $|\sphericalangle BAD| = |\sphericalangle CAD|$ (neboť všechny body A, B, C, D leží na kružnici k .) Polopřímka AD je tedy osou úhlu BAC a bod K je její průsečík s úsečkou BC .

Diskuse. Vysvětlíme, proč pro daný tupý úhel AKS je hledaný trojúhelník ABC jediný (nepřihlížíme-li k možnosti zaměnit označení vrcholů B a C). Protože je úhel AKS tupý, bod D z naší konstrukce zřejmě existuje a přímky p a o jsou různoběžné, takže i bod O je určen jedno-

značně. Zbývá zdůvodnit, proč kružnice k protne přímkou KS ve dvou bodech. Protože bod K je vnitřním bodem základny AD rovnoramenného trojúhelníku ADO , platí $|OK| < |OA| = |OD| = r$, tudíž bod K leží ve vnitřní oblasti kružnice k a přímka KS je nutně její sečnou.

A – III – 5

Hledaná dvoumístná čísla A, B mají tvar $A = az + b$ a $B = bz + a$, kde a, b jsou jejich (*nenuťové!*) číslice, takže $a, b \in \{1, 2, \dots, z-1\}$. Kvadratická rovnice z textu úlohy má dvojnásobný kořen x_0 , právě když platí $2x_0 = A + x_0^2 = B$. Z těchto rovností plyne, že číslo x_0 je kladné a celé. Vzhledem k nerovnosti $x_0^2 = B < z^2$ (B je totiž dvojmístné, zatímco z^2 je trojmístné) navíc platí $x_0 < z$, odkud $A = 2x_0 < 2z$, takže číslo A má jako první číslici jedničku. Platí tedy $a = 1$ a z rovností $2x_0 = z + b$ a $x_0^2 = bz + 1$ vyloučením x_0 dostaneme pro číslici b kvadratickou rovnici $(b-z)^2 = 4$ s dvěma kořeny $b_1 = z-2$ a $b_2 = z+2$. Za číslici b lze ovšem vzít pouze první z nich, takže nutně $b = z-2$.

Dokázali jsme, že čísla A, B musejí být tvaru $A = z + (z-2) = 2z-2$ a $B = (z-2)z + 1 = (z-1)^2$; v soustavě o základě z tedy mají zápisy $A = \overline{1(z-2)}$ a $B = \overline{(z-2)1}$. Provedeme ještě *zkoušku*: kvadratická rovnice $x^2 - (2z-2)x + (z-1)^2 = 0$ má skutečně dvojnásobný kořen $x_0 = z-1$, neboť její levá strana je rovna $(x-z+1)^2$.

Poznámka. Klíčovou rovnost $a = 1$ lze odvodit i bez úvahy o dvojnásobném kořenu x_0 zkoumané rovnice, když zapíšeme podmínku, že její diskriminant $A^2 - 4B$ je roven nule:

$$0 = A^2 - 4B = (az + b)^2 - 4(bz + a) = b^2 + 2z(a-2)b + a(az^2 - 4).$$

Poslední výraz může mít nulovou hodnotu, jen když je činitel $a-2$ záporný, neboť $b \geq 1$, $a \geq 1$, $z \geq 3$ a $az^2 - 4 \geq 3^2 - 4 = 5$. Z nerovnosti $a-2 < 0$ již ovšem plyne $a = 1$. Pro takové a dostáváme rovnici $0 = b^2 - 2zb + (z^2 - 4)$ a závěr je stejný jako v uvedeném řešení.

A – III – 6

K daným kladným číslům a, b, c splňujícím podmínku $abc = 1$ zapíšeme AG-nerovnost pro trojici čísel $a/b, a/b$ a b/c :

$$\frac{1}{3} \left(\frac{a}{b} + \frac{a}{b} + \frac{b}{c} \right) \geq \sqrt[3]{\frac{a}{b} \cdot \frac{a}{b} \cdot \frac{b}{c}} = \sqrt[3]{\frac{a^2}{bc}} = \sqrt[3]{\frac{a^3}{abc}} = a.$$

Platí tedy odhad

$$\frac{2a}{3b} + \frac{b}{3c} \geq a.$$

Ze stejného důvodu platí i odhady

$$\frac{2b}{3c} + \frac{c}{3a} \geq b \quad \text{a} \quad \frac{2c}{3a} + \frac{a}{3b} \geq c.$$

Sečtením těchto tří odhadů dostaneme dokazovanou nerovnost.

Jiné řešení. Platí-li pro kladná čísla a, b, c rovnost $abc = 1$, pak $\max\{a, b, c\} \geq 1$ a $\min\{a, b, c\} \leq 1$. Protože dokazovaná nerovnost se nezmění, zaměníme-li trojici (a, b, c) trojicí (b, c, a) nebo trojicí (c, a, b) , budeme předpokládat, že čísla a a c jsou z trojice (a, b, c) nejmenší a největší (v některém pořadí), takže platí

$$(a - 1)(1 - c) \geq 0. \quad (1)$$

Do dokazované nerovnosti dosadíme $c = a^{-1}b^{-1}$ a provedeme několik ekvivalentních úprav:

$$ab^{-1} + ab^2 + a^{-2}b^{-1} \geq a + b + a^{-1}b^{-1}, \quad / \cdot a^2b$$

$$a^3 + a^3b^3 + 1 \geq a^3b + a^2b^2 + a,$$

$$a^3b^3 - a^2b^2 - a^3b + a^3 - a + 1 \geq 0,$$

$$a^3(b^3 - b^2 - b + 1) + (a^3 - a^2)b^2 - (a - 1) \geq 0,$$

$$a^3(b - 1)^2(b + 1) + (a - 1)(ab - 1)(ab + 1) \geq 0.$$

Poslední nerovnost platí, neboť $(a - 1)(ab - 1) = (a - 1)(ab - abc) = ab(a - 1)(1 - c)$ a takový součin je podle (1) nezáporný.

Jiné řešení. Pro libovolná kladná čísla A, B, C jsou trojice (A^2, B^2, C^2) a (A, B, C) takzvaně *souhlasně uspořádané*, tudíž platí nerovnost

$$A^2 \cdot A + B^2 \cdot B + C^2 \cdot C \geq A^2 \cdot B + B^2 \cdot C + C^2 \cdot A. \quad (2)$$

Dokažme (2) bezprostředně: úpravou dostáváme nerovnost

$$(A - C)^2(A + C) + (B - C)(B^2 - A^2) \geq 0,$$

která zřejmě platí, pokud $B = \max\{A, B, C\}$, čehož lze vždy dosáhnout cyklickou permutací dané trojice čísel.

Zvolíme-li v dokázané nerovnosti (2) hodnoty $A = \sqrt[3]{a/b}$, $B = \sqrt[3]{b/c}$ a $C = \sqrt[3]{c/a}$, obdržíme nerovnost

$$\frac{a}{b} + \frac{b}{c} + \frac{c}{a} \geq \sqrt[3]{\frac{a^2}{bc}} + \sqrt[3]{\frac{b^2}{ca}} + \sqrt[3]{\frac{c^2}{ab}},$$

ze které za předpokladu $abc = 1$ již plyne dokazovaná nerovnost.

Kategorie P

Texty úloh

P – I – 1

Čajovník

Pan Nyi byl dvorním pěstitelem čaje císaře Tiang-tonga. Byl to pěstitel skutečně vyhlášený a jeho čajové lístky putovaly nejen do blízkých šálek císaře Tianga, ale i do dalekých zemí za oceánem. Tajemství Nyiho skvělého čaje spočívalo především v pečlivosti, s jakou se o své čajové keře staral. Nyi byl tak pečlivý, že si o každém svém keři vedl záznamy. Psal si dokonce i to, kolik větvíček vychází z kterého místa keře. Po smrti pana Nyiho byly záznamy rozkradeny a jeho nástupce pan Myi tak měl práci o mnoho těžší. Rozhodl se proto, že záznamy získá zpět. Problémem ale je, že mnoho různých podvodníků mu nabízí záznamy falešné. Ty naštěstí většinou obsahují nesmyslné počty větvení, a tak se dají snadno odhalit. Pana Myiho neustálé ověřování pravosti záznamů už unavuje, a proto vás požádal, abyste mu napsali program, který mu s ověřováním pomůže.

Váš program dostane na vstupu počet významných míst N na údajném čajovníku. Významným místem na čajovníku je buď místo, kde se čajovník větví, nebo místo, kde končí nějaká větev čajovníku. Protože žádné dvě větve čajovníku nemohou srůst, nemohou vznikat „cykly“ z větví. Dále je na vstupu programu zadáno N kladných celých čísel $c_1, c_2, \dots, c_N$, kde c_i určuje počet částí kmene, které vycházejí z i -tého významného místa. Na výstup program vypíše zprávu, zda může existovat čajovník, který bude mít takovéto počty větvení.

Formát vstupu: Vstupní textový soubor `caj.in` obsahuje dva řádky. Na prvním řádku je uvedeno jediné celé číslo N , $1 \leq N \leq 1\,000$. Druhý řádek obsahuje celá čísla $c_1, c_2, \dots, c_N$ oddělená mezerami, $1 \leq c_i \leq N - 1$.

Formát výstupu: Výstupní textový soubor `caj.out` obsahuje jediný řádek tvořený buď slovem EXISTUJE, nebo slovem NEEXISTUJE.

Příklad 1 (obr.):

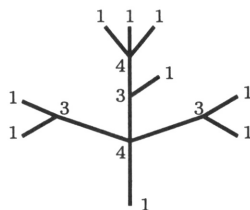
caj.in

14

1 4 3 1 1 3 1 1 3 1 4 1 1 1

caj.out

EXISTUJE



Příklad 2:

caj.in

6

3 3 3 1 1 1

caj.out

NEEXISTUJE

P – I – 2

Knihovna

Knihovnice Míla potřebuje objednat další skříň s poličkami do své knihovny; bohužel však sama neumí spočítat její optimální rozměry. Míla by ráda do nové skříně umístila N knih. Každá kniha má přiřazen jednoznačný číselný kód a tyto kódy určují pořadí knih ve skříně. Kniha s menším kódem se má nacházet na stejné nebo výše umístěné poličce než kniha s větším kódem; na každé poličce mají být knihy s menšími kódy umístěny vlevo od knih s většími kódy. Vstupem vašeho programu bude posloupnost N čísel v_i , $1 \leq i \leq N$, kde v_i je výška i -té knihy (uspořádáno podle rostoucích kódů). Pro zjednodušení můžete předpokládat, že všechny knihy mají stejnou tloušťku 1 cm. Váš program by měl ze zadaných údajů spočítat následující:

- ▷ Šířku skříně — označme ji s .
- ▷ Počet poliček ve skříně — označme ho p .
- ▷ Výšku w_i i -té poličky pro každé $1 \leq i \leq p$.
- ▷ Rozmístění knih do skříně se spočítanými parametry, které respektuje požadavky na pořadí knih zmíněné v zadání tohoto příkladu.

Navíc si knihovnice Míla přeje, aby skříň byla co nejuzší a přitom aby se vešla do místnosti vysoké 250 cm. Rozmístění knih, které váš program nalezne, musí tedy ještě splňovat následující podmínky:

- ▷ Výška libovolné z knih umístěných do i -té poličky je nejvýše w_i .

- ▷ Součet tloušťek knih umístěných do jedné poličky je nejvýše s cm, tj. tato polička obsahuje nejvýše s knih.
- ▷ Výška skříně, která je rovna $\sum_{i=1}^p w_i + (p+1) \cdot 1$ cm (předpokládáme, že šířka desek oddělujících poličky ve skříní je 1 cm), nesmí přesáhnout výšku místnosti 250 cm.
- ▷ s je nejmenší možné.

Příklad: Předpokládejme, že Míla chce do skříně umístit celkem 11 knih, jejichž výšky jsou v pořadí podle jejich kódů následující: 40 cm, 10 cm, 40 cm, 25 cm, 40 cm, 25 cm, 50 cm, 40 cm, 40 cm, 25 cm a 40 cm. Jedno z optimálních řešení by mohlo vypadat následovně: Skříň bude mít šířku pro 3 knihy a celkem 4 poličky s následujícími výškami: 40 cm, 40 cm, 50 cm a 40 cm. Výška skříně je v tomto případě 175 cm. Nalezení jednoho konkrétního možného umístění knih do skříně je snadné.

P – I – 3

Transformace

Jedna z metod zpracování textu používá následující transformační algoritmus:

Na vstupu mějme n -znakový řetězec $C = c_1c_2 \dots c_n$, jehož všechny znaky jsou navzájem různé. Řetězec $C' = c_{k+1}c_{k+2} \dots c_n c_1 \dots c_k$ nazýváme řetězcem C zrotovaným o k (tedy např. eldat je řetězec datel zrotovaný o 3). Vezměme si zadaný řetězec C a napišme si pod sebe C , C zrotovaný o 1, ..., C zrotovaný o $n - 1$. Tím jsme získali tabulku n řetězců. Ty seřídíme v běžném lexikografickém pořadí (tzn. podle abecedy). Z výsledné tabulky si vybereme poslední sloupec S ; dále si také zapamatujeme číslo řádku $\tilde{r}$, na němž se po seřídění nachází náš původní řetězec. Dvojice $(S, \tilde{r})$ je výsledek transformace zadaného vstupu. Jakkoli magicky to vypadá, tyto dva údaje stačí k rekonstrukci původního řetězce.

Příklad: Na vstupu máme slovo **datel**. Transformace probíhá takto:

datel		ateld
ateld		datel *
telda	→	eldat
eldat		ldate
ldate		telda

Výsledkem tedy je slovo **dltea** a informace, že původně zadané slovo je na druhém řádku seříděné tabulky.

Soutěžní úloha. Program dostane na vstupu řetězec S délky n ($1 \leq n \leq 100$), jehož všechny znaky jsou navzájem různé (tj. je-li $S = s_1 s_2 \dots s_n$, pak $s_i \neq s_j$ pro každá i, j , $i \neq j$), a číslo $\check{r}$ ($1 \leq \check{r} \leq n$). Úkolem je najít řetězec C takový, že dvojice $(S, \check{r})$ je výsledkem aplikace výše popsané transformace na řetězec C (máte zaručeno, že takový existuje).

Uvědomte si, že při použití v praxi se délky zpracovávaných vstupů pohybují řádově ve stovkách kilobytů; je tedy nevhodné, aby váš program měl kvadratické časové nebo paměťové nároky.

Formát vstupu: Na prvním řádku vstupního souboru `bw.in` se nachází řetězec S (řetězec neobsahuje mezery). Na druhém řádku je jedno celé číslo $\check{r}$.

Formát výstupu: Výstupní soubor `bw.out` je tvořen jedním řádkem obsahujícím řetězec C (jehož transformací je dvojice $(S, \check{r})$).

<i>Příklad:</i>	<code>bw.in</code>	<code>bw.out</code>
	<code>dltea</code>	<code>datel</code>
	<code>2</code>	

P – I – 4

Reverzibilní výpočty: Políčko pole

Při hledání nových, úspornějších polovodičových technologií se zjistilo, že nejvíce energie se spotřebovává při mazání informací, tudíž že optimální jsou ty výpočty, při nichž se žádné informace neztrácejí. Takovým výpočtům se říká *reverzibilní*, protože díky této vlastnosti mohou probíhat oběma směry — dokáží nejen spočítat ze vstupu výstup, ale také z výstupu jednoznačně určit vstup. Vydejme se proto i my do tohoto zvláštního symetrického světa a prozkoumejme, jak se programuje „ekologicky“.

Začneme tím nejjednodušším, co se v klasických programovacích jazycích vyskytuje, a to je přiřazovací příkaz. Nic takového si bohužel dovolit nemůžeme, ztratili bychom totiž původní obsah proměnné, do níž se přiřazuje. Místo toho zavedeme několik příkazů modifikujících proměnnou vratně:

- ▷ *proměnná += hodnota* — přičte hodnotu k proměnné.
- ▷ *proměnná -= hodnota* — odečte hodnotu od proměnné.
- ▷ *proměnná ^= hodnota* — přixoruje hodnotu k proměnné. (*xor* je bitová operace, která má pro jednobitová čísla výsledek 1 právě tehdy, když jsou oba vstupy různé: $0 \text{ xor } 0 = 1 \text{ xor } 1 = 0$, $0 \text{ xor } 1 = 1 \text{ xor } 0 = 1$).

Vícebitová čísla se xorují po bitech — i -tý bit prvního čísla s i -tým bitem druhého dají i -tý bit výsledku: $5 \text{ xor } 15 = (0101)_2 \text{ xor } (1111)_2 = (1010)_2 = 10$. Obecně pro libovolná čísla x a y platí $x \text{ xor } y = y \text{ xor } x$, $x \text{ xor } x = 0$, $x \text{ xor } 0 = x$ a $(x \text{ xor } y) \text{ xor } z = x \text{ xor } (y \text{ xor } z)$. Podobně lze zavést operace *and* a *or*: $0 \text{ and } 0 = 0$ and $1 = 1$ and $0 = 0$, $1 \text{ and } 1 = 1$, $0 \text{ or } 0 = 0$, $0 \text{ or } 1 = 1$ or $0 = 1$ or $1 = 1$, ale ty nejsou reverzibilní, takže pro nás nebudou tak důležité.)

▷ *proměnná* ::= *proměnná* — prohodí obsah dvou proměnných.

Abychom se vyhnuli problémům s přetečením (co by pak byla inverzní operace?), dohodněme se, že budeme počítat pouze s nezápornými celými čísly v rozsahu $0 \dots \text{maxword}$ (takovým číslům budeme říkat *přirozená*) a všechny operace budou vydávat výsledky modulo $\text{maxword} + 1$, tedy opět přirozené číslo. Příkaz += provedený pozpátku je pak totéž, co -= a opačně; příkazy ^= a ::= jsou inverzní samy k sobě.

Co všechno ale může být *hodnota*? Jistě libovolná konstanta nebo proměnná (ovšem různá od té, do které přiřazujeme, jinak bychom mohli napsat třeba `a -= a`, což určitě reverzibilní není). Také bychom měli povolit nějaké další aritmetické operace — ty samy nemusí být reverzibilní, důležité je, aby se jejich výsledek zpracoval reverzibilně. Každý složitější výraz pak už můžeme přepsat na výrazy s jedinou operací, například `x ^= (a*b)+(c*d)` rozepíšeme takto:

```
t1 += a*b;
t2 += c*d;
x ^= t1+t2;
t2 -= c*d;
t1 -= a*b;
```

Zde `t1` a `t2` jsou pomocné proměnné, které jsou na počátku výpočtu nulové a po dopočítání výrazu se opět k nulovým hodnotám vrátí, takže je můžeme používat pro všechny výrazy v celém programu. Podobně se vypořádáme s každým výrazem — nejdříve si spočítáme všechny mezivýsledky do pomocných proměnných, pak hlavní výsledek použijeme, načež mezivýsledky opět „odpočítáme“. Takže můžeme používat i složité výrazy a spolehnout se na překladač, že je sám rozepíše.

Trik s odpočítáváním mezivýsledků a spouštěním částí programu pozadu je, zdá se, velice šikovný, tak si rovnou nadefinujeme, že *undo příkaz* znamená spustit *příkaz* pozpátku a *wrap příkaz₁ on příkaz₂* provede nejdříve *příkaz₁*, pak *příkaz₂* a nakonec *undo příkaz₁* pro odpočítání mezivýsledků. Náš příklad s výrazem pak snadno zapíšeme takto:

```

wrap begin
    t1 += a*b;
    t2 += c*d
end
on x ^= t1+t2

```

Podmíněné příkazy `if-then-else` můžeme používat bez obav, pokud zaručíme, že po provedení podmíněného příkazu dopadne podmínka úplně stejně jako předtím (třeba proto, že žádná z proměnných, které v ní vystupují, není v podmíněné části programu měněna). Pak totiž i při provádění výpočtu pozpátku rozpoznáme, kterou z větví se výpočet má vydat.

S cykly je situace svízelnější, protože tam si s neměnicími se podmínkami nevystačíme (to by každý cyklus buďto neproběhl nikdy, nebo by se opakoval do nekonečna). Dalo by se to, pravda, zachránit tím, že by každý cyklus měl jednu podmínku, která by fungovala současně jako vstupní i výstupní — sami si rozmyslete, jak by takové cykly vypadaly. My si ale pro naše účely vystačíme s cykly `for`, ty určitě reverzibilní jsou, pokud řídící proměnnou cyklu ani její meze žádný příkaz uvnitř cyklu nemodifikuje, a to se koneckonců nesmí ani v mnoha jiných programovacích jazycích. Navíc abychom nemuseli řešit, co se v řídící proměnné musí vyskytovat před začátkem cyklu a co po jeho konci, domluvíme se, že příkaz `for` si tuto proměnnou sám vytvoří a na konci ji zase zruší.

Příkaz `goto` pro jistotu zakážeme úplně.

Procedury mohou také fungovat reverzibilně, ale musíme se vyhnout kopírování parametrů a výsledků, budeme proto vše vždy předávat odkazem (pascalské `var`). Lokální proměnné budou při spuštění procedury vždy nulové a procedura sama je musí, než skončí, opět do tohoto stavu vrátit. Rekurse je bez problémů.

Nyní již máme vše potřebné, abychom si vybudovali reverzibilní programovací jazyk. Ten náš bude vzdáleným příbuzným Pascalu. Vypadá takto:

Datové typy: K dispozici máme typy `word` (celá čísla bez znaménka), `bit` (jednobitové číslo, tedy 0 nebo 1; používá se rovněž pro pravdivostní hodnoty) a pole `array [x..y] of typ` (x a y udávají meze indexů a jsou to buďto čísla, nebo výrazy, jejichž hodnota se po dobu existence pole nezmění — to si proti Pascalu dovolíme navíc). Prvky polí mohou být také pole, čímž získáme pole vícerozměrná. Svůj vlastní typ si můžete zavést deklarací `type identifikátor = typ`;

Identifikátory slouží k pojmenovávání typů, proměnných a procedur

a jsou to libovolné řetězce písmen, číslic a znaků ‘_’, které nezačínají číslicí a které se neshodují s některým z klíčových slov jazyka (zde sázena *courierem*). Malá a velká písmena se nerozlišují.

```
Procedure se deklarují konstrukcí  
procedure identifikátor ( parametry );  
deklarace lokálních typů, proměnných a procedur  
begin  
příkazy oddělené středníky  
end;
```

Zde *parametry* mají syntaxi *var jméno:typ*, kde *jméno* je identifikátor, jímž se lze na předaný parametr uvnitř *procedury* odkazovat. Pokud má *procedura* parametrů více, oddělují se středníky, jsou-li stejného typu, lze zkracovat, např.: *procedure* X(*var m,n:integer*; *var Z:array* [1..n] of bit); Všechny deklarované objekty (parametry, typy, proměnné i *procedury*) existují pouze během volání této *procedury*, každá *procedura* vidí „své“ lokální proměnné a navíc lokální proměnné všech *procedur*, uvnitř kterých je deklarována (zastiňování se řídí stejnými pravidly jako v *Pascalu* nebo *C*).

Proměnné jsou pojmenovány identifikátory, musí se vytvořit deklarací *var identifikátor : typ*; . Při vstupu do *procedury*, v níž jsou deklarovány, mají nulovou hodnotu (v případě pole ji mají všechny jeho prvky), a než proměnná na konci *procedury* zanikne, musí být opět nulová. Deklaraci více proměnných téhož typu lze zkrátit, např. *var i₁, i₂, ..., i_n : typ*;

Výrazy mohou obsahovat:

- ▷ *konstanty* (přirozená čísla a *maxword* reprezentující maximální dostupné číslo),
- ▷ *proměnné*,
- ▷ *prvky polí* (*pole*[*výraz*]),
- ▷ *číselné operace* (vstupem i výstupem jsou přirozená čísla) +, -, *, div (celá část podílu), mod (zbytek po dělení), and, or, xor (bitové operace viz definice o pár odstavců výše) a not (prohození nulových a jedničkových bitů), výsledky jsou automaticky modulo *maxword*+1,
- ▷ *relační operace* (vstupem jsou dvě čísla, výstupem bitová hodnota 1, když relace platí, 0 pokud nikoliv) <, >, =, <=, >= a <> ,
- ▷ *závorky* (pokud nezávorkujeme, operátory mají své obvyklé priority).

Příkazy existují tyto:

- ▷ *Blok*: *begin příkazy* oddělené středníky *end* — způsobí vykonání všech příkazů, které obsahuje, v daném pořadí.

- ▷ *Modifikační příkazy*: *proměnná* += *výraz* — způsobí vyhodnocení výrazu a přičtení jeho výsledku k dané proměnné (může to být rovněž prvek pole indexovaný nějakým výrazem). Proměnná, kterou příkaz modifikuje, (resp. prvek pole) se již nesmí nikde jinde v témže příkazu vyskytnout. Analogicky příkazy -= a ^=.
- ▷ *Prohazovací příkaz*: *proměnná* := *proměnná* — prohodí obsah dvou proměnných stejného typu. Pokud se jedná o prvky polí, nesmí se ve výrazech určujících indexy používat žádné z těchto polí.
- ▷ *Podmíněný příkaz*: *if podmínka then příkaz₁ else příkaz₂* — vyhodnotí se *podmínka*, což je výraz s bitovým výsledkem, a pokud je roven jedné, vykoná se první z příkazů, jinak druhý. Platnost podmínky musí zůstat po vykonání příkazu nezměněna. Část *else* je možno vypustit, v případech typu *if x then if y then a else b* se pak *else* vztahuje vždy k nejbližšímu předchozímu ještě neukončenému příkazu *if*.
- ▷ *Příkaz cyklu*: *for var identifikátor = d to h do příkaz* — založí novou proměnnou daného jména a daný příkaz vykonává pro tuto proměnnou nabývající postupně hodnot $d, d+1, \dots, h$, načež proměnnou opět zruší. Meze d a h jsou celočíselné výrazy, pokud $d > h$, příkaz se neprovede ani jednou. Příkaz musí zachovávat hodnotu řídicí proměnné, jakož i mezí cyklu (to znamená, že je může modifikovat, ale na konci jednoho průchodu cyklem musí mít obojí opět původní hodnotu). Též je možno použít *h downto d*, tehdy cyklus běží pozpátku, tj. $h, h-1, \dots, d$.
- ▷ *Volání procedury*: *procedura*(*parametr₁*, ..., *parametr_n*) — zavolá proceduru se zadanými parametry, což mohou být buďto proměnné, nebo indexovaná pole (výrazy v indexech ovšem musejí mít po návratu z procedury stejnou hodnotu jako před jejím zavoláním) a jejich počet i typy musejí odpovídat deklaraci procedury.
- ▷ *Příkaz obrácení výpočtu*: *undo příkaz* — provede daný příkaz pozpátku podle následujících pravidel:

<i>undo begin p₁ ; ... ; p_n end</i>	→ <i>begin undo p_n ; ... ; undo p₁ end</i>
<i>undo x += y</i>	→ <i>x -= y</i>
<i>undo x -= y</i>	→ <i>x += y</i>
<i>undo x ^= y</i>	→ <i>x ^= y</i>
<i>undo x := y</i>	→ <i>x := y</i>
<i>undo if x then y else z</i>	→ <i>if x then undo y else undo z</i>
<i>undo for x = d to h do p</i>	→ <i>for x = h downto d do undo p</i>
<i>undo P(x₁, ..., x_n)</i>	→ <i>undo těla procedury (begin ... end)</i>
<i>undo undo p</i>	→ <i>p</i>

Konstrukce *begin p ; undo p end* tedy nevykoná nic, ač může počítat poměrně dlouho.

▷ *Příkaz lokálního výpočtu:* `wrap příkaz1 on příkaz2` je zkratkou za konstrukci `begin příkaz1 ; příkaz2 ; undo příkaz1 end.`

Hlavní program nebudeme zavádět. Abychom se vyhnuli problémům se vstupy a výstupy, budeme vše vždy programovat jako procedury. Ty jako své parametry dostanou jak proměnné, které obsahují vstupní data, tak proměnné, které mají být předepsaným způsobem zmodifikovány podle výsledku.

Časová a prostorová složitost se definuje podobně jako v klasickém programování: časovou složitostí výpočtu je počet vykonaných příkazů modifikujících proměnné, ať již proběhly kterýmkoliv směrem. Množství paměti využité programem v nějakém okamžiku výpočtu spočítáme jako součet velikostí všech lokálních proměnných (typy `bit` a `word` mají jednotkovou velikost, pole má velikost rovnou součtu velikostí svých prvků) a parametrů (ty se všechny počítají jako jednotka, ať už jsou kteréhokoliv typu, protože jsou předávány odkazem) všech právě zavolaných procedur + jednotka navíc za každou takovou proceduru. Prostorovou složitostí programu nazveme pak maximum z využitého množství paměti přes celou dobu běhu programu. (Pozor, jelikož program je pro nás vždy procedurou, jeho vstupy a výstupy se do prostorové složitosti započítávají pouze jednotkově, i když to mohou být velká pole.)

Zbývá maličkost: cokoliv uzavřeného do složených závorek `{` a `}` je *komentářem*, který je počítačem zcela ignorován, jako kdyby na jeho místě byla mezera. Komentář nesmí uvnitř obsahovat složené závorky.

Příklad 1: Procedura pro prohození obsahu dvou proměnných (která ukazuje, že `:=` se dá snadno odvodit pomocí ostatních operací). Časová i prostorová složitost jsou konstantní, tedy $O(1)$.

```
procedure Prohod(var x,y:word);
begin
    { x = X, y = Y (X,Y jsou pův. hodnoty) }
    x ^= y;   { x = X xor Y, y = Y }
    y ^= x;   { x = X xor Y, y = Y xor (X xor Y) = X }
    x ^= y    { x = (X xor Y) xor X = Y, y = X }
end;
```

Příklad 2: Procedura pro výpočet maxima ze zadaných n čísel. Je dáno pole X celých čísel a proměnná max , k níž máme spočtené maximum přičíst. To dokážeme takto: Nejprve si předpočítáme do $M[i]$ maximum z čísel $X[1], \dots, X[i]$, pak přičteme $M[n]$ k max a nakonec $M[i]$ opět vyprázdníme, což snadno zapíšeme pomocí příkazu `wrap`. Časová i prostorová složitost jsou $O(n)$ čili lineární.

```

procedure Maximum(var n:word; var X:array [1..n] of word;
    var max:word);
var M:array [0..n] of word;
begin
    wrap
    for var i=1 to n do
        if X[i]>M[i-1] then
            M[i] += X[i]
        else
            M[i] += M[i-1]
    on max += M[n];
end;

```

Soutěžní úloha. Napište *reverzibilní* proceduru Najdi(var n:word; var X:array [1..n] of word; var co, kde:word). Tato procedura má za úkol v n -prvkovém poli X hledat hodnotu co , a pokud se tam tato hodnota vyskytuje, přičíst k proměnné kde pozici jejího výskytu, tedy i takové, že $X_i = co$. Navíc je známo, že pole X je uspořádáno vzestupně, tedy že pro každé $i < j$ platí $X_i < X_j$; proto také může být výskyt nejvýše jeden. Ve svém řešení se snažte dosáhnout co nejmenší časové i prostorové složitosti.

P – II – 1

Tajemný obraz

V Chile objevili archeologové tajemný obraz z předkolumbovské doby. Obraz vypadá jako několik bodů rozmístěných na pomyslné kružnici, přičemž každé dva body jsou spojeny rovnou čarou. Každá z čar je buď žlutá nebo červená. Archeologové dlouho hledali smysl této malby, ale žádný nemohli nalézt. Až amatérský archeolog a dobrodruh Erik von Kätzinken po dlouhém pátrání vyslovil hypotézu, že obrazec je poselstvím dávných Inků. Počet jednobarevných trojúhelníků s vrcholy na pomyslné kružnici prý udává počet dní od namalování obrazce, po nichž mají na Zemi opět přistát mimozemšťané. Protože obrazec je poměrně velký, rozhodl se Erik určit počet těchto jednobarevných trojúhelníků pomocí počítače.

Vaším úkolem je napsat program, který dostane na vstupu počet bodů nakreslených na obrazci N , $3 \leq N$, a dále seznam dvojic čísel bodů (body si očíslováme od jedné do N), které jsou propojeny červenou čarou (zbylé dvojice bodů jsou tedy propojeny žlutou). Na výstup program vypíše počet jednobarevných trojúhelníků v zadaném obrazci (tzn. takových

trojúhelníků, jejichž všechny tři vrcholy leží v bodech vyznačených na kružnici a jsou spojeny čarami téže barvy).

Příklad. V obrazci s pěti body a červenými čarami mezi body (1, 2), (2, 3), (3, 4) a (2, 4) jsou tři jednobarevné trojúhelníky (jeden červený a dva žluté). Červený trojúhelník má vrcholy v bodech 2, 3, 4 a žluté trojúhelníky v bodech 1, 3, 5 a 1, 4, 5.

P – II – 2

Knihovna

Knihovnice Míla opět potřebuje objednat další skřín do své knihovny. Bohužel však zase sama neumí spočítat, jak by tato skřín měla být široká, a tak vás znovu poprosila o pomoc. Míla by ráda do nové skříně umístila celkem N knih, ale na rozdíl od úlohy P–I–2 z prvního kola je jí jedno, v jakém pořadí knihy do skříně umístí. Vstupem vašeho programu bude posloupnost N čísel v_i , $1 \leq i \leq N$, kde v_i je výška i -té knihy. Pro zjednodušení předpokládejme, že všechny knihy mají stejnou tloušťku — 1 cm.

Váš program by měl ze zadaných údajů spočítat následující:

- ▷ Šířku skříně — označme ji s .
- ▷ Počet poliček ve skříně — označme ho p .
- ▷ Výšku w_i i -té poličky pro každé $1 \leq i \leq p$.
- ▷ Rozmístění knih do skříně se spočítanými parametry.

Rozmístění knih, které váš program nalezne, musí z pochopitelných důvodů splňovat následující:

- ▷ Výška libovolné z knih umístěných do i -té poličky je nejvýše w_i .
- ▷ Součet tloušťek knih umístěných do jedné poličky je nejvýše s cm, tj. tato polička obsahuje nejvýše s knih.
- ▷ Výška skříně, která je rovna $\sum_{i=1}^p w_i + (p+1) \cdot 1$ cm (předpokládáme, že šířka desek oddělujících poličky ve skříně je 1 cm), nesmí přesáhnout výšku místnosti 250 cm.
- ▷ s je nejmenší možné.

Příklad. Předpokládejme, že Míla chce do skříně umístit celkem 14 knih, z nichž devět má výšku 50 cm a pět má výšku 40 cm. Jedno z optimálních řešení by mohlo vypadat následovně: Skřín bude mít šířku pro 3 knihy a celkem 5 poliček — tři z nich budou mít výšku 50 cm a dvě poličky výšku 40 cm. Nalézt jedno konkrétní možné rozmístění knih do skříně je snadné.

Transformace

Jedna z metod zpracování textu používá následující transformační algoritmus: Na vstupu mějme n -znakový řetězec $C = c_1c_2 \dots c_n$. Řetězec $C' = c_{k+1}c_{k+2} \dots c_n c_1 \dots c_k$ nazýváme řetězcem C zrotovaným o k (tedy např. **akaabr** je řetězec **abraka** zrotovaný o 3). Vezměme si zadaný řetězec C a napišme si pod sebe C , C zrotovaný o 1, ..., C zrotovaný o $n - 1$. Tím jsme získali tabulku n řetězců. Ty setřídíme v běžném lexikografickém pořadí (tzn. podle abecedy). Z výsledné tabulky si vybereme poslední sloupec S ; dále si také zapamatujeme číslo řádku $\check{r}$, na němž se po setřídění nachází náš původní řetězec (je-li těchto řádků více, libovolný z nich). Dvojice $(S, \check{r})$ je výsledek transformace zadaného vstupu. Jakkoli magicky to vypadá, tyto dva údaje stačí k rekonstrukci původního řetězce.

Příklad. Na vstupu máme slovo **abraka**. Transformace probíhá takto:

abraka		aabrak
brakaa		abraka *
rakaab	→	akaabr
akaabr		brakaa
kaabra		kaabra
aabrak		rakaab

Výsledkem tedy je slovo **karaab** a informace, že původně zadané slovo je na druhém řádku setříděné tabulky.

Soutěžní úloha. Program dostane na vstupu řetězec S délky n ($1 \leq n \leq 10\,000$) a číslo $\check{r}$ ($1 \leq \check{r} \leq n$). Úkolem je najít řetězec C takový, že dvojice $(S, \check{r})$ je výsledkem aplikace výše popsané transformace na řetězec C (máte zaručeno, že takový existuje).

Poznámka. Budete-li psát program v Pascalu, můžete předpokládat, že do stringu se řetězec této délky vejde.

Uvědomte si, že při použití v praxi se délky zpracovávaných vstupů pohybují řádově ve stovkách kilobytů; je tedy nevhodné, aby váš program měl kvadratické časové nebo paměťové nároky.

Formát vstupu: Na prvním řádku vstupního souboru **bw.in** se nachází řetězec S (řetězec neobsahuje mezery). Na druhém řádku je jedno celé číslo $\check{r}$.

Formát výstupu: Výstupní soubor **bw.out** je tvořen jedním řádkem, obsahujícím řetězec C (jehož transformací je dvojice $(S, \check{r})$).

P – II – 4

Reverzibilní výpočty: Ouřad

(Definice reverzibilních výpočtů je obsažena v textu úlohy P–I–4.)

Oblastní Ouřad sídlí v městě $\mathcal{M}$ v n budovách. Aby si ouředníci sídlící na různých místech mohli rychle posílat všechny ty dopisy, příписy, zápisy, dobropisy, vrubopisy, tiskopisy a vůbec všelijaké spisy s podpisy, vybudovali si potrubní poštu — systém rour spojujících některé dvojice budov. Těmito rourami se pomocí stlačeného vzduchu posílají zásilky. Aby nedocházelo ke kolizím, je každá roura využívána jenom jedním směrem. Nejsou si ale jisti, jestli již postavili všechna potřebná potrubí, a tak je zajímavá, jak zjistit, zda mezi nějakými dvěma zadanými místy je možné dopravit zásilku, a to buďto přímo, nebo s přeložením zásilky v nějaké mezistanici, případně mezistanicích.

Napište *reverzibilní* proceduru `Zkoumej` (`var n:word; var A:array [1..n] of array [1..n] of bit; var x,y,d:word`), která dostane jako vstup počet budov n , matici A popisující v prvku $A[i][j]$, zda vede (1) či nevede (0) roura z i -té do j -té budovy, a čísla budov x a y . Poté do proměnné d přičte, s jakým nejmenším počtem přeložení je možno přepravit zásilku z budovy x do budovy y , případně přičte číslo větší než n , pokud to možné není. Snažte se dosáhnout co nejmenší *prostorové* složitosti výpočtu při zachování polynomiální časové složitosti.

P – III – 1

Hračkářství

V hračkářství Prcek a otec proběhla velká soutěž „O nejhezčí hračku“. Děti měly za úkol nakreslit obrázek své nejoblíbenější hračky. Po ukončení soutěže byla uspořádána výstavka a děti, které nakreslily nejpečnější obrázky, dostaly od hračkářství nějakou hračku. Jak ale asi víte, ne každému dítěti se líbí každá hračka, a tak už před vyhlášením soutěže měl každý malý výtvarník vyhlédnutou tu odměnu, kterou chtěl za svůj obrázek dostat. Tu a žádnou jinou. Svůj názor pak děti po vyhlášení dávaly dost hlasitě najevo. Maminky ječících potomků se tedy rozhodly, že děti si hračky mezi sebou povyměňují tak, aby pokud možno co nejvíce dětí

bylo se svou výhrou spokojeno. Situaci ještě navíc komplikuje skutečnost, že k výměně jsou ochotné pouze ty děti, které nakonec dostanou hračku, po níž touží. S tak náročným úkolem si maminky nevěděly rady, a tak poprosily vás, abyste napsali program, který problém vyřeší.

Váš program dostane na vstupu zadán počet N odměněných dětí a dále pro každé dítě číslo hračky, kterou dostalo, a číslo hračky, kterou by chtělo dostat (protože hraček je stejně jako dětí, očíslováme si je pro jednoduchost od jedné do N). Na výstup program vypíše největší skupinu dětí takovou, že když si děti ve skupině mezi sebou vhodným způsobem vymění hračky, budou všechny spokojené.

P – III – 2

Knihovna

Knihovnice Míla opět potřebuje objednat další skříň do své knihovny, a protože se jí vaše pomoc osvědčila, opět se na vás obrátila, abyste jí pomohli spočítat optimální rozměry nové skříně. Nová skříň má mít P poliček a Míla by do ní ráda umístila celkem N knih. Každá kniha má přiřazen jednoznačný číselný kód a tyto kódy určují pořadí knih ve skříni. Kniha s menším kódem se má nacházet na stejné nebo výše umístěné policiče než kniha s větším kódem; na každé policiče mají být knihy s menšími kódy umístěny vlevo od knih s většími kódy. Vstupem vašeho programu bude celé číslo P a posloupnost N čísel t_i , $1 \leq i \leq N$, kde t_i je tloušťka i -té knihy. Můžete předpokládat, že tloušťka t_i i -té knihy je celé číslo z rozmezí od 1 do 50. Výška každé knihy je taková, že ji lze bez problémů umístit do libovolné z plánovaných P poliček. Váš program by měl ze zadaných údajů spočítat následující:

- ▷ Šířku skříně — označme ji s .
- ▷ Rozmístění knih do skříně se spočítanými parametry.

Rozmístění knih, které váš program nalezne, musí z pochopitelných důvodů splňovat následující:

- ▷ Součet tloušťek knih umístěných do jedné poličky je nejvýše s .
- ▷ Šířka skříně s je nejmenší možná.

Příklad. Předpokládejme, že nová skříň má mít 3 poličky a má být do ní umístěno celkem 6 knih s následujícími tloušťkami (seřazeny vzestupně podle svých kódů): 15, 20, 7, 6, 2 a 4. Minimální možná šířka skříně v tomto případě je 20 — na první poličku se dá pouze první kniha, na druhou pouze druhá kniha a zbylé knihy se umístí na poslední třetí poličku.

P – III – 3

Reverzibilní výpočty: Sčítání

(Definice reverzibilních výpočtů je obsažena v textu úlohy P–I–4.)

Napište *reverzibilní* proceduru `Add(var n:word; var A,B:array [0..n-1] of bit)` sloužící ke sčítání dvou n -bitových čísel zapsaných ve dvojkové soustavě (bit číslo 0 odpovídá řádu jednotek). Tato procedura přičte číslo uložené v poli B k číslu uloženému v poli A . Vstup dostane vždy takový, aby nedošlo k přetečení, tedy součet bude vždy také n -bitový. Snažte se dosáhnout co nejmenší *prostorové* složitosti své procedury.

P – III – 4

Poklad kapitána Flinta

Program: poklad.pas / poklad.c / poklad.cpp

Vstup: poklad.in

Výstup: poklad.out

Kapitán Flint si při svých pirátských výpravách přišel k docela pěkné hromádce zlatáků. Pirátské výpravy jsou však dosti nejisté a štěstěna vrtkavá, a proto kapitán zakopal část svého jmění na pustém ostrově a cestu k pokladu zakreslil na ovčí kůži ve tvaru konvexního N -úhelníku. Celou mapu pak rozřezal na mnoho částí, přičemž každý řez vedl přímo mezi dvěma vrcholy mnohoúhelníku a žádné dva řezy se neprotínaly. Aby si pojistil věrnost posádky svého škuneru, rozhodl se Flint některé části darovat nejzdatnějším pirátům. Protože by se ale námořníci mohli snadno dohodnout, mapu sestavit a poklad vykopat, chce mezi ně kapitán rozdělit části mapy tak, aby žádní dva námořníci neměli sousední díly (tedy takové, které mají společnou hranu). Přitom chce mezi námořníky rozdělit co nejvíce částí mapy. Dokázali byste napsat program, který pomůže kapitánovi vyřešit jeho problém?

Vstup: Na prvním řádku vstupního souboru `poklad.in` dostane program dvě celá čísla N a M oddělená mezerou, $3 \leq N \leq 30\,000$, $0 \leq M \leq 30\,000$ — počet vrcholů mapy a počet řezů. Následuje M řádků popisujících jednotlivé řezy. Každý z těchto řádků obsahuje dvě čísla A a B oddělená mezerou — čísla vrcholů, mezi kterými vede řez (vrcholy číslujeme od jedné do N).

Výstup: Výstupní textový soubor `poklad.out` bude obsahovat jediné číslo udávající maximální počet částí mapy, které lze mezi námořníky rozdělit tak, aby žádní dva námořníci neměli sousední díly mapy.

Příklad: Vstupní soubor

`poklad.in:`

5 2

1 3

3 5

Výstupní soubor

`poklad.out:`

2

P – III – 5

Vážení

Program: `vahy.pas` / `vahy.c` / `vahy.cpp`

Vstup: `vahy.in`

Výstup: `vahy.out`

Mudrc Tlučhuba se přihlásil do konkurzu na královského rádce v jednom nejmenovaném království. Vzápětí však byl zaskočen podmínkami tohoto konkurzu: Jako test svých schopností obdrží N mincí, z nichž některé mají různou a některé stejnou hmotnost. Jeho úkolem bude tyto mince rozdělit do skupin tvořených mincemi stejné hmotnosti a tyto skupiny pak seřadit vzestupně podle hmotností mincí tak, aby v první skupině byly nejlehčí mince a v poslední skupině byly nejtěžší mince. Bude mít k dispozici dvouramenné váhy, na jejichž misky smí v jednom okamžiku položit po jedné minci.

Mudrc Tlučhuba vás požádal o pomoc při plnění tohoto úkolu. Chtěl by, abyste vytvořili program, jenž mu pomůže při rozhodování, které mince zvážit, jak mince rozdělit do skupin a jak vytvořené skupiny uspořádat. Pro účely programu si mince očíslováme od 1 do N . Samotné vážení bude ve vašem programu zastoupeno funkcí `porovnej`.

Mudrc Tlučhuba musí úkol splnit v časovém limitu, který mu byl stanoven (tedy v něm musí úkol splnit i váš program). Kromě toho musí provést všechna *nezbytná* vážení, tj. nesmí existovat dvě či více možných řešení konzistentních s odpověďmi funkce `porovnej`, jinak by byl mudrc upálen jako čarodějník (jak jinak by mohl vědět, které uspořádání je správné?). Na druhou stranu nesmí být provedeno žádné *zbytečné* vážení, tj. takové, jehož výsledek by již (přímo či nepřímo) vyplýval z předchozích odpovědí funkce `porovnej`.

Popis funkce porovnej: Funkce **porovnej** je definována v knihovně **vahy_lib**. Váš program musí obsahovat následující řádek, aby mohl používat funkci **porovnej**:

Pascal: **uses vahy_lib;**

C/C++: **#include "vahy_lib.h"**

Funkce **porovnej** je deklarována takto:

Pascal: **function porovnej(a,b: longint): integer;**

C/C++: **int porovnej(int, int);**

Tato funkce očekává jako vstupní parametry čísla dvou mincí. Vráť hodnotu -1 , pokud mince odpovídající prvnímu parametru je lehčí než mince odpovídající druhému parametru, $+1$, pokud je tomu naopak, a 0 , pokud obě mince mají stejnou hmotnost.

Nezapomeňte, že váš program *nesmí* funkci **porovnej** volat zbytečně, tj. výsledek žádného volání funkce **porovnej** nesmí vyplývat (tedy být jednoznačně určen) z předchozích volání této funkce. Např. pokud jsme voláním funkce **porovnej** zjistili, že mince s číslem 1 je lehčí než mince s číslem 2 a že mince s číslem 2 je lehčí než mince s číslem 3, nelze již funkci **porovnej** zavolat s parametry 1 a 3. Kromě toho váš program může funkci **porovnej** zavolat nejvýše 250 000krát.

Vstup: Vstupní soubor **vahy.in** obsahuje jediný řádek s jediným číslem N , $1 \leq N \leq 10\,000$, které udává počet mincí.

Výstup: Výstupní soubor **vahy.out** musí obsahovat K řádků, kde K je počet různých hmotností mincí. Na každém řádku budou uvedena čísla mincí téže hmotnosti v rostoucím pořadí. Hmotnosti mincí jednotlivých řádků tvoří rovněž rostoucí posloupnost, tzn. první řádek obsahuje všechny nejlehčí mince a poslední řádek všechny nejtěžší mince.

Příklad: Vstupní soubor

vahy.in:

4

Průběh komunikace:

volání **porovnej**(2,4) vrací -1

volání **porovnej**(1,2) vrací 1

volání **porovnej**(3,4) vrací -1

volání **porovnej**(1,3) vrací 0

Výstupní soubor **vahy.out**:

2

1 3

4

Pro řešení úlohy si vypůjčíme terminologii z teorie grafů. Významná místa na čajovníku budeme nazývat *vrcholy*, části kmene čajovníku mezi dvěma významnými místy pak *hrany*. Vrcholy spolu s hranami si pojmenujeme *strom* (pokud bychom chtěli být opravdu přesní, měli bychom jej nazvat *grafem*. My ale víme, že náš čajovník nemá žádný cyklus a graf s touto vlastností se nazývá strom). Počet hran, které vedou z nějakého vrcholu v (tedy vlastně počet částí kmene, které vedou z významného místa v), nazveme *stupněm* vrcholu v .

Nejdříve si všimneme, že každý strom s alespoň dvěma vrcholy má alespoň jeden vrchol stupně jedna (takovýto vrchol se nazývá *list*). Tento vrchol můžeme snadno nalézt následovně. Začneme strom prohledávat v libovolném vrcholu. Pokud ještě nejsme v listu, přejdeme do libovolného sousedního (tzn. připojeného hranou) vrcholu, ve kterém jsme dosud nebyli. Jelikož ve stromu nejsou cykly, musí takovýto vrchol vždy existovat. Protože vrcholů je konečný počet, musíme jednou skončit — a to můžeme pouze v listu.

Nyní ukážeme, že součet stupňů všech vrcholů v libovolném stromu je $2N - 2$ (kde N je počet vrcholů). Naopak platí, že máme-li N kladných celých čísel se součtem $2N - 2$, pak existuje strom s N vrcholy majícími tyto stupně. Z toho už je jasné, že stačí zjistit, zda je součet čísel na vstupu roven $2N - 2$, a podle výsledku vypsát patřičnou zprávu.

První tvrzení dokážeme indukcí podle počtu vrcholů. Strom o dvou vrcholech obsahuje jedinou hranu. Součet stupňů vrcholů je tedy $1 + 1 = 2$ a naše tvrzení platí. Pokud má strom více vrcholů, víme z předchozího pozorování, že má list. Když tento list odebereme (tedy zrušíme vrchol a hranu, která ho připojuje ke zbytku stromu), získáme zřejmě opět strom. Pro něj z indukčního předpokladu platí, že součet stupňů je $2 \cdot (N - 1) - 2 = 2N - 4$. Protože v původním stromu měl jeden vrchol stupeň o jedna vyšší (ten, ke kterému byl připojen list) a byl v něm navíc list, je součet stupňů v původním stromu $2N - 4 + 2 = 2N - 2$. Tím je první tvrzení dokázáno.

Druhé tvrzení dokážeme indukcí dle počtu členů posloupnosti: Necht' máme posloupnost dvou kladných celých čísel, jejichž součet je $2 \cdot 2 - 2 = 2$. Tato čísla tedy mohou být pouze dvě jedničky. Pro ně jsou zřejmě odpovídajícím stromem dva vrcholy spojené hranou. Pokud má posloupnost

více než dvě čísla, musí zřejmě obsahovat alespoň jednu jedničku (jinak by součet N čísel byl alespoň $2N$ a ne $2N-2$). Analogicky musí také obsahovat alespoň jedno číslo větší než jedna. Když z posloupnosti vypustíme jednu jedničku a jedno z čísel větších než jedna snížíme o jedna, získáme posloupnost čísel o jedna kratší se součtem $2N - 2 - 2 = 2 \cdot (N - 1) - 2$. Dle indukčního předpokladu tedy existuje strom na $N - 1$ vrcholech s příslušnými stupni vrcholů. Když do stromu přidáme jeden list a připojíme ho hranou k vrcholu, který odpovídá číslu, jež jsme zmenšovali o jedničku, získáme přesně strom pro naši původní posloupnost. Tím je dokázáno i druhé tvrzení.

Časová složitost algoritmu je $O(N)$, paměťová $O(1)$.

```

program Caj;
var
  N, i, Suma, Casti : Integer;
  vstup, vystup : Text;
begin
  Assign(vstup, 'caj.in');
  Assign(vystup, 'caj.out');
  Reset(vstup);
  Rewrite(vystup);
  Suma := 0;
  ReadLn(vstup, N);
  for i := 1 to N do begin
    Read(vstup, Casti);
    Suma := Suma + Casti;
  end;
  if Suma = 2*(N-1) then
    WriteLn(vystup, 'EXISTUJE')
  else
    WriteLn(vystup, 'NEEXISTUJE');
  Close(vstup);
  Close(vystup);
end.

```

P – I – 2

Předvedeme si dvě možná řešení této úlohy. Obě jsou založena na metodě zvané dynamické programování: Úloha se nejprve vyřeší pro podúlohu velikosti 1. Tohoto řešení se použije pro nalezení řešení podúlohy velikosti 2. Takto nalezených řešení se použije pro vyřešení podúlohy velikosti 3 atd. V našem případě bude velikost podúlohy určená počtem knih, které chceme do skříně umístit.

První řešení je založeno na vytvoření dvojrozměrného pole A o rozměrech $N \times V$, kde N je celkový počet knih, které máme do skříně umístit, a V je maximální výška skříně; V je v našem případě rovno 250 podle

zadání úlohy. Hodnota $A[i, j]$, $0 \leq i \leq N$, $1 \leq j \leq V$, udává minimální možnou šířku skříně výšky j , do které lze umístit prvních i knih. Pokud do skříně výšky j prvních i knih nelze umístit, tj. některá z těchto knih je vyšší než $j - 2$ cm, pak je hodnota $A[i, j]$ rovna nějaké speciální hodnotě, např. -1 . Popíšeme si, jak lze v čase $O(N)$ spočítat hodnotu $A[i_0, j_0]$, máme-li již spočítány hodnoty $A[i, j]$ pro $i < i_0$. Pokud je $i_0 = 0$, pak zřejmě $A[i_0, j_0] = 0$ cm. Pokud existuje i , $1 \leq i \leq i_0$, takové, že výška v_i i -té knihy je větší než $j_0 - 2$ cm, pak prvních i knih nelze do skříně výšky j_0 umístit a $A[i_0, j_0]$ bude rovno -1 . Ve zbylých případech určíme hodnotu $A[i_0, j_0]$ následovně: Pro $0 \leq i < i_0$ zkusíme umístit na poslední policičku skříně $(i + 1)$ -ní až i_0 -tou knihu a prvních i knih dáme na předcházející policičky; výška poslední policičky by tedy musela být alespoň $v = \max_{i+1 \leq k \leq i_0} v_k$ a můžeme předpokládat, že je právě v . Šířka této policičky musí být alespoň $i_0 - i$. Pokud $A[i, j_0 - v - 1]$ je rovno -1 , pak nelze vytvořit skříň výšky j_0 , která by obsahovala prvních i_0 knih a na poslední policiče by z nich měla posledních $i_0 - i$. V opačném případě je nejmenší šířka skříně výšky j_0 , která obsahuje prvních i_0 knih a na poslední policiče má z nich umístěno posledních $i_0 - i$, rovna $\max\{A[i, j_0 - v - 1], i_0 - i\}$. Nejmenší z těchto výrazů pro $0 \leq i < i_0$ bude roven hledané hodnotě $A[i_0, j_0]$. Výše popsany výpočet lze provést v čase $O(N)$, budeme-li postupovat od $i = i_0 - 1$ k $i = 0$; v takovém případě lze $v = \max_{i+1 \leq k \leq i_0} v_k$ spočítat z v pro hodnotu i o 1 větší v konstantním čase. Hodnota pole $A[N, 250]$ je hledanou minimální možnou šířkou skříně. Pokud chceme zároveň nalézt i rozmístění knih do skříně a výšky jednotlivých policiček, zavedeme si ještě pomocné pole $B[i, j]$, $0 \leq i \leq N$, $1 \leq j \leq V$, do jehož položky $B[i_0, j_0]$ si při výpočtu hodnoty $A[i_0, j_0]$ uložíme to i , pro které je šířka skříně minimální při výšce j_0 . Z hodnoty $B[N, 250]$ určíme počet knih, které jsou v optimálním řešení na poslední policiče; tato hodnota nám umožní spočítat výšku skříně bez poslední policičky a počet knih v těchto policičkách. Z příslušné hodnoty v poli B určíme počet knih na předposlední policiče a takto postupujeme, dokud nedosáhneme první policičky. Vzhledem k velikosti pole A jsme si právě popsali algoritmus, jehož časová složitost je $O(VN^2)$ a paměťová složitost $O(VN)$.

Nyní si popíšeme druhé možné řešení. Nejprve si ukážeme, jak lze v čase $O(N^2)$ rozhodnout, zda lze knihy umístit do skříně šířky s a výšky V . K tomu si vytvoříme pomocné pole $A[i]$, $0 \leq i \leq N$, které udává minimální výšku skříně šířky s , do které lze umístit prvních i knih. Pokud $A[N] > V$, pak knihy nelze umístit do skříně šířky s a výšky V ; v opačném případě je lze do skříně s těmito rozměry umís-

tit. K určení hodnot v poli A opět použijeme dynamické programování. Hodnota $A[0]$ je 1 cm, což je speciální případ obecného vztahu „součet výšek políček + (počet políček + 1) $\times$ 1 cm“ pro výšku skříně. Popíšeme, jak lze určit hodnotu $A[i_0]$, pokud známe hodnoty $A[i]$ pro $0 \leq i < i_0$. Zvolme $i_0 - s \leq i < i_0$; na poslední políčku chceme umístit v takovémto případě posledních $i_0 - i$ knih (proto podmínka $i_0 - s \leq i$). Výška skříně je pak rovna $A[i] + 1 + \max_{i < k \leq i_0} v_k$; nejmenší z těchto výrazů pro i , $i_0 - s \leq i < i_0$, je hledaná hodnota $A[i_0]$. Hodnotu $A[i_0]$ lze spočítat v čase $O(N)$, pokud budeme postupovat od $i = i_0 - 1$ k $i = i_0 - s$ (potom lze $\max_{i < k \leq i_0} v_k$ spočítat z hodnoty pro $i + 1$ v konstantním čase). Popsaná procedura v čase $O(N^2)$ s pamětí $O(N)$ rozhodne, zda lze zadaných N knih umístit do skříně šířky s a výšky V . Zbývá popsat, jak lze tuto proceduru použít pro vyřešení původní úlohy. Nejprve zkontrolujeme, že výška všech knih je nejvýše $V - 2 \text{ cm} = 248 \text{ cm}$, a tedy že knihy lze vůbec umístit do nějaké skříně výšky V . K určení minimální šířky s_0 skříně použijeme metodu zvanou půlení intervalu. Budeme si udržovat dvě proměnné $s_1 \leq s_2$, které nám budou ohraničovat možný interval, ve kterém je hledaná šířka s_0 , tj. $s_1 \leq s_0 \leq s_2$. Nejprve položíme $s_1 = 1$ a $s_2 = N$. V každém kroku zvolíme $s = \lfloor \frac{1}{2}(s_1 + s_2) \rfloor$ a pomocí výše popsané procedury zkontrolujeme, zda lze našich N knih umístit do skříně výšky V a šířky s . Pokud knihy lze do takové skříně umístit, položíme $s_2 = s$; v opačném případě položíme $s_1 = s + 1$. Celý postup opakujeme, dokud se hodnoty s_1 a s_2 liší, tj. dokud nenalezneme hledanou hodnotu s_0 . Všimněte si, že v každém kroku se rozdíl $s_2 - s_1$ zmenší alespoň o 1 (kdybychom při volbě s použili horní celou část místo dolní celé části, nebylo by toto tvrzení pravdivé) a tento rozdíl se zmenší zhruba na polovinu. Tedy po $O(\log N)$ krocích nalezneme hledanou optimální šířku skříně s_0 . Výšky políček a rozmístění knih lze nalézt podobně jako v předcházejícím algoritmu zavedením pomocného pole B , do kterého si budeme ukládat počet knih na poslední políčke v optimálním řešení. Celková časová složitost právě popsaného algoritmu je tedy $O(N^2 \log N)$ a paměťová složitost je $O(N)$.

Zbývá vyřešit otázku, který ze dvou popsaných algoritmů je lepší. Odpověď je, že ani jeden není lepší. Vzhledem k zadání úlohy, kde V je omezeno, je časová složitost prvního algoritmu sice $O(N^2)$ a paměťová pouze $O(N)$, ale multiplikativní konstanta skrytá ve „velkém O “ je lineární s V ; na druhou stranu paměťová složitost druhého algoritmu je pouze $O(N)$, kde multiplikativní konstanta je nezávislá na výšce. Dle výše popsaného postupu dokonce druhý algoritmus pracuje s poli, která jsou

250krát menší. Stejně v časové složitosti člen $\log N$ bude menší než člen V vyskytující se v časové složitosti prvního algoritmu. První algoritmus je tedy pro omezenou výšku V asymptoticky lepší, ale ve skutečnosti bude lepší než druhý popsáný algoritmus až pro velmi velké hodnoty N . Lze tedy říci, že druhý algoritmus je použitelnější.

```

program p_1_2;
{ Řešení úlohy P-I-2 verze 1 }
const MAXN=100;
      VYSKA_MISTNOSTI=250;
var vyska: array[1..MAXN] of word; { výšky knih }
    n: word;      { počet knih }
    A: array[0..MAXN,1..VYSKA_MISTNOSTI] of integer;
      { pole minimálních šířek skříně }
    B: array[0..MAXN,1..VYSKA_MISTNOSTI] of word;
      { počty knih na poslední polici v optimálním řešení }
function max(a,b:longint):longint;
begin
    if a<b then max:=b else max:=a
end;
procedure vypis(n: word; v: word);
var i,k:word;
begin
    if n=0 then
        begin
            writeln('Výška skříně: ',VYSKA_MISTNOSTI-v+1,' cm');
            exit;
        end;
    k:=0;
    for i:=n-B[n,v]+1 to n do k:=max(k,vyska[i]);
    vypis(n-B[n,v],v-k-1);
    writeln;
    writeln('Výška poličky: ',k,' cm');
    write('Knihy na polici:');
    for i:=n-B[n,v]+1 to n do write(' ',i,(' ',vyska[i], ' cm'));
    writeln;
end;
var i,j,k: word;
    maxvyska: word;
begin
    readln(n);
    for i:=1 to n do read(vyska[i]);
    for i:=1 to n do
        if vyska[i]>VYSKA_MISTNOSTI-2 then
            begin
                writeln('Pro zadané rozměry knih neexistuje knihovna!');
                halt;
            end;
    for j:=1 to VYSKA_MISTNOSTI do A[0,j]:=0;
    for i:=1 to n do
        for j:=1 to VYSKA_MISTNOSTI do
            begin
                maxvyska:=vyska[i];
                A[i,j]:=-1;

```



```

    for k:=1 to i do
        begin
            maxvyska:=max(maxvyska,vyska[i-k+1]);
            if maxvyska+2>j then break;
            if A[i-k,j-maxvyska-1]=-1 then continue;
            if (A[i,j]=-1) or (A[i,j]>max(A[i-k,j-maxvyska-1],k)) then
                begin
                    A[i,j]:=max(A[i-k,j-maxvyska-1],k);
                    B[i,j]:=k;
                end;
        end;
    end;
end;
end;
if A[n,VYSKA_MISTNOSTI]=-1 then
    begin
        writeln('Pro zadané knihy nelze knihovnu navrhnout.');
        halt;
    end;
    writeln('Optimální šířka skříně je ',A[n,VYSKA_MISTNOSTI],' cm.');
    vypis(n,VYSKA_MISTNOSTI);
end.

program p_1_2;
{ Řešení úlohy P-I-2 verze 2 }
const MAXN=1000;
    VYSKA_MISTNOSTI=250;
var vyska: array[1..MAXN] of word; { výšky knih }
    n: word; { počet knih }
function lze_skrin(s: word; v: word; vypisovat: boolean):boolean;
    var A: array[0..MAXN] of word; { pole s minimálními výškami knihoven }
        B: array[1..MAXN] of word; { pole s počty knih na poličkách }
        maxvyska: word;
        i, j: word;
    begin
        A[0]:=1;
        for i:=1 to n do
            begin
                maxvyska:=vyska[i];
                A[i]:=A[i-1]+maxvyska+1;
                B[i]:=1;
                j:=i-1;
                while (j>0) and (i-j<s) do
                    begin
                        if maxvyska<vyska[j] then maxvyska:=vyska[j];
                        if A[i]>A[j-1]+maxvyska+1 then
                            begin
                                A[i]:=A[j-1]+maxvyska+1;
                                B[i]:=i-j+1;
                            end;
                    end;
                dec(j);
            end
        end;
        lze_skrin:= A[n] <= v;
        if not(vypisovat) then exit;
        i:=n;
        writeln('Výška skříně: ',A[n],' cm.');
```

```

writeln('Výšky poliček ve skříní a jejich naplnění knihami ',
        'od spodu knihovny:');
while i>0 do
  begin
    writeln;
    writeln('Výška poličky: ',A[i]-A[B[i]]-1,' cm');
    write('Knihy v poličce:');
    for j:=B[i]+1 to i do
      write(' ',j,',' ,vyska[j],' cm');
    writeln;
    i:=B[i];
  end
end;
end;
var i:word;
    s1,s2:word;
begin
  readln(n);
  for i:=1 to n do read(vyska[i]);
  for i:=1 to n do
    if vyska[i]>VYSKA_MISTNOSTI-2 then
      begin
        writeln('Pro zadané rozměry knih neexistuje knihovna!');
        halt;
      end;
  s1:=1; s2:=n;
  while s1<s2 do
    if lze_skrin((s1+s2) div 2, VYSKA_MISTNOSTI, false) then
      s2:=(s1+s2) div 2
    else
      s1:=(s1+s2) div 2+1;
  writeln('Optimální šířka skříně je ',s1,' cm.');
```

```

  lze_skrin(s1,VYSKA_MISTNOSTI,true);
end.
```

P – I – 3

Máme zadán poslední sloupec setříděné tabulky. Základní myšlenka celého řešení spočívá v tom, že tímto je dán i první sloupec — stačí setřídít písmena posledního sloupce podle abecedy. Nyní využijeme toho, že jednotlivé řádky tabulky vznikly rotací nějakého řetězce; tedy je-li na některém řádku v prvním sloupci písmeno x a v posledním sloupci písmeno y , znamená to, že v původním řetězci bylo písmeno x za písmenem y (bráno cyklicky — tedy za posledním písmenem následuje první). Vzhledem k tomu, že každé písmeno se v řetězci vyskytuje nejvýše jednou, n řádků tabulky nám již určuje pořadí písmen v původním řetězci až na rotaci. Navíc máme zadáno, na kterém řádku se vyskytuje námi hledané slovo; tím máme určeno jeho první písmeno.

Sestrojit na základě této myšlenky algoritmus je již jednoduché. Písmena zadaného řetězce si setřídíme podle abecedy (vzhledem k tomu, že hodnoty jsou z omezeného rozsahu, nabízí se přihrádkové třídění) a vyrobíme si tabulku, v níž bude každému písmenu přiřazeno jemu odpovídající následující písmeno. Pak začneme od písmene, o němž víme, že je první, a postupujeme od něj po následnících, přičemž rovnou vypisujeme výsledek, dokud se k tomuto písmeni nevrátíme.

Časová i paměťová složitost algoritmu jsou zjevně lineární v délce zadaného řetězce.

```

program bw;
const MAX = 100;
type slovo = array[1..MAX] of char;
var prvni_sloupec, posledni_sloupec : slovo;
    radek : integer;
    delka : integer;
    buckets : array[char] of boolean;
    naslednik : array[char] of char;
    s : string;
    i, l : integer;
    ch : char;

begin
    readln (s);                                     { načtení zadání }
    delka := length (s);
    for i := 1 to delka do
        posledni_sloupec[i] := s[i];
    readln (radek);

    for ch := #0 to #255 do                         { bucket sort }
        buckets[ch] := false;
    for i := 1 to delka do
        buckets[posledni_sloupec[i]] := true;
    l := 0;
    for ch := #0 to #255 do
        if buckets[ch] then
            begin
                inc (l);
                prvni_sloupec[l] := ch;
            end;

    for i:=1 to delka do                             { určení následníků }
        naslednik[posledni_sloupec[i]] := prvni_sloupec[i];

    ch := prvni_sloupec[radek];                       { výpis }
    for i := 1 to delka do
        begin
            write (ch);
            ch := naslednik[ch];
        end;
    writeln;
end.

```

Políčit na pozici příslušného prvku prohledáváním pole políčko po políčku přinese požadovanou proceduru, přesto poněkud pomalou. Pokračujme proto, přátelé, v přemýšlení:

Sestrojíme reverzibilní verzi binárního vyhledávání, místo tradičního zápisu pomocí cyklu `while` ovšem použijeme rekurzi. Zavedeme si podproceduru `Hledej(var l,p:word)`, která bude vyhledávat hodnotu `co` v úseku $X_l, X_{l+1}, \dots, X_p$ a výsledek přičte k proměnné `kde`. Zařídí to tak, že si nejdříve spočte pozici prostředního prvku X_m zadaného úseku (pokud má úsek sudou délku, zaokrouhlíme libovolným směrem) a podle jeho hodnoty zjistí, ve které polovině úseku má hledání pokračovat: pokud $X_m < co$, pak od $m + 1$ do r , je-li $X_m > co$, tak od l do $m - 1$. Na tento úsek pak zavoláme tutéž proceduru rekurzivně, ale nesmíme zapomenout, až se vrátí, m ještě odpočítat. Nastane-li kdykoliv při porovnávání rovnost, právě jsme hodnotu `co` našli a po zvýšení `kde` o m se z procedury vracíme. Dospějeme-li v rekurzi k úseku nulové délky ($r < l$), vracíme se s prázdnou, tedy aniž bychom `kde` jakkoliv měnili.

Podle tohoto algoritmu již snadno vytvoříme program, pro odpočítávání v něm použijeme příkaz `wrap`:

```

procedure Najdi(var n:word; var X:array [1..n] of word; var co,kde:word);
  var one:word;
  procedure Hledej(var l,r:word);
    var m:word;
  begin
    if l<=r then
      wrap m += (l+r) div 2
      on
        if X[m]=co then
          kde += m
        else if X[m]<co then begin
          wrap m += 1
          on Hledej(m,r)
        end
        else begin
          wrap m -= 1
          on Hledej(l,m)
        end
      end;
  begin
    wrap one += 1
    on Hledej(one,n)
  end;

```

Postoupíme-li v rekurzi o úroveň hlouběji, zmenší se prohledávaný úsek minimálně o polovinu, takže po nejvýše $\lceil \log_2 n \rceil$ rekurzivních vo-

láních buďto hledanou hodnotu odhalíme, nebo dospějeme k úseku nulové délky, kde rekurze rovněž končí. Časová složitost tedy činí $O(\log n)$ a paměťová taktéž (pro každou úroveň rekurze spotřebujeme konstantní množství paměti).

P – II – 1

Řešení úlohy nám velmi usnadní následující trik. Nebudeme určovat počet jednobarevných trojúhelníků, ale počet dvoubarevných trojúhelníků. Požadovaný výsledek pak snadno získáme tak, že od počtu všech trojúhelníků s vrcholy v bodech obrazce (těch je $\frac{1}{6}N(N-1)(N-2)$) odečteme počet dvoubarevných.

Počet dvoubarevných trojúhelníků určíme následovně: Uvažujme nějaký bod v , z něhož vede k_v červených hran a $N - 1 - k_v$ hran žlutých. Celkem je tento bod součástí $\frac{1}{2}(N-1)(N-2)$ trojúhelníků (počet způsobů, jak zvolit zbylé dva vrcholy trojúhelníku) a z nich je $k_v(N-1-k_v)$ zaručeně dvoubarevných — to jsou ty, u nichž jsme zvolili jeden vrchol připojený červenou a druhý žlutou čarou. Když sečteme tyto počty zaručeně dvoubarevných trojúhelníků přes všechny vrcholy, dostaneme dvojnásobek počtu dvoubarevných trojúhelníků (každý z dvoubarevných trojúhelníků jsme totiž započítali právě u dvou bodů).

Z výše uvedeného rozboru je algoritmus již zřejmý. Nejdříve si u každého bodu spočítáme červené čáry. Potom podle uvedeného postupu určíme počet všech dvoubarevných trojúhelníků a následně již snadno do počítáme i počet jednobarevných trojúhelníků. Algoritmus má časovou složitost $O(M+N)$, kde M je počet červených čar, a paměťovou složitost $O(N)$.

```
program obraz;
const
  MAXN = 100;
var
  Cervene : Array[1..MAXN] of Integer; {Počet červených čar z bodů}
  N : Integer;                          {Počet bodů}

{Načtení vstupu do pole}
procedure Nacti;
var
  i : Integer;
  A, B : Integer;   {Konce načítané spojnice}
begin
  Write('Pocet bodu: ');
  ReadLn(N);
  for i := 1 to N do
    Cervene[i] := 0;
```

```

while true do begin
  Write('Pocatek cervene cary: ');
  ReadLn(A);
  if A = 0 then {Konec?}
    break;
  Write('Konec cervene cary: ');
  ReadLn(B);
  Inc(Cervene[A]);
  Inc(Cervene[B]);
end;
end;

{Spočte jednobarevné trojúhelníky}
function Spocti : Integer;
var
  i : Integer;
  Dvoj : Integer;  {Počet dvoubarevných trojúhelníků}
begin
  Dvoj := 0;
  for i := 1 to N do
    Dvoj := Dvoj + Cervene[i]*(N-1-Cervene[i]);
  Dvoj := Dvoj div 2;
  Spocti := N*(N-1)*(N-2) div 6 - Dvoj;
end;

begin
  Nacti;
  WriteLn('Pocet jednobarevných trojuhelnu: ', Spocti);
end.

```

P – II – 2

Nejprve učinme následující pozorování: Nechť s_0 je šířka optimální skříně a nechť má tato skříň p policek. Potom existují výšky $w_1 \geq \dots \geq w_p$ policek a rozmístění knih do skříně se šířkou s_0 a policičkami výšky $w_1, \dots, w_p$ takové, že výšky knih v této skříní v pořadí seshora dolů a v každé policičce zleva doprava tvoří nerostoucí posloupnost (první policička je ta nejvýše umístěná).

První část pozorování, o existenci výšek $w_1 \geq \dots \geq w_p$, je jednoduchá — pokud výšky policek ve skříní seshora dolů netvoří nerostoucí posloupnost, stačí policičky (i s jejich obsahem) ve skříní přeuspořádat. Nyní dokážeme, že existuje rozmístění knih ve skříní takové, že výšky knih tvoří nerostoucí posloupnost. Bez újmy na obecnosti můžeme předpokládat, že $v_1 \geq \dots \geq v_N$. Uvažme rozmístění knih do skříně takové, že první policička obsahuje s_0 nejvyšších knih, druhá s_0 nejvyšších knih mezi zbylými knihami atd. a v každé z policek výšky knih tvoří nerostoucí posloupnost. Tvrdíme, že výška nejvyšší knihy v i -té policičce je nejvýše w_i ,

tj. $v_{(i-1)s_0+1} \leq w_i$. Pokud tomu tak není, pak $((i-1)s_0+1)$ -tá kniha musí být v optimálním řešení na jedné z prvních $i-1$ políček, ale pak některá z $(i-1)s_0$ nejvyšších knih (řekněme ta s výškou v_k , $1 \leq k \leq (i-1)s_0$) není v optimálním řešení na jedné z prvních $i-1$ políček — je tedy na j -té políčce, $j \geq i$. Potom ale $w_j \leq v_k$, a tedy $w_i \leq v_{(i-1)s_0+1}$, což je požadovaná nerovnost.

Všimněme si, že jsme v předchozím odstavci vlastně dokázali, že ve výše popsaném optimálním řešení jsou všechny políčky až na tu poslední plné, tj. obsahují přesně s_0 knih. Základem našeho programu bude funkce `existuje(s:integer)`, která pro danou šířku s rozhodne, zda existuje knihovna maximální výšky 250 cm a šířky s , do které lze umístit všechny knihy. Optimální hodnotu s_0 nalezneme pak metodou půlení intervalu, kterou lze nalézt v popisu řešení úlohy P-I-2 domácího kola. Samotná funkce zvolí za výšku i -té políčky výšku $v_{(i-1)s_0+1}$, což je výška nejvyšší knihy, kterou uložíme do i -té políčky v řešení popsaném v minulém odstavci. Naše funkce z výšek jednotlivých políček snadno spočte výšku celé knihovny a ověří, zda je nejvýše 250 cm.

Nyní odhadněme časové a paměťové nároky výše popsaného programu. Nejprve potřebujeme seřadit N čísel, což lze učinit užitím některého ze standardních algoritmů v čase $O(N \log N)$. Časová složitost funkce `existuje` je $O(N/s)$, neboť je v ní potřeba sečíst $\lceil N/s \rceil$ čísel. Odtud již plyne, že časové nároky celého našeho algoritmu jsou majorizovány funkcí $O(N \log N)$. Pokud si uvědomíme, že $s = N/2$ při prvním volání funkce `existuje`, $s = N/4$ při druhém, atd., pak lze časové nároky algoritmu bez úvodního seřazení výšek knih dokonce odhadnout funkcí $O(N)$. Paměťové nároky algoritmu lze odhadnout funkcí $O(N)$, neboť potřebujeme pole velikosti N na uložení výšek jednotlivých knih.

```
program knihovna;
const MAXN=100;
      VYSKA_MISTNOSTI=250;
var vyska: array[1..MAXN] of word; { výšky knih }
    n: word; { počet knih }
procedure utrid_vysky(i1,i2:word); { quicksort }
var pivot: word;
    w: word;
    j1, j2: word;
begin
  if i1>=i2 then exit;
  pivot:=vyska[(i1+i2) div 2];
  j1:=i1; j2:=i2;
  while (j1<j2) do
    begin
      while (vyska[j1]>pivot) do inc(j1);
```

```

        while (vyska[j2]<pivot) do dec(j2);
        w:=vyska[j1]; vyska[j1]:=vyska[j2]; vyska[j2]:=w;
        inc(j1); dec(j2);
    end;
    utrid_vysky(i1,j2);
    utrid_vysky(j1,i2);
end;
function existuje(s:word):boolean;
var v:word;
    i:word;
begin
    v:=1; i:=1;
    repeat
        v:=v+vyska[i]+1;
        i:=i+s;
    until i>n;
    existuje:=v<=VYSKA_MISTNOSTI
end;
var i:word;
    s1,s2:word;
    v:word;
begin
    readln(n);
    for i:=1 to n do read(vyska[i]);
    utrid_vysky(1,n);
    if vyska[1]>VYSKA_MISTNOSTI-2 then
        begin
            writeln('Pro zadané rozměry knih neexistuje knihovna!');
            halt;
        end;
    s1:=1; s2:=n;
    while s1<s2 do
        if existuje((s1+s2) div 2) then
            s2:=(s1+s2) div 2
        else
            s1:=(s1+s2) div 2+1;
    writeln('Optimální šířka skříně je ',s1,' cm.');
```

```

    writeln('Počet poliček ve skříní: ',(n+s1-1) div s1);
    i:=1; v:=1;
    while (i<=n) do
        begin
            v:=v+vyska[i]+1;
            writeln('Výška poličky: ',vyska[i],' cm');
```

```

            write('Výšky knih na poličce:');
            repeat
                if (i>n) then break;
                write(' ',vyska[i],' cm');
```

```

                inc(i);
            until (i mod s1)=1;
            writeln;
        end;
    writeln('Výška skříně: ',v,' cm');
```

```

end.

```


Použijeme myšlenku podobnou té z řešení úlohy P–I–3 domácího kola; problém ovšem je, že když se nám nyní mohou písmena opakovat, následníci nemusí být jednoznačně určeni. Provedeme následující úvahu:

Máme dán poslední sloupec, jeho setříděním dostaneme první sloupec. Dále máme dānu pozici slova, které bylo zakódováno, v setříděné tabulce, tedy známe jeho první písmeno; nechť je to x . Toto písmeno se nám může v prvním sloupci vyskytovat vícekrát, na pozicích odpovídajících slovům $xv_1, xv_2, \dots, xv_k$, kde $xv_1 \leq xv_2 \leq \dots \leq xv_k$. Z toho ovšem plyne také $v_1x \leq v_2x \leq \dots \leq v_kx$, a tedy je-li xv_j zakódované slovo, wx j -té (v abecedním pořadí) slovo končící na x , musí platit $w = v_j$. Nyní můžeme celý postup opakovat (pozice, na níž je první písmeno zbytku zakódovaného slova, je ta, na níž je v posledním sloupci j -té písmeno x).

Algoritmus je již pouze přímočarým přepisem této myšlenky. Implementace tohoto algoritmu je poměrně jednoduchá; místo komplikované práce s dvojicemi (písmeno, pozice) je výhodnější si písmena v posledním sloupci očíslovat (písmenu přiřadíme jeho index v posledním sloupci) a po setřídění (příhrádkovým tříděním, abychom dosáhli lineární časové složitosti) pracovat pouze s těmito indexy.

Časová i paměťová složitost algoritmu jsou opět lineární.

```

program transformace;
const MAX = 10000;
var prvni_sloupec : array[1 .. MAX] of integer;
    posledni_sloupec : string;
    radek, delka, i, l : integer;
    buckets: array[char] of integer;
    ch : char;
begin
    {nacteni a ocislovani}
    readln (posledni_sloupec);
    readln (radek);
    delka := length (posledni_sloupec);
    for ch := #0 to #255 do buckets[ch] := 0;
    for i := 1 to delka do
        inc (buckets[posledni_sloupec[i]]);

    {setrideni}
    l := 1;
    for ch := #0 to #255 do
        begin
            i := l;
            inc (l, buckets[ch]);
            buckets[ch] := i;
        end;
    for i := 1 to delka do
        begin

```

```

    ch := posledni_sloupec[i];
    l := buckets[ch];
    inc (buckets[ch]);
    prvni_sloupec[l] := i;
end;

{vypis}
for i:=1 to delka do
  begin
    write (posledni_sloupec[prvni_sloupec[radek]]);
    radek := prvni_sloupec[radek];
  end;
writeln;
end.

```

P – II – 4

Podobnost úlohy s počítáním vzdálenosti vrcholů (tj. délky nejkratší cesty mezi nimi) v orientovaném grafu jistě není náhodná, držíme se proto i my grafové analogie: Jednotlivé budovy Ouřadu jsou pro nás vrcholy, potrubí mezi nimi orientovanými hranami grafu a A není ničím jiným než maticí sousednosti grafu. Nabízí se použít prohledávání grafu do šířky, ovšem musíme je náležitě upravit, aby bylo reverzibilní.

Vrcholy grafu si rozdělíme do *vrstev*: i -tá vrstva W_i bude obsahovat právě ty vrcholy, jejichž vzdálenost od vrcholu x je rovna i . Vrstev je proto nejvýše n a můžeme je snadno zkonstruovat indukci: do W_0 padne vrchol x a žádný další; když máme sestaveny vrstvy W_0 až W_{i-1} , tak do W_i patří právě ty vrcholy w , do kterých vede hrana z nějakého vrcholu $v \in W_{i-1}$ (tedy existuje cesta délky i z x do w) a $w \notin W_j$ pro $j < i$ (neexistuje žádná kratší cesta).

To je bezpochyby reverzibilní postup — při konstrukci vrstvy nijak neměníme vrstvy už spočítané; nakonec najdeme číslo vrstvy, do které padl vrchol y , to vydáme jako výsledek a všechny informace o vrstvách opět odpočítáme. Tak dostaneme řešení s časovou složitostí $O(n^3)$ a prostorovou složitostí $O(n^2)$. Všimněme si ještě dvou drobností:

1. Ačkoliv vrstev může být až n a v každé z nich až $n - 1$ vrcholů, lze je uložit efektivněji, protože ve všech vrstvách dohromady je nejvýše n vrcholů. Stačí je všechny naskládat za sebe do jednoho pole (říkejme mu třeba V) a nechat druhé pole S ukazovat, kde v poli V která vrstva začíná. Vrcholy ve vrstvě W_i tedy budou uloženy v prvcích V_{S_i} až $V_{S_{i+1}-1}$.
2. Reverzibilita programu není příliš nakloněna značkování vrcholů. Když si totiž budeme v nějakém poli pro každý vrchol pamatovat,

zda jsme v něm již byli, a případně jej pak označujeme, řekněme takto:

```
if UžJsemTamByl[i]=0 then begin
    { objevil jsem nový vrchol a někam si ho zapíšu }
    UžJsemTamByl[i] += 1;
end;
```

dostaneme se do sporu s reverzibilitou podmínky: po ukončení příkazu if nepoznáme, zda byla podmínka splněna či nikoliv, protože UžJsemTamByl[i] bude vždycky jednička. To přesně náš jazyk zakazuje. Naštěstí nás zachrání jednoduchý trik: pokud dokážeme zajistit, abychom v rámci jedné vrstvy na každý vrchol narazili nejvýše jednou, stačí si u každého vrcholu zapamatovat (k tomu budeme používat pole *L*), ve které vrstvě byl objeven, a pokud dosud objeven nebyl, tak nějaké dostatečně velké číslo *inf*. Test se změní na

```
if L[i] >= TatoVrstva then begin
    { objevil jsem nový vrchol a někam si ho zapíšu }
    L[i] -= inf - TatoVrstva;
end;
```

a to už je korektní: platnost podmínky v této vrstvě se totiž přenastavením *L[i]* nezmění, ale v dalších vrstvách již správně poznáme, že vrchol byl zpracován.

Zde je program využívající oba popsané triky:

```
procedure Zkoumej(var n:word; var A:array [1..n] of array [1..n] of bit;
    var x,y,d:word);
var inf,cnt:word;
var L,V,S:array [0..n] of word;
begin
    wrap begin
        inf += n+1;
        for var i = 1 to n do
            L[i] += inf;
        V[0] += x;
        L[x] -= inf;
        S[1] += 1;
        for var i = 1 to n-1 do begin
            S[i+1] += S[i];
            for var w = 1 to n do
                if L[w] >= i then
                    wrap
                    for var j = S[i-1] to S[i]-1 do
                        if A[V[j]][w]=1 then
                            cnt += 1
            on if cnt>0 then begin { ano => přidat do i-té vrstvy }
```

```

        V[S[i+1]] += w;
        S[i+1] += 1;
        L[w] -= inf-i    { L[w] >= i stále platí }
    end
end
end
on d += L[y]                { vrátíme výsledek }
end;

```

Zbývá ještě dodat, že prostorová složitost procedury je lineární a časová kvadratická (inicializace je lineární, vše mimo cyklu řízeného proměnnou j kvadratické a vnitřek zbylého cyklu se provede pro každý vrchol j právě n -krát, takže je dohromady také kvadratický).

Poznámka. Pokud bychom se vzdali polynomiální časové složitosti, existovala by prostorově ještě efektivnější řešení. Jedno z nich je založeno na následující úvaze: hledám-li cestu délky l z x do y , pak je buďto $l < 2$ (tehdy je úloha triviální), nebo cesta musí mít nějaký střední vrchol ve vzdálenosti $\lfloor \frac{1}{2}l \rfloor$. Vyzkouším proto postupně všechny vrcholy a pro každý z nich si rekurzivním zavoláním téže funkce pro obě poloviny cesty a poloviční l ověřím, zda existuje příslušná polovina cesty. Hloubka rekurze je maximálně $\lceil \log_2 l \rceil = O(\log n)$, dosáhneme tedy prostorové složitosti $O(\log n)$ za cenu drastického zpomalení na $n^{O(\log n)}$.

P – III – 1

Při řešení úlohy si nejdříve uvědomíme, že ze zadaných čísel hraček můžeme snadno odvodit, které dítě chce hračku po kterém dítěti. Situaci si představíme jako orientovaný graf, kde vrcholy odpovídají dětem a od vrcholu i vede hrana k vrcholu j , pokud dítě i chce hračku po dítěti j . Protože dítě je ochotno vyměnit hračku pouze tehdy, když dostane tu svou vytouženou, mohou si děti vyměňovat hračky pouze po cyklech — aby se dítě i_1 vzdalo své hračky, musí dostat hračku od i_2 , to od i_3 a tak dále, až nějaké dítě dostane hračku od i_1 . Chceme tedy nalézt v grafu množinu disjunktních kružnic (pro snazší vyjadřování budeme nadále považovat za kružnici i vrchol se smyčkou), které dohromady obsahují co nejvíce vrcholů. Hledání těchto kružnic je usnadněno tím, že každé dvě kružnice v našem grafu jsou disjunktní — kdyby nějaké dvě kružnice měly společný vrchol, musely by se v nějakém místě také od sebe oddělovat. Z příslušného vrcholu by tedy musely vést dvě hrany, což ovšem v našem grafu není možné.

A nyní jak budeme kružnice hledat: Začneme v libovolném vrcholu (třeba prvním) a půjdeme po hranách (z každého vrcholu vede právě

jedna hrana, takže postup je jednoznačný), dokud se nevrátíme do nějakého vrcholu, ve kterém jsme už byli (to poznáme snadno, když si budeme označovat navštívené vrcholy). Tím jsme v grafu našli nějakou kružnici, tu můžeme vypsat a její vrcholy označit za vyřešené. Vrcholy, které jsme prošli předtím, než jsme se dostali na kružnici, pro změnu zaručeně na žádné kružnici neleží (jinak by z nějakého vrcholu musely vést alespoň dvě hrany). Proto se těmito vrcholy už nikdy nemusíme zabývat a můžeme je rovněž označit jako vyřešené. Nyní vezmeme další dosud nevyřešený vrchol a opět se z něj vydáme hledat kružnici. Pokud narazíme na nějaký již vyřešený vrchol, hledání ukončíme a prošlé vrcholy označíme jako vyřešené — nemohou totiž zřejmě ležet na žádné kružnici. Když už nezbude žádný nevyřešený vrchol, máme nalezeny všechny kružnice a výpočet ukončíme. Jediným nedorozumím problémem zůstává, jak rychle hledat dosud nevyřešené vrcholy. To můžeme snadno dělat tak, že při hledání dalšího nevyřešeného vrcholu začneme hledat od naposledy nalezeného vrcholu (před ním jistě žádné nevyřešené již nejsou). Díky tomu s hledáním vrcholů strávíme dohromady čas $O(N)$, a protože na nalezení kružnic potřebujeme dohromady též $O(N)$ (každou hranou projdeme nejvýše jednou), je celková časová složitost $O(N)$. Paměťová složitost je také $O(N)$.

```

/* Hračkářství */
#include <stdio.h>

#define MAXD 100    /* Maximální počet dětí */

int N;              /* Počet dětí */
int Ma[MAXD];       /* Číslo hračky, kterou příslušné dítě má */
int Vlastni[MAXD];  /* Dítě, které vlastní příslušnou hračku */
int Chce[MAXD];     /* Dítě, jehož hračku příslušné dítě chce */
int Hotovo[MAXD];   /* Už jsme dítě řešili? */

/* Načte vstup */
void nacti(void)
{
    int i;

    scanf("%d", &N);
    for (i = 0; i < N; i++) {
        printf("Dítě %d: ", i+1);
        scanf("%d %d", &Ma[i], &Chce[i]);
        Ma[i]--; Chce[i]--;
        Vlastni[Ma[i]] = i;
    }
    /* Převědeme odkazy na hračky na odkazy na děti */
    for (i = 0; i < N; i++)
        Chce[i] = Vlastni[Chce[i]];
}

```

```

/* Projde děti a zjistí největší spokojenou skupinu */
void res(int act)
{
    int start = act;

    /* Projde děti a najde cyklus */
    while (!Hotovo[act]) {
        Hotovo[act] = 1;
        act = Chce[act];
    }
    /* Vypíše cyklus */
    while (Hotovo[act] != 2) {
        Hotovo[act] = 2;
        printf(" %d", act+1);
        act = Chce[act];
    }
    /* Ještě označíme zbylé prošlé vrcholy */
    act = start;
    while (Hotovo[act] != 2) {
        Hotovo[act] = 2;
        act = Chce[act];
    }
}

int main(void)
{
    int i;

    nacti();

    printf("Spokojene deti:");
    for (i = 0; i < N; i++)
        if (!Hotovo[i]) /* Zatím jsme dítě neřešili? */
            res(i);
    printf("\n");
    return 0;
}

```

P – III – 2

Základem našeho řešení bude funkce `existuje(s:integer)`, která pro zadanou šířku s rozhodne, zda existuje knihovna s P políčkami, do které lze umístit všech N knih. Označme T součet tloušťek knih, tj. $T = t_1 + \dots + t_N$. Potom minimální šířka knihovny s P políčkami pro dané knihy je alespoň T/P . Na druhou stranu, určitě existuje knihovna šířky $T/P + t_{\max}$, kde $t_{\max}$ je maximální tloušťka knihy: Knihy rozmístíme na políčky tak, že každých prvních k políček obsahuje nejmenší možný počet knih takový, aby součet tloušťek knih na těchto políčkách byl alespoň kT/P . Snadno nahlédneme, že šířka každé políčky je nejvýše $T/P + t_{\max}$, a tedy existuje knihovna takové šířky. Optimální šířku skříně

pak nalezneme vyzkoušením všech hodnot mezi T/P a $T/P + t_{\max}$ jako možné šířky skříně. Takových hodnot je ale konstantně mnoho kvůli omezení na tloušťku knihy ze zadání úlohy.

Samotná funkce **existuje** bude fungovat následovně: Pro zadané s nalezne největší i_1 takové, že $\sum_{i=1}^{i_1} t_i \leq s$; je jasné, že i_1 je maximální možný počet knih, které lze umístit do první poličky. Poté nalezneme největší i_2 takové, že $\sum_{i=i_1+1}^{i_2} t_i \leq s$, tedy největší možný počet knih i_2 , které lze umístit do prvních dvou poliček, atd. Pokud se nám podaří umístit všechny knihy, tj. $i_P = N$, pak existuje knihovna šířky s , do které lze všechny knihy uložit; v opačném případě taková knihovna zjevně neexistuje.

Zbývá domyslet, jak rychle hledat čísla i_k , $1 \leq k \leq P$, ve funkci **existuje**. Za tímto účelem si nejprve vytvoříme pomocné pole, ve kterém budou uloženy součty tlouštěk prvních j knih pro $1 \leq j \leq N$. Při počítání hodnoty i_k metodou půlení intervalu vyhledáme v tomto pomocném poli největší číslo i' takové, že $\sum_{i=1}^{i'} t_i - \sum_{i=1}^{i_{k-1}} t_i \leq s$; zřejmě i' je hledaná hodnota i_k .

Nyní odhadněme časovou a paměťovou složitost našeho algoritmu. Funkce **existuje** provede P vyhledávání v poli velikosti N , tj. doba jejího běhu je majorizována funkcí $O(P \log N)$. Celková doba běhu našeho programu je tedy $O(N + P \log N)$; čas $O(N)$ spotřebujeme kromě načtení dat také na vytvoření pomocného pole popsaného v minulém odstavci. Pokud by platilo, že $P \log N > N$, lze výše popsanou funkci **existuje** nahradit jednodušší funkcí pracující v čase $O(N)$, která místo P binárních vyhledávání projde pole sekvenčně. Časová složitost našeho programu je tedy majorizována funkcí $O(N)$. Paměťová složitost je $O(N)$ — pole velikosti N je potřeba na uložení tlouštěk jednotlivých knih a stejná je i velikost pomocného pole.

```
program knihovna;
const MAXN=1000;
var tloustka: array[1..MAXN] of word; { tloušťky knih }
    soucet: array[0..MAXN] of word;   { součty tlouštěk knih }
    n: word;                          { počet knih }
    p: word;                          { počet poliček }
function vyhledej(s: word): word;
begin
    i1:=0; i2:=n;
    while i1<i2 do
```

```

        if soucet[(i1+i2+1) div 2]>s then
            i2:=(i1+i2+1) div 2-1
        else
            i1:=(i1+i2+1) div 2;
        vyhledej:=i1
    end;
function existuje(sirka: word):boolean;
var i,j: word;
begin
    i:=0;
    for j:=1 to p do i:=vyhledej(soucet[i]+sirka);
        existuje:=i=n;
    end;
var i: word;
    s1, s2: word;
    tmax: word;
begin
    readln(n,p);
    tmax:=0;
    for i:=1 to n do
        begin
            read(tloustka[i]);
            if tmax<tloustka[i] then tmax:=tloustka[i]
        end;
    soucet[0]:=0;
    for i:=1 to n do soucet[i]:=soucet[i-1]+tloustka[i];
    s1:=soucet[n] div p;
    s2:=soucet[n] div p+tmax;
    while s1<s2 do
        if existuje((s1+s2) div 2) then
            s2:=(s1+s2) div 2
        else
            s1:=(s1+s2) div 2+1;
    writeln('Optimální šířka skříně: ',s1,' mm');
    i:=1;
    while i<=n do
        begin
            write('Knihy na policiče:');
            s2:=0;
            while (i<=n) and (s2+tloustka[i]<=s1) do
                begin
                    write(' ',i,'(',tloustka[i],' mm)');
                    s2:=s2+tloustka[i];
                    inc(i);
                end;
            writeln;
        end
    end
end.

```

P – III – 3

První řešení. Inspirujeme se tradičním algoritmem pro sčítání čísel „pod sebou“ a uvědomíme si, že není závislý na použité číselné soustavě (zvěda-

vější povahy obětují 5 minut na důkaz indukci). Pokud bychom uměli spočítat přenosy mezi řády, je samotné sečtení triviální: $A'_i = A_i \text{ xor } B_i \text{ xor } P_i$, kde P_i je přenos z $(i-1)$ -ního do i -tého řádu (*xor* funguje úplně stejně jako sčítání dvou bitů modulo 2). Pokud jsme ochotni obětovat paměť na všechna P_i , můžeme je spočítat postupně: $P_0 = 0$; pro $i > 0$ je $P_i = 1$, když buďto A_{i-1} a B_{i-1} jsou současně jedničky, nebo když alespoň jedno z nich je jednička a P_{i-1} je jednička. Z toho okamžitě dostáváme program s lineární časovou i prostorovou složitostí:

```

procedure Add(var n:word; var A,B:array [0..n-1] of bit);
var P:array [0..n] of bit;
begin
  wrap
    for var i = 0 to n-1 do
      P[i+1] ^= (A[i] and B[i]) or ((A[i] or B[i]) and P[i])
    on
      for var i = 0 to n-1 do
        A[i] ^= B[i] xor P[i]
      end;
end;
```

My se ovšem s lineárním množstvím paměti nespokojíme a zkusíme být při výpočtu přenosů šetrnější. Celý problém je v tom, že na spočtení P_i potřebujeme P_{i-1} , a to musí být dostupné i v okamžiku, kdy budeme P_i odpočítávat (pokusy o odpočítávání P_i pomocí P_{i+1} selhávají na tom, že když už jsme si jednoho ze sčítanců přepsali výsledkem, nelze určit, zda jednička z výsledku vznikla z jedničky v přepsaném sčítanci nebo z nuly a přenosu z nižšího řádu). Takže si musíme P_{i-1} celou dobu pamatovat a prostorová složitost prostě musí být vždy alespoň lineární a naše první řešení je optimální... a nebo přeci jen ne? Nešlo by na P_{i-1} zapomenout, a až budeme chtít P_i odpočítat, tak si P_{i-1} spočítat znovu? To by fungovalo, ale musíme to provést šikovně, abychom rekurzivním voláním výpočtů předchozích P_j nespotřebovali více paměti, než jsme ušetřili. Tak získáme

Druhé řešení. Sestrojíme si proceduru $\text{Prenos}(i, l, in, out)$, která pro nějaký úsek čísel A a B (konkrétně od i -tého řádu do $(i+l-1)$ -ního) za předpokladu, že přenos do našeho úseku z nižších řádů $P_i = in$, spočítá přenos P_{i+l} do vyšších řádů a přixoruje jej k proměnné out . Pokud je úsek jednoprvkový, udělá to již dobře známým způsobem z našeho prvního řešení (v konstantní paměti). Větší úsek si rozdělí na poloviny, rekurzivně si spočítá přenos mid z nižší poloviny do vyšší, pak rekurzivně spočítá přenos z vyšší poloviny „ven“ a nakonec mid třetím rekurzivním zavoláním odpočítá. To se dá jako obvykle snadno zapsat pomocí příkazu `wrap`:

```

procedure Prenos(var i,l:word; var in,out:bit);
var l1,l2,j:word;
var mid:bit;
begin
  if l=1 then
    out ^= (A[i] and B[i]) or ((A[i] or B[i]) and in)
  else wrap begin
    l1 += l div 2;
    l2 += l-l1;
    j += i+l1;
    Prenos(i,l1,in,mid)
  end
  on Prenos(j,l2,mid,out)
end;

```

Jelikož při každém rekurzivním volání klesne l minimálně na polovinu, je hloubka rekurze nejvýše $\lceil \log_2 l \rceil$, takže procedura `Prenos` dosahuje prostorové složitosti $O(\log l)$. Se složitostí časovou je to trochu obtížnější: Označíme-li čas strávený touto procedurou $T(l)$, bude platit $T(l) = 1 + 3T(l/2)$: procedura vykoná nějakou konstantní práci (jelikož nás složitost zajímá jen asymptoticky, můžeme předpokládat, že jednotkovou), načež třikrát zavolá sama sebe na vstup poloviční délky. Dosadíme-li tento vztah do sebe sama, dostaneme $T(l) = 1 + 3(1 + 3T(l/4)) = 1 + 3 + 9T(l/4)$ a když budeme dosazovat dál, po k krocích dojdeme k $T(l) = 1 + 3 + \dots + 3^{k-1} + 3^k T(l/2^k)$. My ale víme, že $T(1) = 1$, takže pro $k = \log_2 l$ (naše hloubka rekurze) vyjde $T(l) = 1 + 3 + \dots + 3^{\log_2 l - 1} + 3^{\log_2 l}$. To je ovšem geometrická řada se součtem $\frac{1}{2}(3^{k+1} - 1) = O(3^k) = O(3^{\log_2 l})$, což můžeme ještě zjednodušit: $3^{\log_2 l} = (2^{\log_2 3})^{\log_2 l} = 2^{\log_2 3 \log_2 l} = (2^{\log_2 l})^{\log_2 3} = l^{\log_2 3} \leq l^{1,59}$. Z toho plyne, že časová složitost celé procedury je $T(l) = O(l^{1,59})$.

Teď bychom mohli rekurzivní výpočet přenosů zapojit do naší původní sčítací procedury (musíme ovšem sčítat pozadu, abychom si nepřepsali hodnoty, ze kterých budeme přenosy ještě potřebovat) a získat tak sčítání s logaritmickou prostorovou složitostí v čase $O(n \cdot n^{1,59}) = O(n^{2,59})$, ale neuděláme to, protože si všimneme, že každý z blokových přenosů bychom zbytečně počítali mnohokrát.

Místo toho zkonstruujeme podobnou rekurzivní proceduru, která bude provádět současně sčítání a počítání přenosu. Nazveme ji `Secti` a bude mít úplně stejné parametry jako procedura `Prenos`. Nejdříve si zavolá proceduru `Prenos` pro výpočet přenosu z dolní poloviny bloku (ten opět přixoruje k proměnné *mid*), pak rekurzivním zavoláním sebe samé sečte horní polovinu čísla a nakonec rekurzivně zavolá sebe samu

pro dolní polovinu čísla, čímž ji jednak sečte a jednak odpočítá přenos *mid*. Triviální případ sčítání jednobitových čísel opět vyřešíme klasicky.

```

procedure Secti(var i,l:word; var in,out:bit);
var l1,l2,j:word;
var mid:bit;
begin
  if l=1 then begin           { jednobitové sčítání }
    out ^= (A[i] and B[i]) or ((A[i] or B[i]) and in);
    A[i] ^= B[i] xor in
  end
  else wrap begin
    l1 += l div 2;           { opět počítáme, kde jsou poloviny }
    l2 += l-1-l1;
    j += i+l1
  end
  on begin
    Prenos(i,l1,in,mid);     { přenos přes dolní polovinu }
    Secti(j,l2,mid,out);     { sečteme horní polovinu }
    Secti(i,l1,in,mid)       { sečteme dolní a odpočteme přenos }
  end
end;

```

Časová i prostorová složitost naší sčítací procedury bude stejná jako u procedury *Prenos*, protože až na ošetřování triviálních případů, které je konstantní, vypadají obě procedury úplně stejně. Sčítáme tedy v prostoru $O(\log n)$ a čase $O(n^{1,59})$. Program vypadá takto:

```

procedure Add(var n:word; var A,B:array [0..n-1] of bit);
{ Zde jsou vloženy procedury Prenos a Secti }
var zero:word;
var in,out:bit;
begin
  Secti(zero,n,in,out);      { víme, že out vyjde nulový }
end;

```

Třetí řešení. A nešlo by to ještě lépe? Zkusme vyřešit jednodušší problém: jak k danému číslu přičíst jedničku, tedy nalézt maximální souvislý úsek jedniček na nejnižších řádech, tyto jedničky změnit na nuly a bezprostředně předcházející nulu změnit na jedničku. Jinak řečeno změnit ty číslice, za kterými již nenásleduje žádná nula. To se ovšem dá snadno zařídit následujícím trikem: Nejdříve postupujeme od nejnižšího řádu k nejvyššímu a za každou nulu si do počítadla přičteme jedničku, a pak projdeme pole ještě jednou v opačném směru, počítadlo za každou nulu o jedničku snižujeme, a jakmile dospěje do nuly, začneme všechny bity, přes které přejdeme, negovat:

```

procedure AddOne(var n:word; var A:array [0..n-1] of bit);
var i,c:word;
begin
  for var i=0 to n-1 do
    if A[i]=0 then c += 1;

```

```

for var i=n-1 downto 0 do begin
  if A[i]=0 then c -= 1;
  if c=0 then A[i] ^= 1;
end;
end;

```

Dokážeme to tedy v lineárním čase a konstantním prostoru. Jenže když umíme přičíst jedničku, dokážeme přičíst i libovolnou mocninu dvojky — stačí začít u jiného než nejnižšího řádu, a tím pádem také libovolné jiné číslo, protože ho můžeme rozložit na mocniny dvojky a každou přičíst zvlášť:

```

procedure Add(var n:word; var A,B:array [0..n-1] of bit);
var i,j,c:word;
begin
  for var i=0 to n-1 do
    if B[i]=1 then begin
      for var j=i to n-1 do
        if A[j]=0 then c += 1;
      for var j=n-1 downto i do begin
        if A[j]=0 then c -= 1;
        if c=0 then A[j] ^= 1;
      end;
    end;
  end;
end;

```

Tak dosáhneme časové složitosti $O(N^2)$ při prostorové složitosti $O(1)$.

Poznámka na závěr. Řešení v konstantním prostoru těží z toho, že jsme v našem výpočetním modelu nadefinovali prostorovou složitost poněkud nedbale a neměříme ji v bitech, nýbrž ve wordech. Kdybychom počítali opravdu precizně, nebyla by prostorová složitost třetího řešení konstantní, nýbrž logaritmická, zatímco druhé řešení by se dalo snadno upravit tak, aby mělo stále logaritmickou složitost (stačí si uvědomit, že je lze naprogramovat nerekurzivně, čímž se zbavíme závislosti prostoru na lokální proměnné). Ovšem časové složitosti zůstanou zachovány, takže druhé řešení bude pracovat v témže prostoru rychleji.

P – III – 4

Úloha, převedená do podoby v matematice běžnější, zní: Je dán konvexní N -úhelník a M jeho neprotínajících se tětív dělících N -úhelník na díly. Nalezněte maximální počet dílů, z nichž žádné dva nemají společnou stranu.

Uvažujme následující graf G . Vrcholy grafu budou odpovídat jednotlivým dílům N -úhelníku, přičemž dva vrcholy budou spojeny hranou, pokud jim odpovídající díly mají společnou stranu. Graf G zřejmě bude

souvislý a navíc nebude obsahovat žádný cyklus. Uvnitř cyklu by totiž ležela alespoň jedna stěna grafu G . Té musí odpovídat nějaký průsečík v nakresleném N -úhelníku. Tento průsečík však rozhodně nemůže ležet na okraji N -úhelníku, a máme tak spor s tím, že žádné dvě tětivy se neprotínají.

Souvislý graf bez cyklů je strom a naše úloha se tím zjednodušuje na nalezení maximální nezávislé množiny vrcholů (tj. takové množiny vrcholů, že žádné dva vrcholy z této množiny nejsou spojeny hranou) ve stromu. Maximální nezávislou množinu můžeme určit prohledáváním do hloubky. Na počátku si označíme všechny vrcholy jako přijatelné do nezávislé množiny. Začneme v libovolném vrcholu prohledávat strom. Když se vrátíme z nějakého vrcholu, který je označen jako přijatelný, přidáme ho do nezávislé množiny a jeho otce odznačíme. Když takto projdeme celý graf, máme vybranou maximální nezávislou množinu. Nezávislost vybrané množiny je zřejmá. Proč ale bude vybraná množina maximální? Označme si vybranou nezávislou množinu A a dále si vezmeme maximální nezávislou množinu B , která se od naší vybrané množiny liší v nejméně vrcholech. Nyní se podívejme na takový vrchol v , ve kterém se A a B liší a který je nejvzdálenější od vrcholu, ve kterém začalo prohledávání do hloubky. Příklad, kdy v je v B a ne v A , nastat nemůže, protože když jsme nějaký vrchol v nevzali do A , tak pouze proto, že byl sousedem nějakého vrcholu u pod ním zařazeného do A . Protože v je nejvzdálenější vrchol, ve kterém se A a B liší, musí být u obsažen i v B , a tedy B také nemůže obsahovat v . Může tedy nastat pouze situace, že v je obsažen v A a není obsažen v B . Pokud ale v přidáme do B a z B vyřadíme otce v (pokud v ní byl), bude B stále maximální nezávislá množina a přitom se bude lišit v méně vrcholech, což je spor s výběrem B . Vybraná nezávislá množina A musí být proto skutečně maximální.

Zkonstruovat výše popsany graf a na něm pak provést prohledání do hloubky je zbytečně pracné. My budeme graf prohledávat bez jeho explicitní konstrukce. Nejdříve si tětivy zorientujeme tak, aby každá tětiva začínala ve vrcholu s nižším číslem a přidáme pomocnou tětivu začínající v prvním a končící v posledním vrcholu. Tětivy si pomocí přihrádkového třídění setřídíme vzestupně podle jejich počátku, tětivy začínající ve stejném vrcholu pak sestupně podle jejich konce. Nyní postupně procházíme vrcholy N -úhelníku v pořadí od vrcholu s číslem jedna po vrchol s číslem N . Při procházení si udržujeme zásobník s tětivami, od nichž jsme viděli začátek, ale ne konec. U každé tětivy na zásobníku si navíc pamatujeme, zda je přijatelná. Vždy, když začneme zpracovávat nový vrchol,

nejdříve ze zásobníku odebereme tětivy, které v tomto vrcholu končí. Pokud je odebíraná tětiva označena jako přijatelná, zvětšíme velikost nezávislé množiny a odznačíme tětivu pod ní v zásobníku. Po odebrání všech končících tětiv přidáme na zásobník všechny tětivy začínající v daném vrcholu a označíme je jako přijatelné. Pak pokračujeme do dalšího vrcholu N -úhelníku. Výpočet skončíme po průchodu všemi vrcholy N -úhelníku.

Uvedený algoritmus přesně odpovídá dříve popsanému prohledávání do hloubky. Každá tětiva totiž jednoznačně koresponduje s hranou v grafu, která spojuje vrcholy odpovídající dílům odděleným tětivou. Uložení pomocné tětivy $(1, N)$ na zásobník odpovídá vstupu do vrcholu, ze kterého začínáme prohledávání. Uložení další tětivy na zásobník odpovídá přechodu po odpovídající hraně dolů (směrem od vrcholu, ve kterém začalo prohledávání), vybrání tětivy ze zásobníku, pak návratu zpět po hraně. Při prohledávání do hloubky jsme si označovali vrcholy, které lze přidat do nezávislé množiny. V upraveném algoritmu místo vrcholu značíme tu hranu, po které jsme do vrcholu poprvé vstoupili. Algoritmus má časovou i paměťovou složitost $O(N)$ (tětiv nikdy nemůže být více než $N - 3$).

```
#include <stdio.h>
#include <stdlib.h>

#define MAXV 30000 /* Maximální počet vrcholů */
#define MAXR 30000 /* Maximální počet řezů */

/* Struktura pro jeden řez */
struct rez {
    int a, b;
};

int rezu, vrcholu; /* Počet řezů a vrcholů */
struct rez r[MAXR]; /* Jednotlivé řezy */
int vpoc[MAXV]; /* Počty řezů začínajících v jednotlivých vrcholech */

/* Načte vstup */
void nacti(void)
{
    int pom, i;
    FILE *vstup;

    if (!(vstup = fopen("poklad.in", "r")))
        exit(1);
    fscanf(vstup, "%d %d", &vrcholu, &rezu);
    for (i = 0; i < rezu; i++) {
        fscanf(vstup, "%d %d", &r[i].a, &r[i].b);
        r[i].a--; r[i].b--;
        if (r[i].a > r[i].b) {
            pom = r[i].a;

```

```

        r[i].a = r[i].b;
        r[i].b = pom;
    }
}
fclose(vstup);
/* Přidáme ještě fiktivní řez mezi prvním a posledním vrcholem */
r[rezu].a = 0;
r[rezu].b = vrcholu-1;
rezu++;
}

/* Setřídí řezy podle počátku a konce */
void setrid(void)
{
    struct rez r1[MAXR]; /* Jednotlivé přeskládané řezy */
    int vrchind[MAXV];   /* Index, kde začínají řezy z/do daného vrcholu */
    int vrchpoc[MAXV];   /* Počty řezů z/do daného vrcholu */
    int i;

    /* První průchod třídění */
    for (i = 0; i < vrcholu; i++)
        vrchpoc[i] = 0;
    /* Spočteme počty řezů do jednotlivých vrcholů */
    for (i = 0; i < rezu; i++)
        vrchpoc[r[i].b]++;
    vrchind[0] = 0;
    for (i = 1; i < vrcholu; i++)
        vrchind[i] = vrchind[i-1] + vrchpoc[i-1];
    /* Přerovnáme řezy podle cílového vrcholu */
    for (i = 0; i < rezu; i++)
        r1[vrchind[r[i].b]++] = r[i];

    /* Druhý průchod třídění */
    for (i = 0; i < vrcholu; i++)
        vrchpoc[i] = 0;
    for (i = 0; i < rezu; i++)
        vrchpoc[r1[i].a]++;
    vrchind[0] = 0;
    for (i = 1; i < vrcholu; i++)
        vrchind[i] = vrchind[i-1] + vrchpoc[i-1];
    /* Přerovnáme řezy podle zdrojového vrcholu
       (bereme je sestupně podle cílového vrcholu) */
    for (i = rezu-1; i >= 0; i--)
        r[vrchind[r1[i].a]++] = r1[i];
}

/* Spočte, kolik částí mapy může kapitán rozdat */
int spoceti(void)
{
    int casti = 0; /* Počet částí */
    int zasvrch = 0; /* Vrchol zásobníku */
    int zas[MAXR]; /* Zásobník na zpracovávané řezy */
    int zasuzit[MAXR]; /* Značka, že příslušná část mapy může být rozdána */
    int actvrch = 0, actrez = 0; /* Aktuální vrchol a řez mnohoúhelníku */

```

```

while (actvrch < vrcholu) {
    while (zasvrch && r[zas[zasvrch-1]].b == actvrch) {
        if (zasuzit[zasvrch-1]) {
            /* Část oddělená tímto řezem může být použita? */
            casti++;
            if (zasvrch > 1)
                zasuzit[zasvrch-2] = 0;
        }
        zasvrch--;
    }
    while (actrez < rezu && r[actrez].a == actvrch) {
        zas[zasvrch] = actrez++;
        zasuzit[zasvrch++] = 1;
    }
    actvrch++;
}
return casti;
}

int main(void)
{
    FILE *vystup;

    nacti();
    setrid();

    if (!(vystup = fopen("poklad.out", "w")))
        exit(1);
    fprintf(vystup, "%d\n", spocti());
    fclose(vystup);
    return 0;
}

```

P – III – 5

Použijeme upravený třídící algoritmus mergesort. V programu si budeme vytvářet jednosměrné spojové seznamy, jež budou mít svým jednotlivým prvkům přiřazeny mince. Váhy mincí budou od počátku ke konci seznamu tvořit rostoucí posloupnost. Mince stejné hmotnosti budou přiřazeny témuž prvku seznamu. Tento seznam budeme realizovat tak, že každý jeho prvek bude obsahovat ukazatel na následující prvek seznamu a na strom, který obsahuje mince (téže hmotnosti) přiřazené tomuto prvku. Samotný strom bude binární strom, ve kterém má každý prvek žádného nebo dva syny. Čísla mincí ve stromu budou uchovávána v jeho listech a každý uzel tohoto stromu bude obsahovat číslo některé mince ze svého podstromu (v naší implementaci to bude nejmenší číslo mince ve stromu).

Základem bude rekurzivní procedura `vytvor(prvni,posledni)`, která vytvoří jednosměrný spojový seznam popsáný v prvním odstav-

ci. Tento seznam bude obsahovat všechny mince s čísly od první do poslední. Pokud jsou čísla první a poslední shodná, procedura vytvoří jednoprvkový seznam. Jeho jediný prvek bude ukazovat na strom tvořený jedním uzlem, který bude obsahovat číslo první = poslední. Pokud jsou čísla první a poslední různá, procedura nejdříve rozdělí interval tvořený čísly od první do poslední na dva intervaly polovičních délek a na každý z nich se rekurzivně zavolá. Takto získáme dva lineární spojové seznamy s vlastnostmi popsány v prvním odstavci. Z nich naše procedura vytvoří jeden.

Výsledný seznam budeme vytvářet od začátku, a to následujícím způsobem: Na některou z mincí ve stromu hlavy (tj. prvního prvku) prvního ze seznamů a na některou z mincí ve stromu hlavy druhého seznamu zavoláme funkci `porovnej`. Pokud je mince hlavy prvního seznamu lehčí, odpojíme hlavu od prvního seznamu a připojíme ji na konec výsledného seznamu; poté pokračujeme porovnáním hlav nově vzniklé dvojice seznamů. Pokud je naopak mince hlavy druhého seznamu lehčí, připojíme na konec výsledného seznamu hlavu druhého seznamu a pokračujeme s druhým seznamem bez jeho původní hlavy. Zbývá případ, kdy mince obou hlav mají stejnou hmotnost. V tomto případě připojíme na konec výsledného seznamu prvek, který ukazuje na strom, jehož levý podstrom je strom hlavy prvního seznamu a pravý podstrom je strom hlavy druhého seznamu; od obou seznamů následně odpojíme jejich hlavy. Takto pokračujeme, dokud jeden nebo oba z našich dvou seznamů nejsou prázdné. Pokud je jeden z nich neprázdný, nezapomeneme ho připojit na konec výsledného seznamu.

Vytvořit celý program je nyní již snadné: Nejprve ze souboru `vahy.in` načteme počet mincí N . Poté zavoláme proceduru `porovnej` s parametry `prvni = 1` a `posledni = N`. Nakonec vypíšeme čísla v listech stromů (zleva doprava) ve výsledném seznamu; každý strom vypíšeme na samostatný řádek souboru `vahy.out`, a to v pořadí, v jakém stromy odpovídají prvkům seznamu. Čísla na každém řádku jsou seříděna (z konstrukce stromů patřícím prvkům seznamu) a hmotnosti mincí v pořadí dle řádků jsou rostoucí (dle vlastností vytvářeného seznamu). Zbývá si rozmyslet, že náš algoritmus neprovádí zbytečné volání funkce `porovnej`, a určit jeho časovou složitost.

Nejprve dokážeme indukci dle délky intervalu určeného parametry při volání procedury `vytvor`, že náš program neprovádí zbytečné volání funkce `porovnej`. Procedura `vytvor` volá funkci `porovnej` pouze na dvojici prvků z intervalu specifikovaného parametry procedury `vytvor`. Po-

kud je tento interval jednoprvkový, dokazované tvrzení platí z triviálních důvodů. V opačném případě se nejprve vytvoří dva seznamy rekurzivním voláním procedury `vytvor` a ty se následně sloučí. Při slučování dvou seznamů je funkce `porovnej` volána pouze na dvojice mincí z různých seznamů (tedy výsledek takového volání není určen výsledky volání funkce `porovnej` při rekurzi). Vzhledem k tomu, že porovnáváme z každého seznamu minci s nejmenší vahou (a mince s menšími vahami jsme zařadili již do výsledného seznamu), nemůže být vztah hmotností mincí z dotazované dvojice určen předchozími dotazy. Můžeme tedy uzavřít, že žádné volání funkce `porovnej` není zbytečné.

Hloubka rekurzivního volání procedury `vytvor` je $O(\log N)$ (N je počet mincí), neboť při každém volání se délka intervalu specifikovaného parametry funkce zmenší na polovinu. Na sloučení dvou seznamů je třeba čas úměrný délce výsledného seznamu. Protože na každé úrovni volání se libovolná mince vyskytuje právě v jednom seznamu, je čas strávený algoritmem během procedur `vytvor` na jedné úrovni rekurze lineární, tj. $O(N)$. Celková časová složitost je tedy $O(N \log N)$. Libovolná mince se vyskytuje při běhu programu vždy právě v jednom seznamu, a tedy paměťová složitost programu je $O(N)$.

```
#include <stdio.h>
#include <stdlib.h>
#include "vahy_lib.h"

struct tuzel {
    int prvek;
    struct tuzel *levy, *pravy;
};

struct tseznam {
    struct tuzel *strom;
    struct tseznam *dalsi;
};

struct tseznam *vytvor(int prvni, int posledni) {
    struct tseznam *vysledek, *seznam1, *seznam2, **ocas, *pomocna;
    if (prvni==posledni) {
        vysledek=malloc(sizeof(struct tseznam));
        vysledek->dalsi=NULL;
        vysledek->strom=malloc(sizeof(struct tuzel));
        vysledek->strom->prvek=prvni;
        vysledek->strom->levy=vysledek->strom->pravy=NULL;
        return vysledek;
    }
    seznam1=vytvor(prvni, (prvni+posledni)/2);
    seznam2=vytvor((prvni+posledni)/2+1, posledni);
    ocas=&vysledek;
    while (seznam1&&seznam2) {
        switch (porovnej(seznam1->strom->prvek, seznam2->strom->prvek)) {
            case 0:
```

```

        (*ocas)=malloc(sizeof(struct tseznam));
        (*ocas)->strom=malloc(sizeof(struct tuzel));
        (*ocas)->strom->prvek=seznam1->strom->prvek;
        (*ocas)->strom->levy=seznam1->strom;
        (*ocas)->strom->pravy=seznam2->strom;
        pomocna=seznam1; seznam1=seznam1->dalsi; free(pomocna);
        pomocna=seznam2; seznam2=seznam2->dalsi; free(pomocna);
        break;
    case 1:
        *ocas=seznam1; seznam1=seznam1->dalsi;
        break;
    case -1:
        *ocas=seznam2; seznam2=seznam2->dalsi;
        break;
    }
    ocas=&((*ocas)->dalsi);
}
*ocas=seznam1?seznam1:(seznam2?seznam2:NULL);
return vysledek;
}

void vypis_strom(FILE *soubor, struct tuzel *uzel) {
    if (uzel->levy) {
        vypis_strom(soubor,uzel->levy);
        vypis_strom(soubor,uzel->pravy);
    }
    else
        fprintf(soubor,"%d ",uzel->prvek);
}

void vypis_seznam(FILE *soubor, struct tseznam *seznam) {
    while (seznam) {
        vypis_strom(soubor, seznam->strom);
        fprintf(soubor,"\n");
        seznam=seznam->dalsi;
    }
}

int main(void) {
    FILE *soubor;
    int N;
    struct tseznam *seznam;
    soubor=fopen("vahy.in","r");
    fscanf(soubor,"%d",&N);
    fclose(soubor);
    seznam=vytvor(1,N);
    soubor=fopen("vahy.out","w");
    vypis_seznam(soubor,seznam);
    fclose(soubor);
    return 0;
}

```

Přípravná soustředění před 44. MMO

V průběhu 52. ročníku se konalo výběrové soustředění pro přípravu na mezinárodní matematickou olympiádu bezprostředně po skončeném celostátním kole kategorie A, a to od 7. do 11. dubna 2003 v Kostelci nad Černými lesy nedaleko Prahy. Na soustředění bylo pozváno 10 nejlepších řešitelů III. kola kategorie A s výjimkou těch, kteří se rozhodli dát přednost účasti na Mezinárodní fyzikální olympiádě. Soustředění bylo zaměřeno na přípravu reprezentantů a ke konečné nominaci šestičlenného družstva.

Úspěšnost jednotlivých studentů ukazuje následující tabulka:

Jan Moláček	3/4, GJKT Hradec Králové	77
Pavel Čížek	8/8, G Kralupy nad Vltavou	74
Vítězslav Kala	3/4, G Brno, tř. Kpt. Jaroše	70
Marek Krčál	4/4, G Brno, tř. Kpt. Jaroše	69
Pavel Kocourek	2/4, SPŠST, Praha 1, Panská	66,5
Jaromír Kuben	1/4, G Brno, tř. Kpt. Jaroše	65,5
František Konopecký	6/8, G Holešov	61
Martin Káldy	4/4, GChD, Praha 5	56
Tomáš Gavenčiak	3/4, GMK, Bílovec	53
Marek Pechal	5/8, G Zlín, Lesní čtvrť	53

Na základě uvedených výsledků, v nichž jsou započítány i výsledky oblastního a celostátního kola, bylo prvních šest vybráno do reprezentativního družstva a sedmý byl určen jako náhradník. Toto družstvo nás reprezentovalo i na již tradičním střetnutí s družstvy Slovenska a Polska.

Jednotlivé semináře vedli a úlohy připravili:

- dr. Karel Horák (7. 4.),
- dr. Jaroslav Zhouf (8. 4.),
- dr. Martin Panák (9. 4.),
- dr. Jaroslav Švrček (10. 4.)
- a doc. Jaromír Šimša (11. 4.).

Úlohy zadané na přípravném soustředění

1. Je dán tětívový čtyřúhelník $ABCD$. Označme K průsečík přímky BC s tečnou ve vrcholu A a L průsečík přímky AD s tečnou ve vrcholu B ke kružnici opsané danému čtyřúhelníku. Jestliže $|AL| = |AD|$ a $|BK| = |BC|$, je $ABCD$ lichoběžník. Dokažte.

2. Necht' mnohočleny P, Q, R s reálnými koeficienty, mezi nimiž je mnohočlen druhého a mnohočlen třetího stupně, splňují rovnost

$$P^2 + Q^2 = R^2.$$

Dokažte, že pak jeden z mnohočlenů třetího stupně má vesměs reálné kořeny.

3. V rovině je dán konečný počet modrých a červených přímek, přičemž žádné dvě nejsou rovnoběžné a každým průsečíkem dvou přímek téže barvy prochází i přímka druhé barvy. Dokažte, že všechny přímky procházejí jedním bodem.

4. Necht' $n \geq 2$ je přirozené číslo a pro kladná reálná čísla $x_1, x_2, \dots, x_n$ platí rovnost

$$\frac{1}{1+x_1} + \frac{1}{1+x_2} + \dots + \frac{1}{1+x_n} = 1.$$

Dokažte, že

$$x_1 x_2 \dots x_n \geq (n-1)^n.$$

5. Necht' v trojúhelníku ABC je V průsečík výšek a S střed kružnice opsané a necht' přímky AV a AS protínají kružnici opsanou postupně v bodech M a N . Označme postupně P, Q a R průsečíky přímek BC a VN , BC a SM , VQ a SP . Dokažte, že $ASRV$ je rovnoběžník.

6. Je dána množina $M = \{1, 2, 3, \dots, 2002, 2003\}$. Dokažte, že existuje 14 podmnožin množiny M takových, že pro každé $n \in M$ mezi nimi existuje právě sedm množin takových, že n je jejich společným prvkem.

7. Na kruhu je dáno $4n$ bodů střídavě obarvených modře a červeně. Modré body jsou libovolně rozděleny do n párů a body v každém páru jsou spojeny modrou tětivou. Podobně červené body jsou libovolně rozděleny do n párů a body v každém páru jsou spojeny červenou tětivou. Body leží na kruhu tak, že žádné tři tětivy neprocházejí jedním bodem. Dokažte, že existuje aspoň n bodů, v nichž některá modrá tětiva protíná některou červenou tětivu.

8. Dokažte, že $\{n\sqrt{3}\} > \frac{1}{n\sqrt{3}}$ pro libovolné přirozené n . (Symbol $\{n\}$ značí tzv. zlomkovou část čísla n .)

9. Nechť n je přirozené číslo a $f(x) = a_m x^m + \dots + a_1 x + a_0$ je mnohočlen s celočíselnými koeficienty takový, že:

- (i) každé z čísel $a_2, a_3, \dots, a_m$ je dělitelné všemi prvočísly dělícími n ,
- (ii) a_1 a n jsou nesoudělná.

Dokažte, že pro libovolné přirozené k existuje přirozené c tak, že $f(c)$ je dělitelné číslem n^k .

10. Nechť $x_1, x_2, \dots$ je nekonečná náhodná posloupnost nul a jedniček. Co je pravděpodobnější:

- (i) posloupnost 110 se v ní vyskytne před posloupností 010,
- (ii) posloupnost 010 se v ní vyskytne před posloupností 110?

11. Nechť pro kladná čísla x, y, z platí $xyz(x+y+z) = 1$. Určete nejmenší hodnotu výrazu $V = (x+y)(y+z)$.

12. Nechť M je střed strany AB daného trojúhelníku ABC . Sestrojte rovnoběžku p se stranou AB tak, aby její průsečíky E a F po řadě se stranami BC a AC tvořily vrcholy pravoúhlého trojúhelníku EFM s přeponou EF .

13. Nechť T je těžiště a M libovolný bod trojúhelníku ABC . Označme A_1, B_1, C_1 průsečíky přímky MT po řadě s přímkami BC, CA, AB . Dokažte, že platí nerovnost

$$|MA_1| \cdot |MB_1| \cdot |MC_1| \leq |TA_1| \cdot |TB_1| \cdot |TC_1|.$$

14. Nechť $ABCD$ je tětíkový čtyřúhelník se středem S kružnice jemu opsané a P průsečík jeho úhlopříček. Kružnice opsané trojúhelníkům ABP a CDP se protínají v bodě Q ($Q \neq P$). Jestliže jsou body S, Q, P navzájem různé, pak přímky SQ a PQ jsou navzájem kolmé. Dokažte.

15. Určete počet těch čtveřic (a, b, c, d) přirozených čísel, pro které platí

$$1 \leq a < b < c < d \leq 30 \quad \text{a} \quad a + d = b + c.$$

Výsledek uveďte jedním číslem zapsaným v desítkové soustavě.

16. Označme $q(T)$ ten z poměrů délek dvou stran daného trojúhelníku T , který je nejbližší číslu 1. Určete nejmenší kladné číslo C takové, že pro každý trojúhelník T platí nerovnost $|1 - q(T)| \leq C$.

17. Vnitřním bodem daného trojúhelníku ABC vedeme rovnoběžky s jeho stranami. Tyto přímky vytnou na trojúhelníku ABC tři úsečky téže délky. Vyjádřete ji pomocí délek a, b, c stran trojúhelníku ABC . Výsledek zapište ve tvaru podílu dvou mnohočlenů proměnných a, b, c .

18. V rovině je dána konečná množina bodů M a osm kružnic $k_1, k_2, \dots, k_8$ tak, že kružnice k_j prochází právě j body množiny M pro každé $j \in \{1, 2, \dots, 8\}$. Určete nejmenší možný počet prvků (bodů) množiny M .

19. Necht x, y, z jsou navzájem různá celá čísla splňující rovnici

$$(x - y)(y - z)(z - x) = x + y + z.$$

Najděte nejmenší možnou hodnotu $|x + y + z|$.

20. Pro každé $n \geq 4$ určete největší číslo C_n , při kterém nerovnost

$$\frac{a_1}{a_n + a_2} + \frac{a_2}{a_1 + a_3} + \dots + \frac{a_n}{a_{n-1} + a_1} \geq C_n$$

platí pro libovolná kladná čísla $a_1, a_2, \dots, a_n$.

Mezinárodní střetnutí česko-polsko-slovenské

ŽILINA, 16.–17. ČERVNA 2003

V rámci závěrečné přípravy před MMO se uskutečnilo již třetí mezinárodní střetnutí mezi týmy České republiky, Polska a Slovenska. Jednotlivé země reprezentovala šestice účastníků, kteří si vybojovali ve svých zemích postup na 44. MMO v Tokiu.

Soutěž se uskutečnila v termínu 17.–18. 6. 2003 ve slovenské Žilině. Všechna tři reprezentační družstva přicestovala na místo konání již v neděli večer 15. 6. 2003. Organizace a průběh soutěže zůstal zachován z předešlých ročníků — je přizpůsoben stylu III. kola naší MO a podmínkám na MMO. Soutěžícím byly ve dvou dnech předloženy dvě trojice soutěžních úloh, přitom za každou z úloh mohli získat nejvýše 7 bodů, tj. celkově (stejně jako na MMO) 42 body. Na každou trojici úloh měli soutěžící vyhrazeno 4,5 hodiny.

Pořadí	Jméno	Země	body	Součet
1.–3.	Péter Koltai	SVK	777777	42
	Marcin Pilipczuk	POL	777777	42
	Aleksander Zabłocki	POL	777777	42
4.	Paweł Januszewski	POL	727777	37
5.	<i>Jan Moláček</i>	CZE	726772	31
6.–9.	Hana Budáčova	SVK	771771	30
	Tomáš Váňa	SVK	770772	30
	<i>Vítězslav Kala</i>	CZE	777072	30
	<i>Pavel Kocourek</i>	CZE	770772	30
10.–11.	Michal Burger	SVK	720776	29
	Kamil Duszenko	POL	727760	29
12.	<i>Marek Krčál</i>	CZE	725770	28
13.	Witold Rębacz	POL	770750	26
14.	Jakub Závodný	SVK	721771	25
15.–16.	<i>Jaromír Kuben</i>	CZE	727070	23
	Michał Lasoń	POL	727070	23
17.	<i>Pavel Čížek</i>	CZE	726070	22
18.	František Šimančík	SVK	700770	21

Úlohy pro letošní soutěž vybrali slovenští organizátoři — většinou z úloh, jež prošly společnou česko-slovenskou úlohovou komisí. Jejich koordinaci prováděla mezinárodní jury, kterou tvořili *Rafal Lochowski* a *Paulina Domagalska* z Polska, prof. dr. *Jozef Moravčík* a doc. *Oli-ver Ralík* ze Slovenska a dr. *Karel Horák* a dr. *Jaroslav Švrček* za Čes-kou republiku. Na zdárném průběhu celé soutěže, která proběhla vesměs na půdě Vysoké školy dopravní v Žilině, má nepřehlédnutelnou zásluhu doc. *Vojtech Bálint*, vedoucí příslušné katedry matematiky.

Texty soutěžních úloh

1. Nechť $n \geq 2$ je přirozené číslo. V oboru reálných čísel řešte soustavu rovnic

$$\begin{aligned}\max\{1, x_1\} &= x_2, \\ \max\{2, x_2\} &= x_3, \\ &\dots \\ \max\{n-1, x_{n-1}\} &= (n-1)x_n, \\ \max\{n, x_n\} &= nx_1.\end{aligned}$$

2. Je dán ostroúhlý trojúhelník ABC , v němž velikost vnitřního úhlu při vrcholu B je větší než 45° . Nechť D , E , F jsou po řadě paty výšek z vrcholů A , B , C a nechť K je takový bod úsečky AF , že platí $|\sphericalangle DKF| = |\sphericalangle KEF|$. Dokažte, že

- a) takový bod K vždy existuje;
- b) platí rovnost $|KD|^2 = |FD|^2 + |AF| \cdot |BF|$.

3. Jestliže pro reálná čísla p , q , r z intervalu $\langle \frac{2}{5}, \frac{5}{2} \rangle$ platí $pqr = 1$, pak existují dva trojúhelníky o stejném obsahu, z nichž jeden má strany a , b , c a druhý má strany pa , qb , rc . Dokažte.

4. Je dán trojúhelník ABC a jeho vnitřní bod P , který leží na těžnici z vrcholu C . Označme X průsečík přímky AP s přímkou BC a Y průsečík přímky BP s přímkou AC . Je-li čtyřúhelník $ABXY$ tětiový, potom je trojúhelník ABC rovnoramenný. Dokažte.

5. Určete všechna přirozená čísla $n \geq 2$, pro něž jsou všechny binomické koeficienty

$$\binom{n}{1}, \binom{n}{2}, \dots, \binom{n}{n-1}$$

sudá čísla.

6. Určete všechny funkce $f: \mathbb{R} \rightarrow \mathbb{R}$, které pro všechna $x, y \in \mathbb{R}$ splňují rovnici

$$f(f(x) + y) = 2x + f(f(y) - x).$$

Řešení úloh

1. Ukážeme nejprve, že pro každé $i \in \{1, 2, \dots, n\}$ platí $x_i \leq i$. Důkaz provedeme sporem. Předpokládejme, že $x_i > i$ pro nějaké i .

Je-li $x_1 > 1$, plyne z poslední rovnice dané soustavy nerovnost $\max\{n, x_n\} = nx_1 > n$, takže $x_n > n$. Jestliže dále pro nějaké $i > 1$ je $x_i > i$, pak $\max\{i-1, x_{i-1}\} = (i-1)x_i > (i-1)i > i-1$. Je tedy také $x_{i-1} > i-1$. Odtud plyne, že pokud nerovnost $x_i > i$ platí pro některé $i \in \{1, 2, \dots, n\}$, pak platí už pro každé $i \in \{1, 2, \dots, n\}$. V takovém případě má však daná soustava tvar

$$x_1 = x_2, \quad x_2 = 2x_3, \quad \dots, \quad x_{n-1} = (n-1)x_n, \quad x_n = nx_1.$$

Vynásobením těchto rovnic dostaneme $x_1 x_2 \dots x_n = n! x_1 x_2 \dots x_n$, což neplatí pro žádné přirozené $n \geq 2$. Všechna x_i jsou totiž kladná čísla. To je spor.

Pro všechna $i \in \{1, 2, \dots, n\}$ je tudíž $x_i \leq i$. Proto

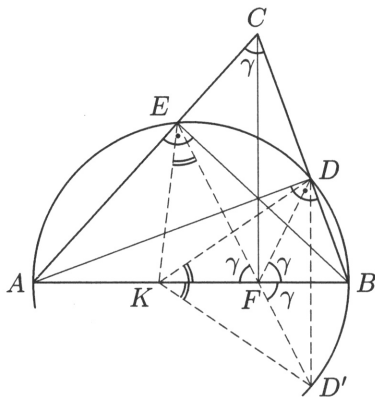
$$i = \max\{i, x_i\} = i x_{i+1}, \quad \text{kde klademe } x_{n+1} = x_1.$$

Odtud již snadno získáme jediné reálné řešení dané soustavy:

$$x_1 = x_2 = \dots = x_n = 1.$$

2. a) Označme velikosti vnitřních úhlů daného trojúhelníku ABC obvyklým způsobem a uvažujme Thaletovu kružnici sestavenou nad průměrem BC . Vzhledem k tomu, že trojúhelník ABC je ostroúhlý, leží paty výšek E, F v polorovině BCA . Z vlastností tětiového čtyřúhelníku $BCEF$ plyne (obr. 36), že je $|\sphericalangle AFE| = \gamma$ a $|\sphericalangle AEF| = \beta$. Podobně z tětiového čtyřúhelníku $AFDC$ plyne, že $|\sphericalangle DFB| = \gamma$. Je-li $K = A$, je $|\sphericalangle DKF| = |\sphericalangle DAF| = 90^\circ - \beta$ a $|\sphericalangle KEF| = |\sphericalangle AEF| = \beta$.

Jestliže se bod K bude spojitě pohybovat po úsečce AF od bodu A k bodu F , poroste velikost úhlu DKF spojitě od hodnoty $90^\circ - \beta$ k hodnotě γ (v ostroúhlém trojúhelníku je $90^\circ - \beta < \gamma$) a současně bude velikost úhlu KEF spojitě klesat, a to od velikosti $\beta > 90^\circ - \beta$ k hodnotě 0° . Existuje proto na úsečce AF bod K , pro který platí $|\sphericalangle DKF| = |\sphericalangle KEF|$.



Obr. 36

b) Nechť D' je obraz bodu D v osové souměrnosti podle přímky AB . Protože $|\sphericalangle AFE| = |\sphericalangle DFB| = \gamma$, leží body E , F a D' na jedné přímce.

Přímka KD' je podle věty o úsekovém úhlu tečnou kružnice k opsané trojúhelníku KFE , neboť $|\sphericalangle D'KF| = |\sphericalangle DKF| = |\sphericalangle KEF|$. Pro mocnost bodu D' ke kružnici k platí

$$\begin{aligned} |KD'|^2 &= |D'F| \cdot |D'E| = |D'F|(|D'F| + |FE|) = \\ &= |D'F|^2 + |D'F| \cdot |FE|. \end{aligned} \quad (1)$$

Nyní stačí využít rovnosti $|FD| = |D'F|$ a $|KD| = |KD'|$, které plynou ze souměrnosti bodů D a D' podle AB , a mocnost bodu F ke kružnici s průměrem AB , která je

$$|EF| \cdot |FD'| = |AF| \cdot |BF|.$$

Dosazením do (1) tak dostaneme $|KD|^2 = |FD|^2 + |AF| \cdot |BF|$, což jsme chtěli dokázat.

3. Ze zadání úlohy plyne, že některá dvě z čísel p , q , r jsou buď nejvýše rovna 1, anebo jsou aspoň 1. Můžeme je proto označit tak, že nastane jeden z následujících dvou případů:

(i) $p \leq q \leq 1 \leq r$,

(ii) $r \leq 1 \leq p \leq q$.

(i) Položme $a = q$, $b = 1$, $c = pq$, potom platí $pa = pq = c$, $qb = q = a$, $rc = pqr = 1 = b$. Trojúhelníky, jejichž strany mají velikosti a , b , c a pa ,

pb , pc , jsou tedy shodné (pokud existují). Ukážeme, že trojúhelník se stranami délek q , 1 , pq existuje. Protože $pq \leq q \leq 1$, stačí ověřit jedinou trojúhelníkovou nerovnost, a to $pq + q > 1$. Ze vztahů

$$pq = \frac{1}{r} \quad \text{a} \quad r \leq \frac{5}{2} \quad \text{plyne} \quad pq \geq \frac{2}{5}.$$

Vzhledem k tomu, že $p \leq q$, platí rovněž

$$q \geq \sqrt{\frac{2}{5}} > \frac{3}{5}, \quad \text{a tedy} \quad pq + q > 1.$$

(ii) Položme opět $a = q$, $b = 1$, $c = pq$. Ukážeme, že i v tomto případě existuje trojúhelník se stranami délek q , 1 , pq . Protože nyní $pq \geq q \geq 1$, stačí ověřit nerovnost $pq < q + 1$. Z nerovnosti $p \leq q$ vyplývá $\sqrt{pq} \leq q$; stačí proto ověřit silnější nerovnost $pq < \sqrt{pq} + 1$, tj. že $t = \sqrt{pq}$ splňuje kvadratickou nerovnost $t^2 - t - 1 < 0$, neboli že $-\frac{3}{2} < \sqrt{pq} < \frac{5}{2}$. Ze vztahů

$$pq = \frac{1}{r} \quad \text{a} \quad r \geq \frac{2}{5} \quad \text{plyne} \quad \sqrt{pq} \leq \sqrt{\frac{5}{2}} < \frac{5}{2},$$

a navíc je $\sqrt{pq} \geq 1$. Tím je důkaz hotov.

4. Označme délky stran trojúhelníku ABC obvyklým způsobem a , b , c a D střed strany AB . Z mocnosti bodu C ke kružnici opsané čtyřúhelníku $ABXY$ dostaneme $|CA| \cdot |CY| = |CB| \cdot |CX|$, tedy $a \cdot |CX| = b \cdot |CY|$. Z Cèvyovy věty pak vyplývá

$$\frac{|AD| \cdot |BX| \cdot |CY|}{|DB| \cdot |XC| \cdot |YA|} = \frac{|BX| \cdot |CY|}{|XC| \cdot |YA|} = 1,$$

takže dosazením $a \cdot |CX| = b \cdot |CY|$ dostáváme dále $a \cdot |BX| = b \cdot |AY|$. Sečtením rovností

$$a \cdot |CX| = b \cdot |CY| \quad \text{a} \quad a \cdot |BX| = b \cdot |AY|$$

dostaneme $a^2 = b^2$, neboli $a = b$. Trojúhelník ABC je tedy rovnoramenný.

Jiné řešení. Je-li čtyřúhelník $ABXY$ je tětíkový, jsou trojúhelníky ABC a XYC podobné (uu), platí proto $a \cdot |CX| = b \cdot |CY|$.

Označme S_{EFG} obsah trojúhelníku EFG . Pro obsahy trojúhelníků zřejmě platí

$$\frac{S_{APC}}{S_{APY}} = \frac{|AC|}{|AY|} = \frac{b}{|AY|} \quad \text{a} \quad \frac{S_{BPC}}{S_{BPX}} = \frac{|BC|}{|BX|} = \frac{a}{|BX|}.$$

Protože bod P leží na těžnici z vrcholu C trojúhelníku ABC , platí také $S_{APC} = S_{BPC}$, což s oběma předešlými vztahy dává

$$\frac{b}{|AY|} S_{APY} = \frac{a}{|BX|} S_{BPX}.$$

Z rovnosti obvodových úhlů AXB a AYB a dále z rovnosti vrcholových úhlů při vrcholu P plyne podobnost trojúhelníků APY a BPX (uu). Platí tedy $S_{APY} : S_{BPX} = |AY|^2 : |BX|^2$, což ve spojení s předchozím vztahem dává $a \cdot |BX| = b \cdot |AY|$. Dále pokračujeme jako v předešlém řešení.

5. Ukážeme, že podmínkám úlohy vyhovují všechna přirozená čísla n , která jsou mocninami čísla 2, tj. všechna přirozená čísla tvaru $n = 2^m$, kde m je přirozené číslo.

Pro každé $k \in \{1, 2, \dots, 2^m - 1\}$ je

$$\binom{2^m}{k} = \frac{2^m \cdot (2^m - 1) \cdot \dots \cdot (2^m - k + 1)}{1 \cdot 2 \cdot \dots \cdot k}. \quad (1)$$

Libovolné přirozené číslo $r \in \{1, \dots, k - 1\}$ lze zapsat ve tvaru $2^\alpha l$, kde l je liché číslo a $\alpha < m$ je celé nezáporné číslo. Proto každý ze zlomků

$$\frac{2^m - r}{r} = \frac{2^{m-\alpha} - l}{l}$$

má po zkrácení v čitateli i jmenovateli lichá čísla. Podobně i číslo k lze zapsat ve tvaru $2^\alpha l$, proto zlomek na pravé straně rovnosti

$$\frac{2^m}{k} = \frac{2^{m-\alpha}}{l}$$

má v čitateli sudé a ve jmenovateli liché číslo. Součin všech těchto zlomků pro $r = 1, 2, \dots, k$ je roven kombinačnímu číslu (1), což je tudíž sudé číslo. Tím jsme dokázali, že každé kombinační číslo tvaru (1) je sudé.

Nechť naopak n není mocninou čísla 2, tj. $n = c \cdot 2^m$, kde $c \geq 3$ je liché číslo. Ukážeme, že kombinační číslo

$$\binom{c \cdot 2^m}{2^m} = \frac{c \cdot 2^m (c \cdot 2^m - 1) \cdot \dots \cdot (c \cdot 2^m - 2^m + 1)}{1 \cdot 2 \cdot 3 \cdot \dots \cdot 2^m} \quad (2)$$

je liché. Podobně jako prve ukážeme, že pro všechna $r \in \{1, 2, \dots, 2^m - 1\}$ má každý ze zlomků

$$\frac{c \cdot 2^m - r}{r} \quad \text{a také} \quad \frac{c \cdot 2^m}{2^m} = c$$

po zkrácení v čitateli i jmenovateli lichá čísla. Součin všech těchto zlomků je roven kombinačnímu číslu (2), které je proto liché.

Dané úloze vyhovují všechna přirozená čísla n , která jsou mocninou čísla 2.

6. Pro každé $c \in \mathbb{R}$ je funkce $f(x) = x + c$ řešením dané funkcionální rovnice (obě její strany jsou pak rovny $x + y + 2c$). Ukážeme, že jiná řešení daná rovnice nemá.

Nejprve dokážeme, že funkce f je surjektivní. Volbou $y = -f(x)$ v dané rovnici dostaneme

$$f(0) - 2x = f(f(-f(x)) - x).$$

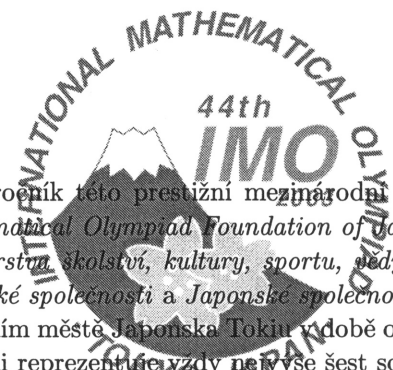
Protože každé reálné číslo lze vyjádřit ve tvaru $f(0) - 2x$, existuje pro každé $y \in \mathbb{R}$ takové $z \in \mathbb{R}$, že platí $y = f(z)$. Speciálně pak existuje $a \in \mathbb{R}$, pro něž platí $f(a) = 0$. Volbou $x = a$ v dané funkcionální rovnici dostaneme

$$f(y) = 2a + f(f(y) - a) \quad \text{tj.} \quad f(y) - a = f(f(y) - a) + a.$$

Protože funkce f je surjektivní, existuje pro každé $x \in \mathbb{R}$ takové $y \in \mathbb{R}$, že $x = f(y) - a$. Odtud plyne, že pro každé x reálné platí $x = f(x) + a$, tj. $f(x) = x - a$.

Tím je úloha vyřešena.

44. mezinárodní matematická olympiáda



V pořadí již 44. ročník této prestižní mezinárodní soutěže uspořádala společnost *Mathematical Olympiad Foundation of Japan* za podpory japonského Ministerstva školství, kultury, sportu, vědy a technologie, *Japonské matematické společnosti* a *Japonské společnosti pro matematické vzdělávání* v hlavním městě Japonska Tokiu v době od 7. do 19. července 2003. Každou zemi reprezentuje vždy nejvýše šest soutěžících; letošního ročníku MMO se zúčastnilo 457 studentů z 82 zemí.

Výběr soutěžících za Českou republiku byl proveden v Kostelci nad Černými lesy na závěrečném soutěžním soustředění deseti nejúspěšnějších účastníků celostátního kola. Vybraní soutěžící se pak ještě zúčastnili trojutkání ve slovenské Žilině mezi Českou republikou, Slovenskem a Polskem, kde soutěžili reprezentanti zúčastněných zemí za podmínek podobných jako při soutěži na MMO. Po této přípravě odjela do Japonska tato šestice soutěžících: *Pavel Čížek* z Gymnázia v Kralupech nad Vltavou, *Vítězslav Kala*, *Marek Krčál* a *Jaromír Kuben* z Gymnázia na tř. Kpt. Jaroše v Brně, *Pavel Kocourek* z SPŠST v Panské ulici v Praze 1 a *Jan Moláček* z Gymnázia J. K. Tyla v Hradci Králové. Vedoucím české delegace byl RNDr. *Karel Horák*, CSc., z Matematického ústavu Akademie věd v Praze, zástupcem vedoucího byl doc. RNDr. *Jaromír Šimša*, CSc., z Masarykovy Univerzity v Brně. Vedoucí delegace přicestoval do Tokia kvůli výběru úloh již 7. července, ostatní čeští účastníci pak o čtyři dny později.

Téměř vše se odehrávalo v moderním areálu budov *Národního olympijského střediska*, postaveném pro obdobná sportovní, kulturní a vzdělávací setkání v místě, kde se v roce 1964 konaly letní olympijské hry. Jen začátek soutěže od příletu soutěžících až do odpoledne druhého soutěžního dne strávila mezinárodní jury složená z vedoucích národních týmů v nedalekém Makuhari.

Den po příletu soutěžících se konalo slavnostní zahájení. Vlastní soutěž pak proběhla v neděli a v pondělí 13. a 14. července. Každý z těchto dnů řešili soutěžící trojici úloh po dobu 4,5 hodiny. Za každou úlohu mohli získat maximálně 7 bodů.

O náročnosti soutěžních úloh svědčí i nízké hranice pro získání medailí: na bronzovou medaili stačilo 13 bodů, stříbro se udělovalo za 19–28 bodů a zlato za alespoň 29 z možného počtu 42 bodů. Výsledky našich jsou uvedeny v následující tabulce:

Umístění		Body za úlohu						Body	Cena
		1	2	3	4	5	6		
269.–291.	Pavel Čížek, 8. roč. gymnázia, Kralupy nad Vltavou	7	1	0	0	1	0	9	HM
231.–246.	Vítězslav Kala, 3. roč. gymnázia, Brno, tř. Kpt. Jaroše	7	0	0	3	1	0	11	HM
179.–196.	Pavel Kocourek, 2. roč. SPŠST, Panská ulice, Praha 1	0	7	0	7	0	0	14	III.
269.–291.	Marek Krčál, 4. roč. gymnázia, Brno, tř. Kpt. Jaroše	7	0	0	2	0	0	9	HM
138.–162.	Jaromír Kuben, 1. roč. gymnázia, Brno, tř. Kpt. Jaroše	7	1	0	7	1	0	16	III.
93.–97.	Jan Moláček, 3. roč. GJKT, Hradec Králové	5	7	0	7	1	0	20	II.
Celkem		33	16	0	26	4	0	79	

Jak je z tabulky vidět, zklamali oba naši maturanti, shodou okolností první dva vítězové letošního celostátního kola. Přitom nejlepší náš účastník, *Jan Moláček* z Gymnázia v Hradci Králové, skončil v celostátním kole až na 11. místě a do reprezentačního výběru se dostal jen díky neúčasti několika vítězů, kteří dali přednost přípravě na Mezinárodní fyzikální olympiádu. Stejně jako vloni získal 20 bodů, kdy to stačilo jen na bronzovou medaili, tentokrát však stejný bodový zisk znamenal stříbro. Podobně i Vítězslav Kala téměř zopakoval svůj loňský výsledek: získal o dva body méně, a zatímco vloni mu k bronzové medaili chyběl jen jeden bod, letos to byly body dva. V celkovém pořadí se však posunul o dvě příčky výše.

Každý z našich soutěžících vyřešil aspoň jednu z úloh za plný počet bodů a získal tak *Honorary mention* (HM), ocenění, které je zvykem udělovat od 29. ročníku MMO. Žádný bod neztratili jen tři soutěžící: *Yunhao Fu* z Číny (ten dosáhl stejného úspěchu už na loňské 43. MMO ve skotském Glasgow) a dva soutěžící *Hung Viet Bao Le* a *Trong Canh Nguyen* z Vietnamu.

	I	II	III	body		I	II	III	body
Bulharsko	6	0	0	227	Norsko	0	1	0	62
ČLR	5	1	0	211	Arménie	0	0	3	61
USA	4	2	0	188	Bosna a Hercegovina	0	0	2	61
Vietnam	2	3	1	172	JAR	0	0	3	60
Rusko	3	2	1	167	Španělsko	0	0	1	59
Korea	2	4	0	157	Makedonie	0	0	2	54
Rumunsko	1	4	1	143	Švédsko	0	0	1	52
Turecko	1	3	1	133	Itálie	0	0	1	50
Japonsko	1	3	2	131	Kirgizie	0	0	2	50
Maďarsko	1	3	1	128	Lotyšsko	0	0	1	50
Velká Británie	1	2	3	128	Litva	0	0	2	49
Kanada	2	0	3	119	Uzbekistán	0	1	1	49
Kazachstán	1	2	2	119	Estonsko	0	0	0	47
Ukrajina	1	2	3	118	Finsko	0	0	1	43
Indie	0	4	1	115	Maroko	0	0	0	43
Tchaj-wan	1	2	2	114	Nový Zéland	0	0	0	43
Írán	0	3	2	112	Macao	0	0	2	40
Německo	1	2	1	112	Rakousko	0	0	0	38
Bělorusko	1	2	2	111	Peru (4)	0	0	1	37
Thajsko	1	1	3	111	Turkmenistán (4)	0	0	1	37
Izrael (5)	0	2	3	103	Island	0	0	1	33
Polsko	1	2	0	102	Trinidad a Tobago	0	0	0	33
Srbsko a Černá Hora	0	3	1	101	Nizozemsko	0	0	0	30
Francie	0	2	2	95	Uruguay (5)	0	0	0	29
Mongolsko	0	1	3	93	Dánsko (5)	0	0	0	27
Austrálie	0	2	2	92	Malajsie (5)	0	0	0	26
Brazílie	0	1	3	92	Švýcarsko	0	0	0	26
Argentina	1	1	2	91	Lucembursko (2)	0	0	1	25
Hongkong	0	2	2	91	Albánie (4)	0	0	0	23
Moldavsko	0	1	2	88	Kypr	0	0	0	23
Řecko	0	1	4	88	Portoriko (3)	0	0	1	23
Gruzie	0	1	2	86	Portugalsko	0	0	0	22
Chorvatsko	0	0	3	80	Irsko	0	0	0	21
Česká republika	0	1	2	79	Slovinsko	0	0	0	18
Slovensko	0	0	4	77	Kuba (1)	0	0	1	14
Singapur	0	0	2	71	Ekvádor	0	0	0	11
Belgie	0	1	1	70	Venezuela (3)	0	0	0	10
Indonézie	0	0	2	70	Filipíny	0	0	0	9
Kolumbie	0	0	3	67	Kuvajt (3)	0	0	0	8
Ázerbájdžán	0	1	1	66	Srí Lanka (4)	0	0	0	4
Mexiko	0	0	3	64	Paraguay (1)	0	0	0	0

Jak je patrné z tabulky zúčastněných států, na první místo v neoficiálním pořadí jednotlivých zemí podle celkového bodového zisku se

tentokrát vyšvihlo Bulharsko, další místa obsadily tradičně výborná družstva Číny, Spojených států, Vietnamu a Ruska. (Případná čísla v závorce upozorňují na nižší počet reprezentantů.)

Texty soutěžních úloh

(v závorce je uvedena země, která úlohu navrhla)

1. Nechť A je podmnožina množiny $S = \{1, 2, \dots, 1\,000\,000\}$ obsahující právě 101 prvků. Dokažte, že v S existují čísla $t_1, t_2, \dots, t_{100}$ taková, že množiny

$$A_j = \{x + t_j : x \in A\} \quad \text{pro } j = 1, 2, \dots, 100$$

jsou navzájem disjunktní.

(Brazílie)

2. Určete všechny dvojice přirozených čísel (a, b) takových, že

$$\frac{a^2}{2ab^2 - b^3 + 1}$$

je přirozené číslo.

(Bulharsko)

3. Je dán konvexní šestiúhelník, jehož libovolné dvě protější strany mají následující vlastnost: vzdálenost jejich středů je $\sqrt{3}/2$ násobek součtu jejich délek. Dokažte, že všechny úhly daného šestiúhelníku jsou stejné.

(Konvexní šestiúhelník $ABCDEF$ má tři dvojice protějších stran: AB a DE , BC a EF , CD a FA .)

(Polsko)

4. Nechť $ABCD$ je tětíkový čtyřúhelník. Označme postupně P , Q a R paty kolmic z bodu D na přímky BC , CA a AB . Dokažte, že $|PQ| = |QR|$, právě když se osy úhlů ABC a ADC protínají na přímce AC .

(Finsko)

5. Nechť n je přirozené číslo a $x_1, x_2, \dots, x_n$ reálná čísla taková, že $x_1 \leq x_2 \leq \dots \leq x_n$.

(a) Dokažte, že

$$\left(\sum_{i=1}^n \sum_{j=1}^n |x_i - x_j| \right)^2 \leq \frac{2(n^2 - 1)}{3} \sum_{i=1}^n \sum_{j=1}^n (x_i - x_j)^2.$$

(b) Ukažte, že rovnost platí, právě když $x_1, x_2, \dots, x_n$ je aritmetická posloupnost.

(Irsko)

6. Nechť p je prvočíslo. Dokažte, že existuje prvočíslo q takové, že pro žádné celé n není číslo $n^p - p$ dělitelné q .

(Francie)

Řešení úloh

1. Vytvořme množinu všech rozdílů $D = \{x - y : x, y \in A\}$. Protože A má 101 prvků, obsahuje D kromě nuly nejvýše $2 \cdot \binom{101}{2} = 101 \cdot 100 = 10\,100$ dalších (kladných i záporných) čísel. Všimněme si, že dvě z uvažovaných množin A_i, A_j jsou disjunktní, právě když $x + t_i \neq y + t_j$ pro libovolná $x, y \in A$, tedy právě když $t_i - t_j \notin D$. Naším úkolem je proto vybrat čísla $t_1, t_2, \dots, t_{100} \in S$ tak, aby žádný jejich rozdíl nepadl do „zakázané“ množiny D .

Zmíněný výběr provedeme induktivně. První číslo t_1 vybereme v S libovolně. Předpokládejme, že jsme již pro některé $k \leq 99$ vybrali čísla $t_1, t_2, \dots, t_k \in S$ tak, že $t_i - t_j \notin D$ pro libovolná různá $i, j \in \{1, 2, \dots, k\}$ (pro $k = 1$ je to splněno triviálně). Číslo t_{k+1} musíme v S zvolit tak, aby platilo $t_{k+1} - t_i \notin D$ pro každé $i \in \{1, 2, \dots, k\}$. Pro pevné i tak má číslo t_{k+1} právě tolik „zakázaných“ hodnot $t_i + d$, kolik je všech čísel $d \in D$. Těch je, jak víme, nejvýše $1 + 101 \cdot 100 = 10\,101$. Pro všechna $i \in \{1, 2, \dots, k\}$ tak dostaneme celkem nejvýše $k \cdot 10\,101$ zakázaných hodnot, což je nejvýše $99 \cdot 10\,101 = 999\,999$ čísel. V množině S je však 10^6 čísel, takže výběr čísla t_{k+1} je možný.

Poznámka. Hodnota $|S| = 10^6$ je zbytečně velká (v předchozím řešení jsme zanedbali skutečnost, že v množině D leží s každým číslem i číslo opačné). Dá se ukázat, že pro libovolnou k -prvkovou podmnožinu A množiny $S = \{1, 2, \dots, n\}$ platí: je-li m přirozené číslo takové, že

$$n > (m-1) \left(\binom{k}{2} + 1 \right),$$

existují v množině S čísla $t_1, t_2, \dots, t_m$ taková, že množiny $A_j = \{x + t_j : x \in A\}$ ($j = 1, 2, \dots, m$) jsou navzájem disjunktní. (Pro $k = 101$ stačí tedy uvažovat množinu $S = \{1, 2, \dots, 500\,051\}$.)

2. (Podle *Jana Moláčka*.) Ukážeme, že řešeními jsou právě všechny dvojice (a, b) tvaru $(8k^4 - k, 2k)$, $(k, 2k)$ a $(k, 1)$, kde $k \in \mathbb{N}$ je libovolné.

Hledáme přirozená čísla a, b, n , pro která platí

$$\frac{a^2}{2ab^2 - b^3 + 1} = n. \tag{1}$$

Rovnost (1) lze upravit do tvaru kvadratické rovnice s neznámou a :

$$a^2 - 2nb^2a + n(b^3 - 1) = 0.$$

Jejími kořeny jsou čísla

$$a_{1,2} = nb^2 \pm \sqrt{(nb^2)^2 - n(b^3 - 1)}. \quad (2)$$

Protože jeden z kořenů $a_{1,2}$ je roven hledanému přirozenému číslu a , odmocněnec ve vzorci (2) musí být „úplný kvadrát“, tedy tvaru

$$(nb^2)^2 - n(b^3 - 1) = d^2 \quad (3)$$

pro vhodné celé $d \geq 0$. Takové číslo d zaručeně existuje, pokud $b = 1$ (pak $b^3 - 1 = 0$, takže $d = nb^2$). Zabývejme se však nejprve obsažnějším případem, kdy $b > 1$. Ukažme, že pro takové b z (3) plynou odhady

$$nb^2 - \frac{b+1}{2} < d < nb^2 - \frac{b-1}{2}. \quad (4)$$

Protože oba krajní výrazy jsou kladné, můžeme obě nerovnosti umocnit; po dosazení d^2 a snadných algebraických úpravách dostaneme dvojici nerovností

$$(b+1)^2 < 4n(b^2+1) \quad \text{a} \quad (b-1)^2 + 4n(b^2-1) > 0,$$

které zřejmě platí, neboť $n \geq 1$ a $b > 1$. Tím jsou odhady (4) dokázány.

Všimněme si nyní, že rozdíl obou krajních výrazů v (4) je roven jedné. Pro liché b by se tyto výrazy dokonce rovnaly dvěma po sobě jdoucím přirozeným číslům, takže by žádné celé d splňující podmínku (4) neexistovalo. Číslo b je proto sudé, tedy $b = 2k$ pro vhodné $k \in \mathbb{N}$; jediné celé d vyhovující nerovnostem (4) je pak tvaru

$$d = nb^2 - \frac{b}{2} = 4nk^2 - k.$$

Pro taková b a d přejde rovnost (3) do tvaru

$$(4nk^2)^2 - n(8k^3 - 1) = (4nk^2 - k)^2,$$

ze které snadno plyne $n = k^2$; vzorce (2) pak dávají vyjádření

$$a_1 = 8k^4 - k \quad \text{a} \quad a_2 = k.$$

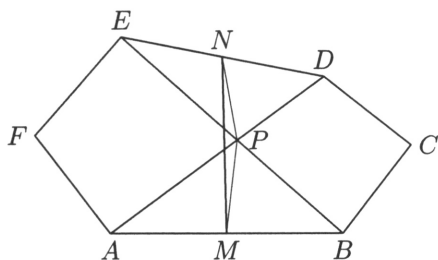
Protože obě vypočtené hodnoty $a_{1,2}$ jsou přirozená čísla (pro každé $k \in \mathbb{N}$), dostáváme dvě (nekonečné) skupiny řešení $(a, b) = (8k^4 - k, 2k)$ a $(a, b) = (k, 2k)$, kde $k \in \mathbb{N}$.

Zbývá rozebrat případ, kdy $b = 1$. Tehdy má zlomek ze zadání úlohy tvar

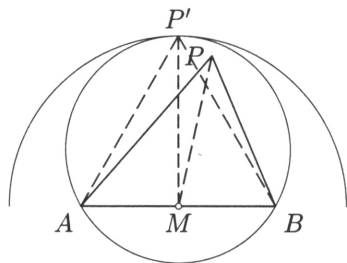
$$\frac{a^2}{2ab^2 - b^3 + 1} = \frac{a^2}{2a} = \frac{a}{2},$$

takže je roven přirozenému číslu, právě když $a = 2k$ pro vhodné $k \in \mathbb{N}$. Třetí (a poslední) skupinou řešení jsou tedy dvojice tvaru $(a, b) = (k, 1)$, kde $k \in \mathbb{N}$.

3. Označme A, B, C, D, E, F vrcholy daného šestiúhelníku a uvažujme tři úhlopříčky AD, BE a FC . Některé dvě z nich nutně svírají úhel aspoň 60° . Bez újmy na obecnosti předpokládejme, že jsou to úhlopříčky AD a BE . Označme P jejich průsečík a M, N středy protějších stran AB a DE (obr. 37).



Obr. 37



Obr. 38

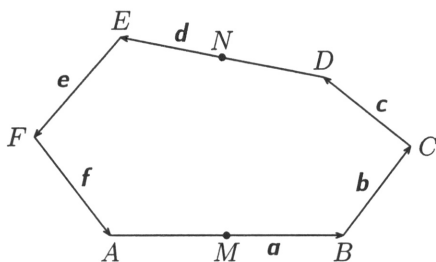
Protože $|\sphericalangle APB| = |\sphericalangle EPD| \geq 60^\circ$, leží bod P uvnitř kružnice opsané rovnostrannému trojúhelníku ABP' (obr. 38), takže platí $|MP| \leq |MP'| = \frac{1}{2}\sqrt{3}|AB|$ s rovností, právě když $P = P'$, neboli právě když je trojúhelník ABP rovnostranný. Podobně odvodíme, že $|NP| \leq \frac{1}{2}\sqrt{3}|DE|$ s rovností, právě když je trojúhelník DEP rovnostranný. Pro vzdálenost středů obou protějších stran AB, DE tak dostáváme odhad

$$|MN| \leq |MP| + |NP| \leq \frac{1}{2}\sqrt{3}(|AB| + |DE|).$$

Z předpokladů úlohy tedy plyne, že oba trojúhelníky ABP a DEP jsou rovnostranné a úhlopříčky AD, BE svírají úhel 60° .

Zbývající úhlopříčka CF musí s jednou z úhlopříček AD, BE svírat úhel aspoň 60° . Opět můžeme bez újmy na obecnosti předpokládat, že se jedná např. o úhlopříčku AD , průsečík úhlopříček CE, AD označme Q . Úplně stejně jako v předchozím případě zjistíme, že trojúhelníky FAQ a CDQ jsou rovnostranné. Označíme-li nakonec R průsečík úhlopříček BE a CF , které dle předchozího svírají nutně úhel 60° , zjistíme, že i trojúhelníky BCR a EFR jsou rovnostranné. Odtud plyne tvrzení úlohy.

Jiné řešení. Označme A, B, C, D, E, F vrcholy daného šestiúhelníku a $\mathbf{a} = \overrightarrow{AB}, \mathbf{b} = \overrightarrow{BC}, \dots, \mathbf{f} = \overrightarrow{FA}$ vektory určené jeho stranami, přitom $\mathbf{a} + \mathbf{b} + \dots + \mathbf{f} = \mathbf{0}$. Označíme-li M, N středy protějších stran AB, DE ,



Obr. 39

můžeme příslušný vektor $\mathbf{MN}$ vyjádřit dvěma způsoby (obr. 39):

$$\mathbf{MN} = \frac{1}{2}\mathbf{a} + \mathbf{b} + \mathbf{c} + \frac{1}{2}\mathbf{d} \quad \text{a} \quad \mathbf{MN} = -\frac{1}{2}\mathbf{a} - \mathbf{f} - \mathbf{e} - \frac{1}{2}\mathbf{d},$$

odkud

$$\mathbf{MN} = \frac{1}{2}(\mathbf{b} + \mathbf{c} - \mathbf{e} - \mathbf{f}). \quad (1)$$

Podle předpokladu platí

$$|\mathbf{MN}| = \frac{\sqrt{3}}{2}|\mathbf{a} + \mathbf{d}| \geq \frac{\sqrt{3}}{2}|\mathbf{a} - \mathbf{d}|. \quad (2)$$

Položme $\mathbf{x} = \mathbf{a} - \mathbf{d}$, $\mathbf{y} = \mathbf{c} - \mathbf{f}$, $\mathbf{z} = \mathbf{e} - \mathbf{b}$, z (1) a (2) tak dostaneme

$$|\mathbf{y} - \mathbf{z}| \geq \sqrt{3}|\mathbf{x}|$$

a podobně

$$|\mathbf{z} - \mathbf{x}| \geq \sqrt{3}|\mathbf{y}|,$$

$$|\mathbf{x} - \mathbf{y}| \geq \sqrt{3}|\mathbf{z}|.$$

Právě uvedené nerovnosti můžeme pomocí skalárních součinů ekvivalentně přepsat jako

$$|\mathbf{y}|^2 - 2(\mathbf{y}, \mathbf{z}) + |\mathbf{z}|^2 \geq 3|\mathbf{x}|^2,$$

$$|\mathbf{z}|^2 - 2(\mathbf{z}, \mathbf{x}) + |\mathbf{x}|^2 \geq 3|\mathbf{y}|^2,$$

$$|\mathbf{x}|^2 - 2(\mathbf{x}, \mathbf{y}) + |\mathbf{y}|^2 \geq 3|\mathbf{z}|^2.$$

Sečtením všech tří nerovností vyjde

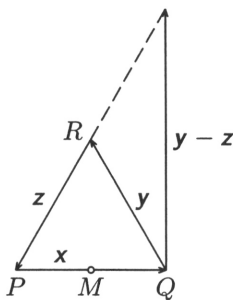
$$-|\mathbf{x}|^2 - |\mathbf{y}|^2 - |\mathbf{z}|^2 - 2(\mathbf{y}, \mathbf{z}) - 2(\mathbf{z}, \mathbf{x}) - 2(\mathbf{x}, \mathbf{y}) \geq 0,$$

neboli $-|x + y + z|^2 \geq 0$. Odtud ovšem plyne, že $x + y + z = 0$ a že ve všech předchozích nerovnostech platí rovnost. Je tedy jednak

$$\begin{aligned} |y - z| &= \sqrt{3}|x|, \\ |z - x| &= \sqrt{3}|y|, \\ |x - y| &= \sqrt{3}|z|, \end{aligned}$$

jednak díky rovnosti v (2) a v dalších dvou analogických nerovnostech i $a \parallel d \parallel x$, $c \parallel f \parallel y$, $e \parallel b \parallel z$.

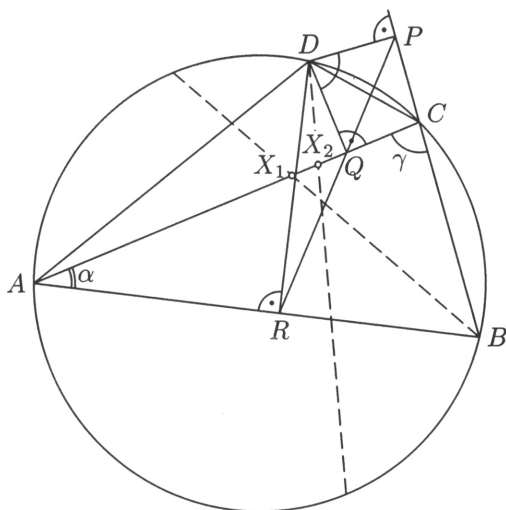
Sestrojíme-li trojúhelník PQR tak, že $PQ = x$, $QR = y$, $RP = z$ (což můžeme díky rovnosti $x + y + z = 0$), bude některý z jeho vnitřních úhlů mít velikost aspoň 60° . Nechť je to např. úhel PRQ (obr. 40). Pro střed M strany PQ pak platí $|MR| = \frac{1}{2}|y - z| = \frac{1}{2}\sqrt{3}|x| = \frac{1}{2}\sqrt{3}|PQ|$, což znamená, že trojúhelník PQR je rovnostranný. Pro vnitřní úhly daného šestiúhelníku to vzhledem k dokázané rovnoběžnosti jeho protějších stran s odpovídajícími vektory x , y a z znamená, že všechny jeho vnitřní úhly mají velikost 120° .



Obr. 40

Poznámka. Z uvedeného řešení je zřejmé, že libovolný šestiúhelník splňující předpoklady úlohy dostaneme tak, že z některého rovnostranného trojúhelníku „odřízneme“ při každém jeho vrcholu shodný rovnostranný trojúhelník.

4. Označme po řadě X_1 , X_2 průsečíky osy úhlu ABC a osy úhlu ADC s úhlopříčkou AC daného tětíivového čtyřúhelníku $ABCD$ (obr. 41). Ze známé vlastnosti osy úhlu plyne jednak $|AX_1|/|CX_1| = |AB|/|CB|$ (v trojúhelníku ABC), jednak $|AX_2|/|CX_2| = |AD|/|CD|$ (v trojúhelníku ACD). Osy úhlů ABC a ADC se tedy protnou na úhlopříčce AC , právě když $X_1 = X_2$, neboli právě když $|AD| \cdot |CB| = |AB| \cdot |CD|$.



Obr. 41

Podle Thaletovy věty leží paty P a Q na kružnici s průměrem CD , takže pro velikost tětiny PQ této kružnice platí

$$|PQ| = |CD| \sin |\sphericalangle PCQ| = |CD| \sin \gamma,$$

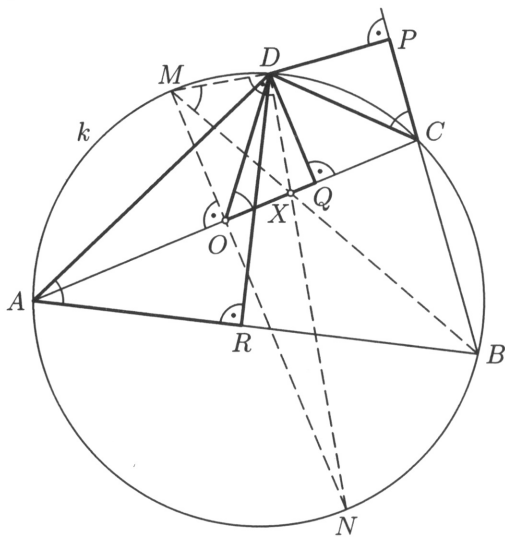
kde $\gamma = \sphericalangle ACB$ (bez ohledu na to, zda pata P padne dovnitř strany BC či nikoli). Podobně leží paty R a Q na kružnici s průměrem AD , takže pro velikost tětiny QR této kružnice platí

$$|QR| = |AD| \sin |\sphericalangle RAQ| = |AD| \sin \alpha,$$

kde $\alpha = \sphericalangle BAC$. Vidíme tedy, že rovnost $|PQ| = |QR|$ je ekvivalentní rovnosti $|CD| \sin \gamma = |AD| \sin \alpha$, což je podle sinové věty pro trojúhelník ABC ekvivalentní s rovností $|AD| \cdot |CB| = |AB| \cdot |CD|$. Tím je tvrzení úlohy dokázáno.

Jiné řešení. (Podle *Marka Krčála*, bohužel až po soutěži.) Označme M a N body, v nichž osa úhlu ABC , resp. osa úhlu ADC protne kružnici k opsanou danému tětívkovému čtyřúhelníku $ABCD$ (obr. 42). Vzhledem k tomu, že každý z bodů M , N pólí příslušný oblouk AC kružnice k , je MN osou úhlopříčky AC a zároveň průměrem kružnice k . Označme O střed úsečky AC . Bez újmy na obecnosti předpokládejme, že úhel BAD není tupý (jinak bychom prohodili označení vrcholů A a C), takže pata P

kolmice z bodu D na BC padne mimo úsečku BC . Z vlastností tětiového čtyřúhelníku plyne $|\sphericalangle DCP| = |\sphericalangle BAD|$ a z rovnosti obvodových úhlů nad tětivou BD rovnost $|\sphericalangle BMD| = |\sphericalangle BAD|$. Jsou tedy trojúhelníky ARD a CPD podobné.



Obr. 42

Předpokládejme, že průsečík X obou zmíněných os úhlů leží na přímce AC . Protože MN je průměr kružnice k , je podle Thaletovy věty $|\sphericalangle XDM| = |\sphericalangle NDM| = 90^\circ$, takže čtyřúhelník $OXDM$ je tětiový. Je tedy také $|\sphericalangle XOD| = |\sphericalangle XMD| = |\sphericalangle BMD|$ a vidíme, že trojúhelník OQD je podobný trojúhelníkům ARD a CPD . Uvažujme spirální podobnost, jež vznikne složením otočení kolem středu D o úhel $90^\circ - |\sphericalangle BAD|$ a stejnoolehlosti se středem D a koeficientem $|DR|/|DA|$. Tato podobnost zobrazí bod A do bodu R , bod C do bodu P a bod O do bodu Q . Protože O je střed úsečky AC , je jeho obraz v této podobnosti, tedy bod Q středem úsečky PR , jež je obrazem úsečky AC .

Obráceně, je-li Q střed úsečky PR , je obrazem bodu O v uvedené podobnosti, takže trojúhelník OQD je podobný trojúhelníkům ARD a CPD (ty jsou podobné vždy). Označíme-li nyní jako X průsečík přímky BM s úhlopříčkou AC , bude $XDMO$ tětiový, a tudíž velikost úhlu XDM bude 90° . Odtud plyne, že bod X leží na ND , ose úhlu ADC .

5. Obě strany dokazované nerovnosti nezmění hodnotu, když ode všech členů x_i odečteme totéž číslo c . Vybereme-li za c aritmetický průměr dané n -tice členů x_i , bude „posunutá“ posloupnost členů $x_i := x_i - c$ splňovat podmínku

$$\sum_{i=1}^n x_i = 0. \quad (1)$$

Dodejme, že zmíněné „posunutí“ zachová rovněž uspořádání čísel x_i podle velikosti a nezmění ani nic na tom, zda dotyčná n -tice tvořila aritmetickou posloupnost či nikoliv.

Za předpokladu (1) upravíme oba součty z dokazované nerovnosti:

$$\begin{aligned} \sum_{i=1}^n \sum_{j=1}^n |x_i - x_j| &= 2 \sum_{1 \leq i < j \leq n} (x_j - x_i) = \\ &= 2 \sum_{i=1}^n \underbrace{((1 + \dots + 1))}_{(i-1) \text{ krát}} - \underbrace{(1 + \dots + 1))}_{(n-i) \text{ krát}} x_i = \\ &= 2 \sum_{i=1}^n (2i - n - 1)x_i, \\ \sum_{i=1}^n \sum_{j=1}^n (x_i - x_j)^2 &= n \sum_{i=1}^n x_i^2 - 2 \sum_{i=1}^n x_i \sum_{j=1}^n x_j + n \sum_{j=1}^n x_j^2 = \\ &= 2n \sum_{i=1}^n x_i^2. \end{aligned}$$

Po dosazení a krácení čtyřmi zjistíme, že máme dokázat nerovnost

$$\left(\sum_{i=1}^n (2i - n - 1)x_i \right)^2 \leq \frac{(n^2 - 1)n}{3} \sum_{i=1}^n x_i^2. \quad (2)$$

Ukažme, že (2) je Cauchyova nerovnost

$$\left(\sum_{i=1}^n x_i y_i \right)^2 \leq \sum_{i=1}^n x_i^2 \sum_{i=1}^n y_i^2 \quad (3)$$

pro n -tici členů $y_i = 2i - n - 1$, $i = 1, 2, \dots, n$. Skutečně, pro takovou n -tici platí

$$\begin{aligned}\sum_{i=1}^n y_i^2 &= \sum_{i=1}^n (2i - n - 1)^2 = 4 \sum_{i=1}^n i^2 - 4(n+1) \sum_{i=1}^n i + n(n+1)^2 = \\ &= 4 \cdot \frac{n(n+1)(2n+1)}{6} - 4(n+1) \cdot \frac{n(n+1)}{2} + n(n+1)^2 = \\ &= \frac{(n^2 - 1)n}{3}.\end{aligned}$$

Tím je důkaz nerovnosti (2) hotov.

Jak je dobře známo, rovnost v Cauchyově nerovnosti (2) nastane, právě když existuje reálné číslo p , pro které platí n -tice rovností

$$x_i = py_i = p(2i - n - 1) \quad (i = 1, 2, \dots, n). \quad (4)$$

Ověřme, že tuto podmínku za předpokladu (1) splňují právě ty konečné posloupnosti $x_1, x_2, \dots, x_n$, jež jsou aritmetické. Skutečně, platí-li rovnost (4), je konečná posloupnost $x_1, x_2, \dots, x_n$ aritmetická s diferencí $2p$. Obráceně, je-li posloupnost $x_1, x_2, \dots, x_n$ aritmetická a značí-li d její diferenci, pak pro každé $i = 1, 2, \dots, n$ platí rovnost $x_i = x_1 + (i - 1)d$ a součet všech členů x_i je dán vzorcem

$$\sum_{i=1}^n x_i = \frac{n(x_1 + x_n)}{2}.$$

Podmínka (1) tudíž znamená, že $x_1 + x_n = 0$, neboli $x_1 + x_1 + (n - 1)d = 0$. Odtud dostáváme $x_1 = \frac{1}{2}d(1 - n)$, proto členy x_i mají pro každé i vyjádření

$$x_i = \frac{d(1 - n)}{2} + (i - 1)d = \frac{(1 - n + 2i - 2)d}{2} = \frac{(2i - n - 1)d}{2},$$

což je (4) pro $p = d/2$.

6. Připomeňme nejdříve vlastnosti mocnin $n^1, n^2, \dots, n^k, \dots$ při dělení prvočíslem q : je-li n celé číslo nesoudělné s q , pak $n^{q-1} \equiv 1 \pmod{q}$ (tzv. *malá Fermatova věta*), navíc množina těch přirozených k , pro která $n^k \equiv 1 \pmod{q}$, je tvořena všemi násobky nejmenšího z nich (což je buď číslo $q - 1$, nebo některý jeho dělitel).

Uvažujme proto rozklad

$$p^p - 1 = (p - 1)S, \quad \text{kde } S = p^{p-1} + p^{p-2} + \dots + p + 1, \quad (1)$$

a za „kandidáta“ na vhodné prvočíslo q vyberme některé z prvočísel dělicích součet S (v pravou chvíli upřesníme, jakou doplňující vlastnost prvočinitele q čísla S budeme ještě potřebovat a proč takové q vůbec existuje). Protože $q \mid S$ a $S \mid (p^p - 1)$, platí $q \mid (p^p - 1)$, tj. $p^p \equiv 1 \pmod{q}$.

Připustíme, že pro vybrané q tvrzení úlohy neplatí, tedy existuje celé n s vlastností $n^p \equiv p \pmod{q}$. Umocněním této kongruence na p dostaneme $n^{p^2} \equiv p^p \pmod{q}$, což spolu s kongruencí ze závěru předchozího odstavce znamená, že $n^{p^2} \equiv 1 \pmod{q}$. Číslo n je tedy nesoudělné s číslem q a podle poznatků připomenutých v úvodu řešení víme, že nejmenší přirozené k s vlastností $n^k \equiv 1 \pmod{q}$ musí být dělitel čísla p^2 , tedy jedno z čísel $1, p, p^2$. Toto číslo musí být zároveň dělitelem čísla $q - 1$ (malá Fermatova věta), takže to nebude číslo p^2 , nebude-li číslo p^2 dělit číslo $q - 1$, tedy pokud $q \not\equiv 1 \pmod{p^2}$. To je právě ona doplňující vlastnost prvočísla q , o které jsme se dříve zmínili; odložme na chvíli důkaz existence takového prvočísla q a dokončeme úvahy o mocninách čísla n .

Pokud tedy $q \not\equiv 1 \pmod{p^2}$, platí kongruence $n^k \equiv 1 \pmod{q}$ pro $k = 1$ nebo pro $k = p$, v obou případech máme $n^p \equiv 1 \pmod{q}$. Porovnáním s kongruencí $n^p \equiv p \pmod{q}$ pak dostaneme $p \equiv 1 \pmod{q}$, takže každá z p mocnin p^j ze součtu S je kongruentní s číslem 1 (modulo q), tudíž $S \equiv p \pmod{q}$. Protože však $q \mid S$, platí $S \equiv 0 \pmod{q}$. Porovnáním vychází $p \equiv 0 \pmod{q}$, což je spor s tím, že $p \equiv 1 \pmod{q}$. Proto žádné celé n s vlastností $n^p \equiv p \pmod{q}$ neexistuje, splňuje-li prvočíslo q podmínku $q \not\equiv 1 \pmod{p^2}$. Existenci takového prvočinitele q (z rozkladu čísla S) nyní dokážeme.

Určeme zbytek součtu S při dělení číslem p^2 : protože $p^2 \mid p^j$ ($j \geq 2$), platí

$$S = p^{p-1} + p^{p-2} + \dots + p + 1 \equiv 0 + 0 + \dots + 0 + p + 1 \pmod{p^2},$$

tedy $S \equiv p + 1 \not\equiv 1 \pmod{p^2}$. Odtud již plyne, že aspoň jeden z prvočinitelů q_j čísla $S = q_1 q_2 \dots q_r$ není kongruentní s 1 (modulo p^2). (Vynásobením r kongruencí $q_j \equiv 1 \pmod{p^2}$ bychom totiž dostali $S \equiv 1 \pmod{p^2}$.)

Důkaz je hotov a úloha vyřešena.

Desátý ročník Středoevropské olympiády v informatice

Ve dnech 5.–12. července 2003 proběhl v německém Münsteru jubilejní 10. ročník Středoevropské olympiády v informatice (Central European Olympiad in Informatics, CEOI 2003).

Vedoucím české reprezentace byl jmenován RNDr. *Tomáš Pitner*, Dr., z Masarykovy univerzity v Brně, Fakulty informatiky. Úloha zástupce vedoucího týmu byla svěřena Mgr. *Martinu Marešovi* z Univerzity Karlovy v Praze, Matematicko-fyzikální fakulty.

Kromě toho byl z rozhodnutí řídicího výboru soutěže organizátory za českou stranu pozván doc. RNDr. Václav Sedláček, CSc., dlouholetý vedoucí českých reprezentací na mezinárodních a středoevropských olympiádách v informatice a dalších soutěžích, prezident (předseda organizačního výboru) CEOI v roce 1999 v Brně a spoluzakladatel iniciativy vedoucí k založení tradice Středoevropských olympiád v informatice. Pozvání bylo vyjádřením uznání dlouholetých zásluh doc. Sedláčka o podporu mladých talentů v informatice a současně potvrzením stabilní pozice České republiky mezi zeměmi střední Evropy, které jsou schopny a ochotny podobné akce organizovat.

Vlastní soutěž probíhala regulérně podle pravidel CEOI za maximální podpory především Spolkového ministerstva školství, vědy a výzkumu, Ministerstva školství spolkové země Nordrhein-Westfalen a složek místní a regionální veřejné správy (Bezirks Regierung Münster). Dalšími hlavními sponzory byly Gesellschaft für Informatik e.V., Fraunhofer Gruppe Informations- und Kommunikationstechnik, sd&m a Siemens Business Services (SBS). Hlavním výkonným organizátorem bylo sdružení Spolková soutěž v informatice (Bundewettbewerb Informatik) vedené Dr. Wolfgangem Pohlem, který byl rovněž prezidentem letošního ročníku CEOI. Na organizaci se dále podílely střední školy z Münsteru a okolí, jejichž učitelé působili v organizačním výboru a studenti/studentky jako průvodci (guides) jednotlivých týmů. Podstatnou roli v odborné části programu sehrál Scientific Committee, tvořený převážně bývalými aktivními účastníky podobných mezinárodních soutěží z hostitelské země.

Vlastní soutěž se konala na gymnáziu Paulinum v Münsteru. Hmotné i personální zabezpečení akce bylo na velmi vysoké úrovni.

Stejně jako v minulém roce bylo při vlastní soutěži (po vzoru IOI) používáno modernizované vývojové a testovací prostředí. Soutěžící pracovali ve třicetidvoubitových prostředích Windows 98 nebo Linux s překladači GNU C++ (gcc) a Free Pascal s vývojovým prostředím RHIDE. Novinkou letošní CEOI bylo použití webového systému pro zasílání řešení k vyhodnocení. Vyhodnocení se provádělo plně automaticky „testovačem“ běžícím v OS Debian Linux pomocí předem připravených sad testovacích dat. Proces vyhodnocení proběhl korektně, maximálně objektivně a bez problémů.

Reprezentační tým ČR na základě výsledků letošního celostátního kola kategorie P tvořili (v abecedním pořadí): *Sven Dražan* a *Kryštof Hoder*, oba z Gymnázia Brno, tř. Kpt. Jaroše; *Daniel Marek* z Gymnázia Ch. Dopplera Praha a *Jiří Schejbal* z Gymnázia Turnov. Členové českého týmu nezískali žádnou z medailí a obsadili místa v poslední třetině startovního pole.

15. mezinárodní olympiáda v informatice

Hostitelem 15. mezinárodní olympiády v informatice IOI 2003 byly Spojené státy americké. Soutěž se uskutečnila ve dnech 16.–23. 8. 2003 v univerzitním kampusu University of Wisconsin-Parkside nedaleko města Kenosha. Organizátoři zvolili termín konání v době univerzitních prázdnin, takže pro potřeby olympiády mohli využít dočasně vyprázdněné studentské ubytovny a jídelny, univerzitní posluchárny pro vlastní soutěž i pro různá probíhající jednání, aulu pro slavnostní zahájení a zakončení soutěže, ale také rozsáhlá sportoviště pro aktivní využití volného času všech účastníků.



Olympiáda byla výborně připravena po stránce organizační, po stránce počítačového vybavení i z hlediska kvality přípravy soutěžních úloh. V každém ze dvou soutěžních dnů řešili soutěžící studenti u počítačů tři poměrně náročné příklady. Se svými počítači i jejich softwarovým vybavením se přitom všichni mohli podrobně seznámit den před vlastní soutěží, kdy probíhalo neoficiální tréninkové předkolo. Při soutěži bylo možné programovat v některém z programovacích jazyků Pascal, C nebo C++, každý si mohl zvolit podle svých předchozích zkušeností pracovní prostředí operačních systémů Windows nebo Linux. O oba uvedené systémy byl mezi účastníky přibližně stejný zájem.

K testování a hodnocení vytvořených programů se na IOI již řadu let používá automatické vyhodnocování pomocí připravené sady vstupních dat. Všechny prováděné testy mají dobu výpočtu omezenu předem známým časovým limitem a jednotlivá testovací vstupní data mají různou velikost a různou složitost, což dohromady umožňuje bodově rozlišit programy podle kvality použitého algoritmu. Za každou úlohu lze získat maximálně 100 bodů, nejčastěji bývá při hodnocení zadáno 20 sad testovacích dat 5 bodech. U některých soutěžních úloh se hodnotí také dosažení optimality nalezeného řešení. Za nalezení nejlepšího možného řešení soutěžící dostane pro daná testovací data plný počet bodů, zatímco o něco horší řešení je hodnoceno méně body podle předem známého klíče (v hodnocení některých úloh se tak objevují i desetiny bodů).

Patnácté mezinárodní olympiády v informatice se zúčastnilo 265 soutěžících studentů ze 69 zemí celého světa. Počtem zúčastněných zemí tak IOI již dostihla ostatní mezinárodní předmětové olympiády středoškóláků a její velikost a význam rok od roku stále narůstá. Českou republiku reprezentovalo čtyřčlenné družstvo ve složení *Pavel Čížek* (absolvent Dvořákova gymnázia v Kralupech nad Vltavou), *Tomáš Gavenčiak* (student gymnázia M. Koperníka v Bílovci), *Jan Kadlec* (absolvent gymnázia Ch. Dopplera v Praze 5) a *Milan Straka* (absolvent gymnázia ve Strakonicih). Naši soutěžící byli vybráni na základě výsledků dosažených v celostátním kole kategorie P (programování) 52. ročníku Matematické olympiády. Vedením družstva byli pověřeni doc. *Pavel Töpfer* a RNDr. *Daniel Král*, oba z Matematicko-fyzikální fakulty Univerzity Karlovy v Praze.

Na mezinárodní olympiádě v informatice se udělují ocenění podle podobného klíče, jaký se používá například také na mezinárodní matematické olympiádě. Nejvýše polovina soutěžících obdrží některou z medailí, přičemž zlaté, stříbrné a bronzové medaile se dělí přibližně v poměru 1 : 2 : 3. Na letošní IOI 2004 bylo rozděleno celkem 132 medailí, z toho 24 zlatých, 45 stříbrných a 63 bronzových. O velké vyrovnanosti světové špičky svědčí skutečnost, že o udělených 24 zlatých medailí se podělili reprezentanti z 19 zemí, mezi nimi i naši. Pouze pět letos nejúspěšnějších států získalo po dvou zlatých medailích (Korea, USA, Rumunsko, Švédsko a Polsko), žádná země nemá tři zlaté medaile.

Naši studenti si vedli v soutěži velmi dobře, získali jednu zlatou, jednu stříbrnou a jednu bronzovou medaili. Podobného výsledku, tj. jedné zlaté, jedné stříbrné a jedné bronzové medaile, dosáhla také reprezentace Slovenska. Mezinárodní olympiáda v informatice je podle svých stanov soutěží jednotlivců, žádné oficiální pořadí zúčastněných zemí není vyhlašováno a není ani stanoveno, podle jakého kritéria by se mělo takové pořadí určovat (zda podle počtu získaných medailí, součtu bodů všech reprezentantů příslušné země či například podle součtu pořadí ve výsledkové listině). Při jakémkoliv způsobu počítání by se naše výprava umístila kolem 12.–13. místa, což je velmi dobrý výsledek. Následující tabulka shrnuje výsledky všech českých studentů v soutěži:

14.	Milan Straka	375,5 bodů	zlatá
43.	Pavel Čížek	306,1 bodů	stříbrná
111.	Tomáš Gavenčiak	195,0 bodů	bronzová
146.	Jan Kadlec	162,5 bodů	–

Příští, v pořadí šestnáctá mezinárodní olympiáda v informatice IOI 2004 se uskuteční v Athénách v první polovině září 2004. Pořadatelé z Řecka již nyní pozvali všechny země zúčastněné na IOI 2003, aby se zúčastnily i příštího ročníku soutěže. V roce 2004 se bude konat také 11. ročník Středoevropské olympiády v informatice CEOI 2004, a to v první polovině července ve Varšavě. Rovněž od polských pořadatelů jsme obdrželi pozvání k účasti.

Texty soutěžních úloh

1. Wisconsinské krávy (interaktivní úloha)

Krávy farmáře Johna se volně pohybují mezi N ($1 \leq N \leq 200$) pastvinami, které jsou očíslovány od 1 do N . Pastviny jsou navzájem odděleny lesem. Krávy udržují systém cest mezi pastvinami tak, aby kdykoliv bylo možné přejít po udržovaných cestách mezi libovolnými dvěma pastvinami. Po všech cestách je možné chodit oběma směry.

Krávy ve skutečnosti cesty samy nevytvářejí. Místo toho používají stezky lesní zvěře. Pro každý týden si vyberou některé ze stezek, které znají, a ty pak udržují jako cesty mezi pastvinami.

Krávy jsou zvířata od přírody velmi zvědavá. Na začátku každého týdne objeví jednu novou stezku. Ze stezek, které znaly z dřívějšíka, a z nově objevené stezky, pak vyberou množinu stezek, které budou v nadcházejícím týdnu udržovat jako cesty. Výběry stezek pro jednotlivé týdny jsou navzájem zcela nezávislé.

Stezku, která je udržována, mohou krávy ihned používat jako cestu mezi pastvinami. Pokud stezka přestane být udržována, není ji již nadále možné používat jako cestu. Krávy chtějí udržovat vždy takovou soustavu cest, aby součet délek udržovaných cest byl nejmenší možný.

Stezky lesní zvěře bývají klikaté. Proto může existovat více stezek různých délek, které spojují stejnou dvojici pastvin. I když se dvě cesty v lese kříží, krávy vždy pokračují v chůzi po původně zvolené cestě.

Váš úkolem je pro každý týden určit nejmenší možný součet délek udržovaných cest. Váš program bude postupně dostávat informace o nových stezkách objevených krávy. Po načtení popisu každé nové stezky musí váš program vypsát optimální součet délek udržovaných cest.

Vstup: standardní vstup (standard input)

- ▷ První řádek obsahuje dvě celá čísla N a W oddělená jednou mezerou. N určuje počet pastvin ($1 \leq N \leq 200$) a W udává počet týdnů, v nichž činnost krav sledujeme ($1 \leq W \leq 6\,000$).

- ▷ Pro každý týden pak následuje samostatný řádek, který popisuje nově objevenou stezku. Tento řádek je tvořen třemi celými čísly oddělenými jednou mezerou, která udávají čísla pastvin spojených novou stezkou a její délku (1 . . . 10 000). Každá stezka spojuje dvě různé pastviny.

Výstup: standardní výstup (standard output)

Popis další stezky nelze načíst, dokud váš program nevypíše řešení pro současnou množinu stezek. Pro každý týden, váš program vypíše jedno celé číslo na samostatném řádku: Toto číslo udává nejmenší možný součet délek cest, které je třeba udržovat, aby byla zachována propojenost všech pastvin. Pokud takový systém cest neexistuje, program vypíše číslo -1.

Program musí skončit po vypsání řešení pro poslední sledovaný týden.

Příklad komunikace:

<i>Vstup</i>	<i>Výstup</i>	<i>Vysvětlení</i>
4 6 1 2 10	-1	Žádná stezka nespojuje pastvinu č. 4 s ostatními pastvinami.
1 3 8	-1	Žádná stezka nespojuje pastvinu č. 4 s ostatními pastvinami.
3 2 3	-1	Žádná stezka nespojuje pastvinu č. 4 s ostatními pastvinami.
1 4 3	14	Systém cest je tvořen stezkami 1 4 3, 1 3 8 a 3 2 3.
1 3 6	12	Systém cest je tvořen stezkami 1 4 3, 1 3 6 a 3 2 3.
2 1 2	8	Systém cest je tvořen stezkami 1 4 3, 2 1 2 a 3 2 3.
<i>konec programu</i>		

<i>Omezení:</i>	Časový limit	1 s CPU
	Paměťový limit	64 MB

Hodnocení. Za každý testovací vstup obdržíte plný počet bodů, pokud váš program vypíše správný výstup. V opačném případě je testovací vstup hodnocen 0 body.

2. Ukradený kód

Společnost Racine Business Networks (RBN) se rozhodla zažalovat společnost Heuristic Algorithm Languages (HAL). RBN tvrdí, že HAL

vykradla část jejího zdrojového kódu RBN UNIXTM a začlenila ho do svého operačního systému HALnix.

Obě společnosti RBN a HAL používají stejný programovací jazyk. Každá instrukce je uvedena na samostatném řádku a všechny mají jednotný formát: $\text{STOREA} = \text{STOREB} + \text{STOREC}$, kde STOREA , STOREB a STOREC jsou jména proměnných. Jméno první proměnné začíná v prvním sloupci, pak následuje jedna mezera, rovnítko a další mezera. Poté je uvedeno jméno druhé proměnné následované jednou mezerou, plusem a další mezerou. Řádek je ukončen jménem třetí proměnné. Jedna proměnná se může na téže řádce vyskytovat několikrát. Jména proměnných jsou tvořena 1 až 8 velkými písmeny anglické abecedy (A, ..., Z).

RBN tvrdí, že programátoři společnosti HAL okopírovali souvislé kusy jejího kódu a provedli pouze následující změny, aby zamaskovali svůj odporný zločin: Programátoři HAL vždy vzali několik po sobě následujících řádků z kódu firmy RBN a v něm změnili jména některých proměnných. Nikdy se však nestalo, že by dvě různé proměnné měly po přejmenování totéž jméno. Programátoři HAL také občas zaměnili pořadí sčítanců na pravé straně instrukce. Tedy instrukci $\text{STOREA} = \text{STOREB} + \text{STOREC}$ nahradili instrukcí $\text{STOREA} = \text{STOREC} + \text{STOREB}$. Pořadí jednotlivých instrukcí však zůstalo zachováno.

Vášim úkolem je v programu společnosti HAL najít nejdelší souvislý kus kódu, který mohl být vykraden z programu společnosti RBN výše popsaným způsobem. Odpovídající si kusy kódu mohou v každém z programů začínat na různých řádcích.

Vstupní soubor: code.in

- ▷ První řádek obsahuje dvě celá čísla R a H oddělená jednou mezerou ($1 \leq R \leq 1000$; $1 \leq H \leq 1000$). R udává počet řádků programu společnosti RBN a H programu společnosti HAL.
- ▷ Následujících R řádků obsahuje kód programu společnosti RBN.
- ▷ Dalších H řádků pak obsahuje kód programu společnosti HAL.

Příklad vstupního souboru: 4 3

```
RA = RB + RC
RC = D + RE
RF = RF + RJ
RE = RF + RF
HD = HE + HF
HM = HN + D
HN = HA + HB
```

Výstupní soubor: code.out

Výstupní soubor musí být tvořen jediným řádkem obsahujícím jedno celé číslo. Toto číslo udává počet řádků nejdelšího souvislého kusu kódu, který mohl být vykraden z programu společnosti RBN.

Příklad výstupního souboru: 2

Řádky 1 a 2 programu společnosti RBN odpovídají řádkům 2 a 3 programu společnosti HAL (RAHM, $RB \rightarrow D$, $RC \rightarrow HN$, $D \rightarrow HA$, $RE \rightarrow HB$). Žádné tři po sobě jdoucí řádky programu společnosti RBN neodpovídají třem řádkům programu společnosti HAL.

Omezení:	Časový limit	2 s CPU
	Paměťový limit	64 MB

Hodnocení. Za každý testovací vstup obdržíte plný počet bodů, pokud váš program vytvoří správný výstupní soubor. V opačném případě je testovací vstup hodnocen 0 body.

3. Posloupnost (open-data úloha)

Vášim úkolem je vytvořit program pro počítač TOM. Počítač TOM má 9 paměťových registrů, jejichž hodnoty lze nastavit na začátku výpočtu. Registry jsou očíslovány čísly od 1 do 9 a každý z nich může uchovávat jedno celé číslo z intervalu $0 \dots 1000$. Počítač má implementovány pouze následující dvě instrukce:

S i j	Do registru j přiřadí hodnotu registru i zvýšenou o 1. Čísla i a j mohou být stejná.
P i	Vytiskne na výstup hodnotu uloženou v registru i.

Program počítače TOM je tedy tvořen počátečním nastavením hodnot registrů a posloupností instrukcí. Vaším úkolem je vytvořit pro zadané celé číslo N ($0 \leq N \leq 255$) program, který vypíše posloupnost čísel $N, N-1, N-2, \dots, 0$. Snažte se, aby počet po sobě následujících S-instrukcí ve výsledném programu byl nejmenší možný.

Příklad programu pro počítač TOM pro $N = 2$ a průběh jeho výpočtu:

Instrukce	Obsah registrů									Výstup
	1	2	3	4	5	6	7	8	9	
Počáteční hodnoty	0	2	0	0	0	0	0	0	0	
P 2	0	2	0	0	0	0	0	0	0	2
S 1 3	0	2	1	0	0	0	0	0	0	
P 3	0	2	1	0	0	0	0	0	0	1
P 1	0	2	1	0	0	0	0	0	0	0

Testovací data jsou očíslována od 1 do 16. Vstupní soubory, které je obsahují, si můžete stáhnout ze soutěžního serveru.

Formát vstupních souborů:

- ▷ První řádek obsahuje jedno celé číslo K , které udává pořadové číslo testovacích dat.
- ▷ Druhý řádek obsahuje číslo N .

Příklad vstupního souboru:

1
2

Formát výstupních souborů:

První řádek výstupního souboru musí obsahovat řetězec „FILE reverse K “, kde K je pořadové číslo testovacích dat.

Druhý řádek má obsahovat 9 celých čísel navzájem oddělených mezerami. Tato čísla představují počáteční hodnoty uložené v registrech (první v registru 1, druhé v registru 2, atd.).

Zbytek výstupního souboru obsahuje kód programu pro počítač TOM. Každý řádek obsahuje právě jednu instrukci. Instrukce na posledním řádku programu by měla vytisknout na výstup číslo 0. Kód programu by měl být ve tvaru jako v následujících příkladech.

Příklad výstupu #1

(část bodů):

FILE reverse 1

0 2 0 0 0 0 0 0 0

P 2

S 1 3

P 3

P 1

Příklad výstupu #2

(plný počet bodů):

FILE reverse 1

0 2 1 0 0 0 0 0 0

P 2

P 3

P 1

Hodnocení. Počet bodů, které obdržíte za testovací data, závisí na správnosti a optimalitě odevzdaného řešení.

Správnost: 20 %

Program počítače TOM je korektní, pokud nevykoná více než 131 po sobě následujících S -instrukcí a zároveň na výstup vypíše postupně $N + 1$ čísel v pořadí od N do 0. Pokud během výpočtu programu některá S -instrukce způsobí přetečení registru, pak je program považován za nekorektní.

Optimalita: 80 %

Úkolem je minimalizovat největší počet po sobě následujících S -instrukcí. Optimalita odevzdaného programu je porovnávána s nejlepším programem pro daný testovací vstup, který je k dispozici.

4. Není kráva jako kráva (interaktivní úloha)

Farmář John chce ustájit svých N ($1 \leq N \leq 50$) krav. Krávy jsou očíslovány celými čísly od 1 do N . Bohužel jsou si navzájem velmi podobné a není jednoduché rozlišit je mezi sebou. Protože je každá kráva zvyklá na své místo ve stáji, musí být John schopný rychle je rozeznávat.

Krávy lze rozeznávat podle P ($1 \leq P \leq 8$) rozlišovacích znaků, např. podle barvy visačky v uchu. Jednotlivé rozlišovací znaky si pro přehlednost očíslováme čísly od 1 do P . Každý z rozlišovacích znaků nabývá u každé krávy jedné ze tří možných hodnot, které pro jednoduchost označme písmeny »X«, »Y« a »Z«. Můžete předpokládat, že každé dvě krávy se liší v aspoň jednom znaku.

Vášim úkolem je napsat program, který farmáři Johnovi pomůže rozpoznat zvolenou krávu z jeho stáda. Program může položit farmáři Johnovi nejvýše 100 otázek typu: „Patří hodnota rozlišovacího znaku s pořadovým číslem T do množiny S ?“, kde S je podmnožina množiny $\{X, Y, Z\}$. Na základě odpovědí na položené otázky váš program pak určí číslo krávy. Snažte se, aby počet otázek potřebných pro určení krávy byl co nejmenší.

Vstupní soubor: `guess.in`

- ▷ První řádek obsahuje dvě celá čísla N a P oddělená jednou mezerou. N ($1 \leq N \leq 50$) je počet krav a P ($1 \leq P \leq 8$) je počet rozlišovacích znaků.
- ▷ Každý z následujících N řádků popisuje jednu krávu ze stáda farmáře Johna. Druhý řádek souboru popisuje krávu s pořadovým číslem 1, třetí řádek krávu s číslem 2, atd. Každý z těchto řádků obsahuje P písmen navzájem oddělených vždy jednou mezerou. První písmeno na řádku udává hodnotu rozlišovacího znaku s pořadovým číslem 1, druhé znaku s číslem 2, atd.

Příklad vstupního souboru:

4	2
X	Z
X	Y
Y	X
Y	Y

Interaktivní komunikace: standardní vstup a výstup (standard input and output)

Komunikace vašeho programu s farmářem Johnem probíhá přes standardní vstup a výstup.

Program položí otázku vypsáním řádku na standardní výstup v následujícím formátu: Prvním znakem řádku je velké písmeno Q následované jednou mezerou, pořadovým číslem rozlišovacího znaku, na který se ptá, a jednou nebo více jeho hodnotami. Všechny hodnoty jsou odděleny na řádku mezerami. Např. „Q 1 Z Y“ reprezentuje otázku „Je hodnota prvního rozlišovacího znaku dané krávy z množiny $\{Z, Y\}$?“ Číslo rozlišovacího znaku musí být z intervalu $1 \dots P$. Každá z možných hodnot znaku může být uvedena v jedné otázce nejvýše jednou a musí být reprezentována jedním z písmen »X«, »Y« a »Z«.

Poté, co program položí otázku, načte odpověď ze standardního vstupu. Odpověď je reprezentována číslem 1 nebo 0: Číslo 1 znamená, že rozlišovací znak má jednu z hodnot uvedených v otázce, zatímco číslo 0 znamená opačnou odpověď.

Nakonec váš program vypíše řádek s pořadovým číslem krávy. Tento řádek musí začínat písmenem »C«, po kterém následuje jedna mezera a pořadové číslo krávy.

Příklad komunikace (pro výše uvedený vstupní soubor):

Vstup	Výstup	Vysvětlení
0	Q 1 X Z	Může být kráva 3 nebo 4. Je to kráva 4!
1	Q 2 Y	
	C 4	
program skončil		

Omezení:	Časový limit	1 s CPU
	Paměťový limit	64 MB

Hodnocení. Správnost: 30 % bodů

Za daný testovací vstup získáte body za korektnost, pokud váš program položí nejvýše 100 otázek, správně určí pořadové číslo krávy a v okamžiku ukončení výpočtu existuje jediná kráva, jejíž rozlišovací znaky jsou konzistentní s odpověďmi na otázky programu.

Počet položených otázek: 70 % bodů

Zbývající body získáte podle počtu položených otázek. Rozhodující je počet položených otázek v nejhorším případě (podle zákona schválnosti všechny případy budou ty nejhorší možné). Část bodů bude přidělena i řešení, jejichž počet otázek bude blízký optimu.

5. Roboti

Stali jste se (ne)šťastnými majiteli dvou robotů, kteří jsou zrovna umístěni ve dvou bludištích. Každé z bludišť je obdélníkového tvaru. Představujme si každé z nich jako čtverečkovou síť tvořenou jednotlivými poli. Pole se souřadnicemi $(1, 1)$ je umístěno v levém horním rohu bludiště.

V i -tém bludišti ($i = 1, 2$) se nachází G_i strážců ($0 \leq G_i \leq 10$), kteří se snaží chytit roboty. Každý ze strážců se stále pohybuje po přímé trase tam a zpět. Trasa strážce je tvořena několika sousedními poli bludiště. Vaším úkolem je napsat program, který nalezne posloupnost příkazů, které vyvedou oba roboty z bludiště, aniž by byli chyceni některým ze strážců.

Na začátku každé minuty vyšlete stejný příkaz oběma robotům. Příkaz udává jeden ze čtyř směrů: nahoru, dolů, doprava, doleva (north, south, east, west). Robot se pak posune v zadaném směru o jedno pole, pokud mu v pohybu nebrání zeď bludiště. Jestliže je v daném směru zeď bludiště, robot žádný pohyb v následující minutě nevykoná, ale instrukce je považována za korektní. Robot opustí bludiště, pokud vykoná krok vedoucí mimo obdélníkovou síť popisující bludiště. Po opuštění bludiště robot další příkazy ignoruje.

Každý ze strážců se na začátku každé minuty posune o právě jedno pole. Výchozí pozice a natočení strážců jsou zadány na vstupu programu. Strážce nejprve vykoná tolik kroků, kolik je počet polí jeho trasy zmenšený o jedna. Na posledním poli se strážce otočí a začne se pohybovat zpět ke svému výchozímu poli, kde se opět otočí o 180° a takto hlídkuje, dokud oba roboti neopustí svá bludiště.

Trasa každého strážce je zvolena tak, že neprotíná zeď bludiště a ani nevychází z bludiště ven. Trasy různých strážců se mohou protínat, ale jejich pohyb je zvolen tak, že se nikdy nesrazí, tj. nikdy nebudou dva strážci na konci některé minuty stát na stejném poli a ani si během některé z minut nevymění vzájemně své pozice. Počáteční pozice strážců jsou zvoleny tak, že se žádný z nich nenachází na poli, kde stojí robot.

Strážce chytí robota, pokud se nachází na konci některé minuty na stejném poli jako robot nebo když si během některé z minut s robotem vymění svou pozici.

Váš program obdrží popis dvou bludišť, každé o rozměrech nejvýše 20×20 polí, spolu s počátečními pozicemi robotů a trasami jednotlivých strážců. Úkolem programu je najít posloupnost instrukcí, společnou pro oba roboty, podle které oba roboti opustí bludiště, aniž by byli chyceni některým ze strážců. Snažte se, aby čas, kdy poslední robot opustí své

bludiště, byl co nejmenší. Čas, kdy první z robotů opustí bludiště, je nepodstatný. Minimalizujte tedy čas, který uplyne od začátku do okamžiku, kdy se oba roboti nacházejí mimo svá bludiště.

Vstupní soubor: robots.in

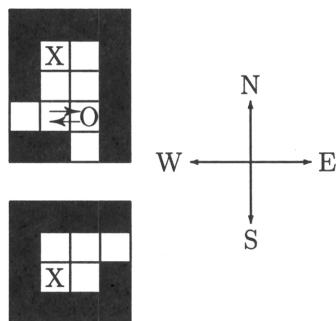
Vstupní soubor je tvořen dvěma částmi. První z nich popisuje první bludiště, pozici robota v něm a trasy strážců. Podobně druhá část souboru popisuje druhé bludiště.

- ▷ První řádek obsahuje dvě celá čísla R_1 a C_1 oddělená jednou mezerou. R_1 je počet řádků prvního bludiště a C_1 je počet jeho sloupců.
- ▷ Následuje R_1 řádků, z nichž každý obsahuje C_1 znaků. Tyto řádky tvoří mapu bludiště. Počáteční pozice robota je označena písmenem »X«, znak ».« představuje volné pole a znak »#« zeď. V popisu bludiště je právě jedno písmeno »X«.
- ▷ Další řádek obsahuje celé číslo G_1 ($0 \leq G_1 \leq 10$). Toto číslo udává počet strážců v prvním bludišti.
- ▷ Následuje G_1 řádků, z nichž každý popisuje počáteční pozici jednoho ze strážců. Každý z těchto řádků obsahuje tři celá čísla a jeden znak navzájem oddělené mezerami. První dvě čísla představují řádkovou a sloupcovou souřadnici strážce, třetí číslo počet polí jeho trasy a poslední znak počáteční nastavení směru pohybu strážce. Počet polí tvořících trasu strážce je 2, 3 nebo 4. Poslední znak na řádku je jedno z písmen »N«, »S«, »E«, »W« (north, south, east, west — nahoru, dolů, doprava, doleva).

Vstupní soubor poté obsahuje popis druhého bludiště ve stejném tvaru.

Příklad vstupního souboru: 5 4

```
####
#X.#
#..#
...#
##.#
1
4 3 2 W
4 4
####
#...
#X.#
####
0
```



Obr. 43

Výstupní soubor: robots.out

První řádek výstupního souboru musí být tvořen právě jedním celým číslem K ($K \leq 10\,000$), které udává délku nalezené posloupnosti instrukcí. Je zaručeno, že pokud existuje posloupnost instrukcí, která vyvede oba roboty z bludišť, potom existuje i taková posloupnost s nejvýše 10 000 instrukcemi. Následujících K řádků bude obsahovat vámi nalezenou posloupnost instrukcí. Každý z těchto řádků je tvořen jedním znakem z množiny »N«, »S«, »E«, »W«. Pokud žádná taková posloupnost instrukcí neexistuje, výstupní soubor bude obsahovat jediný řádek s číslem -1.

Oba roboti se musí po provedení posloupnosti instrukcí uvedené ve výstupním souboru nacházet mimo bludiště. Poslední instrukce nalezené posloupnosti musí být právě ta, po níž poslední robot (nebo oba najednou) opustí bludiště.

Pokud existuje více optimálních řešení, můžete vypsat jedno libovolné z nich.

Příklad výstupního souboru: 8

E
N
E
S
S
S
E
S

Omezení:	Časový limit	2 s CPU
	Paměťový limit	64 MB

Hodnocení. Za testovací vstup, pro který neexistuje řešení, lze získat pouze plný nebo nulový počet bodů. Ostatní testovací vstupy budou hodnoceny, jak je popsáno níže.

Správnost: 20 % bodů

Body za správnost získáte, pokud tvar výstupního souboru odpovídá popisu uvedenému v zadání této úlohy, nalezená posloupnost instrukcí má délku nejvýše 10 000, po jejím provedení se oba roboti nacházejí mimo svá bludiště a poslední instrukce nalezené posloupnosti způsobí, že alespoň jeden robot opustí své bludiště.

Optimalita: 80 % bodů

Body za optimalitu získáte, pokud lze výstup považovat za správný dle minulého odstavce a nalezená posloupnost instrukcí má nejmenší možnou délku. V opačném případě nezískáváte za optimalitu žádné body.

6. Ohrada

Farmář Dan se rozhlíží po ohradě kolem svého čtvercového pole o rozměrech $N \times N$ metrů ($2 \leq N \leq 500\,000$), jehož mapu máte k dispozici. Protilehlé rohy ohrady mají na mapě souřadnice $(0, 0)$ a (N, N) a hranice pole jsou rovnoběžné s X -ovou a Y -ovou osou.

Pletivo ohrady je upevněno na mnoha kůlech. V každém rohu je umístěn jeden kůl a podél každé strany pole jsou kůly umístěny vždy v rozestupech po 1 metru. Celkem tedy ohrada obsahuje $4N$ kůlů. Kůly jsou svislé a jejich průměr považujeme za nulový. Dan chce určit, kolik kůlů uvidí ze zvolené pozice uvnitř svého pole.

Pole si můžeme představovat jako rovinu, na které se nachází R ($1 \leq R \leq 30\,000$) skal, které Danovi omezují výhled. Skály mají podobu kolmých hranolů, jejichž podstavy mají tvar konvexních mnohoúhelníků. Skály jsou na poli umístěny tak, že stojí na své podstavě. Skály se navzájem neprotínají ani nedotýkají a ani se nedotýkají ohrady. Pozice farmáře Dana je zvolena tak, že se nedotýká ani jedné ze skal nebo ohrady a Dan ani na žádné skále nestojí.

Váš program obdrží rozměry pole, pozici farmáře Dana, umístění a tvar jednotlivých skal. Vaším úkolem je spočítat, kolik kůlů lze ze zadané pozice vidět. Farmář se rozhlíží do všech stran, tj. „vidí“ celý interval 360° . Nevidí však ty kůly, mezi nimiž a jím se nachází některá ze skal. Pokud vrchol podstavy skály leží přesně na spojnici kůlu a pozice farmáře Dana, předpokládáme, že Dan tento kůl také nevidí.

Vstupní soubor: `boundary.in`

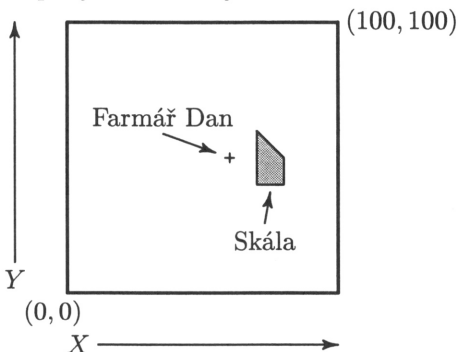
▷ První řádek souboru obsahuje dvě celá čísla N a R oddělená mezerou.

N ($2 \leq N \leq 500\,000$) je délka jedné strany pole a R ($1 \leq R \leq 30\,000$) je celkový počet skal, které se nacházejí na poli.

- ▷ Další řádek vstupního souboru obsahuje dvě celá čísla X a Y oddělená jednou mezerou. X a Y udávají souřadnice pozice farmáře Dana uvnitř jeho pole.
- ▷ Zbytek vstupního souboru tvoří popis R skal, které se nacházejí na poli:
 - ▷ Popis i -té skály začíná řádkem obsahujícím číslo p_i ($3 \leq p_i \leq 20$), které představuje počet vrcholů podstavy této skály.
 - ▷ Následuje p_i řádků, z nichž každý obsahuje dvě celá čísla X a Y oddělená jednou mezerou. Tato čísla udávají souřadnice vrcholů podstavy skály proti směru pohybu hodinových ručiček.

Příklad vstupního souboru:

```
100 1
60 50
5
70 40
75 40
80 40
80 50
70 60
```



Obr. 44

Povšimněte si, že podstava skály obsahuje tři kolineární vrcholy: $(70, 40)$, $(75, 40)$ a $(80, 40)$.

Výstupní soubor: `boundary.out`

Výstupní soubor musí být tvořen jedním řádkem obsahujícím jedno celé číslo. Toto číslo udává počet kúlů, které farmář vidí ze své pozice.

Příklad výstupního souboru: 319

Omezení:	Časový limit	1 s CPU
	Paměťový limit	64 MB

Hodnocení. Za každý testovací vstup obdržíte plný počet bodů, pokud váš program vytvoří správný výstupní soubor. V opačném případě je testovací vstup hodnocen 0 body.

RNDr. Karel Horák, CSc.,
RNDr. Tomáš Pitner, Dr., doc. RNDr. Jaromír Šimša, CSc.,
RNDr. Jaroslav Švrček, CSc., doc. RNDr. Pavel Töpfer, CSc.
RNDr. Jaroslav Zhouf, PhD.

**PADESÁTÝ DRUHÝ ROČNÍK
MATEMATICKÉ OLYMPIÁDY
NA STŘEDNÍCH ŠKOLÁCH**

Vydala Jednota českých matematiků a fyziků,
Žitná 25, 117 10 Praha 1
jako svou publikaci č. 57-551-04
Sazbu programem T_EX připravil Karel Horák
Vytisklo Polygrafické středisko
Univerzity Palackého v Olomouci
1. vydání
Praha 2004

ISBN 80-7015-960-X

matematická olympiáda

MO

ISBN 80-7015-960-X

ISBN 80-7015-960-X



9 788070 159606