ISSN 2783-7599

b·e·b·r·a·s

Informatikos ir informatinio mąstymo uždavinių rinkinys

Nr. 12

ŠVIETIMO, MOKSLO IR SPORTO MINISTERIJA

VILNIAUS UNIVERSITETAS · 1579 · UNIVERSITAS VILNENSIS

Bebras
International Challenge on Informatics and Computational Thinking

Šiame rinkinyje pateikiami XIX informatikos ir informatinio mąstymo konkurso (iššūkio) „Bebras“ II etapo, vykusio 2023 metų pradžioje, užduotys, jų atsakymai ir paaiškinimai, koks informatikos turinys ir konceptai atskleidžiami, kaip ir kuo uždavinys įdomus ugdant informatinį mąstymą. Visi uždaviniai (įskaitant grafiką ir kitą medžiagą) licencijuojami pagal Kūrybinių bendrijų licenciją – „Creative Commons Attribution-ShareAlike 4.0 International License“. Šis uždavinių rinkinys skiriamas 9–12 klasių mokinių informatikos ir informatinio mąstymo gebėjimams ugdyti.

„Bebras“ yra ne tik konkursas, bet ir mokymosi ir atradimų platforma, leidžianti moksleiviams ugdyti savo gebėjimus sprendžiant uždavinius ir gilinti savo žinias informatikos srityje. Tai yra naudinga ne tik moksleiviams, bet ir mokytojams, kurie gali naudoti „Bebro“ užduotis ir išteklius mokymo procese.

Dėkojame Daumilui Ardickui, Nojui Gudinavičiui, Karoliui Jasučiui, dr. Eglei Jasutei, dr. Tatjanai Jevsikovai, Audronei Klupšaitei, Linui Kurtinaičiui, Alvidai Lozdienei, Aleksandrui Maliuginui, Vaidai Masiulionytei-Dagienei, dr. Gabrielei Stupurienei, Gvidui Tupiniui, talkinusiems verčiant ir adaptuojant uždavinius. Taip pat dėkojame tarptautinei „Bebro“ bendruomenei ir uždavinių autoriams.

Sudarė Lina Vinikienė

Konsultavo Valentina Dagienė

Redagavo Viktoras Dagys

Viršelį kūrė Vaidotas Kinčius

Įvadas

Informatinis mąstymas – tai gebėjimas atpažinti, formuluoti ir spręsti aplinkos problemas (uždavinius), logiškai organizuoti ir analizuoti duomenis, taikyti schemas ir modelius, įvertinti problemos išsprendžiamumą, bandyti automatizuoti sprendimą naudojantis skaitmeninėmis technologijomis.

„Bebras“ yra integruotas informatinio ugdymo modelis, kurio tikslas – populiarinti informatikos mokslą ir skatinti informatinį mąstymą (angl. *Computational thinking*) ne tik įvairaus amžiaus mokiniams, bet ir mokytojams bei visuomenei.

Nuo 2004 metų „Bebro“ bendruomenė sukūrė šimtus užduočių. Dauguma jų yra suformuluotos kaip interaktyvūs ir (arba) atvirieji klausimai. Tačiau net ir tada, kai atsakymą reikia pasirinkti iš sąrašo, nėra vienintelio būdo pateikti sprendimą. Užduotys turi būti įdomios ir patrauklios, atitikti konkurso dalyvių amžių, o sprendimui turėtų prireikti vidutiniškai trijų minučių. „Bebro“ užduotyse daugiausia dėmesio skiriama tai informatikos daliai, kurią turėtų išmanyti visi, kas nori suprasti informatikos, kaip mokslo, pagrindus ir tapti ne tik sumaniu technologijų naudotoju, bet ir mokytis moderniais metodais spręsti įvairiausias realaus gyvenimo problemas, tapti naujų technologijų kūrėju.

2022 metų „Bebro“ konkurso uždavinius sprendė 3 295 494 mokiniai 60-yje valstybių. Lietuvoje tais metais konkurse dalyvavo 53 975 mokiniai.

Daugiau informacijos apie „Bebro“ konkursą pateikiama interneto svetainėse:

- Tarptautinė „Bebro“ iššūkio svetainė: www.bebras.org
  (iš čia galite pasiekti visų dalyvaujančių šalių svetaines – spustelėkite šalies vėliavėlę)
- Lietuvos „Bebro“ svetainė: bebras.lt
- Konkurso sistema – vadinamasis „Bebro“ varžybų laukas: lt.bebras.lt

2022 m. „Bebro“ konkurso dalyvių skaičius pagal šalis (60 šalių duomenys)

2023 m. vasario 4 d. Lietuvoje „Bebro“ konkurso II etape jauniai ir kolegos sprendė po 15 uždavinių: trečdalis buvo lengvesnių, trečdalis – vidutinio sunkumo ir trečdalis – sunkesnių. Uždaviniams spręsti skiriama 30 minučių.

Lietuvoje konkurso taškai skaičiuojami taip:

- Prieš pradėdamas spręsti, kiekvienas dalyvis turi 45 taškus (18 uždavinių × 3);
- Už teisingai išspręstą uždavinį skiriama 6, 9 arba 12 taškų (priklausomai nuo uždavinio sunkumo lygio);
- Už neišspręstą uždavinį – 0 taškų;
- Už klaidingą atsakymą atimamas trečdalis uždaviniui skiriamų taškų, t. y., atitinkamai 2, 3 arba 4 taškai.

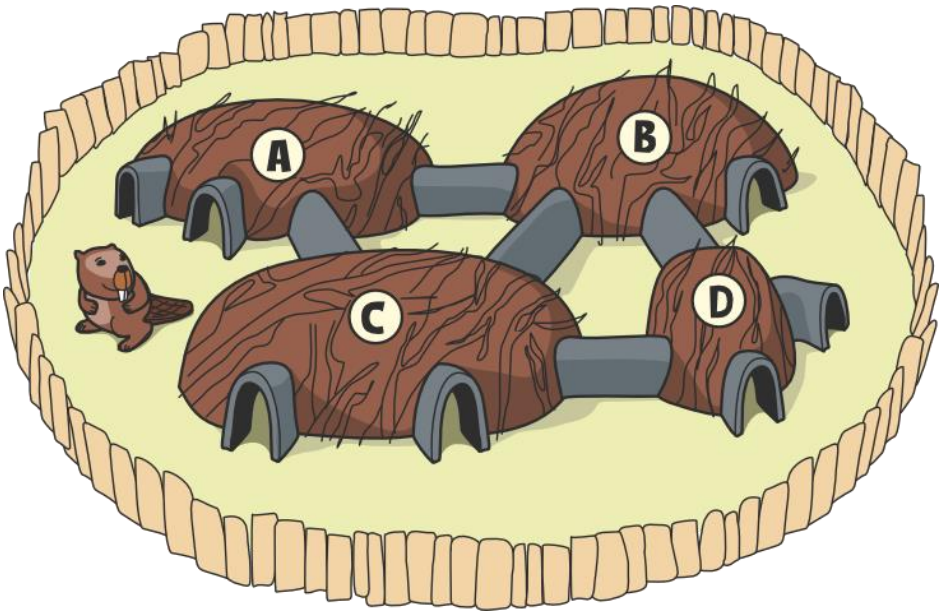
Lietuvoje XIX informatikos ir informatinio mąstymo konkurso „Bebras“ II etape dalyvavo 493 jauniai (9–10 klasių mokiniai) ir 436 kolegos (11–12 klasių mokiniai).

Lentelėje pateikiamas XIX konkurso II etapo uždavinių skirstymas pagal amžiaus grupes.

Kiekvieno uždavinio pradžioje nurodoma, kuriai amžiaus grupei jis skiriamas ir jo sudėtingumo lygis:

- lengvas – 6,
- vidutinis – 9,
- sunkus – 12.

Taip pat pateikiamas uždavinio atsakymas ir paaiškinimas, kaip uždavinys susijęs su informatika.

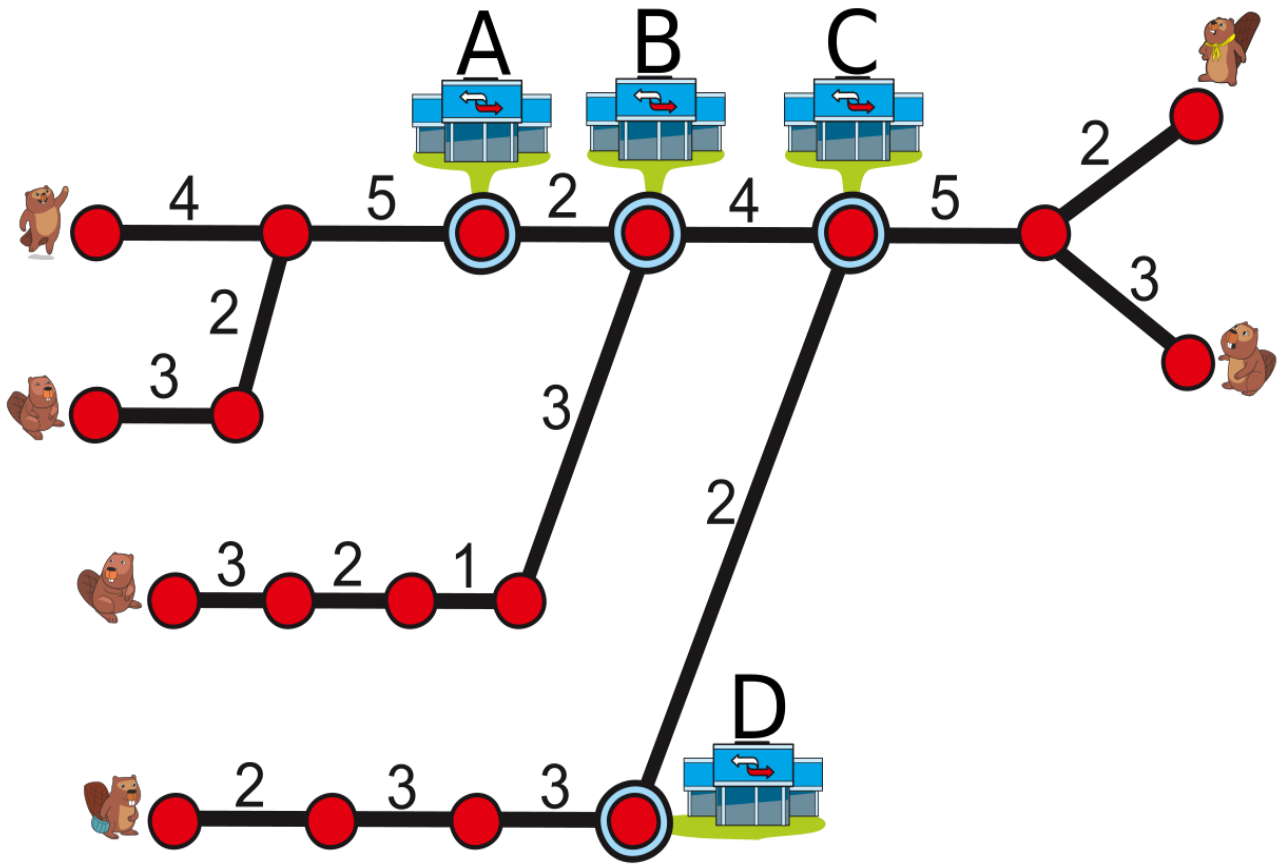


Nr.	*Uždavinio pavadinimas*	*Uždavinio identifikatorius*	*Jauniai*	*Kolegos*
1	Metro traukinių tinklas	2022-SA-04	6	6
2	Ponia Begalybė	2022-IR-03a	6	
3	Rekomendacijų sistema	2022-TW-02	6	
4	Miltų saugykla	2022-RS-04	6	
5	Veržlės ir varžtai	2022-CA-06-II	6	
6	Salų jungimas	2022-LT-01	9	6
7	Spalvotos žvakutės	2022-IE-04c	9	6
8	Daug bebrų	2019-LT-02	9	6
9	Braškės	2022-TW-03	9	
10	Susiliejantys kaimai	2022-LT-05	9	
11	Bebrų duomenys	2022-IT-03a	12	9
12	Muzikos instrumentas	2020-ID-01a	12	9
13	Bebrų dirbtinis intelektas	2022-DE-04	12	12
14	Akmenys	2022-CH-09c	12	12
15	Roboto kelias	2022-FI-03	12	
16	Kipu	2019-JP-03-II		6
17	Varlės spalvinimas	2022-IT-01b		9
18	Dienoraščio paslaptis	2021-CZ-01		9
19	Kengūra	2021-UZ-01d		9
20	Pasivaikščiojimas parke	2018-SK-06		12
21	Monetų kolekcininkas	2018-US-02		12
22	Pramogų bebravietė	2017-LT-01A		12

1. Metro traukinių tinklas

Šeši draugai bebrai gyvena skirtingose Bebrų miesto dalyse. Jie naudojasi požeminių traukinių tinklu ir susitinka vienoje iš pagrindinių stočių: A, B, C arba D. Jie nori atvykti į tą pačią stotį per kuo trumpesnį laiką. Jie visi vienu metu įlipa į traukinius skirtingose stotyse.

Traukinių eismo schemoje parodytos kiekvieno maršruto stotys (raudoni mazgai) ir laikas, per kurį galima nuvažiuoti iš vienos stoties į kitą. Keleivių įlaipinimas ir išlaipinimas kiekvienoje stotyje trunka vieną minutę.



Kurioje stotyje draugai turėtų susitikti, kad kaip įmanoma greičiau pasimatytų?

A. Stotyje A

B. Stotyje B

C. Stotyje C

D. Stotyje D

Paaiškinimas

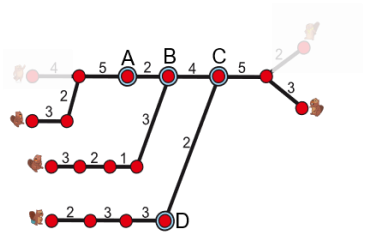
Teisingas atsakymas: B (stotis B).

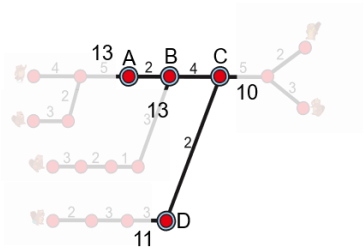
Norėdami išspręsti šią užduotį, turime apskaičiuoti kiekvieno draugo kelionės minutes, pradedant nuo kiekvieno jų pradinės stoties ir baigiant kiekviena iš stočių (A, B, C, D).

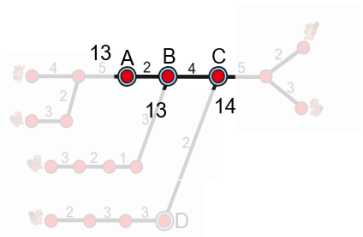
Žinoma, tai galima atlikti apskaičiavus visus galimus kiekvieno draugo kelius ir pažiūrėjus, kiek laiko užtruks, kol paskutinis draugas atvyks į kiekvieną stotį. Tai būtų labai varginantis darbas ir užimtų daug laiko.

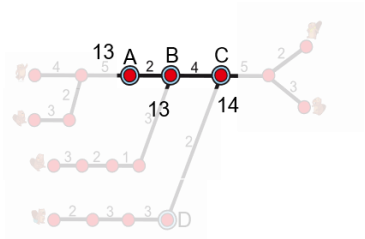
Šiai užduočiai spręsti galime naudoti dekompoziciją (suskaidyti užduotį į mažesnes dalis) ir abstrakciją (nuspręsti, kuriuos maršrutus svarbiausia apskaičiuoti, o į kitus nekreipti dėmesio). Turime atsižvelgti tik į ilgiausią kelionės kiekvienu maršrutu laiką, nes reikia žinoti, per kiek laiko atvyks paskutinis draugas.

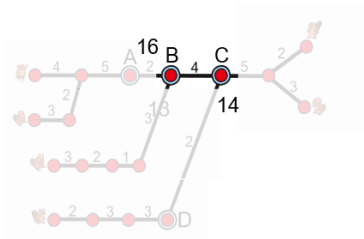
Toliau pateikiame vieną iš galimo šios užduoties sprendimo žingsnių sekų.

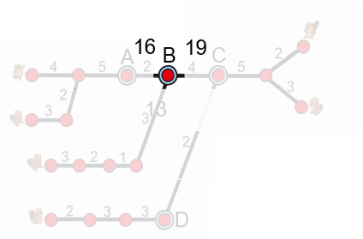












Nagrinėjant dviejų draugų atvykimą į A stotį iš kairės, matome, kad pirmajam draugui reikia 11 minučių (1+4+1+5), o antrajam – 13 minučių (1+3+1+2+1+1). Nuo šio taško toliau turime atsižvelgti tik į lėtesnį atvykimą, t. y. 13 minučių. Panašiai du draugai dešinėje pusėje susitiks C stotyje po 10 minučių (1+3+1+5), bet kadangi trečiasis draugas į C stotį atvyksta per 14 (1+11+2) minučių, nuo šio taško turime atsižvelgti tik į 14 minučių.

Toliau taip pat einame iš vienos stoties į kitą, kaip parodyta paveikslėlyje, visada atsižvelgdami tik į ilgiausią atvykimo laiką, kol pasiekiame galutinę stotį (B). Tai geriausia stotis visiems draugams atvykti per trumpiausią įmanomą laiką – 19 minučių.

Kad patvirtintume savo rezultatą, pateikiame lentelę, kurioje nurodyta minučių skaičius, reikalingas draugams atvykti į kiekvieną stotį iš kiekvienos krypties. Lygindami stotis matome, kad paskutiniam draugui atvykti į A stotį reikia 22 minučių, į B stotį – 19 minučių, į C stotį – 21 minutės, o į D stotį – 24 minučių. Tai patvirtina, kad B stotis yra geriausia stotis susitikti.

Laikas iki A	Laikas iki B	Laikas iki C	Laikas iki D
Iš kairės: 13	Iš kairės: 16	Iš kairės: 21	Iš kairės: 24
Iš dešinės: 22	Iš dešinės: 19	Iš dešinės: 10	Iš dešinės: 11

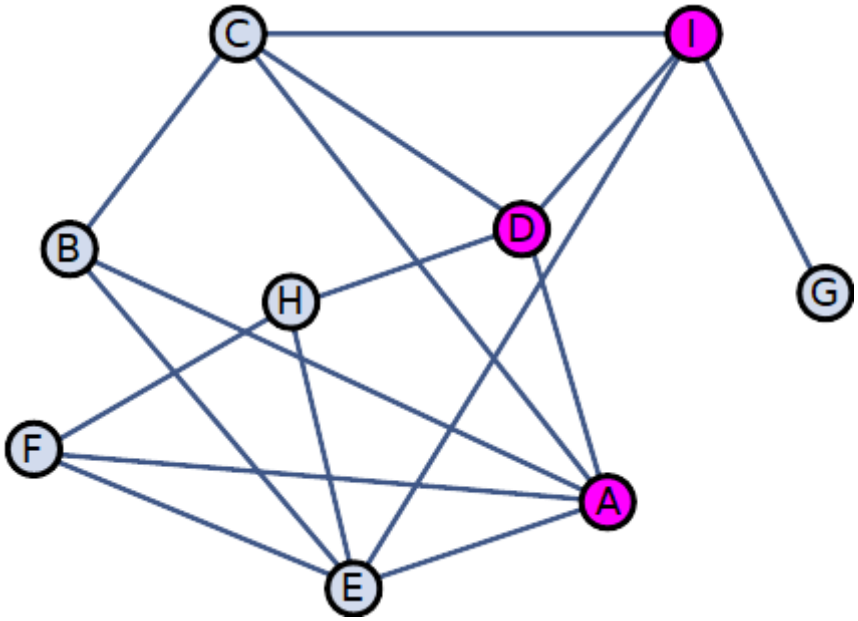
Tai informatika!

Kompiuterių moksle dinaminis programavimas reiškia sudėtingo uždavinio supaprastinimą, suskaidant jį į paprastesnius uždavinius. Dinaminis programavimas naudojamas optimizavimo uždaviniuose (ieškant kokios nors reikšmės maksimumo ir minimumo).

Šio tipo diagramos, naudojamos sprendžiant šią užduotį, vadinamos grafais. Stotys yra grafo mazgai, o keliai – briaunos. Tiksliau, tai yra grafo su svoriais pavyzdys, nes briaunoms priskirta tam tikra reikšmė (kelionės trukmė). Tai reiškia, kad grafas turi svorines briaunas. Svoriai grafuose gali reikšti įvairius dalykus: išlaidas, ilgius, svorius, tūrius.

2. Ponia Begalybė

Mokiniai klasėje su bendraklasiais kalbasi tokia tvarka, kaip parodyta schemoje. Pavyzdžiui, mokinys H per dieną kalbasi tik su D, E ir F. Pirmadienį į klasę atėjo nauja matematikos mokytoja. Mokiniai A, D ir I iš karto mokytoją pradėjo vadinti „Ponia Begalybe“ dėl jos šukuosenos. Slapyvardis tarp mokinių plinta tokiu būdu: jei daugiau nei pusė bendraklasių, su kuriais jie bendrauja, pasako šį slapyvardį, tai šis mokinys jį naudos ir kitą dieną.



Kokia yra pirmoji savaitės diena, kai kiekvienas klasės mokinys naudoja slapyvardį „Ponia Begalybė“?

A) Antradienis

B) Trečiadienis

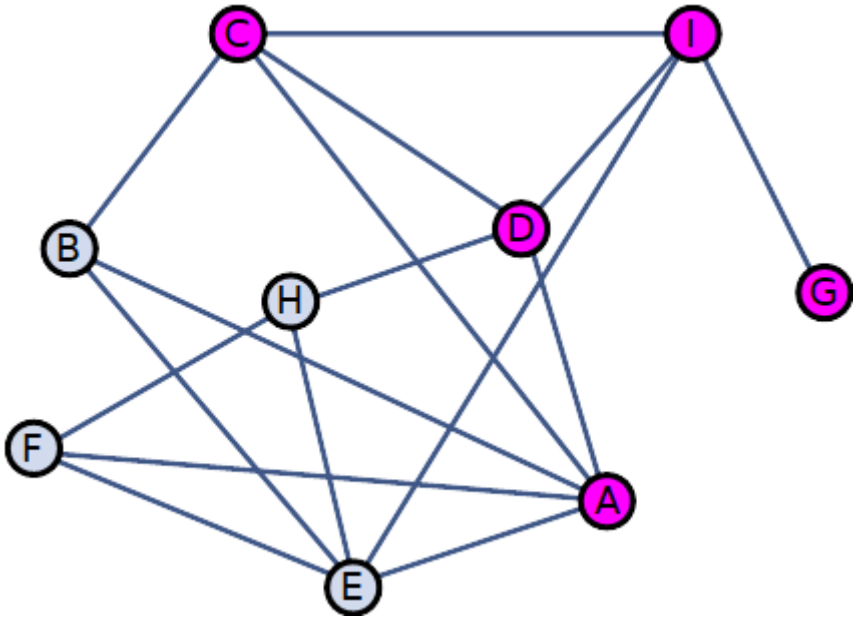
C) Ketvirtadienis

D) Penktadienis

Paaiškinimas

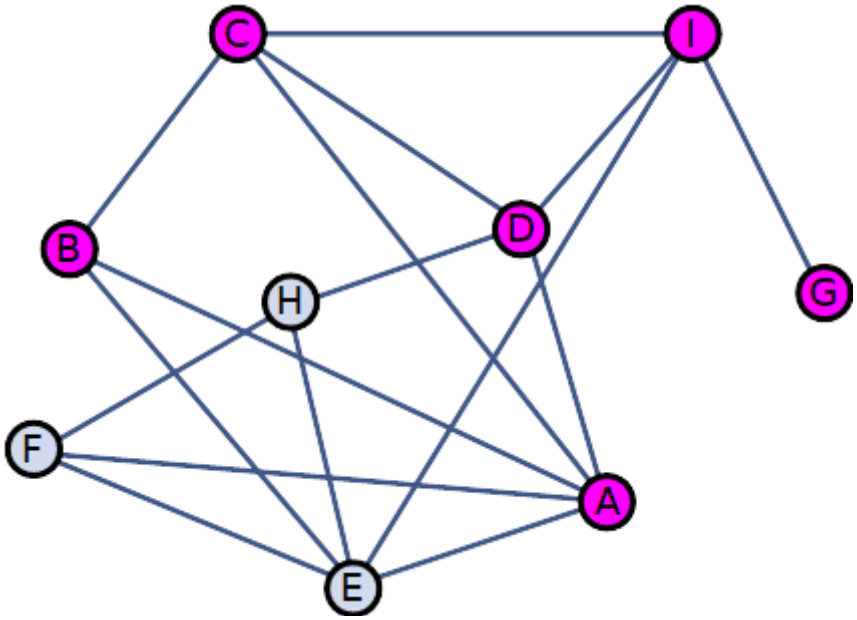
Teisingas atsakymas yra penktadienis.

Dabar galime pasekti, kaip plinta slapyvardis. Mokiniai G ir C pradeda naudoti slapyvardį antradienį.

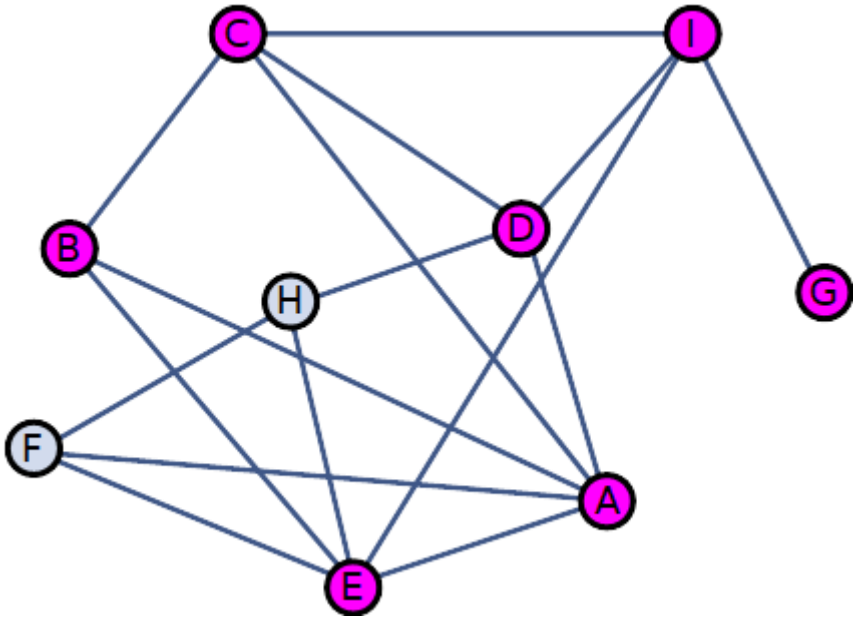


G bendrauja tik su vienu bendraklasiu (I) ir šis bendraklasis slapyvardį naudojo pirmadienį. C bendrauja su 4 bendraklasiais ir 3 iš jų slapyvardį naudojo pirmadienį. 3 iš 4 bendraklasių yra daugiau nei pusė. Taigi C taip pat slapyvardį naudojo antradienį.

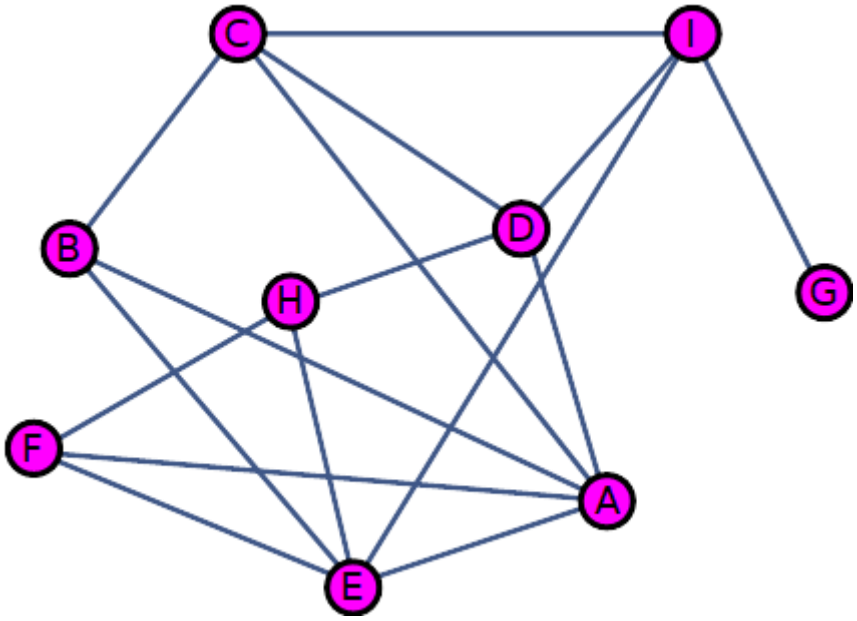
B pradeda naudoti slapyvardį trečiadienį.



E pradeda naudoti slapyvardį ketvirtadienį.



F ir G pradeda naudoti slapyvardį penktadienį.

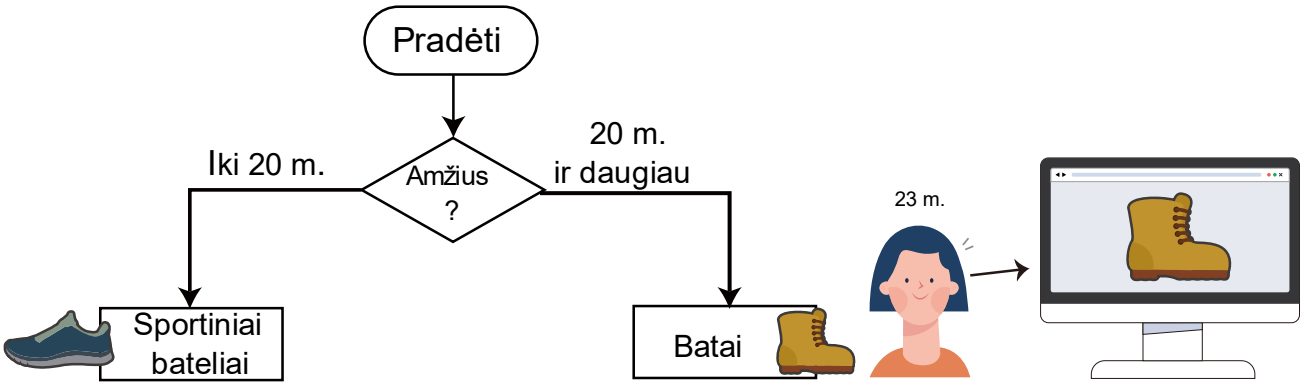


Tai informatika!

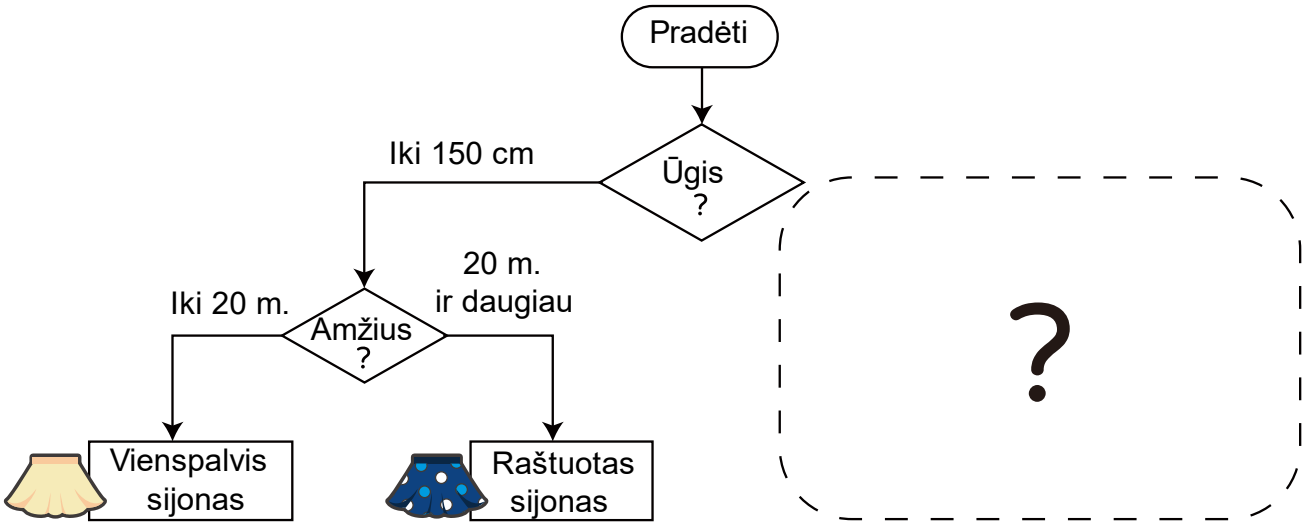
Socialiniai tinklai atlieka svarbų vaidmenį skleidžiant informaciją apie infekcijas, idėjas, daro įtaką savo nariams. Socialiniame tinkle atsiradusi idėja ar naujovė greitai išplinta per socialines sąveikas arba išnyksta. Įtakos sklaidos socialiniuose tinkluose maksimizavimas yra labai aktuali tendencija informatikos, ekonomikos ir vadybos mokslams. Šiame uždavinyje nagrinėjamas specialus sklaidos modelis, vadinamas slenkstiniu modeliu. Jame kiekvienam asmeniui yra nustatytas slenkstis, t. y., dalis jo ryšių, kurie turi būti aktyvūs, kad asmuo taptų aktyvus.

3. Rekomendacijų sistema

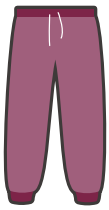
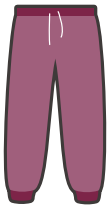
Interneto drabužių ir avalynės parduotuvė naudoja naują rekomendacijų sistemą, pagrįstą asmenine informacija apie klientus. Pavyzdžiui, kai klientas, norintis nusipirkti batus, įveda savo informaciją, programa, vadovaudamasi diagramoje pateikta taisykle, rekomenduoja batų porą.



Vieną dieną dalis prekių rekomendavimo taisyklių netyčia išsitrynė. Toliau pateiktoje diagramoje pavaizduotos sistemoje likusios taisyklės.

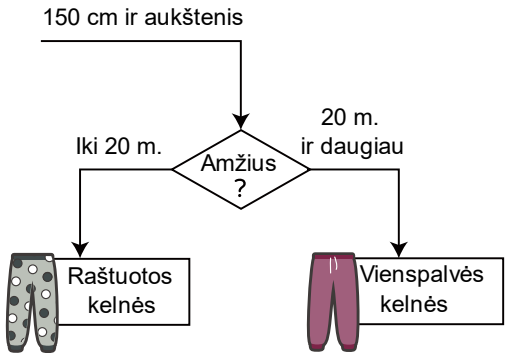


Laimei, ankstesnių pirkimų įrašai saugomi serveryje, kaip parodyta šioje lentelėje:

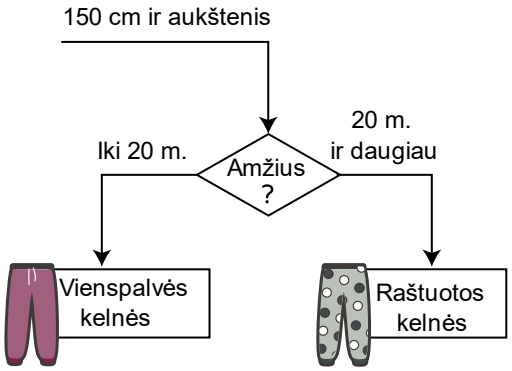
Vartotojas	18 m. 140 cm	32 m. 145 cm	28 m. 155 cm	15 m. 160 cm	10 m. 152 cm
Rekomendacijos	Vienspalvis sijonas 	Raštuotas sijonas 	Raštuotos kelnės 	Vienspalvės kelnės 	Vienspalvės kelnės 

Kuri diagrama atkurta pagal lentelėje pateiktą informaciją?

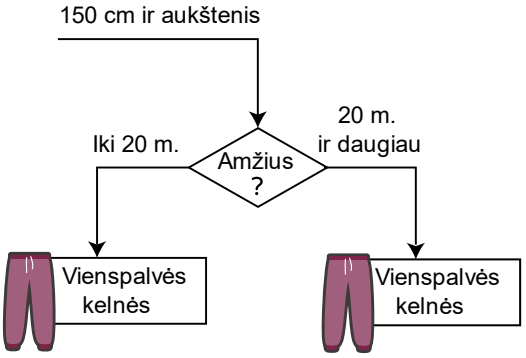
A.



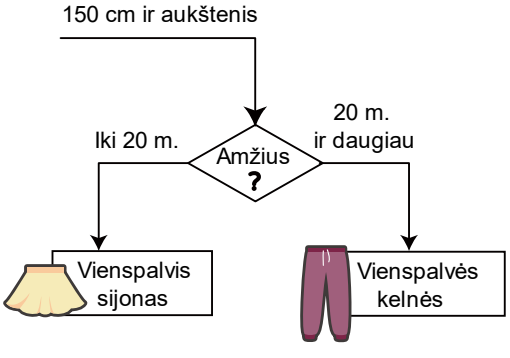
B.



C.



D.



Paaiškinimas

Teisingas atsakymas: B.

Pagal pirmąją sąlygą tikrinamas kliento ūgis, todėl reikia atsižvelgti tik į aukštesnius nei 150 cm ūgio klientus.

Lentelėje matome, kad visiems šiems klientams buvo rekomenduota įsigyti kelnes. Todėl atsakymas D yra neteisingas.

Kadangi visiems šiems jaunesniems nei 20 metų klientams buvo rekomenduotos vienspalvės kelnės, atsakymas A yra neteisingas.

Kadangi visiems šiems 20 metų ir vyresniems pirkėjams buvo rekomenduotos raštuotos kelnės, atsakymas C yra neteisingas.

Tai informatika!

Diagrama, pagal kurią veikia programinė įranga, atliekanti šią užduotį, vadinama srauto diagrama. Informatikoje srauto diagrama gali būti naudojama algoritmui, darbo eigai arba procesui vaizduoti. Srauto diagramą sudaro įvairios figūros, sujungtos rodyklėmis. Figūros vaizduoja skirtingo tipo veiksmus, o rodyklės – tų veiksmų atlikimo tvarką.

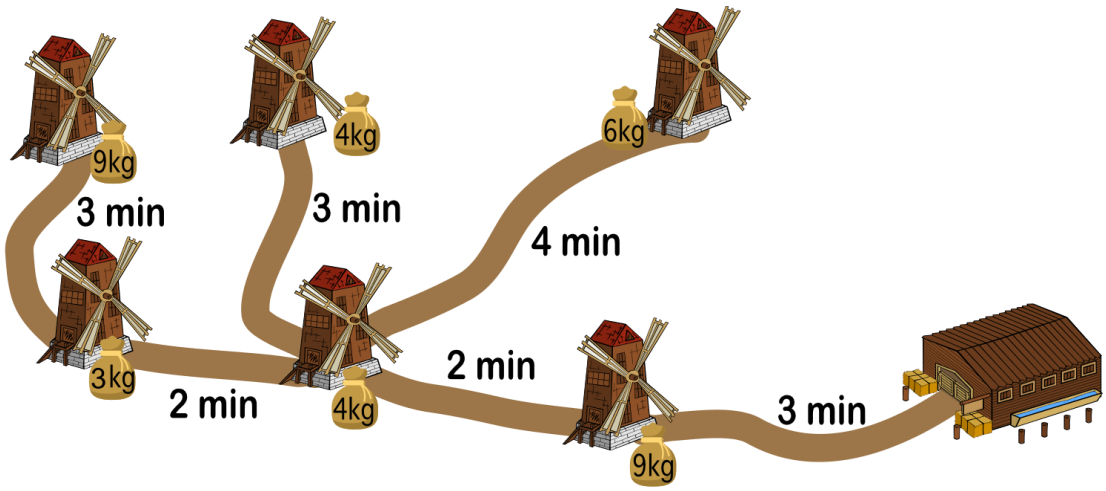
Kadangi srauto diagramos parodo programavimo procesą, inžinieriai dažnai jomis naudojosi projektuodami programą arba rodydami, kaip gauti tam tikrus rezultatus.

Kasdieniame gyvenime kai kurie žmonės naudoja srauto diagramas priimdami sprendimus, pavyzdžiui, spręsdami, ką valgyti pietums, priklausomai nuo turimų nuostatų. Kai kurie jas naudoja avarinei procedūrai pagal įvairias situacijas suformuluoti, o kai kas kaip priemonę, paaiškinančią, kaip turėtų būti vykdomi tam tikri planai.

4. Miltų saugykla

Paveikslėlyje pavaizduoti keli malūnai ir juos su saugykla jungiantys keliai. Kiekvieną vakarą malūnuose paruošiami maišai su miltais ir sudedami šalia malūnų. Bebras Vilius turi surinkti visus maišus ir iki saulėlydžio nugabenti juos į saugyklą. Vilius gali nešti kelis maišus vienu metu, bet bendras maišų svoris negali būti didesnis nei 15 kg.

Laikas, per kurį Vilius nukeliauja nuo vieno malūno iki kito ir iki saugyklos, parodytas paveikslėlyje.



Bebras Vilius yra saugykloje ir nori kuo greičiau sunešti visus maišus į saugyklą. Kiek minučių reikės šiai užduočiai atlikti?

A) 50 minučių

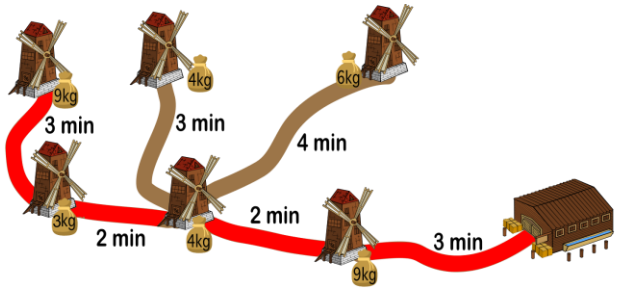
B) 31 minutės

C) 44 minučių

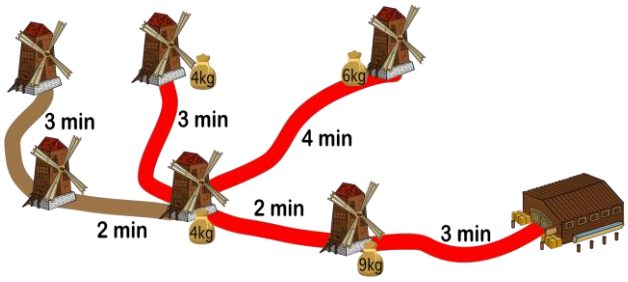
D) 54 minučių

Paaiškinimas

Iš viso reikia paimti 35 kg miltų. Kadangi bebras Vilius per vienu kartu gali nunešti ne daugiau kaip 15 kg, norėdamas sunešti visus maišus jis turės atlikti mažiausiai 3 reisus. Kad sugaištų kuo mažiau laiko, jis turėtų paimti maišus iš gretimų malūnų.

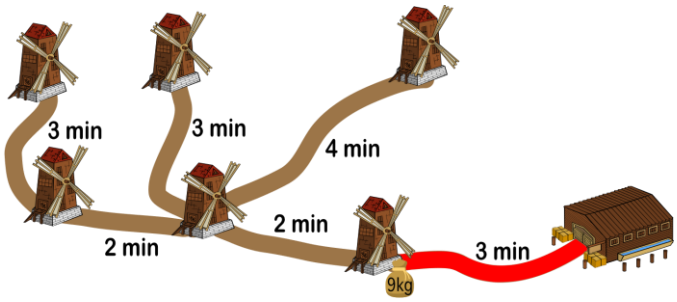


Pavyzdžiui, paėmęs 9 kg maišą iš viršutinio kairiojo malūno, Vilius taip pat gali paimti 3 kg maišą iš gretimo malūno (bendras svoris – 12 kg). Visi kiti maišai sveria daugiau nei 3 kg, todėl jis negali paimti jokio kito maišo. Pirmoji kelionė užtruks 20 minučių (3+2+2+3+3+2+2+2+3).



Norint paimti likusius maišus iš likusių keturių malūnų, antrojo reiso metu reikėtų paimti tris tolimiausius maišus: 6 kg maišą iš viršutinio dešiniojo malūno ir du 4 kg maišus iš vidurinio malūno (bendras svoris – 14 kg). Antroji kelionė truktų 24 minutes (3+2+4+4+3+3+3+2+3).

Trečiuoju reisu iš arčiausiai saugyklos esančio malūno bus paimti likusieji 9 kg ir tai užtruks 6 minutes (3+3). Taigi iš viso kelionė truktų 50 minučių.



Pirmojo, antrojo ir trečiojo reisų eilės tvarką galima keisti; svarbu, kad būtų suskirstyti į grupes maišai, kuriuos reikia paimti.

B variantas negali būti tinkamas atsakymas, nes 9 kg maišui iš viršutinio kairiojo malūno paimti jau prireiktų 20 minučių (žr. anksčiau aprašytą pirmąją kelionę). Likusių 10 minučių užteks tik kelionei iki antrojo malūno ir atgal, todėl Vilius negalės paimti dviejų maišų iš viršutinių malūnų.

C variantas negali būti teisingas atsakymas, nes 44 minučių užtenka tik pirmajai ir antrajai kelionei, todėl 9 kg artimiausiame malūne nebūtų paimti.

D atsakymo variantas irgi negali būti tinkamas, nes jo laikas ilgesnis, nei A varianto.

Tai informatika!

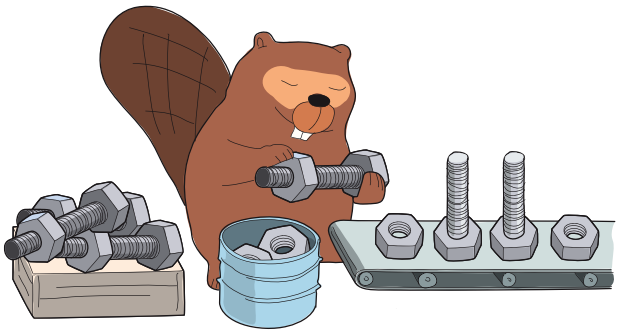
Informatikos moksle optimizavimo uždaviniais vadinami tokie, kuriais siekiama rasti geriausią sprendimą iš visų galimų. Paprastai tai susiję su funkcijos didžiausios arba mažiausios reikšmės radimu. Pavyzdžiui, šioje užduotyje prašoma rasti mažiausią bendrą laiką.

Uždavinio sprendimas kompiuteriu taip pat dažnai susijęs su optimizavimu. Pavyzdžiui, norime, kad kompiuteris galėtų atlikti užduotis per trumpą laiką ir naudodamas minimalius išteklius, pavyzdžiui, atmintį.

Spręsdami šią užduotį, mes (arba kompiuteris) galime rasti geriausią sprendimą, išbandydami visus galimus maršrutus ir tada pasirinkdami maršrutą, kurio bendras laikas yra mažiausias. Šis metodas vadinamas brutalios jėgos metodu. Jis tinka sprendžiant nedidelės apimties uždavinius. Jei uždavinys apima daug dydžių, brutalios jėgos metodas veiks lėtai, nes reikės išnagrinėti labai daug galimų sprendinių.

Yra kitų metodų, kuriais galima efektyviau rasti panašių uždavinių sprendimus, pavyzdžiui, godūs algoritmai ir dinaminis programavimas.

5. Veržlės ir varžtai



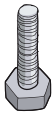
Bebras Benas dirba gamykloje „Bebro konstrukcijos“ prie veržlių ir varžtų surinkimo linijos.

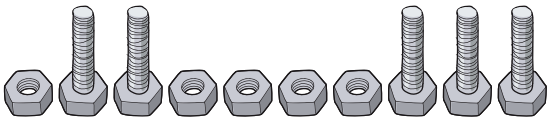
Benas dirba taip:

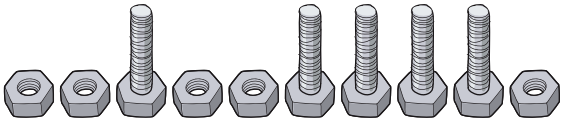
- Benas stovi ilgos konvejerio juostos, kurioje yra veržlių ir varžtų eilė, pradžioje .
- Beno užduotis – nuiminėti nuo konvejerio visas veržles ir varžtus po vieną.
- Jei Benas paima veržlę nuo konvejerio, jis ją įdeda į šalia esantį kibirą.
- Jei Benas paima varžtą nuo konvejerio, jis paima veržlę iš šalia esančio kibiro, užsuka veržlę ant varžto ir junginį padeda ant didelės dėžės.

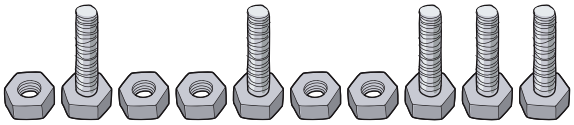
Dirbant Benui gali nutikti dvi nemalonios situacijos:

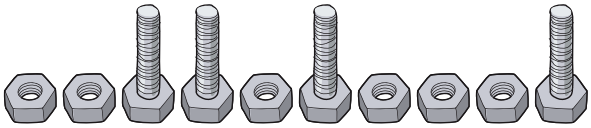
1. Jis paima varžtą nuo konvejerio juostos, o kibire nėra nė vienos veržlės.
2. Konvejerio juosta jau tuščia, o kibire vis dar yra veržlių.

Pradžioje kibiras yra tuščias. Kurioje veržlių  ir varžtų  eilėje Benas, atlikdamas darbą iš kairės į dešinę, nuims visas detales ir nepatirs nemalonios situacijos?

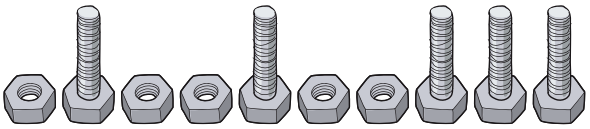
A. 

B. 

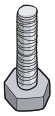
C. 

D. 

Paaiškinimas

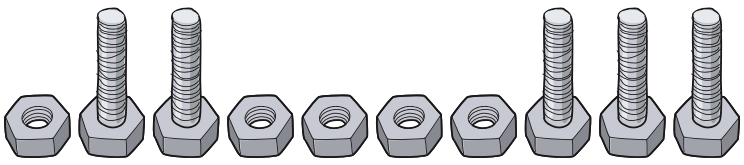
Teisingas atsakymas yra  .

Galime stebėti varžtų ir veržlių judėjimą konvejeriu iš kairės į dešinę.

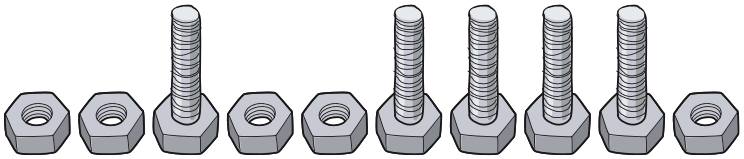
Pažymėkime N =  , B = 

Kibiras	*Konvejeris*
tuščias	N B N N B N N B B B
N	B N N B N N B B B
tuščias	N N B N N B B B
N	N B N N B B B
N N	B N N B B B
N	N N B B B
N N	N B B B
N N N	B B B
N N	B B
N	B
tuščias	tuščias

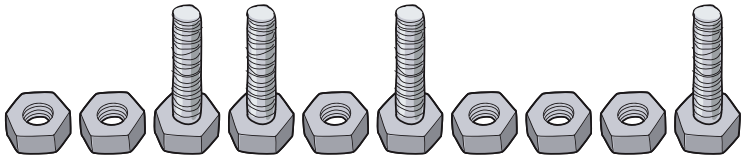
Panagrinėkime kitus atsakymus. Sutarkime nemalonią situaciją, kai Benas negali atlikti jokio veiksmo, vadinti klaida.

A. 

Nuėmus 3-ą detalę (varžtą) bus klaida, nes kibiras bus tuščias.

B. 

Nuėmus paskutinį penktą varžtą bus klaida, nes nuimtos 4 veržlės bus panaudotos ir kibiras bus tuščias.

D. 

Konvejeryje yra 6 veržlės ir 4 varžtai, todėl bus klaida, nes kibire liks nepanaudotos 2 veržlės, nors visos detalės nuo konvejerio bus nurinktos.

Tai informatika!

Šioje užduotyje akcentuojamas nuleidžiamo automato (PDA) naudojimas. PDA – tai būdas aprašyti algoritmą, kuris remiasi dabartine būsena, bet taip pat turi neribotą atmintį – dėklą (angl. *stack*). Šioje užduotyje būsena yra arba veržlė, arba varžtas ant konvejerio juostos, o dėklas – kibiras, kuriame laikomos veržlės.

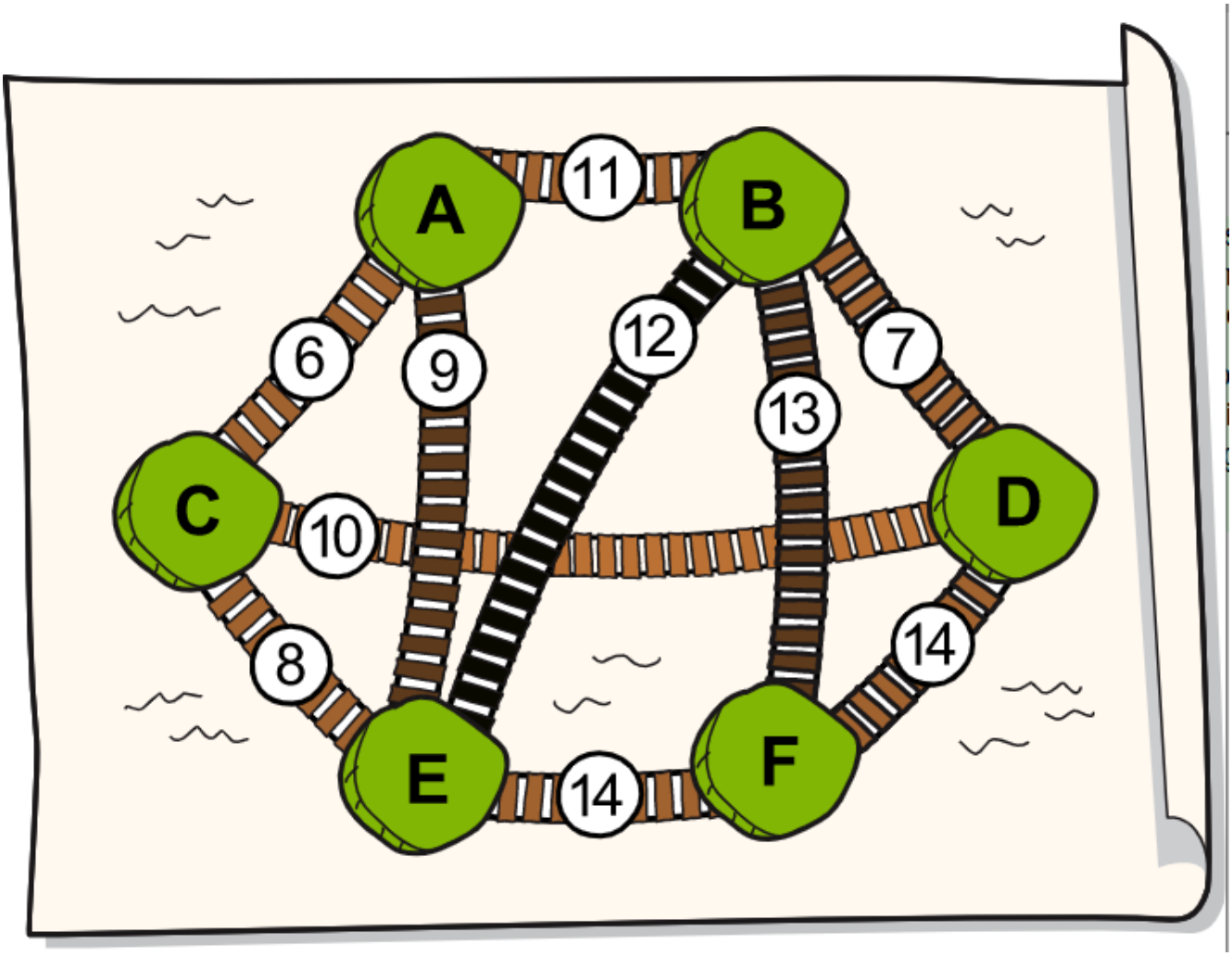
PDA gali būti naudojamas atpažinti arba analizuoti laisvo konteksto kalbas. Atpažinti arba analizuoti kalbą reiškia nustatyti, ar tam tikra simbolių seka priklauso kalbai. Šiuo atveju veržles ir varžtus galime įsivaizduoti kaip subalansuotų skliaustų atvaizdą, čia N=(ir B=). Paprastai subalansuoti skliaustai naudojami aritmetiniuose reiškiniuose. Nesubalansuotų skliaustų sekos gali būti (((() arba ())(. Subalansuotų skliaustų aptikimas svarbus kompiliatoriams, nes daugelyje programavimo kalbų skliaustai naudojami įterptinėms sritims ir aritmetiniams reiškiniams nurodyti.

6. Salų jungimas

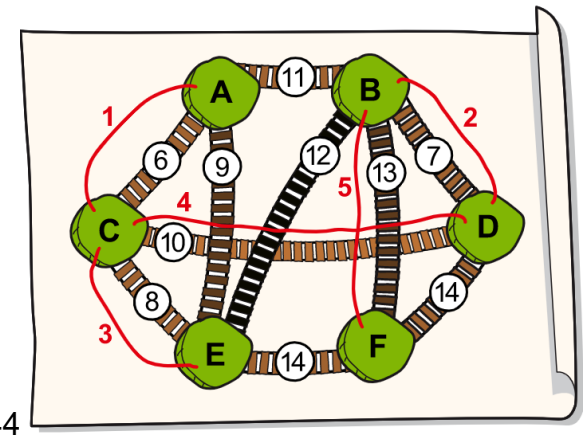
Šešiose salose gyvenanti džiunglių bendruomenė nori sujungti šias salas tiesdami skliautinių tiltų tinklą. Ji sudarė tiltų jungimo planą. Tiltai nesusikerta vienas su kitu. Skaičiai rodo galimo salų jungimo tiltais statybos išlaidas.

Bendruomenė nori sujungti visas salas taip, kad iš bet kurios salos būtų galima keliauti į bet kurią kitą salą tiesiogiai arba netiesiogiai per vieną ar daugiau salų. Taip pat bendruomenė nori statyti tiltus kuo pigiau.

Koks yra pigiausias būdas sujungti visas šešias salas?



Paaiškinimas



6 + 7 + 8 + 10 + 13 = 44

Algoritmas yra paprastas ir intuityvus (žinomas kaip Kruskalio algoritmas):

1. Pradėkime nuo pigiausio tilto AC, kurio kaina 6.

2. Tada renkamės antrą pigiausią tiltą – BD (7).

3. Trečias pigiausias tiltas – CE (8).

4. Kito pigiausio tilto kaina 9. Tačiau jei jį pastatysime, gausime jungtį AECA. Vadinasi, šis tiltas mums visai nereikalingas – salas A, C ir E galime pasiekti nestatydami tilto AE.

5. Ketvirtas statomas tiltas yra CD (10).

6. Kitas tiltas AB, kurio kaina 11, vėl yra jungtis: ABDCA.

7. Panašiai ir tiltas BE (12) sudaro jungtį ACDBEA.

8. Penktas tiltas, kurį turime pastatyti, yra BF (13).

Dabar turime sujungtas visas šešias salas, o penkis jungiamuosius tiltus statome pigiausiu būdu: kiekviename žingsnyje pasirikome pigiausią įmanomą tiltą.

Tai informatika!

Sprendžiant šią užduotį reikia rasti grafo minimalų jungiamąjį medį. Minimalus jungiamasis medis – tai duoto grafo pografis, turintis lygiai tas pačias viršūnes ir kai kurias briaunas, taigi jis irgi yra medis. Žinomas Kruskalio algoritmas visada randa minimalų jungiamąjį medį svertiniam grafui. Algoritmas grindžiamas mažiausią svorį turinčių briaunų pasirinkimu po vieną, vengiant jungčių su jau pasirinktomis briaunomis.

7. Spalvotos žvakutės

Simonas turi skaitmenų nuo 0 iki 9 formos žvakučių. Yra po dvi kiekvieno skaitmens žvakutes. Kiekvienam gimtadieniui jis turi tokį žvakučių komplektą. Žvakutės yra trijų spalvų: oranžinės, raudonos ir mėlynos. Abi skaitmens 0 formos žvakutės yra oranžinės, abi skaitmens 1 žvakutės – raudonos ir t. t. (žr. lentelę). Kasmet per savo gimtadienį Simonas uždeda ant torto tokias žvakutes, kad sudarytų skaičių, kiek metų jam sueina. Šiandien yra 11-asis Simono gimtadienis. Kadangi abi žvakutės ant torto yra tos pačios spalvos, Simonas iš šeimos gauna papildomą dovaną. Kito tokio gimtadienio, kai abi žvakutės bus tos pačios spalvos, Simonui reikės laukti trejus metus, kol jam sukaks 14 metų. Po to – dar trejus metus, kol sueis 17 metų, o tuomet – penkerius metus iki 22-ojo gimtadienio.

Skaitmuo	*Spalva*
0	Oranžinė
1	Raudona
2	Mėlyna
3	Oranžinė
4	Raudona
5	Mėlyna
6	Oranžinė
7	Raudona
8	Mėlyna
9	Oranžinė

Jei Simonas taikytų tokią pačią taisyklę nuo šios dienos iki kol jam sukaks 99 metai, koks būtų ilgiausias laikotarpis metais, kurį Simonui reikėtų laukti tarp gimtadienių, kol abi žvakutės, žyminčios metus, vėl būtų tos pačios spalvos?

A. 5 B. 6 C. 7 D. 8

Paaiškinimas

Teisingas atsakymas yra 5.

Tarkime, kad XY yra dabartinis Simono amžius, čia X ir Y yra skaitmenys nuo 0 iki 9, sudarantys metų skaičių, ir taip pat X ir Y žymi tą pačią spalvą pagal sąlygoje pateiktą lentelę.

Sakykime, kad Y yra vienas iš skaitmenų 0, 1, 2, 3, 4, 5 arba 6. Tuomet Simonui reikėtų laukti trejus metus, kol jam sukaks X(Y+3) metų ir abi žvakutės bus tos pačios spalvos.

Toliau atskirai panagrinėkime atvejus, kai Y yra 7, 8 arba 9.

Jei Y=7, tuomet X, kad taip pat būtų raudona skaitmens žvakutė, turi būti lygus 1, 4 arba 7. Tokiu atveju Simonui tektų laukti 5 metus, kol jam sukaks (X+1)2 metų, kad abi žvakutės vėl būtų tos pačios spalvos.

Jei Y=8, X turėtų būti mėlyna žvakutė, taigi, 2, 5 arba 8, o kitas gimtadienis su tos pačios spalvos žvakutėmis bus po dvejų metų, kai Simonui sueis (X+1)0 metų.

Jei Y=9, X skaitmens žvakutė bus oranžinė (0, 3, 6 arba 9), o Simonas turės laukti dvejus metus, kol jam sueis (X+1)1 metų, ir metų skaičiaus žvakutės vėl bus tos pačios spalvos.

Taigi, didžiausias laukimo laikas yra 5 metai. Simonui niekada neteks laukti 6 ar daugiau metų.

Tai informatika!

Šiame uždavinyje dviženklių dešimtainių skaičių seka susieta su spalvų porų seka. *n*-tasis šios abstrakčios sekos elementas yra dviejų žvakučių, žyminčių skaičių *n*, spalva. Kompiuterijoje dažnai nurodomos sekos sudarymo taisyklės, o tuomet reikia generuoti sekos elementus tol, kol bus pasiektos tam tikros sąlygos. Uždavinyje pasitelkti informatinio mąstymo metodai naudojami bent keliose kompiuterijos srityse, pavyzdžiui, algoritmavime, atvaizdavime ar šablonų atpažinime.

Algoritmai. Šis uždavinys yra algoritmo analizės pavyzdys. Uždavinyje pateikiamas algoritmas, generuojantis spalvų porų seką – tai dviženklius skaičius žyminčių žvakučių spalvų seka. Sugeneruoti visą seką tam, kad nustatytume tik tam tikrą elementą, užtruktų per ilgai. Todėl reikia išanalizuoti algoritmo savybes, galbūt generuojant tik trumpus sekos fragmentus.

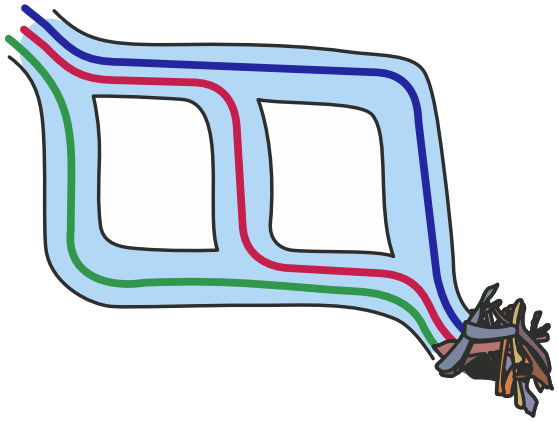
Atvaizdavimas. Šiame uždavinyje skaičiai yra nevienareikšmiškai vaizduojami spalvų poromis. Kitaip sakoma, jog skaičiai susieti su spalvų poromis. Sprendžiant uždavinį, reikia

vis pritaikyti tą patį atvaizdį skirtingiems skaičiams, kad rastume tokius skaičius, kurie vaizduojami tos pačios spalvos žvakučių pora.

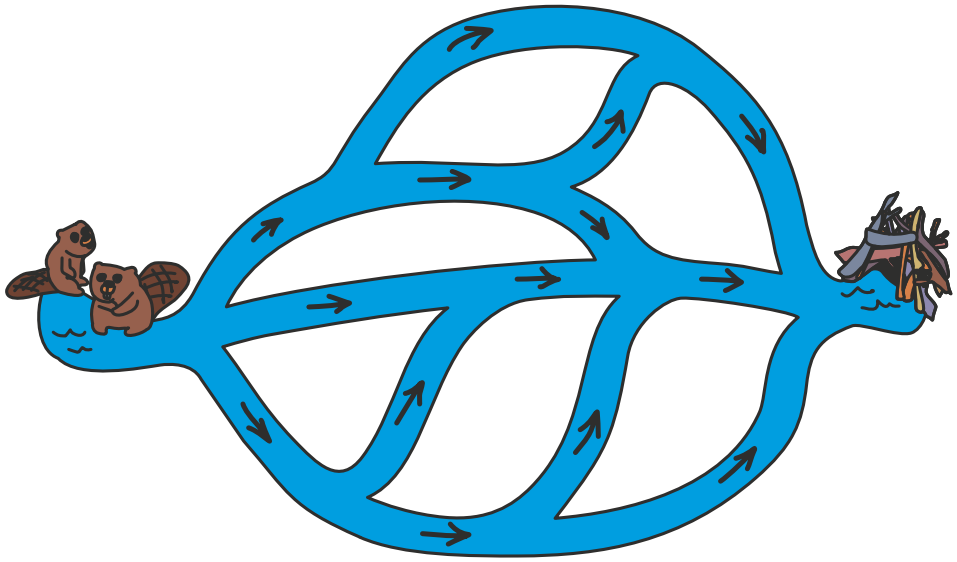
Šablonų atpažinimas. Netiesiogiai šį uždavinį galima laikyti ir šablonų atpažinimo pavyzdžiu. Uždavinyje pateiktas šablonas – spalvotų žvakučių porų seka, spalvos nustatomos paprastu algoritmu. Uždaviniui išspręsti reikia numatyti kitą šablono elementą, kad žinotume, kuris kitas sekos elementas turės mums reikiamą savybę.

8. Daug bebrų

Bebrai yra labai ambicingi: kiekvienas bebras turi turėti savo kelią į namus, tačiau jie tingi plaukti prieš srovę. Pavyzdžiui, jei turime paveikslėlyje pavaizduotą upės struktūrą, nuo pradžios iki pabaigos yra daugiausiai trys skirtingi keliai. Taigi šioje upėje gali gyventi daugiausiai trys bebrai.



Dabar panagrinėkime kitą upę, kurioje gyvena bebrų pora:



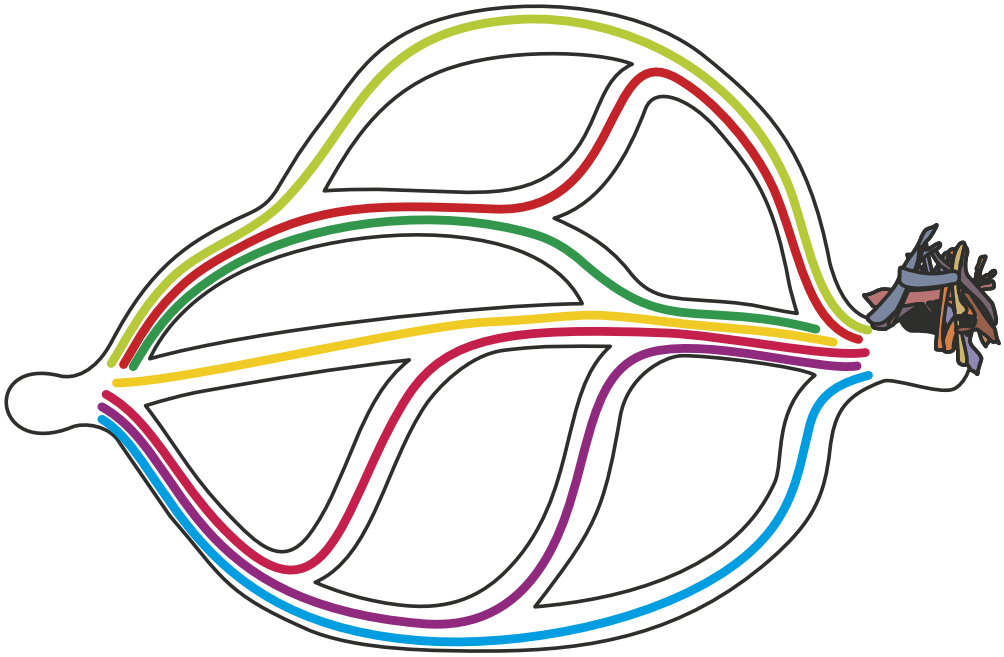
Bebrų pora nusprendė pagausinti savo šeimą. Kiek daugiausiai vaikų jie gali susilaukti, jei kiekvienas šeimos narys nori turėti savo kelią į namus?

Paaiškinimas

Atsakymas: 5

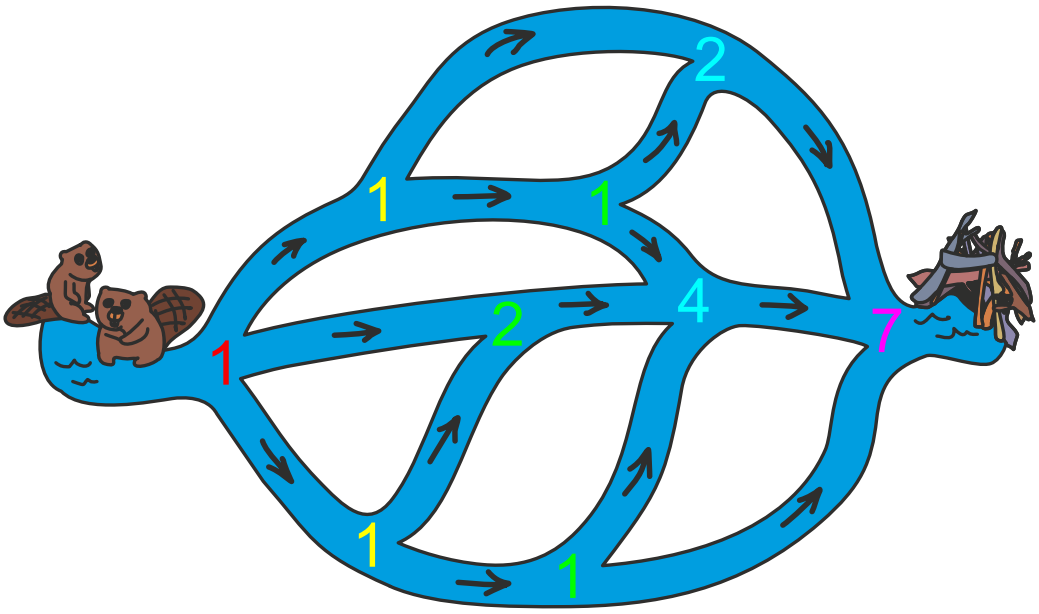
Į bebrų namus veda septyni skirtingi keliai, iš kurių du skirti tėvams. Šiuos kelius galime skaičiuoti dviem skirtingais būdais.

Vienas būdas – ieškoti visų paveikslėlyje esančių kelių ir nupiešti juos skirtingomis spalvomis:



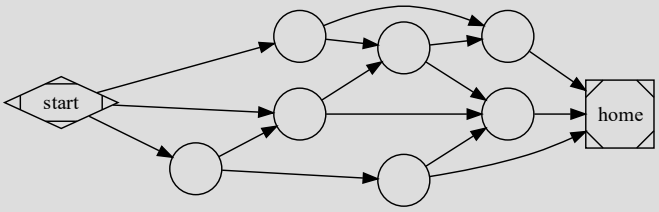
Kitas būdas – naudoti dinaminio programavimo metodą. Ieškome visų paveikslėlyje pavaizduotų sankryžų. Suskaičiuojame, kiek kelių veda į kiekvieną sankryžą, ir užrašome atitinkamą skaičių. Einame iš kairės į dešinę ir sumuojame skaičius ant sankryžų, kurios tiesiogiai sujungtos su dabartine sankryža įeinančiomis rodyklėmis.

Šiuo atveju pradedame nuo to, kad į pradinę sankryžą (raudoną) įrašome 1, nes iš pradinės bebrų padėties yra tik vienas kelias. Tada išsiaiškiname, kokius skaičius rašyti į geltonąsias sankryžas – jie taip pat lygūs 1, nes vienintelė galima ankstesnė sankryža yra raudonai pažymėtoji. Po to apskaičiuojame žaliųjų sankryžų vertes: pavyzdžiui, norėdami gauti žaliąjį 2, sudedame raudonąjį 1 ir geltonąjį 1. Toliau apskaičiuojame mėlynųjų sankryžų vertes, o tada – galutinę sankryžą: 7 gauname sudėję ankstesnių sankryžų skaičius 2, 4 ir 1.



Tai informatika!

Skirtingoms užduotims reikia skirtingai pateikti duomenis. Vienas iš būdų – naudoti grafus. Paveikslėlyje pateiktą vandens takų sistemą galima pavaizduoti orientuotu grafu. Briaunos vaizduoja upės atkarpas, o mazgai – sankryžas. Mūsų užduočiai skirtas grafas atrodo taip:

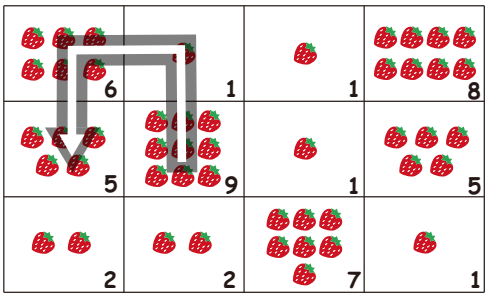


Orientuotasis grafas – tai patogi duomenų struktūra, kuri gali padėti nustatyti, ar galime pereiti iš vieno mazgo į kitą. Kiekvienos upės atkarpos ilgį galime įrašyti į atitinkamą briauną ir gauti svorinį grafą. Svorinius grafus naudojame ieškodami optimalaus kelio tarp pradinės ir galutinės padėties. (Daugiau: https://en.wikipedia.org/wiki/Graph)

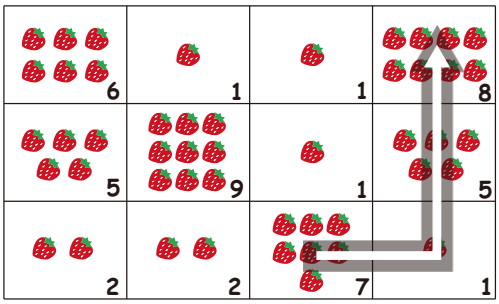
Sisteminis metodas, kai, naudodamiesi ankstesniais etapais, žingsnis po žingsnio kuriame savo sprendimą, vadinamas dinaminiu programavimu. Šis metodas taikomas daugeliui šiuolaikinių problemų spręsti. Dinaminį programavimą galima taikyti tik tuo atveju, kai kiekvienas tolesnis žingsnis priklauso tik nuo esamos situacijos, o ne nuo to, kaip ją pasiekėme. (Daugiau: https://en.wikipedia.org/wiki/Dynamic_programming)

9. Braškės

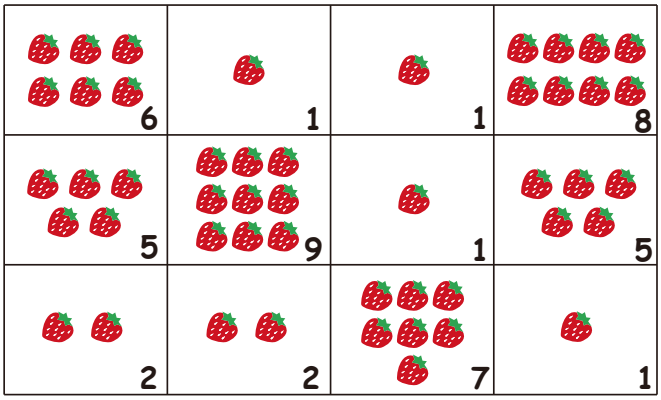
Bebrai mėgsta žaidimą, kurį jie vadina Braškyčių medžiokle. Į kiekvieną tinklelio laukelį dedamas skirtingas skaičius braškių. Bebras gali pradėti nuo bet kurio laukelio ir žengti 3 žingsnius. Žengdamas kiekvieną žingsnį, jis gali pasirinkti gretimą laukelį aukštyn, žemyn, kairėn, dešinėn. Pasirinkęs kelią bebras aplanko keturis laukelius ir suvalgo visas juose esančias braškes. Pirmajame pavyzdyje bebras suvalgys 9 + 1 + 6 + 5 = 21 braškę.



Antrajame pavyzdyje bebras taip pat suvalgys 7 + 1 + 5 + 8 = 21 braškę.



Mažasis Bebras žaidžia šiame tinklelyje:



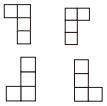
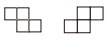
Kiek daugiausiai braškių gali suvalgyti Mažasis Bebras?

Paaiškinimas

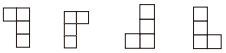
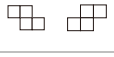
Teisingas atsakymas yra 23.

Nors yra 51 galimas žaidimo pasirinkimas (žr. lentelę toliau), nebūtina tikrinti visų atvejų.

Žinome, kad kiekvienas žaidėjas gali pasiekti 4 iš eilės einančius gretimus laukelius. Šie 4 laukeliai gali sudaryti 14 figūrų:

Figūra	Kvadratas	Atkarpa	L figūra	Horizontali L figūra	Z figūra	Vertikali Z figūra
Forma						
Galimybės	6	3	12	16	8	6

Dabar patikrinsime tik tas figūras, kuriose yra laukelis su 9 arba 8 braškėmis (arba abu laukeliai). Kadangi tikrinant laukelių kombinacijas su 9 arba 8 braškėmis bus tikrinami ir laukeliai su 5, 6 arba 7 braškėmis, todėl bus tikrinami visi laukeliai su didesniu braškių skaičiumi.

	Braškių skaičius, kurį Mažasis Bebras gali suvalgyti		
	kartu su 9 braškių laukeliu	*kartu su 8 braškių laukeliu*	*šių laukų maksimumas*
Kvadratas 	 9 + 5 + 6 + 1 = 21	 8 + 5 + 1 + 1 = 15	21
Atkarpa	 5 + 9 + 1 + 5 = 20	 6 + 1 + 1 + 8 = 16	20
L figūra 	 7 + 2 + 9 + 1 = 19	 8 + 5 + 1 + 7 = 21	21
Horizontali L figūra 	 9 + 1 + 5 + 8 = 23	 9 + 1 + 5 + 8 =23	23
Z figūra 	 5 + 9 + 2 + 7 = 23	 8 + 1 + 1 + 9 = 19	23
Vertikali Z figūra 	 6 + 5 + 9 + 2 = 22	 8 + 5 + 1 + 7 = 21	22

Atkreipkite dėmesį, kad be 8 ir 9 laukelių didžiausia braškių suma bet kuriame kitame keturių laukelių derinyje yra 7 + 6 + 5 + 5 = 23. Taigi ir be 8 ir 9 braškių laukelių negalima viršyti 23. Todėl Mažasis Bebras gali suvalgyti ne daugiau kaip 23 braškes.

Tai informatika!

Daugelyje informatikos uždavinių ieškoma maksimali arba minimali reikšmė. Pavyzdžiui, reikia atlikti didžiausią darbą arba išleisti mažiausią pinigų sumą. Iš galimų variantų reikia rasti tuos, kurių reikšmės yra didžiausios arba mažiausios. Tokie sprendiniai vadinami optimaliais.

Yra įvairių optimalaus sprendinio paieškos metodų.

Vienas iš jų – išvardinti visas galimybes, vieną po kitos apskaičiuoti jų reikšmes ir pasirinkti tas, kurių reikšmės yra geriausios. Šį metodą vadiname grubia paieška (jėgos metodas) arba išsamia (nuodugnia) paieška (angl. *brute-force search*, *exhaustive search*). Tačiau, jei yra labai daug galimybių, šis metodas gali užtrukti labai ilgai. Todėl naudinga išanalizuoti užduotį ir rasti apribojimus, kurie sumažintų galimybių skaičių ir padėtų greičiau rasti optimalų sprendinį. Daugeliui uždavinių informatikos mokslininkai taiko įvairius metodus, padedančius rasti optimalius sprendinius. Yra uždavinių, kurių neįmanoma efektyviai išspręsti.

10. Susiliejantys kaimai

Bėgant metams Kopūstėnų kaimas  , Braškyčių kaimas  ir Morkadvario kaimas  plėtėsi ir pradėjo susilieti. Kai statomas naujas namas, kuriam kaimui jis bus priskirtas gyventojai nusprendžia laikydamiesi tokios taisyklės:

Namas priskiriamas tam kaimui, kuriam priklauso dauguma iš X artimiausių kaimynų. Lygiųjų atveju naujasis namas priskiriamas tam kaimui, kuriam priklauso jo artimiausias kaimynas.

Dabar statomi du nauji namai ir priskiriami kaimams naudojant tą pačią X reikšmę. 1-as namas pastatomas ir priskiriamas anksčiau nei 2-as namas.

Kokia turi būti mažiausia X reikšmė, kad 2-asis namas būtų priskirtas Braškyčių kaimui  ?

Paaiškinimas

Teisingas atsakymas: 5.

Jei X=1, ir 1-as, ir 2-as namas priskiriami Morkadvario kaimui  .

Jei X=2, 1-as namas priskiriamas Morkadvario kaimui  , nes artimiausias iš dviejų kaimynų yra iš Morkadvario kaimo; 2-asis namas taip pat priskiriamas Morkadvario kaimui, nes abu artimiausi jo kaimynai yra iš Morkadvario kaimo.

Jei X=3, tai 1-as namas priskiriamas Kopūstėnų kaimui  , nes du jo kaimynai yra iš Kopūstėnų kaimo, o 2-asis namas priskiriamas Morkadvario kaimui  , nes visi jo kaimynai yra skirtingi, tačiau artimiausias kaimynas yra iš Kopūstėnų kaimo.

Jei X=4, tai 1-as namas priskiriamas Morkadvario kaimui  , kadangi jis turi du kaimynus iš Kopūstėnų kaimo  ir du iš Morkadvario kaimo, tačiau arčiausias yra iš Morkadvario kaimo. Taigi 2-asis namas taip pat priskirtas Morkadvario kaimui, nes jis turi du kaimynus iš Morkadvario kaimo ir du iš Braškyčių kaimo  , tačiau arčiausias yra iš Morkadvario kaimo.

Jei X=5, tai 1-as namas priskiriamas Morkadvario kaimui  , panašiai kaip tuomet, kai X=4, o 2-asis namas priskiriamas Braškyčių kaimui  , nes trys iš penkių jo kaimynų yra iš Braškyčių kaimo.

Kai X=6 arba X=7, tai 2-asis namas taip pat priskiriamas Braškyčių kaimui, tačiau tai nėra mažiausios X reikšmės.

Tai informatika!

Kiekvieno namo priskyrimo kaimui taisyklė yra klasifikavimo algoritmo, vadinamo k artimiausių kaimynų algoritmu (angl. *k-nearest neighbors*, k-NN), pavyzdys. Šis algoritmas kiekvieną naują duomenų elementą klasifikuoja pagal k panašiausių duomenų elementų, kurie jau buvo klasifikuoti. Šiame uždavinyje kintamasis X atlieka k vaidmenį.

Mašininio mokymosi procese k artimiausių kaimynų algoritmas dažnai naudojamas, nes jį lengva įgyvendinti ir nereikia duotiems duomenims pritaikyti sudėtingo modelio.

11. Bebrų duomenys

Bebrų kaime gyvena keliolika šeimų. Bebras informatikas Bronius sukūrė kaimo gyventojų duomenų bazę, įrašydamas duomenis apie kiekvieną bebrą 16 bitų seka nuo b15 (kairėje) iki b0 (dešinėje):

- b15 _ b12: keturi bitai šeimos numeriui;
- b11: vienas bitas lyčiai (0 = moteris, 1 = vyras);
- b10 _ b4: septyni bitai svoriui, išmatuotam kilogramais (kilogramų skaičius);
- b3 _ b2: du bitai darbuotojo kvalifikacijai (00 = namelių statyba, 01 = užtvankų statyba, 10 = maisto sandėliavimas, 11 = jaunų bebrų ugdymas);
- b1 _ b0: du bitai mėgstamam valgiui (00 = medžių žievė, 01 = vandens augalai, 10 = žolės, 11 = viksvos).

Pavyzdžiui, seka 0100 0 0100101 10 01 reiškia, kad tai bebrė ir ji priklauso 4-ai šeimai, yra patelė (moteriškos lyties), sveria 37 kilogramus, yra įgudusi maisto sandėliavimo darbininkė ir mėgsta maitintis vandens augalais.

Informatikas Bronius užklausas duomenų bazėje formuluoja loginiais reiškiniais (0 = *false*, 1 = *true*).

Duotas toks reiškinys:

b11 and not (b10) and b9 and b8 and not (b3 and b2)

Kuris bebrų aprašas atitinka šį reiškinį?

A. Patelės, sveriančios ne mažiau kaip 16 kg, kvalifikuotos maisto sandėliavimo darbuotojos.

B. Patinai, sveriantys ne mažiau kaip 64 kg, kvalifikuoti namelių ar užtvankų statybininkai.

C. Patinai, sveriantys nuo 40 iki 63 kg, kvalifikuoti namelių ar užtvankų statybininkai arba maisto sandėliavimo darbuotojai.

D. Patinai, sveriantys ne daugiau kaip 39 kg, kvalifikuoti užtvankų statybininkai.

Paaiškinimas

Teisingas atsakymas: C.

b11 = 1 reiškia „vyras“ (patinas), b10 = 0 reiškia „sveria ne daugiau kaip 63 kg“, b9 = 1 ir b7 = 1 reiškia „sveria ne mažiau kaip (32 + 8) kg“, b3 = 0 arba b2 = 0 rodo, jog neįtraukta tik jaunų bebrų ugdymas (b3 = 1 ir b2 = 1).

(De Morgano dėsnis nusako not (b3 and b2) ekvivalentiškumą reiškiniui not (b3) or not (b2)).

Atsakymas A neteisingas, nes jau pirmas reiškinio elementas b11 = 1 netinka, todėl ir visas reiškinys, nepriklausomai nuo kitų elementų, yra neteisingas (false).

Atsakymas B neteisingas, nes b2 = 0, kas reiškia, jog tinka ne tik kvalifikuoti namelių arba užtvankų statybininkai, bet ir maisto sandėliavimo darbuotojai.

Atsakymas D neteisingas, nes neteisingai aprašyta ne tik kvalifikacija, bet ir svoris, nes b9 = 1 ir b7 = 1 reiškia „sveria ne mažiau kaip 40 kilogramų“.

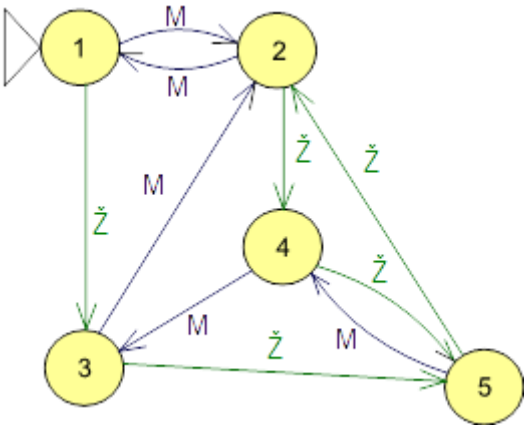
Tai informatika!

Aprašytoje paprastos duomenų bazės užklausoje gali būti naudojamas loginis reiškinys (teiginių logikos formulė), t. y., jis gali būti laikoma paieškos raktu. Paprastai tokia formulė identifikuoja duomenų poaibį (tokias situacijas galima grafiškai pavaizduoti Veno diagramomis). Pavyzdžių galima rasti 1943–1944 m. Konrado Cūzės (Konrad Zuse) darbuose. (Konradas Cūzė buvo vokiečių statybų inžinierius, tačiau didžiausias jo pasiekimas – pirmasis programa valdomas elektromechaninis kompiuteris, pradėjęs veikti 1941 m.)

Dauguma šiuolaikinių programavimo kalbų turi unarinį loginį operatorių *not* ir dvinarinius loginius operatorius *and*, *or*. Daugelio kompiuteriuose naudojamų skaitmeninių komponentų (multipleksorių, sumatorių, aritmetinių loginių blokų ir t. t.) pagrindas yra loginės grandinės (schemos), sudarytos iš atitinkamų loginių elementų. 1913 m. H. Shefferis (Henry Maurice Sheffer) parodė, kad visus loginius operatorius galima išvesti iš vieno (*nand*): *A nand B* yra ekvivalentus *not (A and B)*, taigi *A nand A* yra ekvivalentus *not (A)*. 1880 m. Čarlzas Pirsas (Charles Sanders Peirce) pirmasis tyrinėjo loginius reiškinius, siedamas juos su semiotika. Kaip ir *nand* elementai, *nor* elementai (*A nor B* ekvivalentus *not (A or B)*, taigi *A nor A* ekvivalentus *not (A)*) yra universalūs loginiai elementai; juos galima sujungti į bet kokius kitus loginius elementus. Daugelis elektroninių grandinių buvo sukurtos tik iš *nand* arba *nor* elementų.

12. Muzikos instrumentas

Bebras sukūrė ypatingą muzikos instrumentą, kuriuo garsai išgaunami tik trimis klavišais: raudonu (R), mėlynu (M) ir žaliu (Ž).



Šiais klavišais išgaunamos 5 skirtingos natos: 1, 2, 3, 4 ir 5.

Pradedant groti instrumentu visada pirmiausia nuspaudžiamas raudonas klavišas R ir suskamba pirmoji nata 1. Kita nata išgaunama priklausomai nuo prieš ją suskambėjusios natos.

Paveiksle pavaizduota galima natų skambėjimo schema. Žinoma, kad nuspaudus klavišą R, visada skamba nata 1. Teisinga iš natų skambėjimo sukurta melodija visuomet baigiasi skambant dviem natomis 1.

Pavyzdžiui, atliekant instrumentu seką „R-Ž-M-M-R“, skamba tokios natos: 1-3-2-1-1, o atliekant seką „R-Ž-M-R-M“ skamba tokios: 1-3-2-1-2.

Kuria seka galima išgauti teisingą melodiją?

A. R-M-M-Ž-M-R

B. R-Ž-Ž-Ž-M-R

C. R-M-Ž-M-Ž-R

D. R-Ž-Ž-M-Ž-R

Paaiškinimas

Teisingas atsakymas: B.

Atsakymą galima surasti nagrinėjant schemos veikimą.

Tačiau galima nagrinėti ir sekas. Kiekvienoje sekoje klavišas R spaudžiamas du kartus – pradžioje ir pabaigoje. Diagramoje nėra rodyklės, kuri rodytų į 1 ir būtų žalios spalvos. Taigi galima atmesti C ir D, nes šios sekos nesibaigia du kartus nata 1.

Pirmoje sekoje (atsakymas A) priešpaskutinė skamba nata 2, prieš paspaudžiant klavišą R.

Atsakymas B yra teisingas, nes šia klavišų seka bus gaunama tokia melodija: 1-3-5-2-1-1.

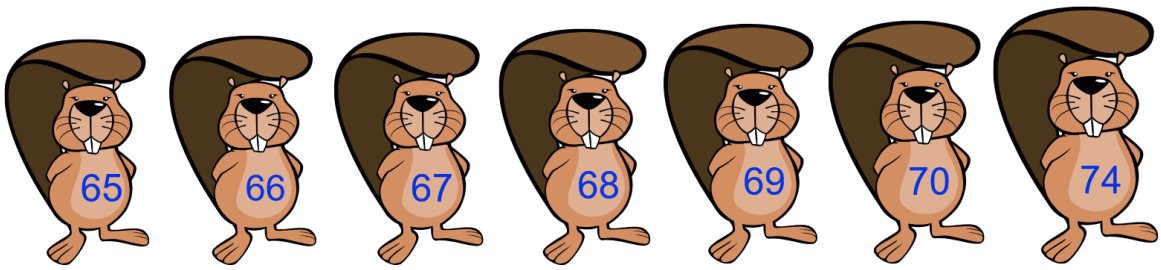
Tai informatika!

Ši užduotis remiasi baigtinio automato idėja.

Daugelį paprastų ir labai sudėtingų uždavinių galima išspręsti naudojant baigtinio automato idėją. Baigtinio automato veikimo principas dažnai naudojamas programavimo kalbų kompiliatoriuose – programose, skaitančiose ir tikrinančiose kompiuterio kodo teisingumą.

Geras baigtinio automato pavyzdys buityje yra kavos aparatas. Prieš gamindamas kavą jis turi sulaukti būsenos, kai yra įdėta pakankamai monetų. Monetos įdėjimas gali turėti skirtingą poveikį aparatui: jis arba suskaičiuos, kiek sumokėta, ir lauks, arba parodys, kad sumokėta pakankamai ir padarys kavos. Tas pats veiksmas, skirtingas poveikis – priklausomai nuo to, kas buvo padaryta prieš tai (kaip ir šioje užduotyje aprašyto muzikos instrumento atveju).

13. Bebrų dirbtinis intelektas



Bebrai konstruoja dirbtinio intelekto sistemą. Ši sistema skirta matuoti gyvūno dydį ir tuo remiantis nuspręsti, ar tas gyvūnas yra bebras, ar ne. Dirbtinio intelekto sistema mokosi priimti sprendimus iš jai pateikiamų pavyzdžių.

Pirma, dirbtinio intelekto sistema mokosi iš tokių dydžių gyvūnų pavyzdžių:

- 65, 66, 67, 68, 69 => bebras
- 11, 101, 110, 120, 130 => nėra bebras

Tada bebrai leidžia dirbtinio intelekto sistemai spręsti. Sistema pateikia tokį atsakymą:

- 70, 74 => bebras
- 86, 38 => nėra bebras
- 40, 80 => bebras

Dirbtinis intelektas padarė klaidą, kadangi 40 ir 80 dydžio gyvūnai nėra bebrai! Dirbtinis intelektas aptiko, kad gyvūnas, kurio dydis 11, nėra bebras, o gyvūnas, kurio dydis 65, yra bebras. Žiūrint į gyvūnų dydžių skirtumus, dirbtinio intelekto sistema padarė išvadą, kad bebrai yra tik tie gyvūnai, kurie didesni už 38 ir mažesni kaip 85.

Dirbtinio intelekto sistemai pagerinti bebrai pateikė jai dar vieną pavyzdį: gyvūnas, kurio dydis 42, nėra bebras.

Kaip dabar dirbtinio intelekto sistema klasifikuos du gyvūnus, kurių dydžiai 48 ir 84?

A. bebras, bebras

B. bebras, nėra bebras

C. nėra bebras, bebras

D. nėra bebras, nėra bebras

Paaiškinimas

Teisingas atsakymas – C.

Kaip dirbtinio intelekto (DI) sistema pasirenka dydžių intervalą 38–85 bebrams identifikuoti? Iš tiesų, tiksliai nežinome. Bet kadangi tarp DI nagrinėtų pavyzdžių „11 – ne bebras“, o „65 – bebras“, DI paskaičiavo ribą, kada gyvūnas yra bebras, kada – ne, ir nustatė ją kažkur tarp 11 ir 65. Tinkamas taškas ribai gali būti vidurkis (11+65)/2 = 38. Panaši taisyklė gali būti taikoma kitam intervalo galui rasti: (69+101)/2 = 85. Naudojant tą pačią procedūrą, pamačius pavyzdį „42 nėra bebras“, riba 85 neturėtų pasikeisti, bet nauja kairioji riba turėtų būti (42+65)/2 = 53,5. Taigi gyvūnas, kurio dydis 48, būtų identifikuotas, jog „nėra bebras“, o 84 dydžio gyvūnas būtų laikomas bebru.

Tai informatika!

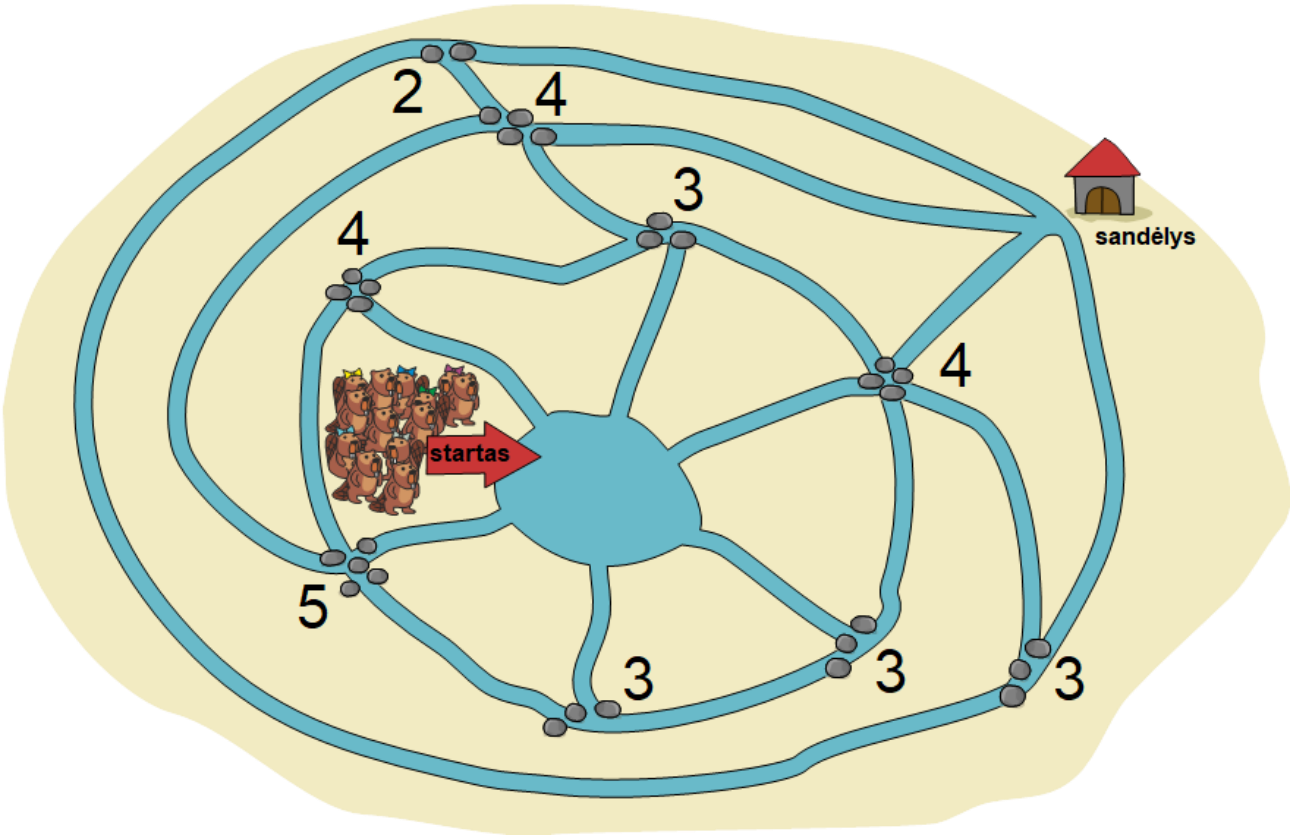
Šios užduoties dirbtinio intelekto sistemoje taikomas mašininio mokymosi algoritmas. Mašininio mokymosi srityje informatikai tyrinėja kompiuterių algoritmus, kurie galėtų automatiškai tobulinti savo sprendimų priėmimo procesą. Mašininio mokymosi algoritmai formuoja modelį remdamiesi pavyzdžių duomenimis, vadinamais apmokymo duomenimis, tam, kad prognozuotų arba priimtų sprendimus be tiesioginio programavimo tai daryti. Tokių algoritmų kokybė labai priklauso nuo naudotų apmokymo duomenų kokybės, – tai galime matyti šioje užduotyje.

Mašininio mokymosi algoritmų taikymo spektras labai didelis. Jie naudojami medicinoje, el. pašto laiškų filtravimui, šnekos atpažinimui, kompiuterinei regai, – tada, kai sudėtinga arba netikslinga projektuoti tradicinius algoritmus numatytiems uždaviniams atlikti.

Mašininis mokymasis laikomas priklausančiu dirbtinio intelekto sričiai. Mašininio mokymosi veikimo principą galite išbandyti interneto svetainėse, pvz., https://machinelearningforkids.co.uk arba „Dirbtinis intelektas vandenynui“ (AI for Oceans).

14. Akmenys

Bebrų šeimynos sodyboje yra iš 21 kanalo susidedanti vandens sistema. Kai kurių kanalų sankirtose yra akmenų. Iš viso yra 31 akmuo ir juos visus bebrai, plaukiodami kanalais, turi sunešti į sandėlį.



Akmenys sunkūs, todėl vienu metu bebras gali plukdyti tik vieną ar du akmenis, bet ne daugiau. Kad nukeliautų nuo vienos kanalų sankirtos iki kitos, bebrai visada plaukia lygiai vieną valandą. Pradėję vienu metu bebrai turi per keturias valandas sunešti į sandėlį visus akmenis.

Koks mažiausias bebrų skaičius gali šį darbą atlikti?

14

18

20

24

Paaiškinimas

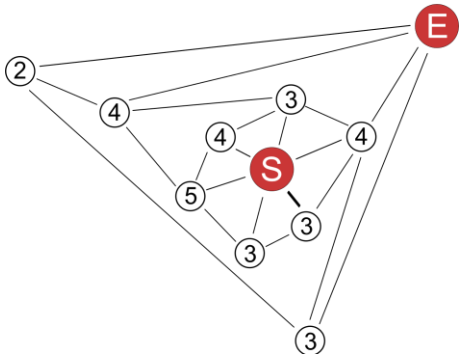
Teisingas atsakymas: 14 bebrų.

Trumpas paaiškinimas

Yra tik vienas pasirinkimas, kai pradėję plaukti iš starto vietos bebrai pasiekia sandėlį per 2 valandas. Šiuo atveju kanalų sankirtoje yra 4 akmenys. Šiuos 4 akmenis iki sandėlio gali nunešti 2 bebrai, kurie gali grįžti, ir vėl nunešti dar 4 akmenis iki sandėlio. Lieka 23 (31–8) akmenys ir reikia dar mažiausiai 12 bebrų, nes likusiems akmenims nunešti reikia 3 arba 4 valandų, tad nebelieka laiko atnešti daugiau akmenų. Likusius 23 akmenis į sandėlį gali atnešti 12 bebrų, iš kurių 11 neša po 2 akmenis, o vienas – tik vieną akmenį.

Detalus paaiškinimas

Iš pradžių visi akmenys yra kanalų sankirtose. Vienoje sankirtoje yra ir sandėlis (E). Paveikslėliuose kanalai vaizduojami tiesiomis linijomis, o skaičiai reiškia akmenų skaičių atitinkamoje sankirtoje.

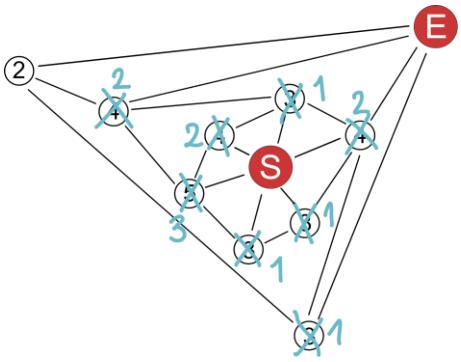


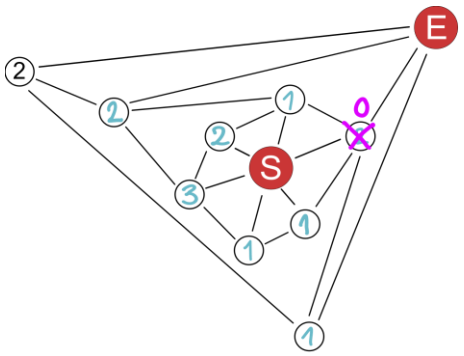
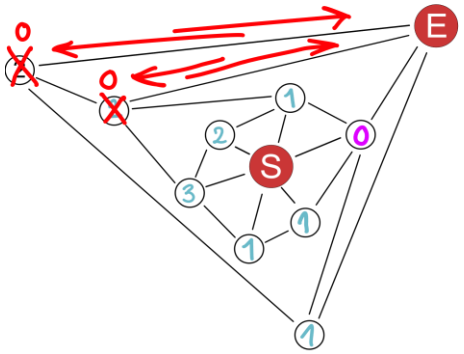
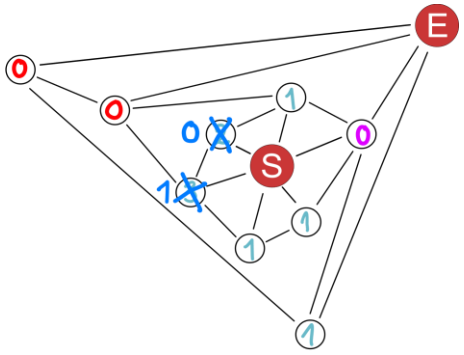
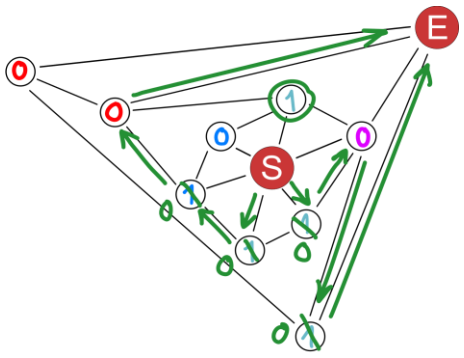
Kelio trukmę tarp pradinio taško ir sandėlio matuojame skaičiuodami kanalus, kuriais bebrai turi plaukti.

Nėra atvejų, kai vienas kanalas jungia startą ir sandėlį (t. y., nėra 1 valandos trukmės kelių).

Yra 2 valandų, 3 valandų ir 4 valandų trukmės kanalais keliai.

Bebrai akmenis turi rinkti ne ilgiau kaip keturias valandas. Ilgesnių nei keturių valandų trukmės kelių nagrinėti nereikia.

Pirmiausiai nagrinėjame visas kanalų sankirtas, kuriose yra bent trys akmenys. Kiekvienoje tokioje sankirtoje bebras paima du akmenis ir atneša juos į saugyklą. Tam reikia 8 bebrų. Jie surenka 16 akmenų. Dar lieka 31–16 = 15 akmenų. Visi bebrai, išskyrus vieną, užtrunka kelyje mažiausiai tris valandas. Jie negali grįžti prie kanalų ir paimti daugiau akmenų (t. y., padirbėti pakartotinai). Vienas bebras per dvi valandas surenka du akmenis. Jis turi pakankamai laiko grįžti iki sankirtos, paimti dar du	

akmenis ir nunešti juos į sandėlį. Todėl šis bebras gali padirbėti pakartotinai.	
Kitas bebras per dvi valandas surenka taip pat du akmenis. Šiuo bebru taip pat galima pasinaudoti pakartotinai. Dar lieka 15–2 = 13 akmenų. Jau pasinaudojome 9 bebrais. Du iš jų dar gali padirbėti pakartotinai, nes jie sugaišo tik po dvi valandas.	
Jau dviem bebrais iš pasirinktų 9 bebrų pasinaudojome pakartotinai, kiekvienas iš jų atnešė po du akmenis. Lieka 13–4 = 9 akmenys. Tačiau jau atidirbusių bebrų skaičius toks pat – 9. Nė vienu iš jų jau nebegalime pasinaudoti.	
Toliau taikoma ta pati strategija ir nagrinėjamos visos kanalų sankirtos, kuriose yra bent du akmenys. Reikalingi 2 bebrai. Jie surenka 4 akmenis. Dar lieka 9–4 = 5 akmenys. Jau pasinaudota 9+2 = 11 bebrų.	
Galutinis žingsnis – rasti sprendimą (galbūt bandymų ir klaidų metodu), kuris leistų naudoti kuo mažiau bebrų. Dar du bebrai surenka po du akmenis. O paskutinis bebras paima paskutinį akmenį. Geriausias sprendimas yra A: 11+3 = 14 bebrų.	

Yra 31 akmuo, tik du bebrai gali atnešti po 2 akmenis ir grįžę pasiimti daugiau, kiti bebrai gali atnešti ne daugiau kaip 2 akmenis, todėl reikia mažiausiai 13,5 bebro. Geriausiam sprendimui reikia 14 bebrų.

Tai informatika!

Pirmasis veiksmas – atrinkti informaciją, kuri būtina uždaviniui išspręsti, ir atsisakyti perteklinės.

Reikšmingą informaciją sudaro:

- Kanalai ir kanalų sankirtos (akmenys, starto vieta ir sandėlis).
- Matematiniai apribojimai: bebras nuo vienos kanalų sankirtos iki kitos plaukia lygiai 1 valandą; bebrai turi sunešti akmenis į sandėlį per 4 valandas; jie turi surinkti 31 akmenį; jie gali vienu metu nešti vieną arba du akmenis.
- Reikia surinkti visus 31 akmenį naudojant kuo mažiau bebrų.

Tiksli kanalų sistema ir jų išsidėstymas, bebrų atsiradimas reikiamoje sankirtoje ir panaši informacija yra perteklinė. Šią informaciją galima praleisti.

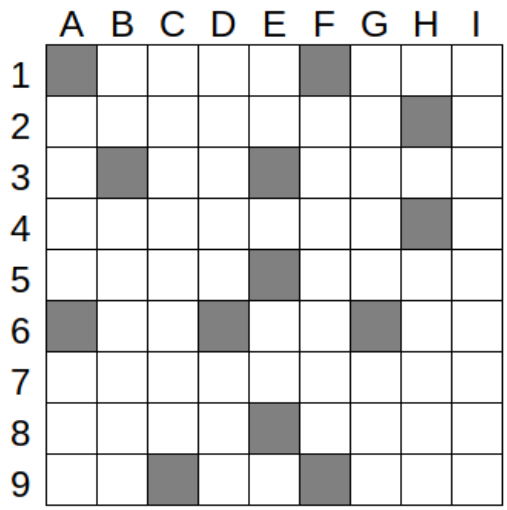
Informatikoje toks formalus uždavinio vaizdavimas vadinamas grafu. Grafą sudaro mazgai ir briaunos tarp mazgų. Atstumą tarp dviejų mazgų galima išreikšti keliais būdais, pavyzdžiui, suskaičiuojant, kiek briaunų turime pereiti, kad patektume nuo vieno mazgo iki kito. Grafai yra viena iš pagrindinių informatikos uždavinių sprendimo metodų. Navigacijos sistemos yra populiarus taikomųjų programų, kuriose naudojami grafai, pavyzdys.

Šioje užduotyje prašoma surinkti visus akmenis. Gali būti, kad yra daugiau nei vienas būdas šiam tikslui pasiekti. Galima įsivaizduoti, kad reikia pasinaudoti 14, 18, 20 ar net 31 skirtingais bebrais. Tačiau užduotis aiškiai reikalauja rasti sprendimą, kuriam būtų panaudota kuo mažiau bebrų.

Uždaviniai, kurie reikalauja geriausio įmanomo sprendimo, labai dažni. Informatikoje jie vadinami optimizavimo uždaviniais. Pavyzdžiui, tikimasi, kad navigacinė sistema parinks geriausią (pvz., optimalų) būdą nuvykti iš taško A į tašką B. Žinoma, tikslią geriausio sprendimo reikšmę reikia apibrėžti a *priori*. Navigacinės sistemos atveju kriterijumi galėtų būti atstumo kilometrais arba laiko, kurio reikia automobiliui pasiekti tikslą, minimizavimas.

15. Roboto kelias

Dronas pradeda kelionę baltame pavaizduotos lentelės langelyje. Jis stovi pasisukęs viena iš keturių galimų judėjimo krypčių.



Dronas aplanko 8 kitus baltus langelius tokia tvarka:

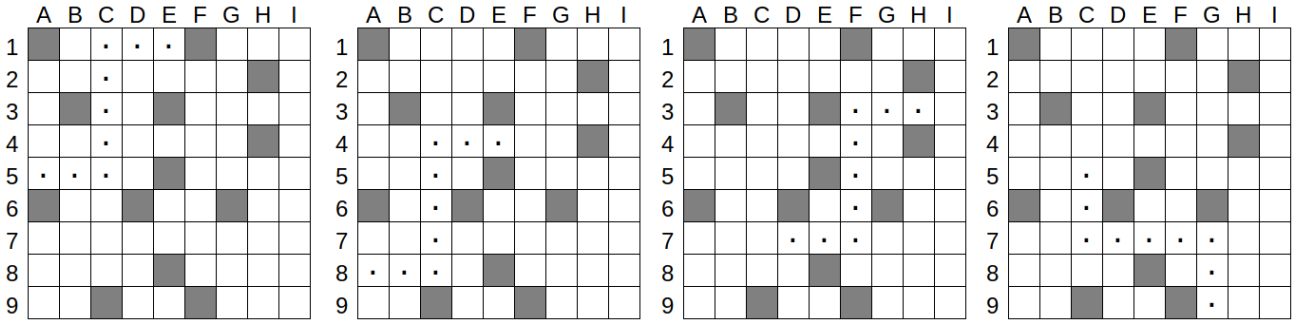
1. Pajuda per 2 langelius pirmyn;
2. Pasisuka 90 laipsnių kampu kairėn (esamame langelyje);
3. Pajuda per 4 langelius pirmyn;
4. Pasisuka 90 laipsnių kampu dešinėn (esamame langelyje);
5. Pajuda per 2 langelius pirmyn.

Kiek lentelėje yra langelių, iš kurių dronas galėtų pradėti tokią kelionę?

Paaiškinimas

Teisingas atsakymas yra 8.

Spręsdami uždavinį, pirmiausia išsiaiškinkime, jog yra keturi galimi drono judėjimo keliai:



Šiuos kelius, ko gero, lengviausia rasti ieškant penkių gretimų baltų langelių, o radus tikrinant, ar šios atkarpos galuose galima statmenai penkių langelių atkarpai į priešingas puses pridėti dar po du gretimus baltus langelius. Kad nesuklystumėte tikrindami, geriausia tai daryti nuosekliai – pavyzdžiui, nuo viršutinio kairiojo lentelės kampo iki apatinio dešiniojo lentelės kampo.

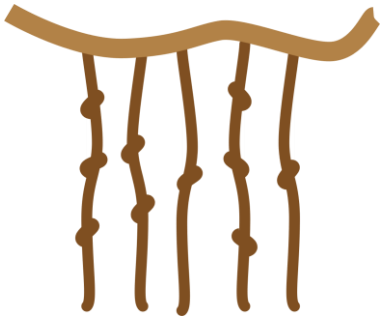
Drono judėjimo kelias yra simetriškas, tad dronas gali judėti nuo bet kurio kelio galo. Taip gauname 8 langelius, iš kurių dronas gali pradėti kelionę.

Tai informatika!

Šis uždavinys – tai paprastas sukimo atžvilgiu invariantinių paveikslų (šablonų) atitikties pavyzdys. Šis metodas yra tik maža kur kas platesnės kompiuterijos srities – paveikslų apdorojimo – dalis. Uždavinio sąlygoje yra aprašytas šablonas (1–5 žingsniai), kurio reikia ieškoti, o paveikslas, kuriame to šablono ieškoma – tai bandymų tinklelis. Šablonų atitiktis taikoma įvairiose srityse, pavyzdžiui, veido atpažinimo technologijose ar medicininėms nuotraukoms apdoroti. Ieškomas šablonas kartais yra vadinamas keliu.

16. Kipu

Karalienė riša mazgus ant kabančių virvių (vadinamų *kipu*), norėdama pranešti naujienas savo karalystei. Pavyzdžiui, pateikta *kipu* reiškia „Švęskime“.



Pranešimo reikšmė priklauso nuo virvių išsidėstymo ir mazgų skaičiaus ant kiekvienos virvės.

Ant kiekvienos virvės gali būti 0, 1, 2 ar 3 mazgai.

Karalienė turi 50 skirtingų pranešimų, kuriuos gali perduoti virvėmis.

Kiek mažiausiai virvių reikia karalienei?

2

3

4

5

Paaiškinimas

Jei būtų tik viena virvė, tuomet būtų galima perduoti 4 skirtingus pranešimus, nes ant virvės gali būti 0, 1, 2 arba 3 mazgai. Dvejomis virvėmis, kurių kiekviena gali turėti 4 mazgų variantus, būtų galima perduoti 4×4=16 skirtingų pranešimų. Tačiau to dar nepakanka perduoti visus karalienės pranešimus. Jei pridedama trečia virvė, galima perduoti 4×4×4=64 skirtingus pranešimus. Kadangi 64 yra daugiau nei 50, trijų virvių pakanka, kad perduoti visus karalienės pranešimus.

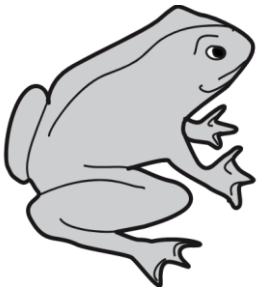
Tai informatika!

Ši užduotis yra pozicinės sistemos pavyzdys. Sudarant *kipu* svarbu virvės padėtis ir mazgų skaičius ant kiekvienos virvės. Kadangi kiekvieną virvę galima surišti keturiais būdais, *kipu* yra ketvirtainės sistemos pavyzdys.

Kai žmonės atlieka skaičiavimus, dažniausiai naudojama dešimtainė sistema. Dešimtainiai skaitmenys (nuo 0 iki 9) yra kaip mazgai ant *kipu*, o skaitmenų pozicija (atitinkanti 10 keliant laipsniais) yra kaip virvių padėtis *kipu*. Pavyzdžiui, 427=4×100+2×10+7, su trimis dešimtainiais skaitmenimis galime sudaryti 10×10×10=1000 skirtingų natūraliųjų skaičių (0–999).

Kompiuteriai naudoja dvejetainę sistemą. Tokioje sistemoje kiekvienas skaitmuo (kompiuterinėse sistemose vadinama bitais) gali turėti vieną iš dviejų reikšmių: 0 arba 1. Visi kompiuterio duomenys ir nurodymai yra saugomi remiantis šiuo principu.

17. Varlės spalvinimas



Kompiuterio ekrane pasirodo varlės piešinys. Varlės spalva nenurodyta. Spalvai pakeisti yra keturios programos, tačiau paleisti galima tik vieną iš jų. Kiekvieną programą sudaro komandų **jei-tai-priešingu atveju** arba **jei-tai** seka; kiekvienas kita komanda turi būti vykdomas po to, kai baigiama prieš tai buvusi.

Įvykdžius programą varlė turi būti žalios spalvos, nepriklausomai nuo to, kokios spalvos ji buvo prieš programos paleidimą.

Tik vienu atveju, kai pradžioje varlė yra tam tikros spalvos, įvykdžius vieną (ir tik vieną) iš keturių programų, varlė nebus nuspalvinta žalia spalva.

Kuri programa, ją įvykdžius, nenuspalvina varlės žaliai?

A **jei** varlė yra raudona, **tai** nuspalvina geltonai, **priešingu atveju** nuspalvina žaliai;
jei varlė yra geltona, **tai** nuspalvina raudonai;
jei varlė ne geltona, **tai** nuspalvina žaliai.

B **jei** varlė raudona, **tai** nuspalvina geltonai;
jei varlė ne raudona, **tai** nuspalvina žaliai;
jei varlė geltona, **tai** nuspalvina raudonai, **priešingu atveju** nuspalvina žaliai.

C **jei** varlė geltona, **tai** nuspalvina žaliai;
jei varlė ne geltona, **tai** nuspalvina raudonai;
jei varlė raudona, **tai** nuspalvina žaliai, **priešingu atveju** nuspalvina geltonai.

D **jei** varlė geltona, **tai** nuspalvina žaliai, **priešingu atveju** nuspalvina raudonai;
jei varlė raudona, **tai** nuspalvina žaliai **priešingu atveju** nuspalvina geltonai.

Paaiškinimas

Teisingas atsakymas – programa D.

Jei iš pradžių varlė yra geltona, įvykdžius pirmąją komandą *jei-tai-priešingu atveju* ji nuspalvinama žaliai, po to sąlyga „varlė yra raudona" tampa klaidinga, todėl ji vėl nuspalvinama geltonai.

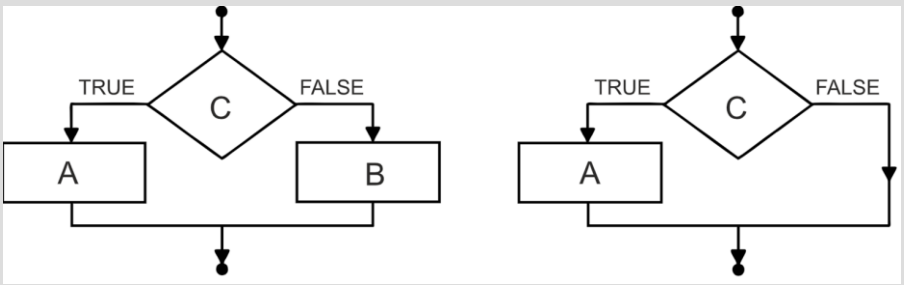
Atliekant programą A, įvykdžius komandą *jei-tai-priešingu atveju*, varlė nuspalvinama arba geltonai, arba žaliai; įvykdžius antrąjį sakinį ji nuspalvinama arba raudonai, arba žaliai, po to sąlyga „varlė ne geltona" yra teisinga, todėl ji bus nuspalvinta žaliai.

Atliekant programą B, įvykdžius antrąją komandą, sąlyga „varlė ne raudona" tikrai bus teisinga, todėl varlė nuspalvinama žaliai, o po trečios komandos (*jei-tai-priešingu atveju*) vėl nuspalvinama žaliai.

Atliekant programą C, įvykdžius antrąją komandą, sąlyga „varlė ne geltona" tikrai bus teisinga, todėl varlė bus nuspalvinta raudonai, o po trečios komandos (*jei-tai-priešingu atveju*) nuspalvinama žaliai.

Tai informatika!

Pagal imperatyvinio programavimo paradigmą (taip pat ir objektinio programavimo atveju) galite parašyti sąlyginių sakinių (komandų) seką, kurie turi būti vykdomi tam tikra tvarka. Sąlyginis sakinys turi formą **jei C, tai A, priešingu atveju B** arba formą **jei C, tai A**, kaip pavaizduota schemose:



Atkreipkite dėmesį, kad sakinių sekos

jei C, **tai** A, **priešingu atveju** B

ir

je C, **tai** A;

jei ne C, **tai** B

nėra ekvivalenčios. Tai reiškia, kad jų rezultatas ne visada sutampa.

Iš tikrųjų, antruoju atveju (kai sąlyga tikrinama du kartus), įvykdžius sakinį A sistemos būsena gali pasikeisti taip, kad sąlyga C taps klaidinga ir iš karto po to būtų įvykdytas sakinys B. Tačiau pirmuoju atveju vykdomas tik vienas iš dviejų sakinių.

Nagrinėjamoje užduotyje skirtingas rezultatas gaunamas vykdant 4 ir 3 programas.

Daroma prielaida, kad sąlygos C tikrinimas nekeičia sistemos būsenos ir kad kiekvieno sakinio vykdymas galiausiai baigiasi.

Bet kuriuo atveju, sakinys

jei C, **tai** A, **priešingu atveju** B

gali būti keičiamas į seką

nauja_sąlyga := C;

jei nauja_sąlyga, **tai** A;

jei ne nauja_sąlyga, **tai** B

čia *nauja_sąlyga* yra „naujas“ *loginis* kintamasis, kuriam pirmoji eilutė priskiria loginę C reikšmę.

Funkcinio programavimo paradigmoje nėra komandų; bet kokią apskaičiuojamą funkciją galima nurodyti naudojant sąlyginius reiškinius ir rekursiją. Sąlyginis reiškinio pavidalas yra toks: jei C, tai E1, priešingu atveju E2,

čia E1 ir E2 yra to paties tipo reiškiniai.

Pavyzdžiui, *n*!, natūraliojo skaičiaus *n* faktorialą, galima apskaičiuoti patikrinus sąlyginį reiškinį

jei $n = 0$, tai 1, priešingu atveju $n * (n - 1)!$

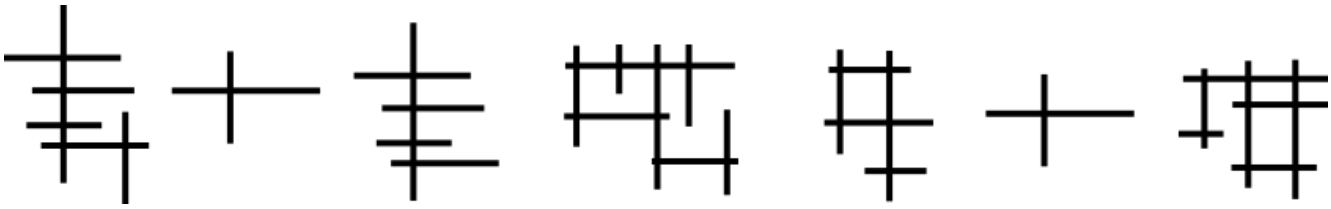
Atkreipkite dėmesį, kad naudojama rekursija. Priešingo atvejo sąlygos negalima praleisti, nes rezultatas turi būti nurodytas ir tada, kai sąlyga netenkinama.

18. Dienoraščio paslaptis

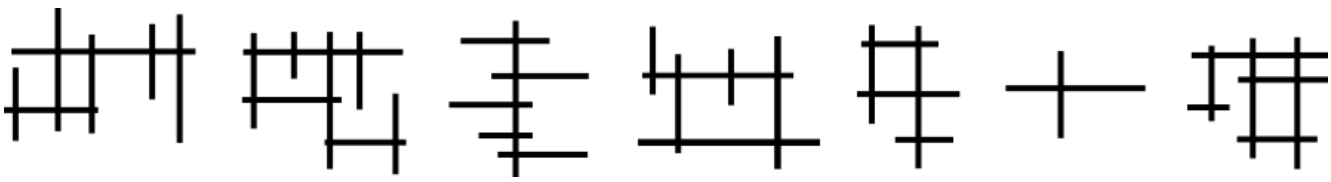
Paulina ir Joana rado slaptą savo klasės draugės Liucijos dienoraštį. Liucija užšifravo savo dienoraščio tekstą simboliais, sudarytais iš horizontalių ir vertikalių linijų, naudodama šią raidžių lentelę:

A	B	C	D	E
F	G	H	I	J
K	L	M	N	O
P	R	S	T	U
V	W	X	Y	Z

Abi merginos pastebi, kad užšifruotame tekste yra daug daugiau nei 25 skirtingi simboliai. Jos taip pat sėkmingai iššifruoja šiuos septynis simbolius, kuriais užšifruotas Liucijos brolio vardas RAPOLAS:



Iššifruokite Liucijos vaikino vardą, kuris dienoraštyje užrašytas šiais simboliais:



Paaiškinimas

Liucijos vaikino vardas yra JOVARAS.

Užšifruotuose simboliuose esančios horizontalios ir vertikalios linijos turi prasmę. Horizontalių linijų skaičius atitinka raidžių lentelės eilutės numerį. Analogiškai vertikalių linijų skaičius atitinka stulpelio numerį. Raidė, randama eilutės ir stulpelio susikirtimo vietoje, yra simboliu užšifruota raidė.

Pavyzdžiui, pirmasis simbolis turi 2 horizontalias linijas ir 5 vertikalias linijas. Raidė, randama 2-os eilutės ir 5-o stulpelio sankirtoje, yra J.

Naudodami šį metodą kiekvienam likusiam simboliui iššifruojame vardą JOVARAS.

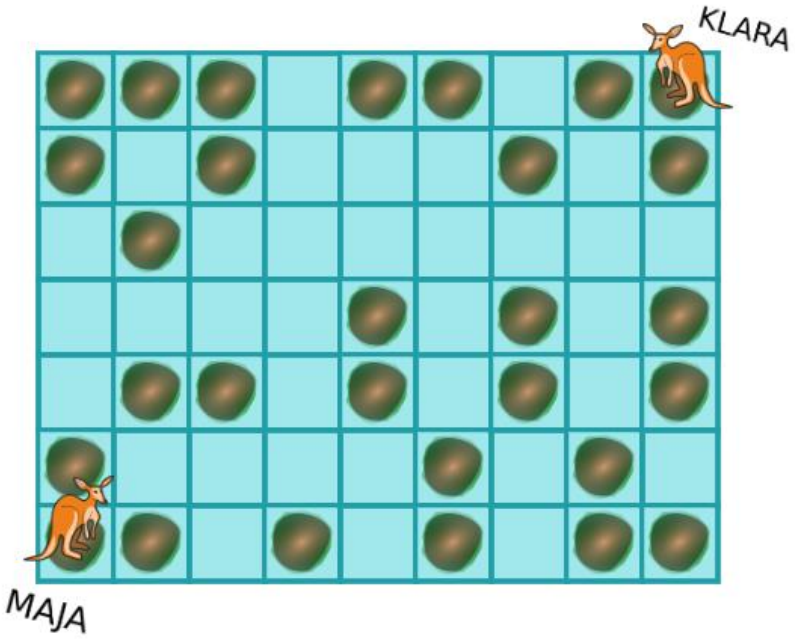
Tai informatika!

Teksto užšifravimas ir iššifravimas yra labai svarbi informatikos dalis. Komunikuojant internetu įprasta reikalauti, kad informacija būtų perduodama saugiai ir patikimai.

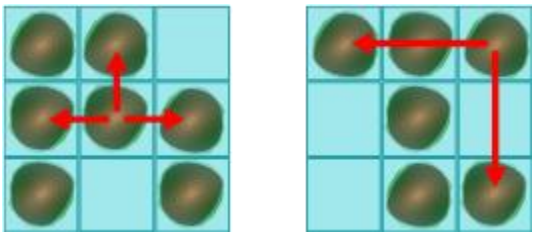
Šioje užduotyje panaudotas šifravimo metodas nėra naudojamas kompiuterijoje, tačiau užduotis parodo kai kuriuos įprastus informatinio mąstymo įgūdžius. Pirma, norint rasti ryšį tarp simbolių ir raidžių, reikia loginio mąstymo. Antra, reikia abstrahuoti, kad būtų galima nustatyti svarbius požymius. Linijų ilgis arba jų susikirtimo vieta nėra svarbus požymis ir gali būti suabstraktintas. Svarbu yra linijų skaičius ir tai, ar jos yra horizontalios, ar vertikalios.

19. Kengūra

Kengūra Maja nori nušuoliuoti pas kengūrą Klarą. Kelias tarp šių kengūrų eina salelėmis. Pelkę ir saleles galima pavaizduoti, kaip parodyta paveiksle.



Maja gali atlikti tik dvejopus šuolius: trumpą ir ilgą. Trumpu šuoliu kengūra iš vieno langelio gali nušokti į bet kurį iš keturių besiribojančių langelių. Ilgu šuoliu Maja gali šokti tiesiai ir peršokti bet kurį iš besiribojančių keturių langelių. Žemiau kairiajame paveikslėlyje pavaizduoti trumpi šuoliai, dešiniajame – ilgi.

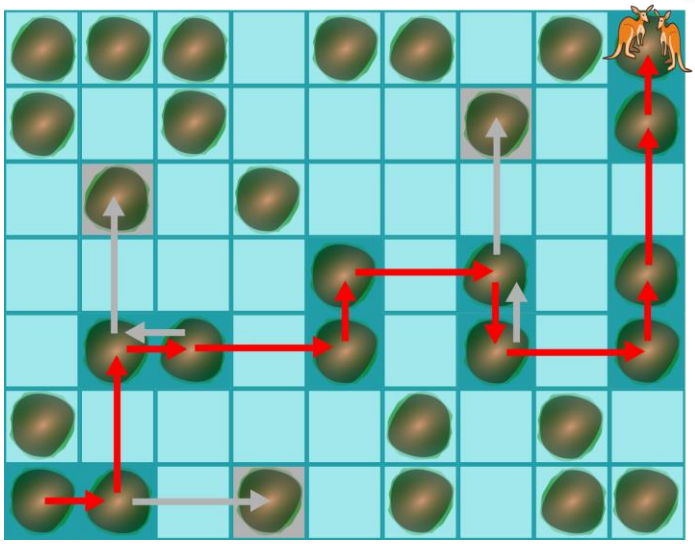


Maja negali atlikti jokių kitų šuolių: šokti įstrižai, peršokti dviejų ar daugiau langelių ir pan. Ilgi šuoliai vargina labiau ir yra pavojingesni už trumpus, todėl Maja negali atlikti dviejų ilgų šuolių iš eilės.

Raskite kelią, kuriuo Maja galėtų pasiekti Klarą.

Paaiškinimas

Teisingas atsakymas pavaizduotas paveiksle.



Galimi ir kitokie sprendimai, kai Majai tektų į kai kurias saleles šokti po keletą kartų.

Yra tik trys salelės (pažymėtos pilka rodykle), kurias Maja gali pasiekti, tačiau jos nepatenka į Majai reikalingą kelią – iš šių salelių bandant eiti toliau, Majai tektų atlikti po du ilgus šuolius iš eilės. Atmetus šias tris saleles, iš likusių salelių, kurias Maja gali pasiekti, suformuojamas reikiamas kelias.

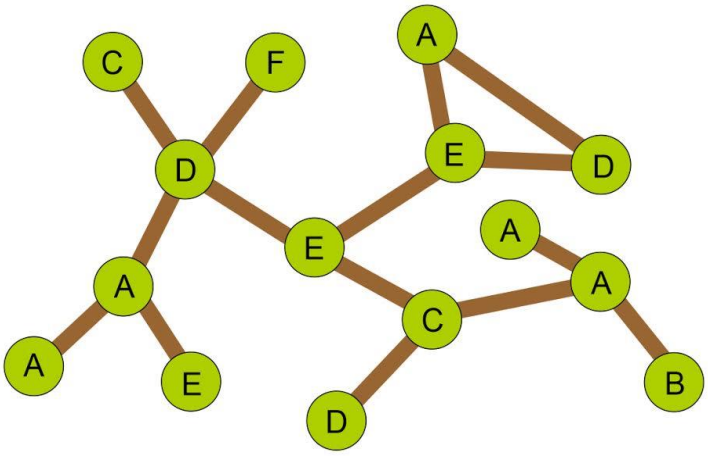
Tai informatika!

Užduotyje naudojama paieška į plotį – modifikuotas Lee algoritmas. Lee algoritmas imituoja bangos judėjimą užliejant gretimus langelius. Šiame uždavinyje langeliai laikomi gretimais ir tuomet, kai kengūra gali juos pasiekti ir dviem šuoliais: pirma ilgu, po to – trumpu.

Užduoties sąlyga draudžia du ilgus šuolius iš eilės, todėl uždavinio negalima išspręsti naudojant dinaminį programavimą.

20. Pasivaikščiojimas parke?

Duotas parko planas:



Žalios spalvos skrituliai su raidėmis vaizduoja parko medžius, o rudos spalvos linijos – parko takus. Kai kurios raidės naudojamos daugiau kaip vienam medžiui žymėti. Pasivaikščiojimas nuo medžio F iki medžio B būtų aprašomas raidžių seka **F D E C A B**.

Praeitą sekmadienį parke vaikščiojo dvi šeimos.

Binkių šeimos pasivaikščiojimo kelias buvo **B A A A C E D E E D A**.

Mekų šeimos kelias buvo **F D C D A E A D E D A**.

Laikykime, kad abi šeimos pradėjo savo pasivaikščiojimus tuo pačiu laiku, o nueiti nuo vieno medžio iki kito medžio užima tiek pat laiko. Kiek kartų dvi šeimos susitiko prie medžių?

A. Vieną kartą

B. Du kartus

C. Tris kartus

D. Nesusitiko nei vieno karto

Paaiškinimas

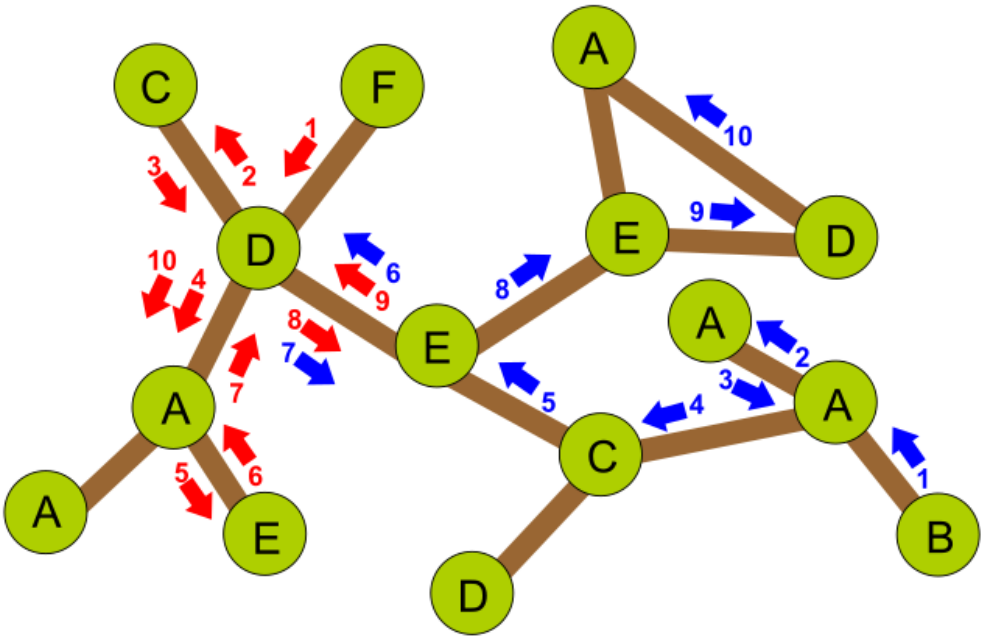
Teisingas atsakymas yra: *Jie nesusitiko nei vieno karto*.

Sprendžiant šį uždavinį neužtenka pastebėti tą pačią raidę toje pat pozicijoje (abi šeimos prie tokia raide pažymėto medžio buvo tuo pačiu laiku, kadangi ta pati raidė gali žymėti skirtingus tos pačios rūšies medžius. Pavyzdžiui, abi šeimos baigė savo maršrutus prie medžio, pažymėto raide A, tačiau nagrinėdami abu pasivaikščiojimo kelius žingsnis po žingsnio, matome, kad maršrutai baigiasi prie skirtingų medžių.

Galime nesunkiai sekti kiekvienos šeimos kiekvieną žingsnį, nes kiekvieno medžio kaimynai (t. y., medžiai, kuriuos jungia tas pats takas) visada pažymėti skirtingomis raidžių sekomis.

Visgi lygiagrečiai sekti abiejų šeimų pasivaikščiojimo maršrutus nėra taip paprasta (nors ir įmanoma).

Nupieškime Binkių šeimos pasivaikščiojimo kelią (pradžia B taške, žymima mėlyna spalva) ir sunumeruokime medžius pagal jų aplankymo laiką (mėlynos spalvos skaičiai nuo 1 iki 11). Analogiškai nupieškime Mekų šeimos kelią raudonos spalvos rodyklėmis ir sužymėkime skaičiais.



Dvi šeimos susitiks prie tų pačių medžių tik tuo atveju, jei tie medžiai bus pažymėti tuo pačiu raudonu ir mėlynu skaičiumi. Tokių medžių mūsų schemoje nėra.

Pastebėkime, kad yra tik du medžiai, kuriuos aplankė abi šeimos. Tad galime kreipti dėmesį tik į juos. Medis D (esantis parko kairėje pusėje) Binkių šeimos buvo septintas aplankytas, o Mekų šeimai jis buvo 2-as, 4-as, 8-as ir 10-as. Medis E (kaimyninis parko kairėje esančiam D

medžiui) Binkių šeimai buvo 6-as ir 8-as aplankytas, tačiau Mekų šeimai jis buvo 9-as aplankytas medis.

Tai informatika!

Informatikos specialistai, programuotojai dažnai taiko grafus (ir kalba apie grafo viršūnes ir briaunas, o ne apie medžius ir juos jungiančius parko takus). Grafų teorija – svarbi ir įdomi tiek matematikos, tiek informatikos sritis. Abu šie mokslai ją naudoja savaip ir skirtingais tikslais.

Dar vienas įdomus šio uždavinio aspektas yra pasivaikščiojimo kelių parke vaizdavimas. Nepaisant to, kad kai kurie medžiai (viršūnės) pažymėti ta pačia raide, pasivaikščiojimo maršrutus, kurie prasideda iš B arba F, galima vienareikšmiškai apibūdinti pagal raidžių seką pasivaikščiojimo kelyje. Tai reiškia, kad viena raidžių seka apibūdina tik vieną pasivaikščiojimo maršrutą. Taip yra todėl, kad kiekvieno medžio kaimyniniai medžiai (medžio X kaimyniniai medžiai – tai tiesiogiai taku su X sujungti medžiai) visada žymimi skirtingomis raidėmis. Todėl, jei žinome, kur esame tam tikru pasivaikščiojimo momentu, ir matome kitą raidę pasivaikščiojimą vaizduojančioje schemoje, tada nekyla abejonių, kurį medį turėtume aplankyti toliau.

21. Monetų kolekcininkas

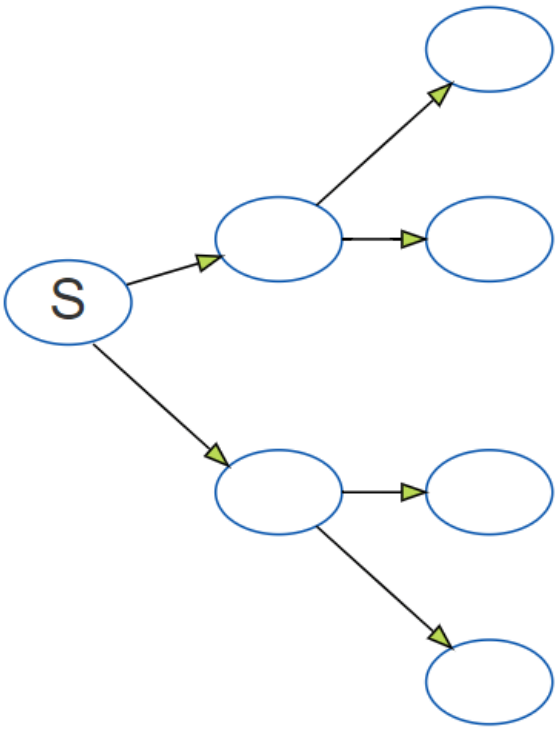
Alisa nori žaisti su savo draugu. Ji miške nutiesė kelis takus, iš šonų juos papuošė dideliais akmenimis. Prie kiekvieno akmens Alisa padėjo po maišelį auksinių monetų. Kiekviename maišelyje yra nuo 1 iki 6 monetų.

Alisos draugas nori surinkti kuo daugiau aukso monetų. Jis pradeda nuo akmens, pažymėto raide „S“. Prie kiekvieno akmens Alisos draugas gali pasirinkti tolesnį akmenį – arba esantį kairėje kelio pusėje, arba dešinėje pusėje. Tada jis eina prie pasirinkto akmens ir pasiima monetas.

Kiekvieną kartą, kai Alisos draugas nori pasirinkti akmenį, jis gali matyti, kiek monetų yra maišelyje prie kiekvieno iš tų dviejų akmenų. Alisos draugas godus – jis nusprendė visada rinktis tą akmenį, prie kurio yra daugiau monetų.

Alisa nori pamokyti savo draugą, kad jo strategija nėra gera ir ne visada padeda gauti daugiausiai monetų.

Pateiktame Alisos išdėliotų akmenų paveiksle nuvilkite maišelius su monetomis ant akmenų taip, kad jos draugas, nesilaikydamas savo strategijos, iš tikrųjų galėtų gauti daugiau monetų.



Paaiškinimas

Yra daug teisingų šios užduoties sprendimų. Vienas iš jos sprendimo būdų – priversti draugą pačioje proceso pradžioje priimti neteisingą sprendimą. Pavyzdžiui, padarykite taip, kad pirmasis pasirinkimas būtų tarp „1“ ir „2“. Draugas dėl savo godumo pasirinks „2“. Dabar visus maišelius su didesniu kiekiu monetų padėkite už pasirinkimo „1“, o visus maišelius su mažiau monetų už pasirinkimo „2“. Akivaizdu, kad tai yra sprendimas.

Tai informatika!

Godus algoritmas – toks algoritmas, kai kiekvieną kartą pasirenkamas optimaliausias vietos atžvilgiu variantas. Tai reiškia, kad nežiūrite į platesnį vaizdą, o tik į informaciją, kuri yra iš karto šalia jūsų. Kartais godūs algoritmai veikia gerai. Tačiau yra atvejų (pvz., ši užduotis), kai godus algoritmas netinka.

Informatikos moksle svarbu savo algoritmams sukurti bandomuosius duomenis. Tai reiškia, kad sukursite tam tikrus duomenis, kuriuos pateiksite savo algoritmui, kad patikrintumėte, ar jis teisingas. Gerų bandomųjų duomenų kūrimas yra labai svarbus įgūdis. Naudojant netinkamus bandomuosius duomenis jūsų algoritmas gali atrodyti teisingas, tačiau taip bus tik tuo konkrečiu atveju.

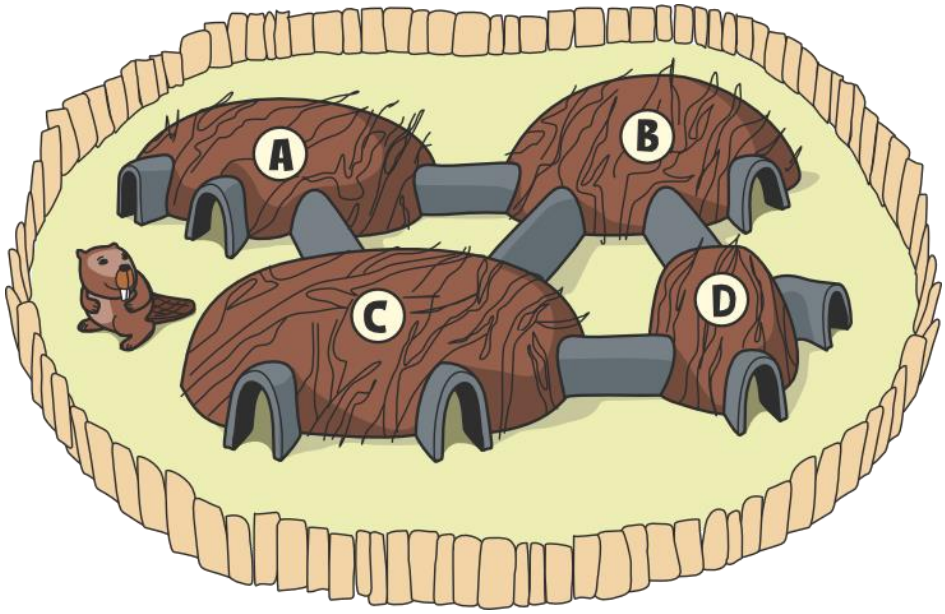
Atkreipkite dėmesį, kad „godus algoritmas“ nėra algoritmas – tai algoritme naudojamas metodas.

Daugiau: https://en.wikipedia.org/wiki/Greedy_algorithm

22. Pramogų bebravietė

Bebro šeima įrengė pramogų buveinę – 4 kambarius, kuriuos jungia 5 tuneliai ir iš kurių yra 7 durys į kiemą.

Bebro vaikai pastebėjo, kad įmanoma perbėgti visus tunelius ir pro visas duris, visur lankantis tik po vieną kartą.



Iš kurio kambario reikėtų pradėti bėgti?

A) Iš A

B) Iš B

C) Iš C

D) Iš D

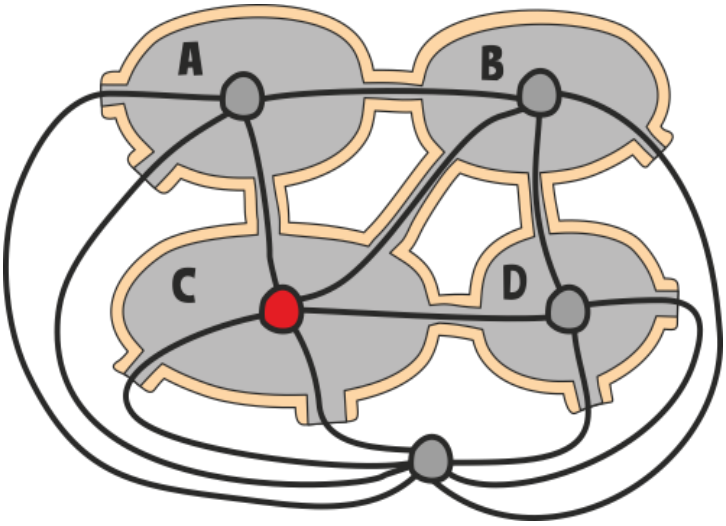
Paaiškinimas

Teisingas atsakymas yra C.

Pramogų bebravietę galima pavaizduoti grafu. Kiekvienas kambarys tampa viršūne, kiemas žymimas atskira viršūne. Dvi viršūnės sujungiamos, jei tarp kambarių yra tuneliai arba iš kambario yra durys į kiemą.

Norimas maršrutas įmanomas, jei yra dvi viršūnės, turinčios nelyginį briaunų skaičių. Tada viena iš šių viršūnių yra maršruto pradžia, o kita – pabaiga.

Mūsų atveju tokia viršūnė, turinti nelyginį kelių skaičių, yra C (3 tuneliai + 2 durys).



Tai informatika!

Tai grafų teorijos uždavinys. Vadinamasis Oilerio kelias kaip tik ir eina per visas grafo viršūnes ir tik po vieną kartą. Būtina Oilerio kelio sąlyga: viena arba dvi viršūnės turi turėti nelyginį briaunų skaičių. Kodėl keliama tokia sąlyga, galime suprasti pamąstę: keliaudami į kiekvieną kambarį turime įeiti (tuneliu ar pro duris) ir taip pat iš jo išeiti (tuneliu arba pro duris), išskyrus maršruto pradžią ir pabaigą.

https://en.wikipedia.org/wiki/Eulerian_path

https://lt.wikipedia.org/wiki/Oilerio_mar%C5%A1rutas