$\alpha + \omega$, 2002, Nr. 2, 88–94

Aštuntosios Baltijos šalių informatikos olimpiados uždaviniai

Valentina Dagienė
dagiene@ktl.mii.lt

Straipsnyje pateikiami šeši varžybų uždaviniai. Jų testus galima rasti Lietuvos moksleivių informatikos olimpiadų svetainėje: http://aldona.mii.lt/pms/olimp/.

Olimpiados metu mokiniai turėjo išspręsti septynis uždavinius: vieną pasitreniravimui, susipažinimui su programine įranga ir šešis per varžybas (po tris per dvi dienas).

1. Trikampiai (Latvija)

Užduotis. Plokštumoje yra N stačiųjų lygiašonių trikampių. Kiekvieną trikampį nusako trys sveikieji skaičiai: x, y, m ($m > 0$). Trikampio viršūnės yra taškai $(x; y)$, $(x+m; y)$ ir $(x; y+m)$.

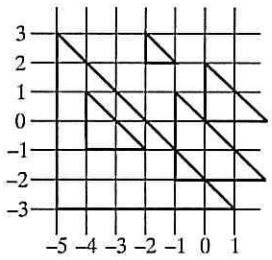
Parašykite programą, kuri suskaičiuotų bendrą visų trikampių užimamą plotą.

Pradiniai duomenys. Pradiniai duomenys įrašyti byloje `tr.in`. Pirmoje eilutėje įrašytas vienas sveikasis skaičius n ($n \leqslant 2000$).

Likusiose n bylos eilučių įrašyta po tris tarpais atskirtus sveikuosius skaičius (vienam trikampiui skirta viena eilutė): x_i, y_i ir m_i ($1 \leqslant n$, $-10^7 \leqslant x_i \leqslant 10^7$, $-10^7 \leqslant y_i \leqslant 10^7$, $0 < m_i \leqslant 1000$).

Pavyzdžiai

Pradiniai duomenys	Rezultatai
5 -5 -3 6 -1 -2 3 0 0 2 -2 2 1 -4 -1 2	24.5



Rezultatai. Pirmoje rezultatų bylos `tr.out` eilutėje įrašykite bendrą visų trikampių užimamą plotą — vieną realųjį skaičių su lygiai vienu skaitmeniu po kablelio.

2. Tenisas (Estija)

Norėdamas susirasti naujų narių teniso klubas suorganizavo *Teniso savaitę.* Žaisti parodomuosiuose susitikimuose buvo pakviesta daug teniso žvaigždžių. Kiekvienas pakviestasis žaidėjas nurodė, keliuose susitikimuose jis sutinka žaisti. Kad pakviestiesiems žaidėjams nebūtų nuobodu, organizatoriai nutarė sudaryti tokį žaidimo tvarkaraštį, kad bet kurie du žaidėjai tarpusavyje susitiktų ne daugiau kaip vieną kartą.

Užduotis. Parašykite programą, kuri padėtų organizatoriams suskirstyti pakviestuosius žaidėjus poromis taip, kad kiekvienas žaidėjas žaistų tiek susitikimų, kiek jis buvo nurodęs pradžioje, ir su jokiu žaidėju nežaistų daugiau kaip vieną kartą. Be abejo, žaidėjas negali žaisti pats su savimi.

Pradiniai duomenys. Pirmoje pradinių duomenų bylos `tennis.in` eilutėje įrašytas žaidėjų skaičius N ($2 \leqslant N \leqslant 1000$). Likusiose N eilučių įrašyta po vieną skaičių — kiek susitikimų G_i ($1 \leqslant G_i < N$) sutiko žaisti atitinkamas žaidėjas. Tariama, kad žaidėjai sunumeruoti nuo 1 iki N taip, kaip pateikti jų pageidavimai pradinių duomenų byloje.

Rezultatai. Pirmoje rezultatų bylos `tennis.out` eilutėje įrašykite `NO SOLUTION`, jei neįmanoma sudaryti tokio tvarkaraščio, kad visų kviestinių žaidėjų norai būtų patenkinti, arba `SOLUTION`, jei tokį tvarkaraštį įmanoma sudaryti.

Jei sprendinys egzistuoja, jį įrašykite į tolesnes N eilučių. Kiekvienoje eilutėje įrašykite žaidėjo, kurio norimų žaisti žaidimų skaičius buvo nurodytas atitinkamoje pradinių duomenų eilutėje, varžovų numerius. Eilutėje nurodomi numeriai turi būti pateikti didėjančia tvarka ir vienas nuo kito atskirti vienu tarpu.

Jei galimi keli sprendiniai, pateikite vieną iš jų.

Pavyzdžiai

Pradiniai duomenys	Rezultatai
3 1 2 1 3 2 2 1	SOLUTION 2 1 3 2 NO SOLUTION

Vertinimas. Šiam uždaviniui taškai už sprendinio neturinčius testus (t. y. kai teisingas rezultatas yra `NO SOLUTION`) buvo skiriami tik tada, jei programa teisingai įveikė bent pusę testų, kuriems sprendinys yra galimas.

3. Monetų krūvelės (Vokietija)

Draugas Jums paaiškina paprastą žaidimą. Jis sukrauna keletą krūvelių iš vienodų monetų. Jūs galite perkelti tam tikrą skaičių monetų iš vienos krūvelės (pažymėkime ją A) į kitą krūvelę (pažymėkime ją B) tik tada, jeigu skaičius monetų, kurias Jūs paimate iš krūvelės A, yra lygus skaičiui monetų krūvelėje B prieš jų perkėlimą, t. y. ėjimas padvigubina skaičių monetų krūvelėje B. Jūsų užduotis — išlyginti krūveles (t. y. pasiekti, kad monetų visose krūvelėse būtų vienodas skaičius) atliekant kuo mažiau ėjimų.

Kadangi Jums nelabai aišku, kaip žaisti, draugas pateikia pavyzdį.

Yra trys krūvelės, kuriose yra atitinkamai 4, 6 ir 14 monetų. Galima išlyginti krūveles perkėlus 6 monetas iš trečios krūvelės į antrąją (dabar krūvelėse yra 4, 12 ir 8 monetos), o tada 4 monetas iš antros krūvelės į pirmąją. Dabar visos krūvelės yra vienodo aukščio (8 monetos) — baigta! Kadangi krūvelių negalima išlyginti atlikus mažiau ėjimų, minimalios ėjimų sekos ilgis — 2.

Užduotis. Žinoma, kiek yra krūvelių ir koks monetų kiekvienoje krūvelėje skaičius. Parašykite programą, kuri suskaičiuotų, per kiek mažiausiai ėjimų galima išlyginti krūveles (krūvelių skaičius negali sumažėti) ir rastų minimalią ėjimų seką.

Pradiniai duomenys bus tokie, kad sprendinys visada egzistuos (t.y. bendras visose krūvelėse esančių monetų skaičius dalysis be liekanos iš krūvelių skaičiaus) ir krūveles bus galima išlyginti atlikus 8 ar mažiau ėjimų ($M \leqslant 8$).

$\alpha + \omega$

Pradiniai duomenys. Pradinių duomenų bylą `stacks.in` sudaro dvi eilutės. Pirmoje eilutėje įrašytas krūvelių skaičius S ($1 \leqslant S \leqslant 8$). Krūvelės yra sunumeruotos nuo 1 iki S. Antrojoje eilutėje įrašyta S skaičių H_i ($1 \leqslant H_i \leqslant 70$, $1 \leqslant i \leqslant S$). H_i nusako i-osios krūvelės aukštį (t. y. ją sudarantį monetų skaičių) žaidimo pradžioje. Jūsų draugas nėra labai turtingas, taigi jis nenaudos žaidimui daugiau nei 200 monetų, t. y. $\sum_{i=1}^{S} H_i \leqslant 200$.

Rezultatai. Pirmoje rezultatų bylos `stacks.out` eilutėje įrašykite minimalų ėjimų, kurių reikia krūvelėms išlyginti, skaičių M. Toliau įrašykite M eilučių, aprašančių ėjimus ta tvarka, kuria jie yra atliekami. Kiekvienoje eilutėje įrašykite po du sveikuosius skaičius F ir T. F — tai krūvelės, iš kurios imate monetas, numeris, o T — krūvelės, į kurią padedate paimtas monetas, numeris.

Pavyzdžiai

Pradiniai duomenys	Rezultatai
3 4 6 14	2 3 2 2 1

4. Greičio ribojimai (Švedija)

Šiuolaikiniame pasaulyje dažniausiai renkamės ne trumpiausią kelią, o tą, kuriuo važiuodami užtrunkame mažiausiai laiko. Štai todėl važiuojant automobiliu greičio ribojimai keliuose yra gana svarbūs. Įsivaizduokite tokią situaciją, kad tam tikru momentu kai kurie greitį ribojantys ženklai keliuose paslaptingai dingsta. Kadangi negalima tikėtis iš vairuotojo, kad jis prisimins greičio ribojimus, peršasi vienintelė logiška išvada: pravažiavęs vietą, kur anksčiau buvo kelio ženklas, vairuotojas laikysis to paties greičio ribojimo, kokio jis laikėsi prieš atvažiuodamas į tą vietą.

Yra kelių tinklo aprašas. Dėl paprastumo tarkime, kad kelių tinklas susideda iš *kelių* ir *sankryžų*. Visi keliai yra vienpusiai, jungia lygiai dvi sankryžas ir juose yra ne daugiau kaip vienas greitį ribojantis ženklas, kuris yra (jei yra) pačioje kelio pradžioje. Laikykite, kad greitis pakinta akimirksniu ir nėra jokių kitų veiksnių, kurie gali daryti įtaką Jūsų kelionei. Ir, žinoma, jokiu būdu negalima važiuoti greičiau, negu leidžia paskutinysis Jūsų pravažiuotas ženklas.

Užduotis. Parašykite programą, kuri, pasinaudodama tuo, kad kai kurie ženklai yra dingę, rastų mažiausiai laiko reikalaujantį maršrutą.

Pradiniai duomenys. Pradiniai duomenys įrašyti byloje `speed.in`. Pirmoje eilutėje yra trys sveikieji skaičiai N, M ir D; čia N ($2 \leqslant N \leqslant 150$) yra skaičius sankryžų, kurios numeruojamos nuo 0 iki $N-1$, M — kelių skaičius, o D žymi sankryžą, į kurią Jums reikia patekti iš nulinės sankryžos.

Kiekvienoje tolesnių M eilučių aprašytas vienas kelias. Kelią aprašanti eilutė sudaryta iš keturių sveikųjų skaičių A ($0 \leqslant A < N$), B ($0 \leqslant B < N$), V ($0 \leqslant V \leqslant 500$) ir L ($1 \leqslant L \leqslant 500$); čia A ir B rodo, kad kelias tęsiasi nuo sankryžos A iki sankryžos B, V yra greičio ribojimas kelyje, L — kelio ilgis.

Jei V lygus nuliui, tai greičio ribojimo ženklas šiame kelyje yra dingęs. Kelyje sugaištas laikas T apskaičiuojamas pagal formulę $T = \frac{L}{V}$, jei $V \neq 0$, priešingu atveju $T = \frac{L}{V_{\text{sen}}}$; čia V_{sen} yra greičio ribojimas, kurio laikėtės iki tol, kol atvažiavote į šią sankryžą.

Atkreipiame dėmesį, kad operacijos turėtų būti atliekamos su slankaus kablelio skaičiais, kad būtų išvengta nereikalingo apvalinimo. Pradiniu momentu esate sankryžoje 0 ir laikotės 70 vienetų greičio ribojimo. Pradiniai duomenys tokie, kad kelias, kuriame užtrunkama trumpiausiai, yra tik vienas.

Rezultatai. Rezultatų bylą `speed.out` turi sudaryti viena eilutė. Joje turi būti išvardytos sankryžos, per kurias reikia važiuoti, norint kelyje iš sankryžos 0 į sankryžą D sugaišti mažiausiai laiko. Sankryžos turi būti surašytos tiksliai tokia tvarka, kokia per jas yra važiuojama. Taigi pirmoji sankryža turėtų būti 0, o paskutinė — D.

Pavyzdžiai

Pradiniai duomenys	Rezultatai	Paaiškinimas
6 15 1 0 1 25 68 0 2 30 50 0 5 0 101 1 2 70 77 13 35 42 2 0 0 22 2 1 40 86 2 3 0 23 2 4 45 40 3 1 64 14 3 5 0 23 4 1 95 8 5 1 0 84 5 2 90 64 5 3 36 40	0 5 2 3 1	Kelionėje sugaištas laikas šiuo atveju yra lygus 2,628.

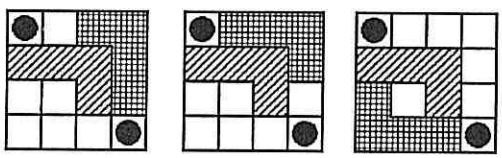
5. L žaidimas (Švedija)

L žaidimą sukūrė Edvardas de Bono, kuriam patiko žaisti stalo žaidimus ir kuris nemėgo žaidimų su daug figūrų. Jis ketino sukurti kuo paprastesnį žaidimą, kuriam žaisti reiktų daug intelekto. L žaidimas ir yra šių ketinimų rezultatas.

Abu žaidėjai turi po vieną raidės L formos figūrą (*L figūrą*). Dar yra dvi neutralios (t. y. nepriklausančios nė vienam žaidėjui) šaškės. Lentos matmenys yra 4×4. Žaidimo tikslas: įstumti priešininką į tokią poziciją, kad jis negalėtų pajudinti savo *L figūros*.

Iš pradžių abiejų žaidėjų *L figūros* ir abi šaškės užima lentoje tam tikras pozicijas. Pirmasis žaidėjas (arba tas žaidėjas, kurio eilė) pirmiausia paeina su *L figūra*, t. y. pakelia *L figūrą* ir padeda ją į bet kurią kitą laisvą vietą. Prieš dedant figūrą galima apversti (t. y. pakeisti L orientaciją). Padėta figūra privalo užimti keturis langelius.

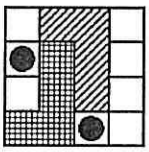
Paėjęs su *L figūra*, žaidėjas ***gali*** perkelti bet kurią (tik vieną) šaškę į bet kurį laisvą lentos langelį. Atkreipkite dėmesį, kad paeiti su šaške nebūtina!



Paveikslėliuose pateiktos trys pradinės situacijos. Iš bet kurios pateiktos situacijos žaidėjas, kuriam priklauso languota *L figūra*, gali paeiti su ja į dvi skirtingas pozicijas (tos pozicijos parodytos dviejuose likusiuose paveikslėliuose). Paėjęs *L figūra*, žaidėjas gali paeiti bet kuria šaške į bet kurį iš likusių laisvų 6 langelių arba visai nejudinti nė vienos šaškės. Iš viso yra $2 \times (6 + 6 + 1)$ galimų ėjimų.

$\alpha + \omega$

Žaidėjas laimi, jei priešininkas nebegali paeiti savo *L figūra*. Toliau pateiktame pavyzdyje atėjo eilė eiti žaidėjui, kurio figūra yra languota. Tačiau nebėra kur eiti, todėl jis pralaimi.



Žaidimas paprastas ir kartu sudėtingas, nes yra labai daug ėjimų. Iš viso yra daugiau kaip 18 000 variantų, kaip išdėstyti *L figūras* ir šaškes šioje mažoje lentoje, ir bet kuriuo žaidimo momentu gali būti iki 195 galimų skirtingų ėjimų, iš kurių tik vienas sėkmingas.

Užduotis. Duota tam tikra situacija lentoje, taip pat žinoma kuris žaidėjas turi paeiti. Pavadinkime šį žaidėją *pirmuoju*, o kitą — *priešininku*. Parašykite programą, kuri nuspręstų, ar pirmasis žaidėjas turi *laimintį ėjimą*, ir jei taip, jį rastų. Jei egzistuoja keli laimintys ėjimai, reikia nurodyti bet kurį vieną. Jei laiminčio ėjimo nėra, programa turi nuspręsti, ar žaidimas baigiasi *lygiosiomis* (turint omenyje, kad abu žaidėjai žaidžia optimaliai), ar pirmasis žaidėjas *pralaimi*.

Ėjimas vadinamas *laiminčiuoju*, jei pirmajam žaidėjui atlikus šį ėjimą ir toliau optimaliai žaidžiant jo priešininkas būtinai pralaimi nepriklausomai nuo to, kaip jis žaidžia.

Ėjimas vadinamas *pralaiminčiuoju*, jei pirmasis žaidėjas, nesvarbu kaip žaisdamas, *pralaimi*, optimaliai žaidžiant jo priešininkui.

Jeigu nė viena iš šių dviejų sąlygų netenkinama (t. y. nė vienas žaidėjas negali užsitikrinti, kad optimaliai abiem žaidžiant laimės, žaidimas laikomas pasibaigusiu *lygiosiomis*.

Pradiniai duomenys. Pradinių duomenų byloje `lgame.in` įrašytos keturios eilutės, turinčios po keturis simbolius. Jos apibūdina situaciją žaidimo lentoje.

Simbolis · atitinka tuščią lentos langelį, `x` žymi neutralią šaškę, `#` — pirmojo (jo eilė paeiti) žaidėjo *L figūrą*, `*` — priešininko *L figūrą*. Pradinė situacija yra korektiška, ir pirmasis žaidėjas gali padaryti bent vieną ėjimą.

Rezultatai. Rezultatus įrašykite į bylą `lgame.out`. Jei egzistuoja laimintis ėjimas, išveskite jį tokiu pat formatu kaip pradinių duomenų byloje.

Priešingu atveju pirmoje rezultatų bylos eilutėje išveskite `No winning move exist`. Antroje eilutėje išveskite arba `Draw`, jei žaidimas baigsis lygiosiomis (laikant, kad abu žaidėjai žaidžia optimaliai), arba `Losing`, jei pirmasis žaidėjas pralaimės.

Pavyzdžiai

Pradiniai duomenys	Rezultatai

`. * * *`
`# * . x`
`# # # .`
`x . . .`
	`. * * *`
	`x * # x`
	`# # # .`
	`. . . .`

`. . . x`
`# # # .`
`# * * *`
`x . . *`
	`No winning move`
	`exist`

`. # # #`
`x # * x`
`* * * .`
`. . . .`
	`No winning move exist`
	`Losing`

6. Judantys robotai (Lietuva)

Yra keletas robotų, kurie gali judėti visoje dvimatėje plokštumoje. Plokštuma padalyta į vienetinius kvadratėlius, kurių koordinatės (x, y) yra sveikieji skaičiai.

Roboto būseną apibūdina jo padėtis ir judėjimo kryptis. Padėtis nusakoma sveikųjų skaičių pora (x, y), o kryptis — laipsniais. Yra keturios galimos kryptys — 0, 90, 180 ir 270 laipsnių.

Robotai gali vykdyti dviejų rūšių — *posūkio* ir *žingsnio* komandas. Posūkio komanda turi vieną parametrą D, kurio galimos reikšmės yra 90, 180 ir 270. Ši komanda pakeičia roboto kryptį per D laipsnių: jei dabartinė roboto kryptis yra C, tai ji pakeičiama į $(C + D)$ **mod** 360. Žingsnio komanda parametrų neturi. Ją atlikdamas robotas pajuda per vieną žingsnį judėjimo kryptimi. Vienas žingsnis 0 laipsnių kryptimi pakeičia roboto padėtį (atitinkamas koordinates) per $(1, 0)$, 90 laipsnių kryptimi — per $(0, 1)$, 180 laipsnių kryptimi — per $(-1, 0)$, o 270 laipsnių kryptimi — per $(0, -1)$.

Kiekvienas robotas turi jam skirtą baigtinio ilgio seką komandų, kurias atlieka iš eilės. Kai visos komandos atliktos, robotas sustoja savo galutinėje padėtyje. Roboto judėjimas visiškai nepriklauso nuo kitų robotų, tą pačią padėtį (langelį) plokštumoje gali užimti bet koks skaičius robotų.

Prieš pradedant robotams judėti, valdymo centras gali įsakyti pašalinti tam tikras komandas iš kai kurių robotų komandų sekos. Taip valdymo centras gali keisti robotų judėjimo trajektorijas bei galutines padėtis.

Valdymo centras nusprendė, kad visi robotai turi susirinkti į vieną vietą (langelį) apžiūrai, taigi reikia, kad visų robotų galutinės padėtys sutaptų. Taip pat centras nori, kad tai būtų įgyvendinta pašalinus kuo mažiau komandų.

Užduotis. Yra R robotų. Kiekvienas robotas turi pradinę padėtį, pradinę judėjimo kryptį ir jam paskirtą komandų seką.

Parašykite programą, kuri nustatytų, kiek mažiausiai komandų iš visų robotų komandų sekų reikia pašalinti, kad visų robotų galutinės padėtys sutaptų. Šią padėtį pavadinkime *bendra galutine padėtimi*. Jeigu galima parinkti kelias bendras galutines padėtis, pakanka nurodyti bet kurią.

Pradiniai duomenys. Pradiniai duomenys įrašyti byloje `robots.in`. Pirmoje eilutėje įrašytas robotų skaičius R ($2 \leqslant R \leqslant 10$). Toliau yra R grupių eilučių — po vieną kiekvienam robotui.

Pirmoje grupės eilutėje yra keturi tarpu atskirti skaičiai: pradinė roboto padėtis (x, y), pradinė roboto judėjimo kryptis d ($d = 0$, 90, 180 arba 270) ir roboto komandų sekos ilgis n ($1 \leqslant n \leqslant 50$). Toliau grupėje yra n eilučių, aprašančių roboto komandų seką. Kiekvienoje eilutėje yra po vieną komandą.

Žingsnio komandai skirtoje eilutėje įrašytas vienintelis simbolis — raidė S. *Posūkio* komandai skirtoje eilutėje įrašyta raidė T ir posūkio parametras D ($D = 90$, 180 arba 270). Raidę T nuo posūkio parametro skiria tarpas.

Rezultatai. Rezultatus įrašykite į bylą `robots.out`. Jeigu neįmanoma robotų surinkti į tą pačią galutinę padėtį (langelį), į vienintelę rezultatų bylos eilutę įrašykite skaičių -1 (minus vienas).

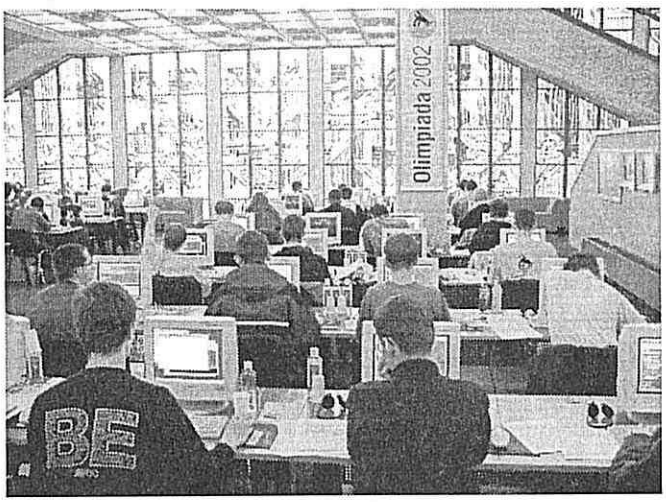
Priešingu atveju į pirmą eilutę įrašykite mažiausią skaičių komandų, kurias reikia pašalinti. Antroje eilutėje tuomet įrašykite du tarpu atskirtus skaičius: bendros galutinės padėties koordinates x ir y.

Pavyzdžiai

Pradiniai duomenys	Rezultatai	Paaiškinimai
2 2 0 270 5 S T 180 S S S 1 -1 0 8 S S T 90 S T 270 S T 90 S	2 2 1	Turime du robotus. Pirmojo pradinė padėtis (2, 0), judėjimo kryptis 270, jam duotos 5 komandos. Antrojo pradinės padėties koordinatės (1, −1), judėjimo kryptis 0, jam duotos 8 komandos. Mažiausias skaičius komandų, kurias reikia pašalinti, lygus 2. Tuomet robotų galutinės padėtys sutaps ir bendros galutinės padėties koordinatės bus (2, 1). Tai galima gauti, pvz., pašalinus pirmojo roboto 3-ąją komandą ir antrojo roboto 5-ąją komandą.

Olimpiados akimirkos

Informatikų olimpiada Seimo rūmuose



Švietimo ir mokslo ministras apdovanoja geriausių rezultatų pasiekusį estą Martin Pettai

Olimpiados šeimininkai ir svečiai (iš kairės į dešinę): A. Truu (Estija), J. Koivisto (Suomija), H. Strömberg (Švedija), G. Babravičius (LR Seimas), M. Sapagovas (MII), V. Dagienė (MII), J. Zalatorius (IVP komitetas)