ISSN 2783-7599

b·e·b·r·a·s

Informatikos ir informatinio mąstymo uždavinių rinkinys

Nr. 8

ŠVIETIMO, MOKSLO IR SPORTO MINISTERIJA

VILNIAUS UNIVERSITETAS ·1579· UNIVERSITAS VILNENSIS

Bebras
International Challenge on Informatics and Computational Thinking

Šiame rinkinyje pateikiami 2021 metų XVII informatikos ir informatinio mąstymo konkurso (iššūkio) „Bebras“ II etapo uždaviniai, 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.

Dėkojame Daumilui Ardickui, dr. Tatjanai Jevsikovai, Audronei Klupšaitei, Alvidai Lozdienei, Vaidai Masiulionytei-Dagienei, Modestui Rimkui, dr. Dovilei Stumbrienei, dr. Gabrielei Stupurienei, Tomui Šiauliui, talkinusiems verčiant ir adaptuojant užduotis. Taip pat dėkojame tarptautinei „Bebro“ konkurso bendruomenei ir užduočių autoriams.

Parengė Lina Vinikienė

Konsultavo Valentina Dagienė

Redagavo Viktoras Dagys

Viršelį kūrė Vaidotas Kinčius

Užduočių rinkinys platinamas pagal kūrybinių bendrijų licenciją nekomerciniais tikslais (*Creative Commons Attribution–NonCommercial–ShareAlike*)

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.

2006 metais informatinio mąstymo idėją ėmėsi aktyviai propaguoti Jeannette M. Wing (*Computational thinking. Communication of ACM, 49, 33–35*), tuo metu JAV Kolumbijos universiteto Duomenų mokslo instituto informatikos profesorė, viena iš „Microsoft Research“ vadovų. Tačiau pradinė informatinio mąstymo idėja priklauso daugeliui Lietuvos pedagogų žinomos knygos „Minčių audros: vaikai, kompiuteriai ir veiksmingos idėjos“ (*Mindstorms: Children, Computers, and Powerful Ideas*) autoriui Seimūrui Papertui (Seymour Papert). Jis jau prieš pusšimtį metų rašė, kad turime mokyti vaikus mąstyti apie savo mąstymą, gerinti jį pasitelkę įvairias priemones, taip pat ir kompiuterį, turime spręsti realaus gyvenimo uždavinius ir jų sprendimui pasinaudoti technologijomis.

„Bebras“ – tai ne tik konkursas (anglų kalba jis vadinamas *challenge* – iššūkiu). Tai daugybė įvairių veiklų, kurios vyksta ištisus metus. Susikūrė stiprus pasaulinis „Bebro“ konkurso mokslininkų ir mokytojų tinklas, kasmet kuriami nauji uždaviniai, ieškoma aktualių temų. Konkursui vykdyti reikalingos modernios sistemos, dalis šalių kuria ir tobulina šias sistemas pačios, kitos naudojasi kai kurių šalių paslaugomis. „Bebro“ edukacines veiklas savo šalyse įgyvendina aukštosios mokyklos: Oksfordo universitetas (Didžioji Britanija), Masačusetso technologijos institutas (JAV), Federalinis technologijos institutas (Šveicarija), Vaterlo universitetas (Kanada) ir daugelis kitų. „Bebro“ veiklas Jungtinėje Karalystėje parėmė net kompanija „Google“.

Mokiniai, spręsdami įdomius, informatikos konceptais grįstus uždavinius, susipažįsta su informatikos sąvokomis, išsiugdo gebėjimą taikyti jas praktiškai, uždavinių sprendimus aptaria su bendraamžiais ir mokytojais. Didelė dalis uždavinių yra iš algoritmų ir programavimo srities. Juos perpratus, skatinama mokytis praktinio programavimo. „Pateikiami uždaviniai nėra skirti išmokyti naudotis informacinėmis technologijomis, teikti konkrečias technines žinias, o labiau – motyvuoti, pritraukti ir parodyti informatikos mokslo gilumą, patrauklumą, ugdyti gebėjimus pastebėti dėsningumus, perkelti sprendimo būdus iš vieno uždavinio į kitą, vieną informaciją sieti su kita“, – sako konkurso sumanytoja prof. Valentina Dagienė. Mokytojams uždaviniai – didaktiniai ištekliai, idėjos savo pamokoms pajvairinti. Mokslininkai kurdami uždavinius stengiasi perteikti esminius informatikos mokslo

principus. „Bebro“ bendruomenę labiausiai vienija patrauklių, įdomių informatikos uždavinių paieška, informatikos konceptų išreiškimas žaidybinėmis užduotimis.

Vykdant konkursą kaupiami mokinių sprendimai – daugybė duomenų surinkta, galima atlikti tyrimus, stebėti ir pan. Įvairių šalių mokslininkai, remdamiesi „Bebro“ bendruomenės sukauptais duomenimis, rengia ir publikuoja mokslinius straipsnius (žr. https://www.bebras.org/publications.html). „Vokiečiai, čekai, austrai, šveicarai, italai kasmet leidžia „Bebro“ uždavinių knygutes mokiniams ir knygas mokytojams. Olandai taip ištobulino „Bebro“ technologinę infrastruktūrą, kad dabar teikia technologines paslaugas ir kitoms šalims narėms. Jie įsteigė „Bebro“ kompaniją savo šalyje ir įregistravo „Bebro“ mokymo paslaugas JAV. Prancūzijoje ne tik rengiami konkursai, bet sukurtos ir mokytojų mokymosi platformos, kur mokomasi kartu su mokiniais. Australijoje nacionalinėse informacinių technologijų ugdymo programose vyrauja „Bebro“ konkurso metodika. Beje, prancūzai, be „Bebro“, išplėtojo ir kitą sistemą, vadinamą „Algorea“ (programavimo mokymosi sistema)“, – pasakoja „Bebro“ idėjos pradininkė.

2020 metų „Bebro“ konkurso uždavinius sprendė 2 479 870 mokinių 57-iose valstybėse. Lietuvoje tais metais konkurse dalyvavo 32 107 mokiniai.

Daugiau informacijos apie „Bebro“ konkursą pateikiama interneto svetainėse:

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

www.bebras.org

bebras.lt

lt.bebras.lt

2020 m. „Bebro“ dalyvių skaičius pagal šalis (56 šalių duomenys)

Nemažai šalių geriausiems konkurso dalyviams rengia akivaizdinį antrąjį etapą. Lietuvoje antrasis etapas rengiamas universitetuose ir kolegijose sausio mėnesio pabaigoje ar vasario pradžioje.

2021 m. vasario 20 d. Lietuvoje 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 54 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 už uždavinį skiriamų taškų, t. y., atitinkamai 2, 3 arba 4 taškai.

Lietuvoje XVII informatikos ir informatinio mąstymo konkurso „Bebras“ II etape dalyvavo 297 jauniai (9–10 klasių mokiniai) ir 243 kolegos (11–12 klasių mokiniai).

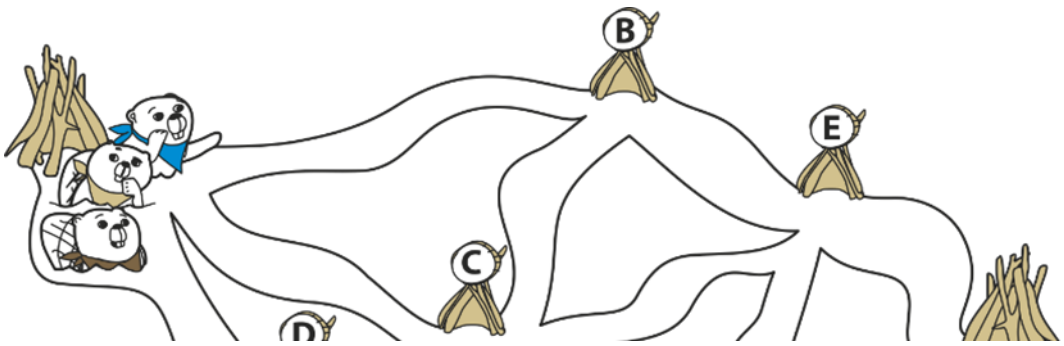
Lentelėje pateikiamas XVII konkurso II etapo uždavinių skirstymas pagal amžiaus grupes.

Nr.	*Uždavinio pavadinimas*	*Uždavinio identifikatorius*	*Jauniai*	*Kolegos*
1	Stebuklingas gėrimų aparatas	2020-KR-08	6	
2	Ypatingos sankryžos	2020-NL-06	6	
3	Slaptoji žinutė	2020-IE-01a	6	
4	Elektromobiliai	2020-IE-03	6	
5	Šviečiančios plytelės	2020-JP-03	6	
6	Pietų stalas	2020-IN-01	9	6
7	Siurblys-robotas	2020-CZ-03	9	6
8	Jogurto fasavimo aparatas	2014-RU-06	9	6
9	Užtvankos tuneliai	2017-CH-07b	9	6
10	Šokinėjimas rąstais	2014-CH-02	9	6
11	Trys darbininkai	2020-TH-03	12	9
12	Supynės iš rąsto	2020-DE-04a	12	9
13	Dykuma	2018-HR-08	12	9
14	Rodyklių spalvinimas	2018-HU-07	12	9
15	Knygų rikiavimas	2018-CN-02	12	9
16	Šnabždesių žaidimas	2020-IN-24		12
17	Matematinė mašina	2020-DE-06b		12
18	Efektyvi gamykla	2020-CN-08		12
19	Elektromobilių eilės	2020-IE-04		12
20	Medžiai ant terasų	2020-LT-17_CH		12

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

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

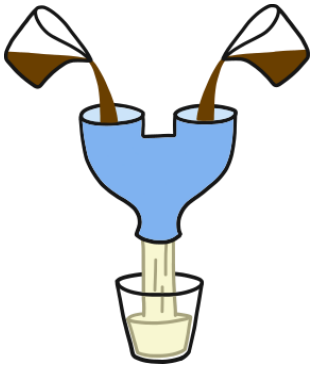
Taip pat pateikiamas uždavinio atsakymas ir paaiškinimas, kaip uždavinys susijęs su informatika.



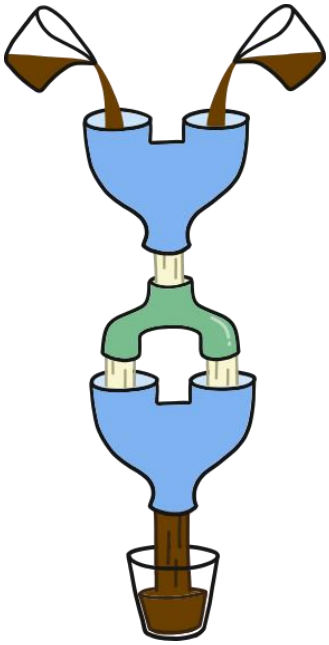
1. Skaičių skirtumų automatas

Bebras Kobis turi paslaptingą mėlyną gėrimų maišymo aparatą, pavaizduotą 1-ame paveiksle. Aparatas turi du piltuvėlius.

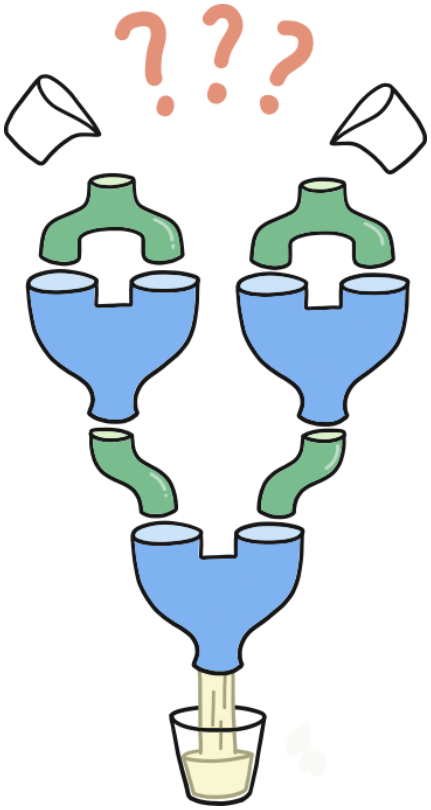
- Jei bebras į abu piltuvėlius pila šokoladinį pieną, gaunamas baltas pienas.
- Jei bebras į bent vieną iš dviejų piltuvėlių pila baltą pieną, gaunamas šokoladinis pienas.
- Jei bebras sujungia du aparatus ir į abu piltuvėlius pila šokoladinį pieną, kaip parodyta 2-ame paveiksle, gaunamas šokoladinis pienas. Žalia jungiamoji detalė neturi jokios įtakos gėrimo spalvai.



1 pav.



2 pav.



3 pav.

Kokį pieną reikia pilti į piltuvėlius, kad būtų gaunamas baltas pienas, kai trys gėrimų maišymo aparatai sujungti taip, kaip pavaizduota 3-ame paveiksle?

A) baltą pieną, baltą pieną

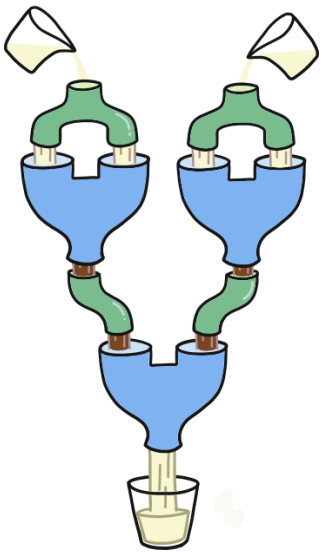
B) baltą pieną, šokoladinį pieną

C) šokoladinį pieną, baltą pieną

D) šokoladinį pieną, šokoladinį pieną

Paaiškinimas

Teisingas atsakymas: A.



4 pav.

Kad maišymo pabaigoje gautume baltą pieną, į abu apatinio aparato piltuvėlius reikia pilti šokoladinį pieną, kaip pavaizduota 4-ame paveiksle. Taigi, šokoladinis pienas turėtų išbėgti iš abiejų viršutinių aparatų.

Kad iš abiejų viršutinių aparatų gautume šokoladinį pieną, į kiekvieno iš viršutinių aparatų bent vieną piltuvėlį reikėtų pilti baltą pieną. Vis dėlto, bendra konstrukcija tokia, kad į viršutinių aparatų abu piltuvėlius gali tekėti tik tos pačios spalvos pienas. Taigi, kad maišymo pabaigoje gautume baltą pieną, į abu viršutinius piltuvėlius reikia pilti baltą pieną.

Tai informatika!

Elektroninė grandinė, kurios pradiniai duomenys yra viena ar daugiau loginių reikšmių, o atlikus logines operacijas gautas rezultatas – viena loginė reikšmė, yra vadinama logine grandine. Loginės operacijos yra AND (konjunkcija), OR (disjunkcija), NOT (inversija) ir XOR (griežtoji disjunkcija). Loginė operacija NAND aprašoma taip:

A NAND B = NOT (A AND B)

Įdomu tai, kad visas logines operacijas galima užrašyti naudojant tik NAND operatorių.

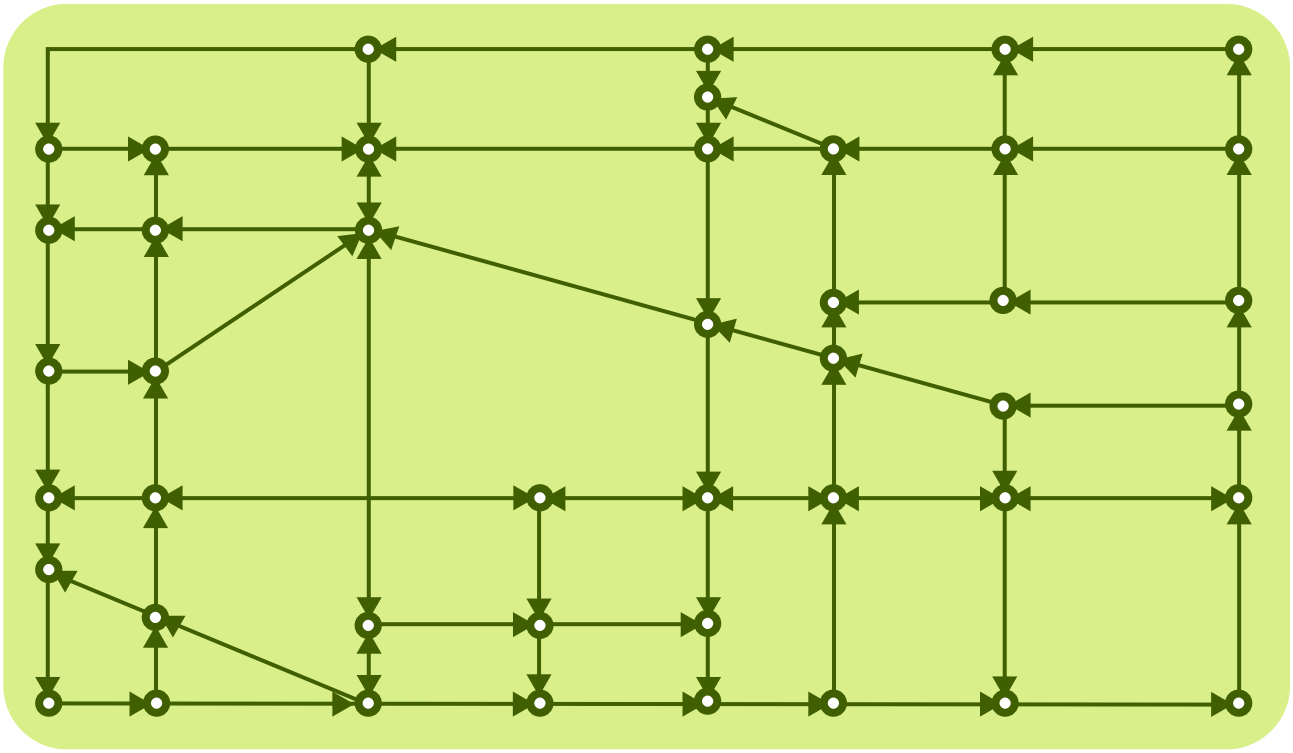
NOT A = A NAND A

A AND B = (A NAND B) NAND (A NAND B) → 2 pav.

A OR B = (A NAND A) NAND (B NAND B) → 3 pav.

Uždavinio sąlygoje baltas pienas žymi reikšmę 0, šokoladinis pienas – reikšmę 1, maišymo aparatas – operaciją NAND. 2-ame paveiksle pavaizduota operacija AND naudojant tik operatorių NAND, o 3-ame paveiksle – operacija OR naudojant tik operatorių NAND.

2. Ypatingos sankryžos



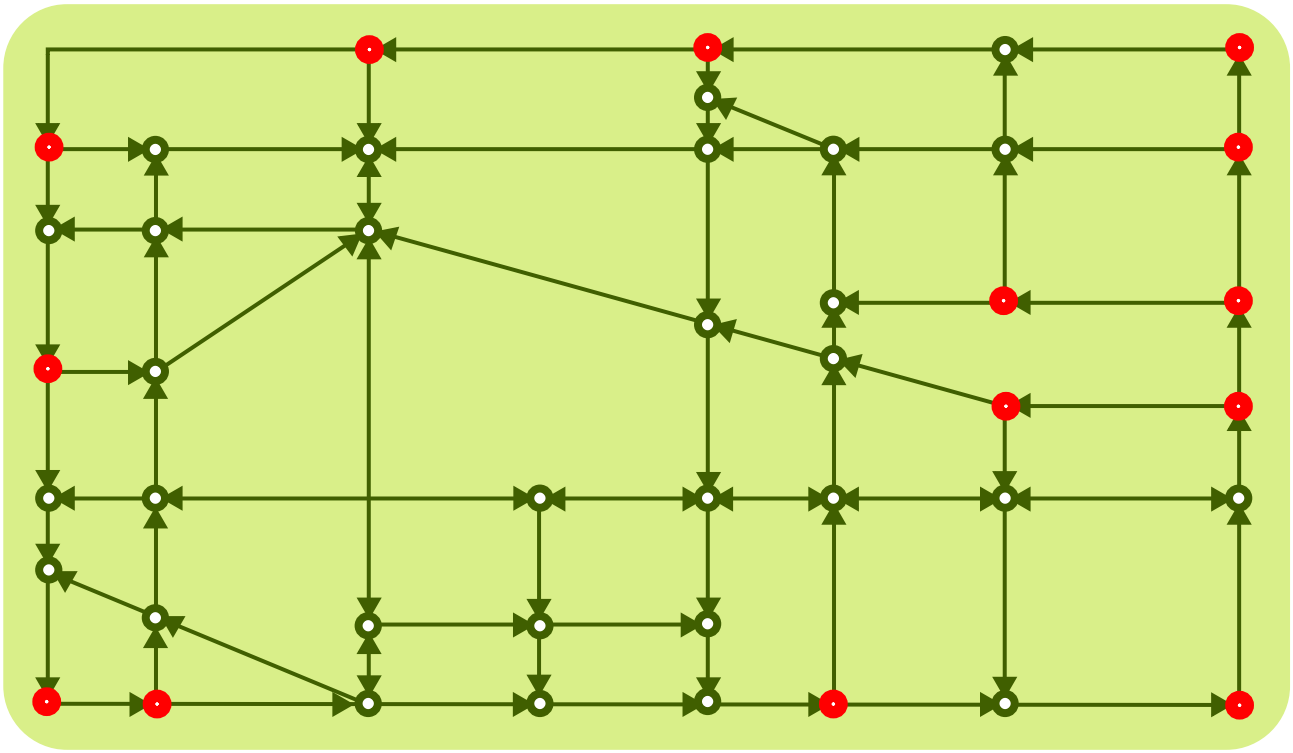
Šio miesto žemėlapyje dauguma gatvių yra vienakryptės – žymimos vienos krypties rodyklėmis. Keletas gatvių yra dvikryptės – žymimos dvipusėmis rodyklėmis.

Sankryža vadinama ypatinga, jei į ją galima įvažiuoti tik iš vienos gatvės.

Kiek yra ypatingų sankryžų pateiktame miesto žemėlapyje?

Paaiškinimas

Šiame mieste yra 14 ypatingų sankryžų, jos pažymėtos raudonais pilnaviduriais skrituliais.

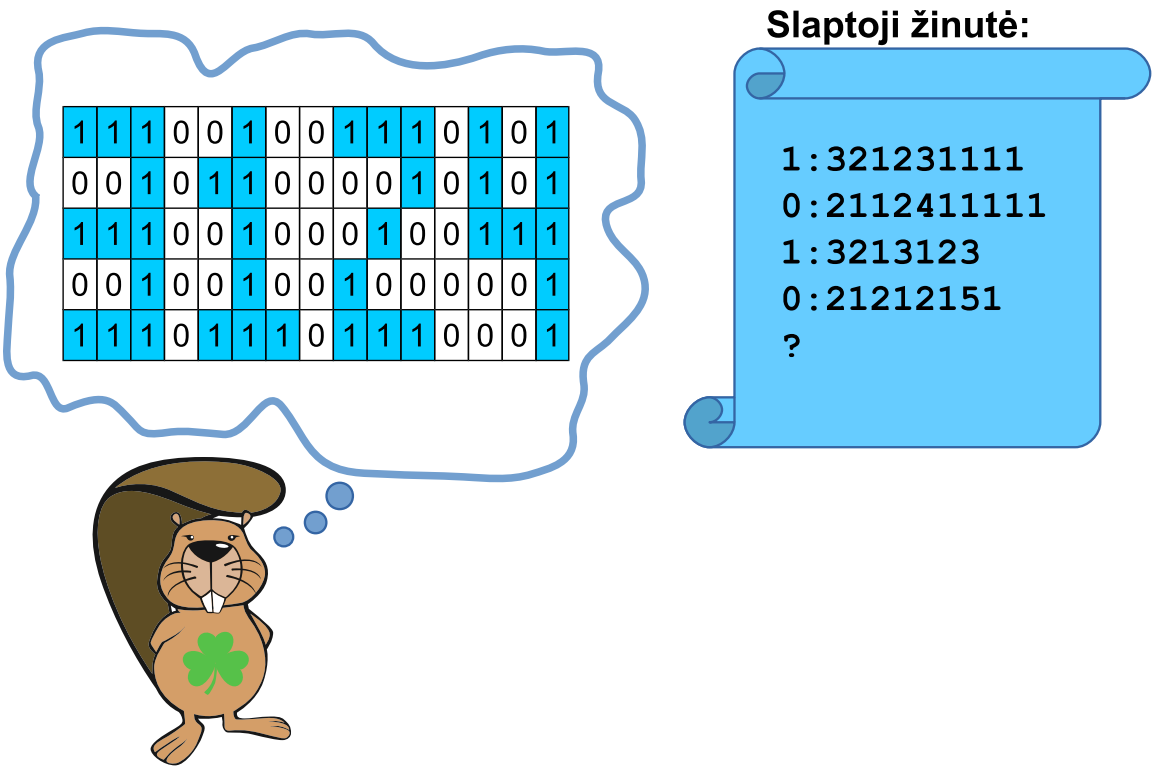


Tai informatika!

Šio žemėlapio struktūra vadinama tinklu. Ypatingos sankryžos yra problemiškos: jas lengva užblokuoti ir tada kai kurios vietos gali būti nepasiekiamos. Analizuojant tinklą galima nustatyti šias pavojingas sankryžas ir iš anksto pasirengti, pavyzdžiui, geriau apsaugoti. Taigi, žinoti ypatingas sankryžas yra svarbi tinklo informacija. Tinklas yra orientuotasis grafas, į kiekvieną viršūnę (sankryžą) įeina keletas kelių – jų skaičius vadinamas įeinančiųjų laipsniu. Viršūnė yra ypatinga, jei jos laipsnių (įeinančių kelių) skaičius lygus 1. Nepasiekiama viršūnė yra tokia, kurios laipsnių (įeinančių kelių) skaičius lygus 0, tokių sankryžų pateiktame tinkle nėra.

3. Slaptoji žinutė

Bebrė Lina ir jos draugai naudoja specialų kodą siųsdami vieni kitiems žinutes. Lina ruošiasi slaptam vakarėliui ir nori išsiųsti įėjimo kodus draugams.



Kokia bus paskutinė slaptosios žinutės eilutė?

A) 0:3131331

B) 1:321231111

C) 1:3131331

D) 1:231231111

Paaiškinimas

Teisingas atsakymas: C.

Kiekvienos eilutės pirmasis skaitmuo nurodo, kuriuo skaitmeniu (arba kokios spalvos langeliu) prasideda atitinkama paveikslo eilutė: jei 1 (mėlynu langeliu) – tai 1, jei 0 (baltu langeliu) – tai 0. Po dvitaškio einantys skaitmenys nusako, kiek yra nulių arba vienetų (baltų ar mėlynų langelių).

Paskutinė paveikslo eilutė prasideda 1 (mėlynu langeliu). Po dvitaškio eina skaitmuo, nurodantis vienetų skaičių (3). Po to eina vienas nulis, trys vienetai, vienas nulis ir t. t. Tad teisingas atsakymas yra:

1 : 3 1 3 1 3 3 1

Arba aiškinant spalvomis: pirmasis langelis mėlynas: 3m 1b 3m 1b 3m 3b 1m.

A variantas negali būti teisingas, nes prasideda 0: tai reikštų, kad pirmasis skaitmuo turi būti nulis (baltas langelis). Iš tikrųjų ši eilutė vaizduoja priešingą paveikslo eilutę nei bebrė nori.

B variantas negali būti teisingas, nes iš tikrųjų apibūdina eilutę, identišką pirmajai paveikslo eilutei.

D negali būti teisingas, nes tokiu atveju paskutinė eilutė turėtų prasidėti dviem vienetais (dviem mėlynais langeliais).

Tai informatika!

Šioje užduotyje slypi keletas duomenų kodavimo konceptų:

- pirma, slaptažodis (tekstas) pateikiamas kaip paveikslas (vaizdas);
- antra, vaizdas užkoduojamas naudojant bitus: nulius ir vienetus (taškinis arba dvejetainis atvaizdas);
- trečia, dvejetainis atvaizdas tampa slaptąja žinute panaudojant posekių ilgio algoritmą.

Tos pačios informacijos pateikimas keliais skirtingais būdais ir poreikis konvertuoti iš vienos vaizdavimo formos į kitą yra įprasti veiksmai apdorojant duomenis. Šioje užduotyje taikomas posekių ilgio kodavimo būdas (angl. *run-length encoding*, RLE) yra labai paprastas duomenų glaudinimo neprarandant reikšmių algoritmas. Jo esmė – rasti posekius, kurių reikšmės kartojasi keletą kartų iš eilės. Kiekviena tokia pasikartojanti seka koduojama tik vienu skaičiumi, kuris nurodo pasikartojimą. Daugeliu atvejų išvedamų duomenų eilutė yra mažesnė už įvedimo, tačiau išskirtiniais atvejais gali būti ir didesnė.

Šiame uždavinyje praleistos reikšmės nurodymas laikant, kad kiekviename žingsnyje posekį sudarančios reikšmės yra priešingos ankstesnei. (Ar galite sugalvoti pavyzdį, kad šis požiūris sukeltų Linai problemų? Ar tokie pavyzdžiai gali vaizduoti slaptažodžius?)

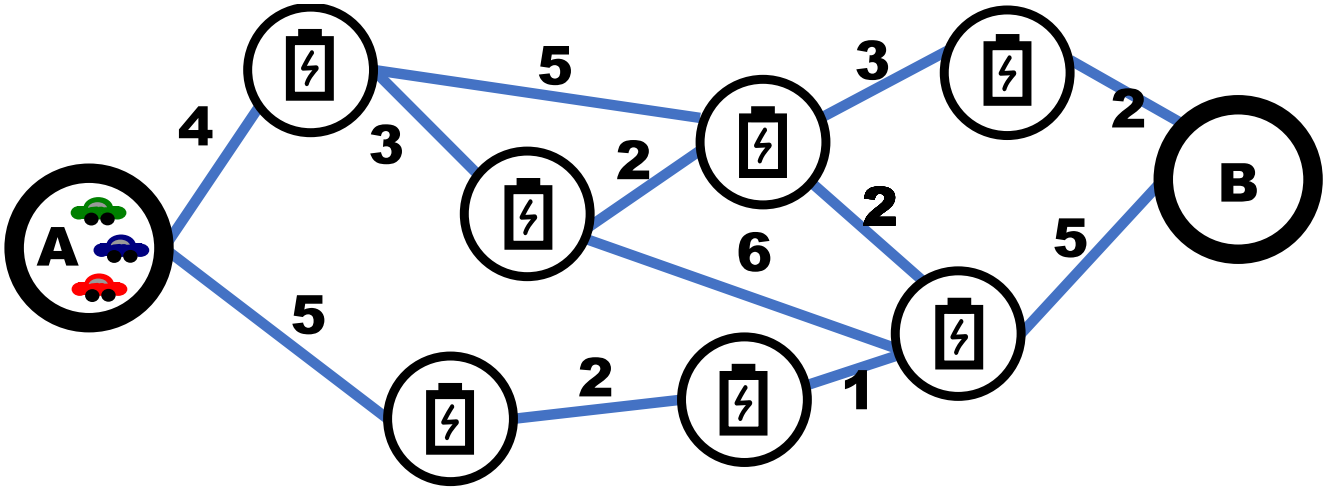
Užduotis reikalauja iš mokinio apibendrinimo ir algoritminio mąstymo įgūdžių. Pirmiausia mokinys turi atrasti RLE algoritmą, palygindamas atitinkamas paveikslo ir pranešimo eilutes pavyzdyje, nes užduotyje algoritmas nėra išreikštai įvardinamas. Toliau mokinys turi pritaikyti šį algoritmą, kad gautų paskutinę pranešimo eilutę.

Reikia pažymėti, kad RLE nėra stiprus šifravimo algoritmas. Pamačius keletą slaptų pranešimų pavyzdžių, nebus labai sunku perprasti algoritmą net nežinant atitinkamų slaptažodžių. Taigi, tai neturėtų būti naudojama jokioms svarbioms paslaptims koduoti.

4. Elektromobiliai

Mėlynas elektromobilis išsikrauna, kai nuvažiuoja 4 km, jį įkrauti trunka 3 minutes. Žalias elektromobilis išsikrauna, kai nuvažiuoja 5 km, jį įkrauti trunka 4 minutes. Raudonas elektromobilis išsikrauna, kai nuvažiuoja 6 km, o įkrauti jį trunka 5 minutes.

Jei elektromobilis neišsikrovęs, tai jis nebūtinai sustoja kiekvienoje įkrovimo stotelėje. Jei elektromobilis sustoja, tai jo įkrovimo laikas trunka nurodytas minutes, nepaisant to, kiek jis buvo įkrautas prieš pradedant krauti iš naujo. Starto taške A visi trys elektromobiliai yra visiškai įkrauti, kelionę pradeda vienu metu, pasirinkę optimalius maršrutus link taško B. Visose maršruto atkarpose jie važiuoja tuo pačiu vienodu greičiu.



Paveiksle rodomi visi galimi keliai ir įkrovimo stotelės tarp starto taško A ir finišo taško B. Skaičiai nurodo atstumą (km) tarp kiekvienos įkrovimo stotelės.

Kuris elektromobilis iš taško A pasieks finišo tašką B greičiausiai?

A. Raudonas

B. Mėlynas

C. Žalias

D. Visi pasieks vienu metu

Paaiškinimas

Teisingas atsakymas: B (žalias).

Žalias elektromobilis gali keliauti maršrutu {5, 2, 1, 5} ir nuvažiuos iš viso 13 km, sugaišdamas pirmoje ir trečioje įkrovimo stotelėje 8 minutes.

Greičiausias mėlyno elektromobilio maršrutas bus {4, 3, 2, 3, 2} ir jis tuo maršrutu nuvažiuos 14 km, o baterijos įkrovimui sugaiš 12 minučių, nes jis kiekvienoje stotelėje privalo įkrauti bateriją.

Greičiausias raudono elektromobilio maršrutas bus {5, 2, 1, 5} ir jis tuo maršrutu nuvažiuos 13 km, sugaišdamas pirmoje ir trečioje įkrovimo stotelėje 10 minučių.

Jei laikysime, kad kiekvieno elektromobilio greitis yra 1 km/min, tai žalias elektromobilis iš viso kelyje užtruks 21 minutę, mėlynas – 26 minutes, o raudonas – 23 minutes.

Tai informatika!

Trumpiausias maršrutas (kelias) tarp taškų yra vienas iš populiarių informatikos uždavinių. Šiuolaikinių technologijų pagalba norime su bet kokiame Žemės taške esančiu kolega bendrauti kaip įmanoma greičiau. Trumpiausio maršruto tarp dviejų taškų radimas gali užtikrinti greičiausią komunikavimą (bendravimą).

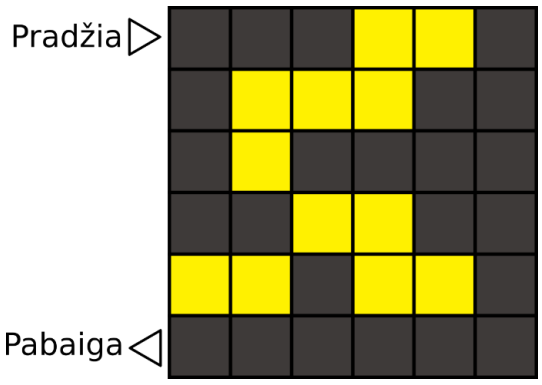
Metams bėgant buvo sukurti įvairūs algoritmai, padedantys rasti trumpiausią maršrutą bet kuriame tinkle. Vieną iš tokių algoritmų sukūrė olandų informatikas Edsgeras Deikstra (Edsger Wybe Dijkstra), šis algoritmas dabar vadinamas jo vardu. Algoritmas suranda trumpiausią kelią nuo pradinio mazgo iki kiekvieno kito mazgo.

5. Šviečiančios plytelės

Bebrų karalystės karalius valstybinei šventei miesto aikštėje nusprendė pastatyti milžinišką meno kūrinį iš šviečiančių plytelių. Šis meno kūrinys buvo sukurtas naudojant 36 šviečiančias plyteles, kaip parodyta paveikslėlyje. Kiekviena plytelė įsijungia ir išsijungia, kai ant jos kas nors užlipa. Šviečiantis raštas pradedamas kurti nuo įėjimo plytelės ir pabaigiamas kurti ties išėjimo plytele, einant po vieną plytelę kairėn, dešinėn, į viršų arba į apačią. Ant tos pačios plytelės galima užlipti keletą kartų.

Išjungta plytelė

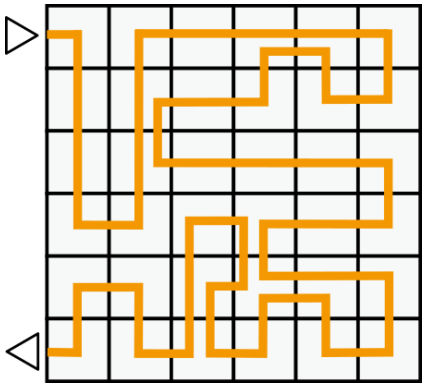
Įjungta plytelė



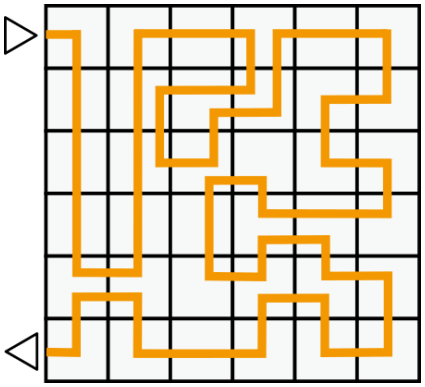
Karalius sukūrė raštą, kuris pavaizduotas paveikslėlyje, tačiau jam jis nepatiko, taigi karalius nusprendė įjungti visas plyteles, kad galėtų pradėti iš pradžių.

Pasirinkite, **kuriuo keliu jis turėtų eiti, norėdamas įjungti visas plyteles.**

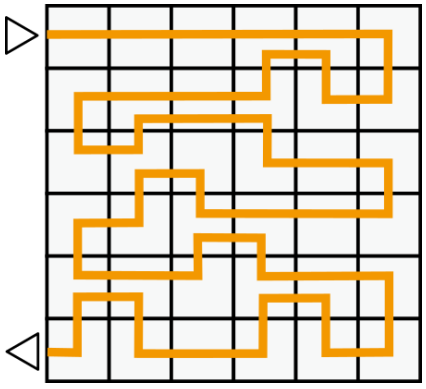
a)



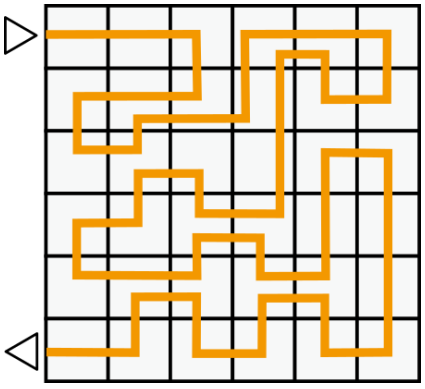
b)



c)

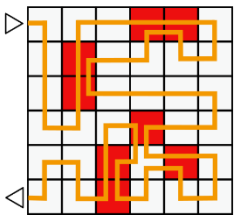


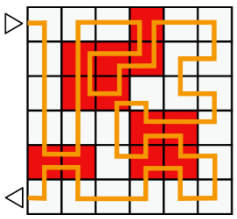
d)

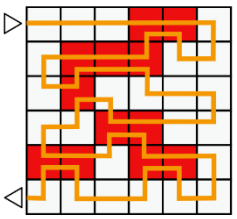


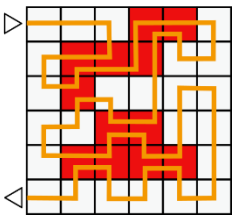
Paaiškinimas

Teisingas atsakymas yra: C.

a) 

b) 

c) 

d) 

Jei pasirinksite kelią, kuriame ant šviečiančios plytelės užlipama du kartus, o ant kitų – po vieną kartą, bus įjungtos visos plytelės. Patikrinkite, ar raštas, sudarytas iš plytelių, ant kurių užlipta du kartus, atitinka pradinį raštą.

Tai informatika!

Gebėjimas sekti procedūrą yra svarbus programavime. Šios užduoties atsakymą randame sekdami, kaip įjungiamų ir išjungiamų plytelių raštas (tai yra, programos apdorojami duomenys) kinta priklausomai nuo pasirinkto kelio.

Tačiau sekti visus pasikeitimus užtrunka laiko, taigi, taip pat svarbu atmesti netinkamus atsakymus supratus plytelių įjungimo ir išjungimo principą.

Užduotyje naudojamos plytelės reprezentuoja mažiausią ekrano laukelį – pikselį: https://en.wikipedia.org/wiki/Pixel

6. Pietų stalas

Matas organizuoja šventinius pietus. Jis pakvietė savo draugą Robertą ir dvi bebrų šeimas. Jonų šeima: mama ir tėtis su sūnumi Geraldu ir dukra Rože, ir Petrų šeima: mama ir tėtis su dukromis Lilija ir Daina.

Matas sugalvojo keletą taisyklių, kaip susodinti svečius aplink apvalų pietų stalą, kad būtų paskatinti nauji pokalbiai.

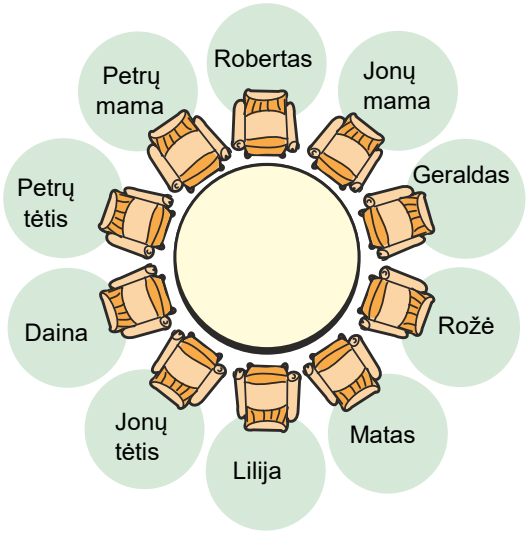
1) Dvi bebrės negali sėdėti šalia.

2) Tos pačios bebrų šeimos mama ir tėtis negali sėdėti šalia.

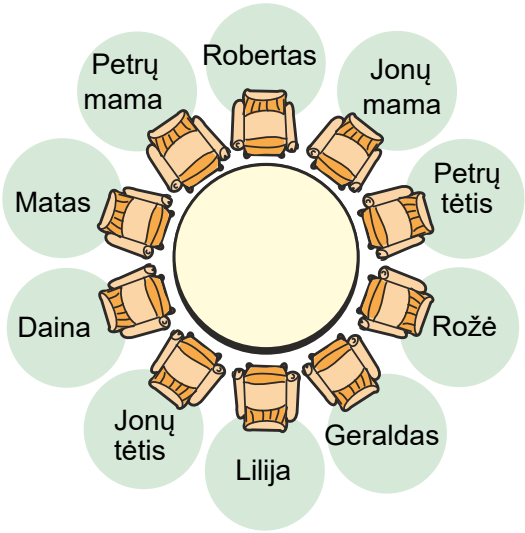
3) Jauni bebrai negali sėdėti šalia savo tėvų.

Kuris iš pateiktų susodinimo aplink stalą variantų yra teisingas?

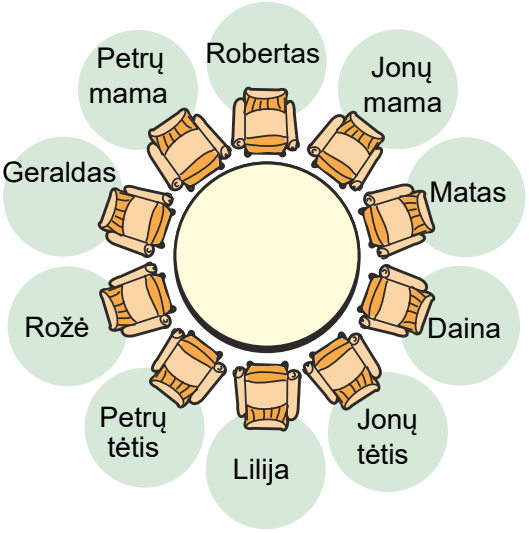
A.



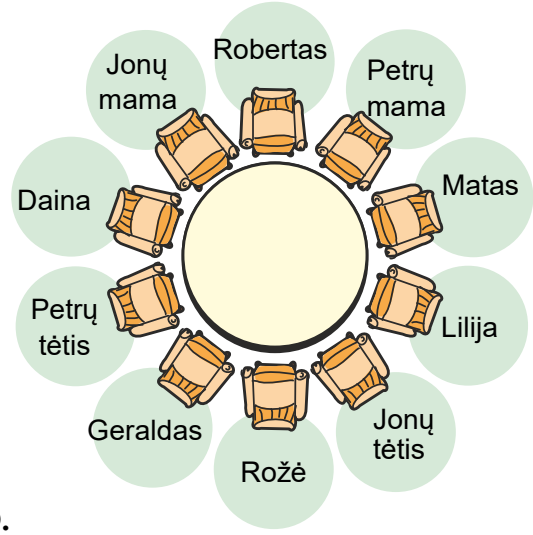
B.



C.



D.



Paaiškinimas

Teisingas atsakymas: B.

A variante Petrų šeimos mama ir tėtis sėdi vienas šalia kito. Tai pažeidžia antrą taisyklę. Daina taip pat sėdi šalia savo tėčio, Petrų tėčio, o tai neatitinka trečios taisyklės.

C variante Lilija sėdi šalia Petrų šeimos tėčio, t. y., savo tėčio. Tai pažeidžia trečią taisyklę.

D atveju Daina sėdi šalia Jonų šeimos mamos, o tai pažeidžia pirmą taisyklę. Rožė sėdi šalia savo tėčio (Jonų šeimos tėčio), o Daina – šalia savo tėčio (Petrų šeimos tėčio). Tai netinka pagal antrą taisyklę.

Visos taisyklės (apribojimai) tenkinamos B variante, kuris yra teisingas atsakymas.

Tai informatika!

Pietų stalo uždavinys – tai supaprastintas variantas bendresnių apribojimo uždavinių, kai turima apribojimų aibė (taisyklės ir ribojimai), reikšmių sritis, o reikia rasti sprendinį, kuris tenkina visus apribojimus. Pasirenkame kintamuosius, nurodome, kokias reikšmes jie gali įgyti, ir sprendžiame, kokius apribojimus reikia taikyti. Šiame uždavinyje kintamieji – tai kiekvieno bebro užimamos vietos prie stalo. Tada bandome ieškoti sprendinio žingsnio, priskirdami tinkamas reikšmes kiekvienam kintamajam ir išplėsdami šį procesą kitiems kintamiesiems. Kai kuriais atvejais, jei priskirti nepavyksta, grįžtame atgal, priskiriame kitas reikšmes ir bandome iš naujo.

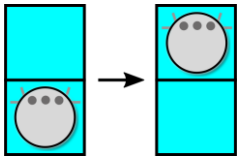
Šiame uždavinyje viskas paprasčiau: yra pateiktos reikšmės atsakymų variantuose, o mums tereikia patikrinti, ar tai tenkina taisykles (apribojimus).

	Jauniai	Kolegos
	9	6

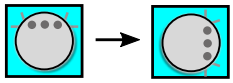
7. Siurblys-robotas

Dulkių siurblys-robotas  juda 6×7 kvadratinių plytelių kambaryje, apsuptame sienų. Robotas visuomet juda iš vienos plytelės vidurio į kitos plytelės vidurį ir yra visuomet atsisukęs vienos iš sienų kryptimi. Jis gali judėti pagal tokias komandas:

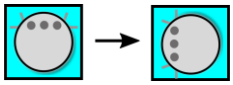
- PIRMYN: pajudėti į gretimą plytelę roboto žiūrėjimo kryptimi



- DEŠINĖN: vietoje pasisukti 90 laipsnių pagal laikrodžio rodyklę



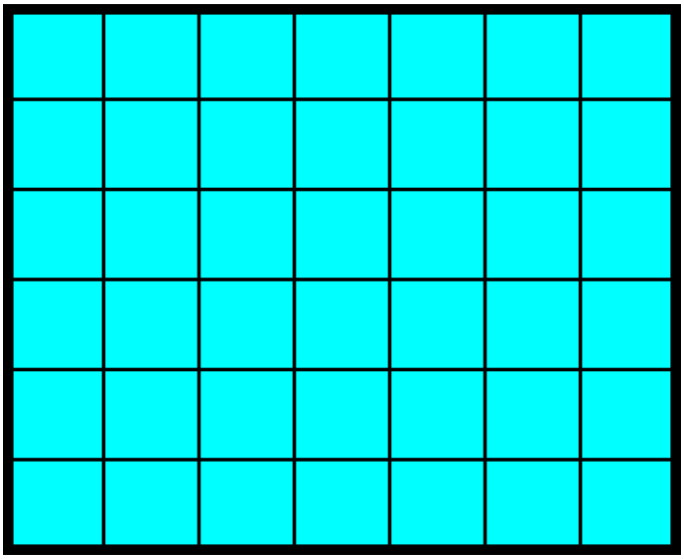
- KAIRĖN: vietoje pasisukti 90 laipsnių prieš laikrodžio rodyklę



Dulkių siurblys-robotas turi įvykdyti tokią programą:

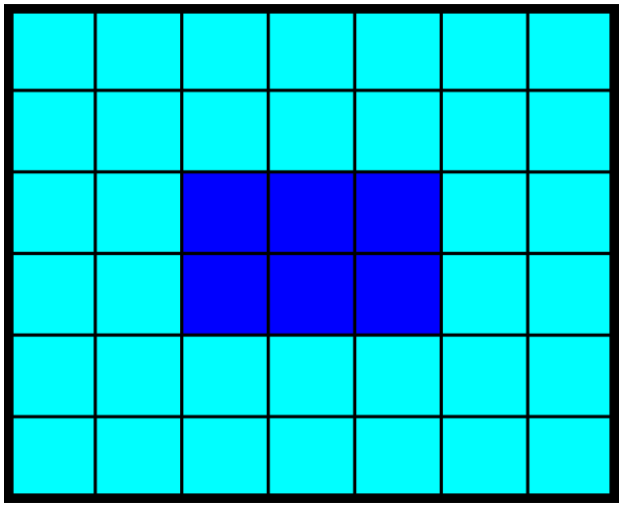
PIRMYN KAIRĖN PIRMYN DEŠINĖN PIRMYN

Keliose plytelėse dulkių siurblys-robotas gali pradėti šią programą taip, kad ją visą įvykdytų neatsitrenkdamas į sieną nepriklausomai nuo pradinės krypties?

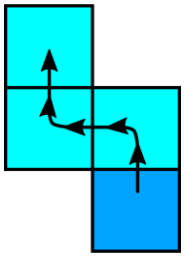


Paaiškinimas

Teisingas atsakymas: 6. Atsakymas parodytas paveikslėlyje.



Vykdydamas programą robotas nueis du langelius į priekį ir vieną langelį į kairę:



Reikia atkreipti dėmesį, kad tam, jog langelis tiktų atsakymui, šiems roboto judesiams vietos turi būti visomis keturiomis kryptimis.

Jei robotas pradės programą vienoje iš sieną liečiančių plytelių (pavadinkime tokias plyteles kraštinėmis plytelėmis) arba vienoje iš kraštinę plytelę liečiančių plytelių, tuomet bus bent viena kryptis (nukreipta į artimiausią sieną), kuria judėdamas robotas būtinai atsitrenks į sieną.

Pažymėtos plytelės yra saugios, nes yra pakankamai toli nuo sienų.

Tai informatika!

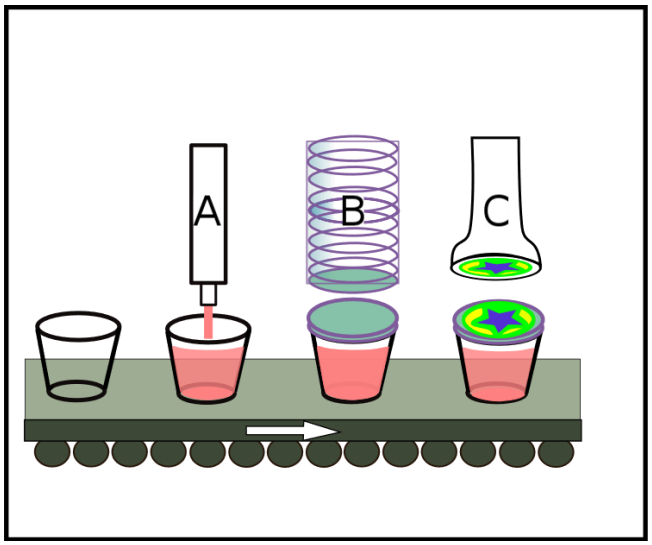
Kurdami programas programuotojai turi galvoti apie situacijas, iškylančias vykdant programą. Jie turi apsvarstyti veiksnius, vedančius į kompiuterio ar roboto, vykdančio programą, gedimus ar klaidingą elgesį. Taigi, programuotojai privalo patikrinti ir apibrėžti pradines sąlygas, kuriomis programa veiks. Sakome, jie turi gebėti *testuoti* programą.

Mūsų siurblys-robotas elgiasi taip, kaip vienos pirmųjų programavimo kalbų vaikams herojus Karelas. Jo vardas turėtų priminti čekų rašytoją Karelą Čapeką, pirmą kartą pavartojusį žodį robotas apibūdinti protingus mechaninius darbininkus 1921 metais išleistoje pjesėje „R.U.R“. Šio rašytojo futuristinė pjesė apibūdina gamyklą, kurioje robotai buvo kuriami *dirbti žmonėms*, o ne *vietoje žmonių*, tikslas buvo atnešti rojų į žemę. Tačiau niekas nesitęsia amžinai ir net robotai gali pradėti galvoti patys.

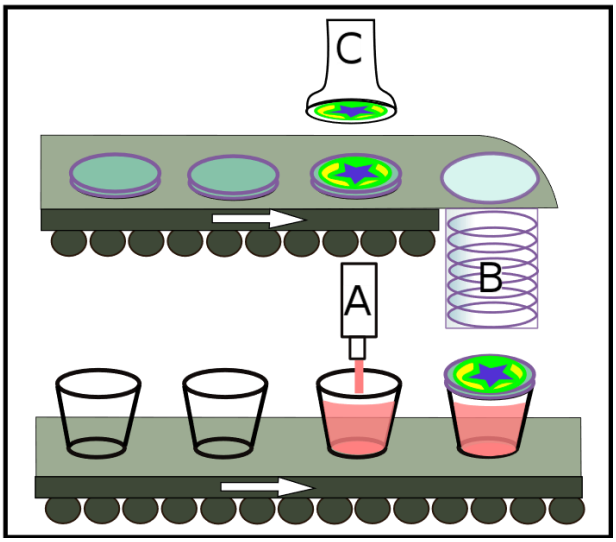
8. Jogurto fasavimo aparatas

Trys robotai fasuoja jogurtus: robotas A užpildo indelį jogurtu, robotas B uždeda dangtelį, o robotas C užklijuoja lipduką ant dangtelio. Kiekviena operacija trunka vieną sekundę.

Technologas Bobas nusprendžia pagaminti du skirtingus automatus. Pirmajame automate veikia viena konvejerio linija, o antrajame dvi (žr. paveikslus). Visų konvejerių judėjimo greitis pastovus ir vienodas. Laikas pradedamas skaičiuoti nuo tada, kai pirmasis indelis pasiekia pirmąjį robotą.



1 automatas



2 automatas

Reikia sufasuoti 120 jogurto indelių. **Kiek sekundžių bus sutaupyta naudojantis antruoju automatu?**

Paaiškinimas

Atsakymas: 1.

Lengva pastebėti, kad tik fasuojant pirmąjį indelį antrajame automate bus laimėta viena sekundė lyginant su pirmuoju automatu. Toliau fasuojant laikas jau nebus taupomas, nepriklausomai nuo to, kiek indelių reikia sufasuoti.

Tai informatika!

Šiame uždavinyje atskleidžiama, kad ne visada lygiagretieji algoritmai pagreitina uždavinio sprendimą. Šis uždavinys puikiai demonstruoja lygiagrečiojo algoritmo nulinį efektyvumą, kai lygiagretieji procesai naudoja tą patį išteklių.

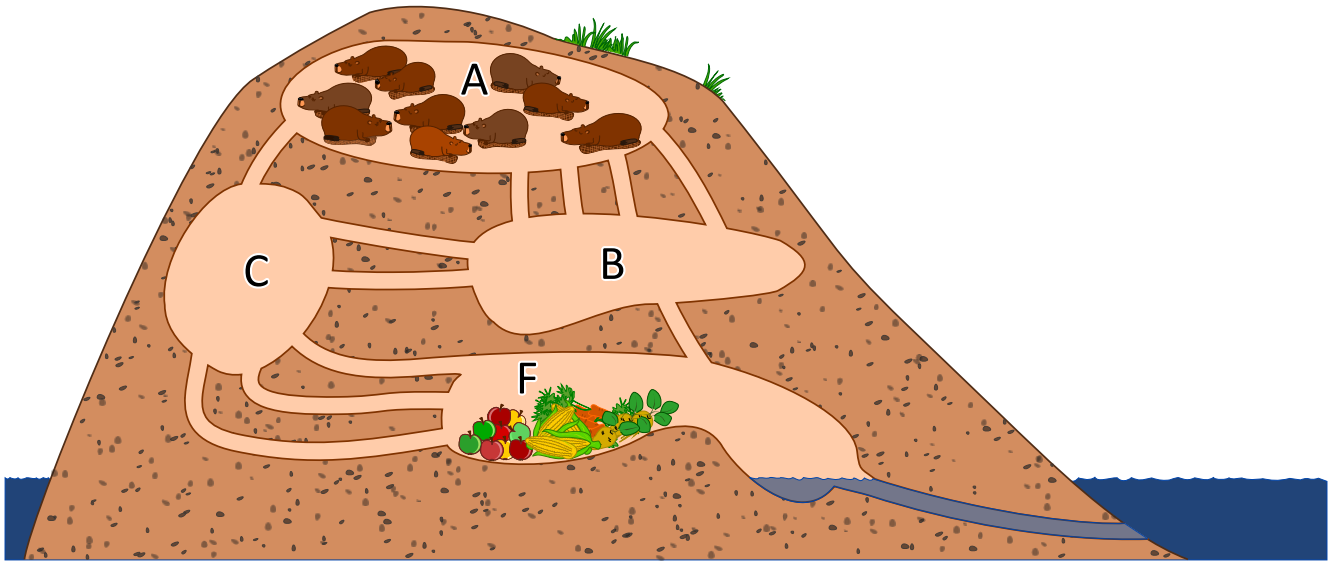
http://www.ims.mii.lt/EK%C5%BD/enciklo.html?word=lygiagretusis%20algoritmas

https://lt.wikipedia.org/wiki/Lygiagretusis_programavimas

http://en.wikipedia.org/wiki/Parallel_algorithm

9. Užtvankos tuneliai

Užtvankos dugne iškasti tuneliai, jungiantys keturis urvus A, B, C, F. Pirmieji trys urvai (A, B ir C) yra gyvenamieji, ketvirtasis urvas (F) yra maisto sandėlis (žr. paveikslą).



Urve A gyvena 10 bebrų. Kai bebrai išalksta, nori kuo greičiau patekti į maisto sandėlį F.

Keliavimas bet kuriuo tuneliu trunka 1 minutę. Vienu metu tunelyje gali būti tik vienas bebras. Tuneliai tarp urvų pasiskirstę taip:

- tarp A ir B yra 4 tuneliai,
- tarp A ir C yra 1 tunelis,
- tarp B ir C yra 2 tuneliai,
- tarp B ir F yra 1 tunelis,
- tarp C ir F yra 3 tuneliai.

Urvai yra dideli, todėl bet kuriame iš jų gali tilpti visi bebrai.

Kiek minučių trunka greičiausias visų bebrų perėjimas iš urvo A į maisto sandėlį F?

Paaiškinimas

Geriausiu atveju visi bebrai maisto sandėlį pasieks per 4 minutes. Tunelių ir urvų visuma sudaro grafą. Grafas turi du trumpiausius maršrutus, kurių abiejų pralaidumas yra po 1 bebrą, tad abi kelionės trunka 2 minutes:

- nuo A iki B ir nuo B iki F
- nuo A iki C ir nuo C iki F

Yra ir pralaidesnis maršrutas (2 bebrai), bet juo kelionė trunka 3 minutes:

- nuo A iki B, nuo B iki C ir nuo C iki F.

Norint optimizuoti bendrą laiką, pralaidumas tarp A ir B turi būti sumažintas. Optimalus sprendimas – iš A į B siųsti tris bebrus pirmąją minutę ir tris antrąją minutę. Atsižvelgiama į tai, kad iš B urvo išeina tik 3 tuneliai. Todėl, jei A ir B urvų jungtis naudojama, kai pralaidumas didžiausias (4 bebrai), B ir C, B ir F urvų jungtyse susidaro spūstis, todėl vienas bebras lieka laukti B urve. Pateiktoje lentelėje aiškinami veiksmai, atliekami kiekvieną minutę, kol visi bebrai pereina į maisto sandėlį. Šį optimalų uždavinio sprendimą (4 minutės), galima gauti keliais būdais. Pasirenkame tą būdą, kuriuo keliaujantiems bebrams nereikėtų laukti urve B.

Veiksmas (situacija)	*Bebrų skaičius urvuose (atlikus veiksmą)*			
	A	B	C	F
Pradinė situacija	10	0	0	0
3 bebrai pereina iš A į B (kai pralaidumas nėra didžiausias)				
1 bebras pereina iš A į C				
Situacija po 1 minutės	6	3	1	0
3 bebrai pereina iš A į B (kai pralaidumas nėra didžiausias)				
1 bebras pereina iš B į F				
2 bebrai pereina iš B į C				
1 bebras pereina iš C į F				
1 bebras pereina iš A į C				
Situacija po 2 minučių	2	3	3	2
1 bebras pereina iš A į B (pasirenkamas trumpiausias kelias)				
1 bebras pereina iš B į F				
2 bebrai pereina iš B į C				
1 bebras pereina iš A į C (pasirenkamas trumpiausias kelias)				
3 bebrai pereina iš C į F				
Situacija po 3 minučių	0	1	3	6
1 bebras pereina iš B į F				
3 bebrai pereina iš C į F				
Situacija po 4 minučių	0	0	0	10

Tai informatika!

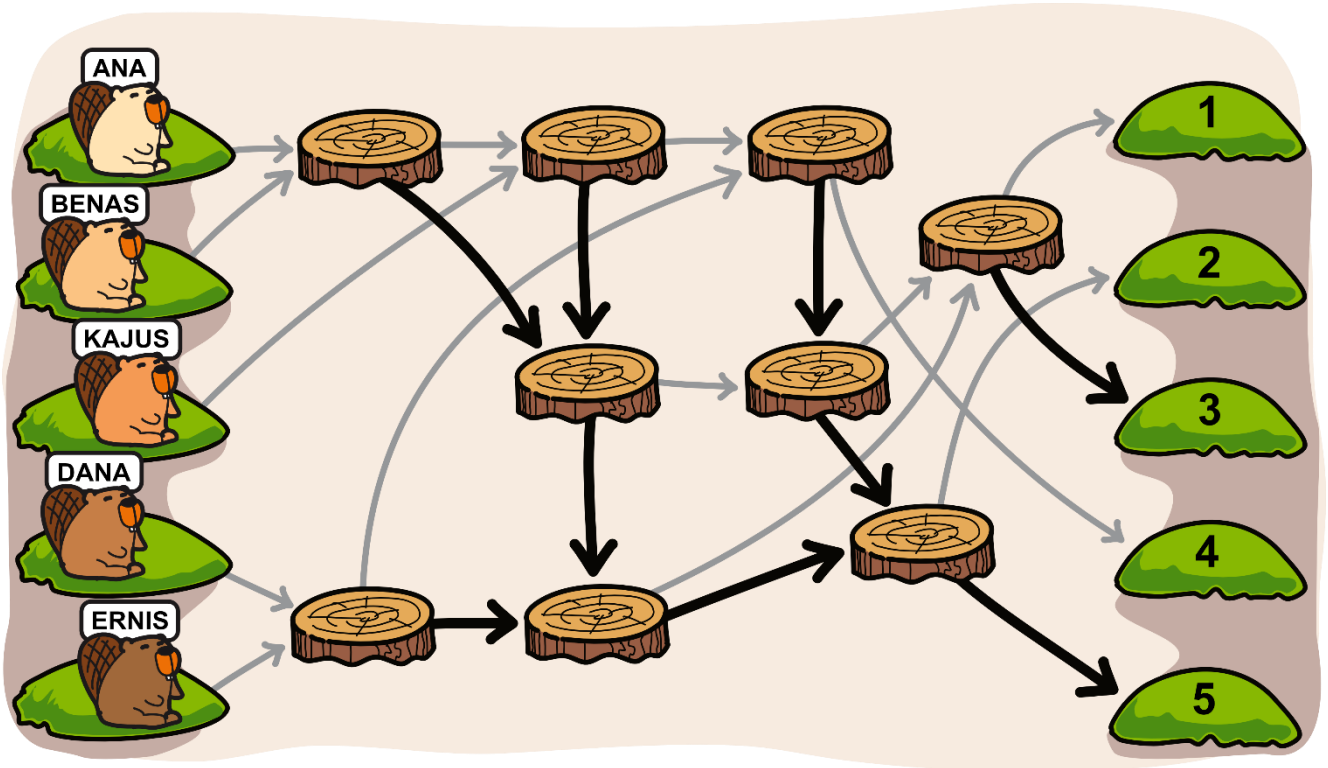
Grafų teorijoje tunelių tinklą galima įsivaizduoti kaip srautą tinkle. Iš tiesų, tai orientuotasis grafas, kurio kiekviena briauna (tunelis) yra tam tikro pralaidumo (didžiausias tunelių ar keliaujančių bebrų skaičius tarp urvų). Kiekvienos briaunos srautas negali viršyti jos pralaidumo.

Siekiama optimizuoti bebrų srautą tinkle, kad visi bebrai kuo greičiau pasiektų tikslą. Pavyzdžiui, toks tinklas gali būti naudojamas eismui modeliuoti transporto kelių sistemoje. Šioms problemoms spręsti gali būti naudojami keli algoritmai, vienas kurių yra Fordo ir Fulkersono algoritmas.

Mūsų sprendžiamas uždavinys taip pat yra vienas iš šios problemos atvejų: jei nėra galimybės toliau keliauti, leidžiama laukti urvuose B ir C. Klasikinio srauto tinklo uždavinio formuluotėje laukimas nenumatomas. Viskas, kas ateina, turi iš karto pasiskirstyti per kitus kanalus. Be to, paprastai formuluojant šią problemą nurodomos kiekvieno tunelio kryptys. Čia galima laisvai pasirinkti, kuria kryptimi einama tuneliais.

10. Šokinėjimas rąstais

Penki bebriukai šokinėja pelkėje plūduriuojančiais rąstais. Bebriukų amžius: Ana – 7 metų, Benas – 8 metų, Kajus – 9 metų, Dana – 10 metų ir Ernis – 11 metų. Šokinėjama pagal tokią taisyklę: ant bet kurio rąsto užšokęs pirmas bebriukas turi laukti, kol užšoks dar vienas. Tuomet vyresnis bebriukas turi peršokti ant rąsto, į kurį rodo stora juoda rodyklė, o jaunesnis šoka plonos pilkos rodyklės kryptimi.



Kaip išsidėstys bebriukai, kai baigs šokinėti?

a)
1: Ana
2: Benas
3: Kajus
4: Dana
5: Ernis

b)
1: Ernis
2: Dana
3: Kajus
4: Benas
5: Ana

c)
1: Benas
2: Dana
3: Kajus
4: Ana
5: Ernis

d)
1: Benas
2: Kajus
3: Dana
4: Ana
5: Ernis

Paaiškinimas

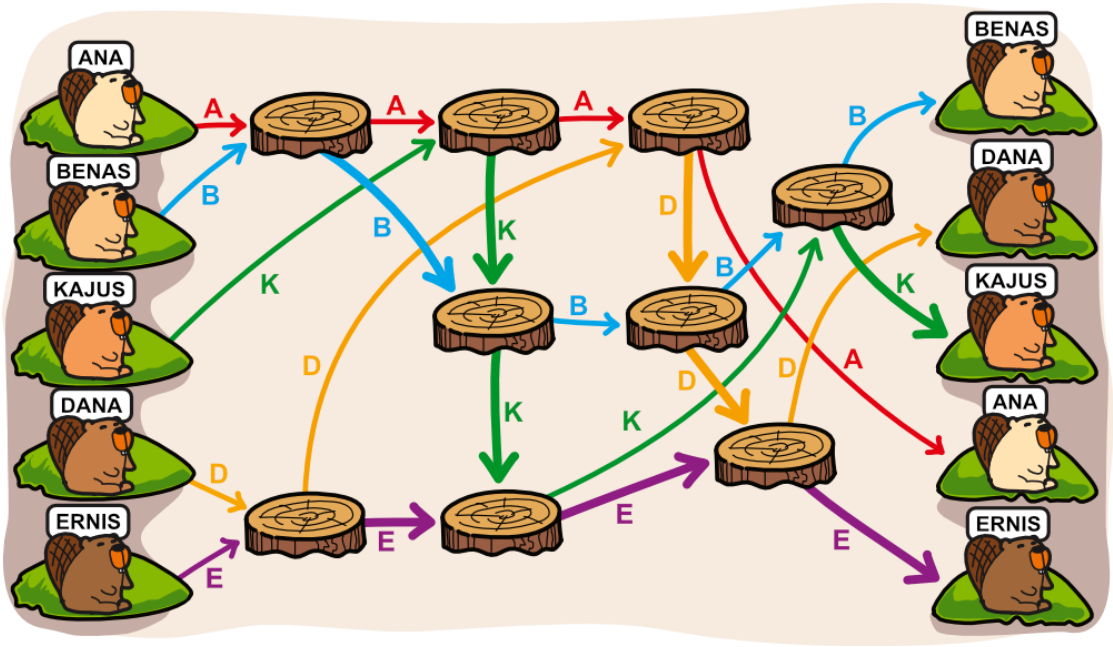
Atsakymai *a* ir *b* neteisingi. Jauniausia bebrė Ana visada šoks ten, kur rodo plona pilka rodyklė, ir baigs šokinėti 4-ame išėjime. Be to, *b* neteisingas dar ir todėl, kad vyriausias bebriukas Ernis visada šoks ten, kur rodo stora juoda rodyklė, ir baigs 5-ame išėjime.

Teisingas atsakymas: c.

Atsakymas *d* neteisingas.

Kadangi Ana baigs šokinėti 4-ame išėjime, o Ernis– 5-ame, tai teisingas turėtų būti vienas iš paskutinių dviejų *c* arba *d* atsakymų. Reikėtų atkreipti dėmesį, kad bebrė Dana šoka plonos pilkos rodyklės kryptimi tik tuomet, kai susitinka su vyresniu bebriuku Erniu. Visais kitais atvejais ji šoka ten, kur rodo juoda stora rodyklė. Taigi, plona rodykle ji šoka nuo pirmojo rąsto kairiajame apatiniame piešinio kampe. Toliau šoka storos rodyklės kryptimi, vėl su Erniu susitinka ant paskutiniojo rąsto dešiniajame apatiniame piešinio kampe ir baigia šokinėti 2-ajme išėjime.

Paveiksle kiekvieno bebriuko kelias pažymėtas vis kita spalva ir vardo raide.



Tai informatika!

Šis bebriukų šokinėjimas nurodytomis kryptimis iš tiesų atitinka kompiuterių tinkluose vykstantiems rūšiavimo procesams. Įsivaizduokime, kad tinklą sudaro laidai ir komparatoriai, kurie lygina dviejų įeinančių laidų vertes. Jei šiame tinkle yra komparatorių, kurie sujungti tam tikru būdu, tai tokiu atveju galima surikiuoti nesurūšiuotų verčių seką. Toks tinklas yra vadinamas rikiavimo tinklu. Tokie rikiavimo tinklai veikia tuo pačiu metu ir gali būti labai efektyvūs rikiavimo uždaviniams spręsti.

Pateiktame pavyzdyje tinklas nėra skirtas rikiavimui, nes komparatoriai nėra sujungti pagal tam tikras taisykles, todėl tik parodomas duotos sekos perrikiavimas.

11. Trys darbininkai

Gamykloje dirba tik 3 darbininkai: Agnė, Benas ir Cezaris. Jie eina į darbą pagal nustatytas taisykles. Kiekvieną pirmadienį privalo dirbti lygiai 2 darbininkai. Kiekvieną kitą savaitės dieną dirbama šitaip:

- 1 taisyklė: jei Agnė ėjo į darbą, tai kitą dieną Benas turi likti namie.
- 2 taisyklė: jei Benas ėjo į darbą, tai ir Cezaris ėjo į darbą, bet kitą dieną Cezaris turi likti namie.
- 3 taisyklė: jei Cezaris nėjo į darbą, tai Agnė kitą dieną turi likti namie.

Kuris iš pateiktų savaitės darbo grafikų tenkina šias taisykles?

a)

Pirmadienis	*Antradienis*	*Trečiadienis*	*Ketvirtadienis*	*Penktadienis*
2	3	1	1	2

b)

Pirmadienis	*Antradienis*	*Trečiadienis*	*Ketvirtadienis*	*Penktadienis*
2	1	3	2	1

c)

Pirmadienis	*Antradienis*	*Trečiadienis*	*Ketvirtadienis*	*Penktadienis*
2	1	3	1	0

d)

Pirmadienis	*Antradienis*	*Trečiadienis*	*Ketvirtadienis*	*Penktadienis*
2	1	0	3	2

Paaiškinimas

Pažymėkime Agnę raide A, Beną – raide B, Cezarį – raide C. Galime pradėti nuo tikrinimo, kurie du darbininkai gali dirbti pirmadienį. Dviejų darbininkų komandos gali būti tokios: A-B, A-C ir B-C. Iš karto atmetame komandą A-B, nes pagal 2-ąją taisyklę kartu turėtų dirbti ir C, bet pirmadienį dirba tik 2 darbininkai, o ne 3. Todėl nagrinėjame komandas A-C ir B-C:

1 atvejis:

	Pirmadienis	*Antradienis*	*Trečiadienis*	*Ketvirtadienis*	*Penktadienis*
Darbe	A-C	?	?	?	?
Namie	B	B	?	?	?

2 atvejis:

	Pirmadienis	*Antradienis*	*Trečiadienis*	*Ketvirtadienis*	*Penktadienis*
Darbe	B-C	?	?	?	?
Namie	A	C	A	?	?

1-uoju atveju B neleidžiama dirbti antradienį pagal 1-ąją taisyklę. 2-uoju atveju C neleidžiama dirbti antradienį pagal 2-ąją taisyklę. Abiem atvejais antradienį negali dirbti 3 darbininkai. Todėl galima atmesti atsakymą *a*.

2-uoju atveju matome, kad pritaikoma ir 3-ioji taisyklė: A turi likti namie kitą dieną po to, kai C nėjo į darbą. Taigi, jeigu savaitėje yra tokia diena, kai nedirba nei vienas darbininkas, tai kitą dieną negali dirbti visi 3 darbininkai, nes A turi likti namie. Todėl galima atmesti atsakymą *d*.

Grafikas *b* tinka tik iki trečiadienio: tarkime, kad pirmadienį dirba A-C (1 atvejis), tai antradienį gali dirbti C , o trečiadienį gali dirbti visi 3 darbininkai. Todėl iki trečiadienio vienintelis galimas sprendimas yra toks:

	Pirmadienis	*Antradienis*	*Trečiadienis*	*Ketvirtadienis*	*Penktadienis*
Darbe	A-C	C	A-B-C	?	?
Namie	B	A-B	—	B-C	A

Bet remiantis taisyklėmis, jeigu visi 3 dirba trečiadienį, tai ketvirtadienį turi likti namie B (1-oji taisyklė) ir C (2-oji taisyklė). Taigi, ketvirtadienį gali dirbti tik A. Todėl atmetame ir grafiką *b*.

Lieka vienintelis galimas grafikas *c*. Vienintelis galimas darbininkų paskirstymas, kuris tenkina visas taisykles, yra toks:

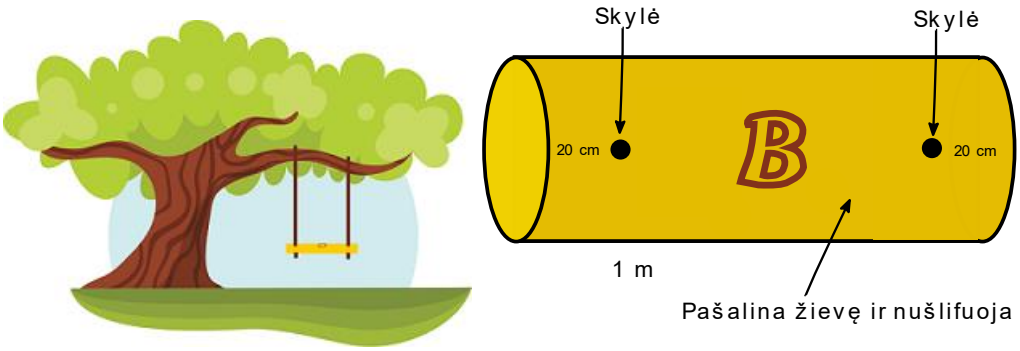
	Pirmadienis	*Antradienis*	*Trečiadienis*	*Ketvirtadienis*	*Penktadienis*
Darbe	A-C	C	A-B-C	A	—
Namie	B	A-B	—	B-C	A-B-C

Tai informatika!

Šiam uždaviniui išspręsti gali būti panaudojamos kelios strategijos. Pateiktame atsakyme demonstruojamas grįžimo atgal (angl. *backtracking*) metodas: iškeliame hipotezę (pavyzdžiui, „A ir B dirba pirmadienį“) ir tikriname, ar ji neprieštarauja taisyklėms, arba naudojant techninę terminologiją – ar ji tenkina apribojimus (angl. *constraint satisfying*). Jei taip, toliau samprotaujame galbūt iškeldami daugiau tolesniam sprendimui reikalingų hipotezių. Jei ne, grįžtame atgal, iškeliame kitas hipotezes ir bandome išspręsti uždavinį remdamiesi šiomis naujomis hipotezėmis.

Žmonėms sunku sekti, kurioje sprendimo proceso vietoje esame, ir būti tikriems, kad sistemingai keliame hipotezes – tai užtikrintai leistų pasiekti sprendimą, jei jis egzistuoja. Geras hipotezių kėlimo pavyzdys yra sudoku galvosūkių sprendimas. Sudoku yra sudėtingas, sunku rasti atsakymą, nes reikia kelti daug hipotezių ir jas atmetinėjant ieškoti sprendimo. Kompiuteriai labai gerai tinka sistemingam tyrimui. Per daugelį metų buvo sukurta daugybė konkrečių algoritmų uždaviniams spręsti, paremtų grįžimo atgal metodu.

12. Supynės iš rąsto



Gauta nauja ilgų rąstų siunta. Keturių robotų komanda iš šių rąstų pagal pateiktą brėžinį gamins sūpynių skersinius. Kiekvienas robotas gali atlikti tik vieną veiksmą.

Pjovėjas: pjausto rąstus į 1 metro ilgio ruošinius.

Gręžėjas: išgręžia po dvi skyles nušlifuoto ruošinio galuose – 20 cm nuo kairiojo ir dešiniojo galo.

Spausdintojas: nušlifuoto ruošinio viduryje spausdina ar išdegina įmonės logotipą.

Nužievintojas: nuvalo ruošinio paviršių (pašalina žievę ir nušlifuoja).

Kiekvienas robotas vienu metu dirba tik su vienu ruošiniu ir tik vienas pats jį apdoroja. Robotas baigia darbą tik tada, kai apdoroja visus ruošinius.

Robotas pradeda darbą, kai gauna valdančios programos signalą. Robotas negali dirbti, jei jam nėra paruoštų, jo atliekamam veiksmui tinkamų, ruošinių. Valdančios programos veiksmai atliekami nuosekliai.

Parengtos 4 valdymo programos. **Tik viena jų yra tinkama sūpynių skersiniams paruošti. Kuri?**

A)	`Paleisti Pjovėją`
	`Paleisti Nužievintoją`
	`Laukti, kol robotai baigs darbą`
	`Paleisti Gręžėją`
	`Paleisti Spausdintoją`
		C)	`Paleisti Nužievintoją`
			`Paleisti Spausdintoją`
			`Laukti, kol robotai baigs darbą`
			`Paleisti Pjovėją`
			`Paleisti Gręžėją`

B)	`Paleisti Pjovėją`
	`Paleisti Spausdintoją`
	`Laukti, kol robotai baigs darbą`
	`Paleisti Nužievintoją`
	`Paleisti Gręžėją`
		D)	`Paleisti Pjovėją`
			`Paleisti Spausdintoją`
			`Paleisti Nužievintoją`
			`Laukti, kol robotai baigs darbą`
			`Paleisti Gręžėją`

Paaiškinimas

Teisingas atsakymas: A.

Komandas prieš laukimą galima sukeisti vietomis. Po laukimo – taip pat.

Nužievinimo ir pjovimo veiksmai vienas kitam netrukdo, todėl robotai gali veikti lygiagrečiai. Gręžėjui reikia lygiai 1 m ilgio ruošinių. Spausdintojui taip pat reikia 1 m ruošinių, tik nužievintų ir nušlifuotų. Todėl reikia sulaukti, kol pirmoji pora baigs darbą.

B ir D atvejais spausdinimas gali būti atliekamas su nenužievintu ruošiniu, tai užblokuotų sistemą. C atveju spausdinantis robotas gauna neatpjautus ruošinius, todėl negali nustatyti ruošinio vidurio.

Tai informatika!

Ši užduotis iliustruoja du pagrindinius lygiagretaus skaičiavimo metodus: 1) susiejimą ir 2) procesų sinchronizavimą.

1. Susiejimas yra alternatyva pakartojimui (iteracijai). Apibrėžiate funkciją ir priskiriate ją rinkiniui. Rinkinys yra objektų aibė, pavyzdžiui, šioje užduotyje rinkinys yra rąstų siunta. Tai reiškia, kad funkcija yra atliekama kiekvienam rinkinio objektui, tačiau nesvarbu, kokia tvarka tai vyksta. Kadangi funkcija gali būti vykdoma lygiagrečiai (vykdymo tvarka neturi reikšmės), todėl šiai funkcijai įvykdyti galime panaudoti kelis procesorius. Matome, kad susiejimas gali būti greitesnis nei pakartojimas.

Pavyzdys (*Python* programavimo kalba):

```
>>> fruits = ['apple', 'orange', 'cherry']
>>> s = map(str.upper, fruits)
>>> print(list(s))
['APPLE', 'ORANGE', 'CHERRY']
```

Šiame pavyzdyje matome, kad metodas `upper()` yra susietas su tekstiniu objektų masyvu.

2. Šioje užduotyje esantys robotai gali vienu metu lygiagrečiai dirbti su rąstų rinkiniu. Valdymo programa užtikrina, kad Gręžėjas ir Spausdintojas nepradėtų dirbti tol, kol Pjovėjas ir Nužievintojas nebaigs savo darbo. Tai vadinama proceso sinchronizacija. Kompiuteryje keli procesai gali veikti vienu metu, kad atliktų užduotį – tai reiškia, kad eilės tvarka, kuria kiekvienas atlieka savo darbą, neturi reikšmės. Uždavinio sprendimas vienu metu naudojantis keliais procesais vadinamas lygiagrečiuoju apdorojimu. Lygiagrečios kompiuterinės programos veikia greičiausiai, kai jose naudojami lygiagretūs procesai pradeda veikti vienu metu.

	Jauniai	Kolegos
	12	9

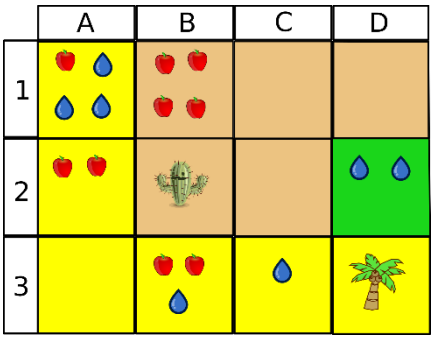
13. Dykuma

Benas keliauja aplink pasaulį. Kelionę dykumoje pradeda iš langelio A1 ir nori pasiekti palmę langelyje D3. Einama kairėn, dešinėn, aukštyn ir žemyn, bet negalima eiti įstrižai. Per tą patį langelį galima eiti tik vieną kartą. Langelius su kaktusu būtina apeiti.

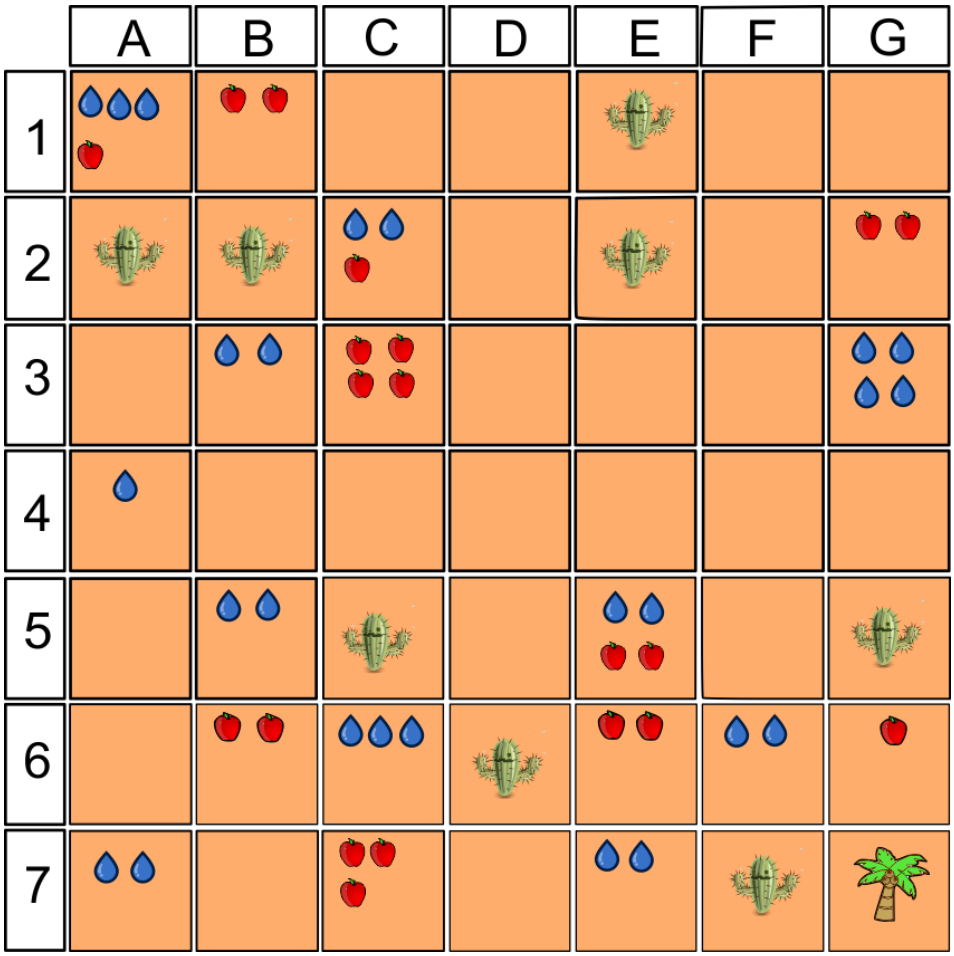
Langeliuose reikia rinkti obuolius ir vandens lašus. Pereinant iš vieno langelio į kitą netenkama vieno lašo vandens ir vieno obuolio. Jei Benas neturi vandens ir obuolio, negali toliau keliauti.

Pavyzdyje geltona spalva pažymėtas teisingas dykumos perėjimo kelias:

- A1: yra 3 lašai ir 1 obuolys
- A2: liko 2 lašai ir 2 obuoliai
- A3: liko 1 lašas ir 1 obuolys
- B3: liko 1 lašas ir 2 obuoliai
- C3: liko 1 lašas ir 1 obuolys
- D3: tikslas pasiektas!

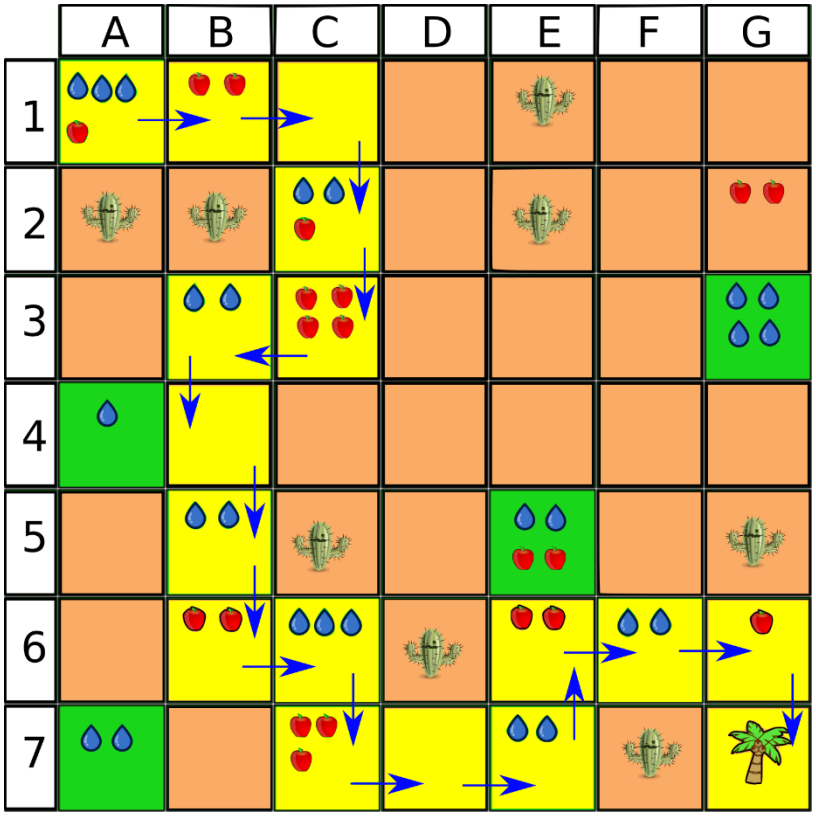


Padėkite Benui pereiti dykumą ir pasiekti langelį su palme.



Paaiškinimas

Yra du galimi keliai: vienas pavaizduotas paveiksle, kitas toks pat, tik dar papildomai aplankant E5 ir F5 langelius. Visuose kituose keliuose pritrūktų vandens arba obuolių.



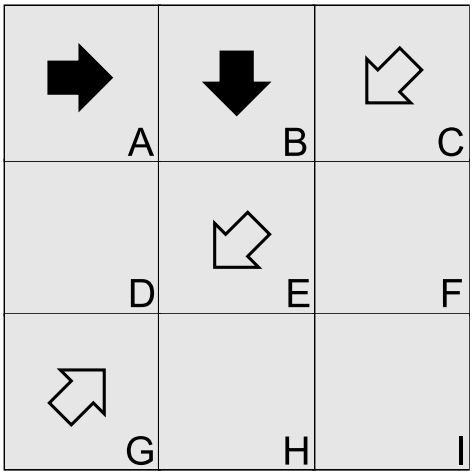
Tai informatika!

Sprendimas paremtas grafo paieškos į gylį algoritmu.

Paieška į gylį (angl. *depth-first search*, DFS) – paieška grafe arba grafo apėjimo būdas, kai pasirinkus pradinę viršūnę einama grafo briaunomis kiek įmanoma giliau, renkantis vis naują viršūnę; kai nebėra neaplankytos viršūnės, tuomet grįžtama iki ankstesnės neaplankytos viršūnės ir vėl ieškoma tol, kol bus pasiektas tikslas arba kol bus aplankytos visos grafo viršūnės.

Sprendžiant šį uždavinį vienu metu apdorojami keli veiksmai (skaičiuojami obuoliai ir lašai, numatoma, kur eiti).

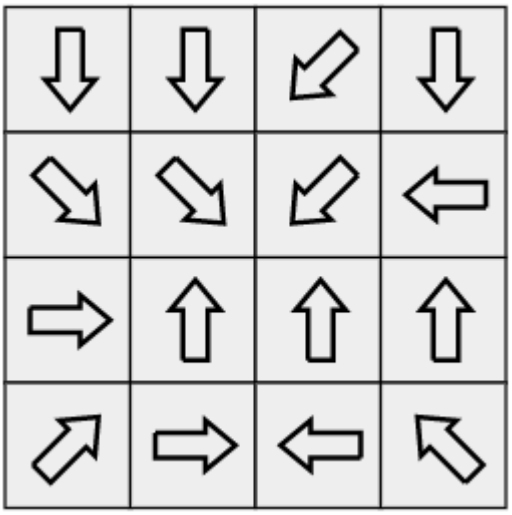
14. Rodyklių spalvinimas



A langelyje esanti nuspalvinta rodyklė rodo į vieną nuspalvintą rodyklę B langelyje ir į vieną nenuspalvintą rodyklę C langelyje.

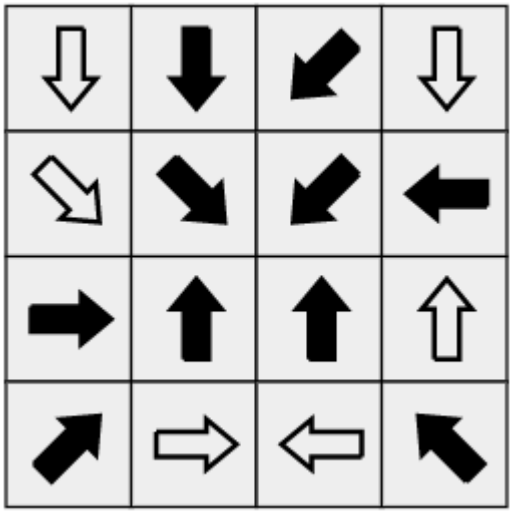
Nenuspalvinta rodyklė C langelyje rodo į dvi nenuspalvintas rodykles E ir G langeliuose.

Spustelėdami rodykles nuspalvinkite dalį paveiksle pateiktų rodyklių taip, kad kiekviena nenuspalvinta rodyklė rodytų į vieną nenuspalvintą rodyklę, o kiekviena nuspalvinta rodyklė rodytų į dvi nuspalvintas rodykles.



Paaiškinimas

Galimas sprendimas:



Tai informatika!

Sprendžiant uždavinį galima pradėti paiešką vykdant žingsnį po žingsnio, o jei atsiduriama aklavietėje (sprendimas nepavyko), grįžtama vienu žingsniu atgal ir bandoma atlikti kitą galimą žingsnį.

Informatikoje tai vadinama grįžimų metodu. Grįžimų metodas gali būti taikomas sprendžiant galvosūkius, aštuonių valdovių uždavinį, sudoku ar kombinatorinio optimizavimo uždavinius, pavyzdžiui, kuprinės uždavinį (žr. 18 uždavinio „Efektyvi gamykla“ skyrelį „Tai informatika!“).

Kai kuriose loginio programavimo kalbose, pavyzdžiui, *Icon*, *Planner*, *Prolog*, generuojant atsakymus taikomas grįžimų metodas.

Daugiau informacijos:

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

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

	Jauniai	Kolegos
	12	9

15. Knygų rikiavimas

Trys bebrai darbuojasi vienas greta kito paupio laukeliuose. Kiekvieno bebro laukelyje padėta po dvi knygas. Knygų tvarka sumaišyta ir bebrai nori jas surikiuoti atlikdami keitimo žingsnius.

Yra du žingsnių tipai.

- Esant pirmojo tipo žingsniui kiekvienas bebras gali sukeisti vietomis savo laukelio knygas
(A pavyzdys).
- Esant antrojo tipo žingsniui bebrai gali apsikeisti knygomis su kaimyniniais laukeliais (B pavyzdys).

Bebrai pradeda pirmuoju žingsniu keisdami vietomis knygas savo laukeliuose.

Kiek mažiausiai žingsnių reikės bebrams norint išrikiuoti knygas nuo 1-os iki 6-os?

A. 3 žingsnių

B. 4 žingsnių

C. 5 žingsnių

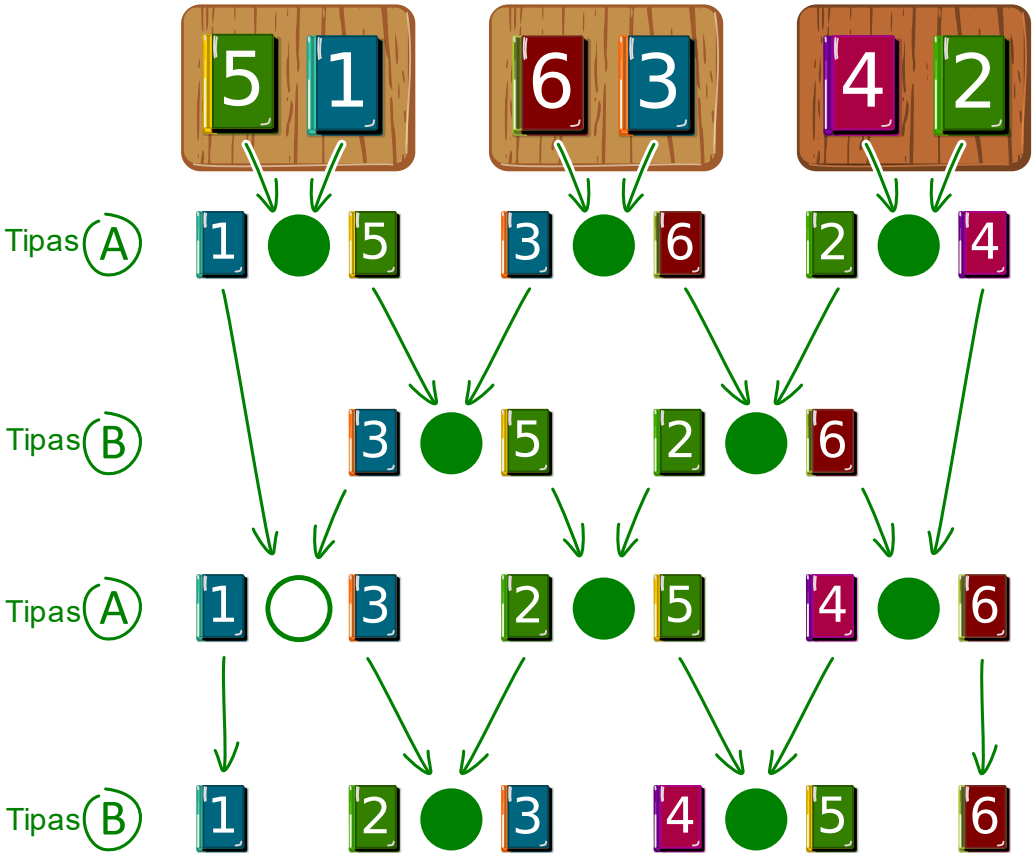
D. 6 žingsnių

Paaiškinimas

Teisingas atsakymas: B.

Šis uždavinys pagrįstas lygiagretaus tinklo rikiavimo algoritmu. Pirmiausia bebrai sukeičia vietomis knygas savo laukeliuose, kaip nurodyta uždavinio sąlygoje. Perrinkimo būdu nusprendžiama, ar atlikti sukeitimą, ar ne. Antruoju žingsniu visi bebrai stengiasi apsikeisti knygomis tarp savo laukelių. Tada vėl atliekamas pirmasis žingsnis – keičiamos vietomis knygos laukeliuose ir t. t.

Iš tiesų naudojamas godusis algoritmas (kas atrodo, jog geriausiai tuo metu tinka), jis intuityvus ir šiuo atveju optimalus. Taikant šį algoritmą, galima lengvai rasti atsakymą. Šio godžiojo algoritmo (ar strategijos) veiksmus pavaizduosime rikiavimo tinklu (žr. paveikslą).



Tai informatika!

Rikiavimas – objektų išdėstymas eile pagal kurį nors jų parametrą, pavyzdžiui, skaičių – pagal jų reikšmes, žodžių – pagal raidžių rikiavimo eilę abėcėlėje. Rikiuoti galima dvejopai: didėjančiai arba mažėjančiai (taip pat nedidėjančiai ir nemažėjančiai). Jeigu objektai skaidomi į kelias grupes, sakoma, kad jie rūšiuojami. Rūšiavimas gali būti panaudotas kaip pagalbinis rikiavimo veiksmas. Objektai surūšiuojami į grupes (rūšis), atliekamas rikiavimas, po to grupės sujungiamos ir gaunama surikiuotų objektų eilė.

Šiame uždavinyje rikiavimo procesas vyksta sparčiau, rikiuojama mažomis grupelėmis lygiagretinant, po to grupelės apjungiamos ir vėl rikiuojama.

16. Šnabždesių žaidimas

Penki draugai stovi eilutėje: Aušra, Benas, Cilė, Domas ir Egis. Aušra paraidžiui šnabžda 9 raidžių žodį Benui į ausį (pavyzdžiui. M-A-T-O-M-U-M-A-S). Tada Benas paraidžiui šnabžda žodį Cilei, bet padaro jame vieną klaidą. Klaida gali būti pakeista raidė (pavyzdžiui, M-A-L-O-M-U-M-A-S) arba praleista raidė (pavyzdžiui, M-A-T-O-U-M-A-S). Tuomet Cilė šnabžda žodį Domui, taip pat padarydama vieną klaidą. Tą patį padaro ir Domas.

Kiekviename šnabždesyje yra lygiai viena klaida, ne daugiau ir ne mažiau, tačiau ta pati raidė gali būti pakeista ar praleista daugiau nei vieną kartą.

Jeigu Aušros sužnabždėtas Benui žodis yra P-A-N-A-Š-U-M-A-S, tai kurie iš šių žodžių gali būti sužnabždėti Egiui? [Nurodykite visus, kurie tinka]

A) N-A-M-A-S

B) P-A-N-A-Š-U-M-A-I

C) N-A-Š-U-M-A-S

D) N-E-Š-I-M-A-S

E) P-A-N-A-M-A

Paaiškinimas

Teisingi atsakymai: B, C ir E variantai.

Aušra paraidžiui šnabžda teisingą žodį Benui. Benas šnabžda žodį su viena klaida Cilei. Cilė šnabžda žodį Domui su dar viena klaida. Galiausiai Domas šnabžda žodį Egiui su dar viena klaida. Taigi Egiui sušnabždėtame žodyje yra lygiai 3 klaidos.

A variantas klaidingas. Šiame žodyje yra tik 5 raidės, taigi buvo praleistos 4 raidės. Tačiau galima praleisti daugiausiai 3 raides.

B variantas teisingas. Raidė S galėjo būti pakeista raide I padarius pirmąją klaidą. Tada toliau šnabždant (antra klaida) pakeičiama bet kuri raidė (tarkime, A), o trečioji klaida – ši raidė vėl pakeičiama į pradinę raidę (A), t. y., P-A-N-A-Š-U-M-A-S → P-A-N-A-Š-U-M-A-I → P-A-N-A-Š-U-M-U-I → P-A-N-A-Š-U-M-A-I.

C variantas teisingas. Iš pradžių viena iš pirmų dviejų raidžių galėjo būti pakeista į bet kurią kitą, tuomet pirmos dvi raidės praleidžiamos, t. y., P-A-N-A-Š-U-M-A-S → F-A-N-A-Š-U-M-A-S → A-N-A-Š-U-M A-S → N-A-Š-U-M-A-S.

D variantas klaidingas. Žodyje yra 7 raidės, taigi 2 raidės buvo praleistos. Be to, žodyje yra 2 naujos raidės (E ir I), taigi buvo padaryti 2 keitimai. Tam reikėjo padaryti daugiau negu 3 klaidas.

E variantas teisingas. Šis žodis gautas praleidus 3 raides (Š, U ir S) ir daugiau nieko nepakeitus.

Tai informatika!

Šis uždavinys yra pavyzdys sąvokos, vadinamos informacijos perdavimo triukšmu. Triukšmas šiuo atveju – tai raidžių praleidimas ir pakeitimas. Žinome, kad triukšmas čia yra apribotas 3 raidėmis. Todėl galime būti tikri, kad galutinis žodis, kuris pasieks Egį, turės mažiausiai 7 raides, kurios sutaps su pradinio žodžio raidėmis ir bus savo vietoje.

Realiame gyvenime toks triukšmas dažniau pasireiškia perduodant skaitmeninius signalus, sudarytus iš bitų, t. y., signalus, kuriuos sudaro tik nuliai ir vienetai. Tokiuose signaluose gali įvykti tokios pat klaidos – simbolio pakeitimas ir praleidimas, tačiau šiuo atveju pakeitimas reiškia tik bito perjungimą, t. y., 0 tampa 1 ir atvirkščiai.

Klodas Šenonas (Claude Shannon), vadinamas informacijos teorijos tėvu, atliko keletą novatoriškų tyrimų, kaip pasiųsti skaitmeninį signalą beveik be klaidų, tam nagrinėdamas įtaką kanalų, kuriais signalas yra siunčiamas.

17. Matematinė mašina

Bebrai sukūrė matematinę mašiną. Ji priima natūraliuosius skaičius ir juos pertvarkiusi pateikia rezultatus. Mašinos viduje naudojami komponentai, kurie visi veikia vienodai. Kiekvienas komponentas priima po tris skaičius ir atlieka tokius veiksmus:

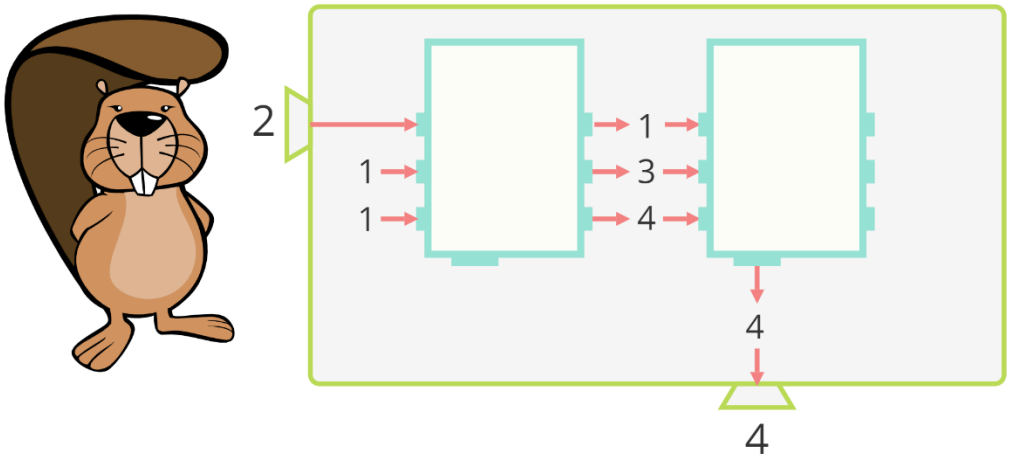
Jei pirmas skaičius yra 1, tai trečias skaičius yra mašinos pateikiamas rezultatas.

Priešingu atveju:

- pirmasis skaičius vienetu sumažinamas ir tampa nauju pirmuoju skaičiumi;
- antrasis skaičius padidinamas 2 ir tampa nauju antruoju skaičiumi;
- naujas antrasis skaičius sudedamas su trečiuoju ir ši suma tampa nauju trečiuoju skaičiumi;
- visi trys naujieji skaičiai siunčiami į naują komponentą tokia pačia tvarka.

Kai tik mašina gauna skaičių, ji siunčia šį skaičių komponentui kaip pirmą skaičių, o likusius du skaičius komponentui pateikia vienetus. Galutiniu mašinos rezultatu laikomas bet kurio komponento pateiktas galutinis rezultatas.

Paveiksle pavaizduota, kaip matematinė mašina atlieka pertvarkymus, gavusi skaičių 2, šiuo atveju naudodama tik du komponentus.



Kurie iš šių skaičių gali būti matematinės mašinos galutiniai rezultatai?

A. 1

B. 50

C. 100

D. 250

E. 400

Paaiškinimas

Teisingi atsakymai yra 1, 100 ir 400.

Mašina, gavusi pradinį duomenį n, sudeda pirmus n nelyginius skaičius, pradėdama 1, todėl galutinis rezultatas yra n^2.

Tai informatika!

Programavime dažnai didelis uždavinys suskaidomas į mažesnes dalis, sprendžiančias tarpusavyje panašius uždavinius. Šis programavimo būdas vadinamas „skaldyk ir valdyk“ (angl. *divide and conquer*). Rekursija yra viena iš svarbių informatikos mokslo idėjų. Rekursija naudojama, kai funkcijos ar procedūros kreipiasi pačios į save.

Rekursijos galia akivaizdžiai slypi galimybėje apibrėžti begalinę objektų aibę baigtiniu teiginiu. Taip begalinis veiksmų skaičius gali būti aprašomas baigtine rekursine programa, net jei šioje programoje nėra aiškių pasikartojimų. Uodegos rekursija (angl. *tail call*) yra ypač naudinga ir ją lengva įgyvendinti. Uodegos rekursija gali būti vykdoma nepridėjus naujo programos dėklo kadro (angl. *stack frame*) į dėklą.

18. Efektyvi gamykla

Automobilių gamykla gavo užsakymą pagaminti penkias detales. Detalės pirmiausia yra gaminamos A padalinyje, o po to dažomos ir dekoruojamos B padalinyje. Kiekvienas padalinys vienu metu gali užsiimti tik su viena detale. Penkių detalių pagaminimo laikas valandomis A ir B padaliniuose pavaizduotas lentelėje.

Kodas	*Detalė*	*Padalinys A*	*Padalinys B*
1	Galinė kėbulo dalis	3 val.	6 val.
2	Priekinė kėbulo dalis	5 val.	2 val.
3	Ratai	8 val.	1 val.
4	Durys	7 val.	4 val.
5	Rėmas	10 val.	9 val.

Detales reikia gaminti tokia tvarka, kad visam užsakymui atlikti būtų sugaišta mažiausiai laiko.

Kiek trumpiausiai gali užtrukti šio užsakymo detalių pagaminimas?

A. 33 val.

B. 34 val.

C. 36 val.

D. 40 val.

Paaiškinimas

Kadangi padalinys B detales sutvarko greičiau (per 22 valandas), negu padalinys A (per 33 valandas), todėl labai paprasta detalių gamybą išdėlioti norima tvarka. Svarbu atkreipti dėmesį, kad kodą 3 turinti detalė B padalinyje gaminama 1 valandą, todėl geriausiu atveju reikės visam užsakymui sugaišti mažiausiai 34 valandas. (Padalinys B gali tik tęsti detalės gamybą.)

Nesunku matyti, kad galimi keli atsakymai. Pavyzdžiui, jei detalės gaminamos tokia tvarka:

1-5-4-2-3

arba

5-1-2-4-3

arba

4-1-5-2-3

bus sugaišta 34 valandos (33 valandos + 1 valanda).

Tai informatika!

Šio uždavinio sprendimo optimizavimo idėja paremta godžiuoju algoritmu.

Optimizavimo algoritmas vadinamas godžiuoju, jei kiekviename žingsnyje, kur tik galima rinktis, pasirenkamas tuo momentu (tame žingsnyje) geriausias variantas, neatsižvelgiant į tai, koks bus galutinis rezultatas.

Geras pavyzdys – kuprinės uždavinys. Yra ribojimas, kokį svorį galima sudėti į kuprinę, ir duoti galimų nešulių svoriai. Reikia kuprinę sukrauti taip, kad nešamas svoris būtų kuo didesnis. Pavyzdžiui, turime kuprinę, į kurią galima iš viso galima sudėti 20 kg krovinio ir tris nešulius: 15 kg, 9 kg ir 8 kg. Godusis algoritmas pradėtų kuprinę „krauti“ nuo sunkiausiojo nešulio ir apsiriktų.

Beje, įrodoma, kad kuprinės uždavinys išsprendžiamas tik visiško perrinkimo būdu.

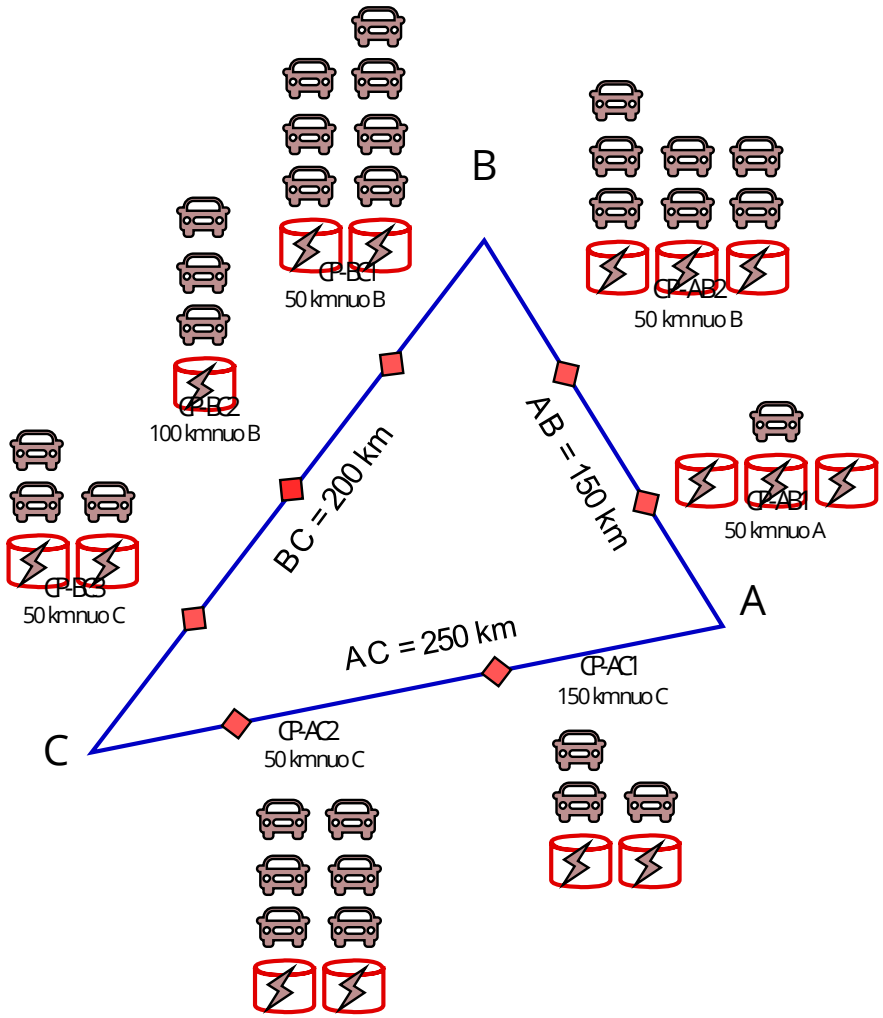
https://bebras.lt/3s/informatikos-mokymas-mokytojams/algoritmas-ir-programavimas/godusis-algoritmas/

19. Elektromobilių eilės

Elektromobilis, kuris važiuoja 100 km per valandą greičiu, visiškai įkrautas gali nuvažiuoti 200 km. Jis gali būti įkraunamas bet kuriuo metu. Tam, kad elektromobilis būtų visiškai įkrautas, reikia 1 valandos, nepaisant to, kiek jis buvo įkrautas prieš pradedant krauti iš naujo.

Paveiksle parodyti keliai, jungiantys A, B ir C. Kelionę pradedate taške A ir norite aplankyti taškus B ir C per kuo trumpesnį laiką. Galite pasirinkti bet kurį maršrutą: į C per B arba į B per C. Kai važiuojate per tašką B arba C, turite sustoti 30 minučių. Šiuo metu elektromobilis neišsikrauna.

Keliuose yra įkrovimo stočių su įkrovimo taškais. Prie įkrovimo taškų gali būti laukiančių eilėje kitų elektromobilių. Prieš pradėdami įkrauti savo elektromobilį turite palaukti, kol jie baigs įkrovimo procedūrą. Pavyzdžiui, įkrovimo stotyje CP-AB1 yra trys įkrovimo taškai, iš kurių du yra laisvi, o prie trečio stovi vienas elektromobilis. Kiekvieną elektromobilį įkrauti užtrunka 1 valandą. Kiekvienoje eilėje esantis pirmas elektromobilis pradėtas krauti tada, kai Jūs išvykote iš taško A. Kelionę pradedate visiškai įkrautu automobiliu.



Kiek mažiausiai reikia laiko, per kurį galima aplankyti taškus B ir C, pradėjus taške A?

A.	5 valandų	B.	6 valandų
C.	6 valandų 30 minučių	D.	7 valandų

Paaiškinimas

Teisingas atsakymas: 6 val.

Šį rezultatą galima pasiekti, pasirinkus maršrutą iš A į B ir į C tokiu būdu. Pradedame taške A ir važiuojame 50 km iki stoties CP-AB1 (30 min.). Įkrauname automobilį CP-AB1 (1 val.). Važiuojame 100 km iki B (1 val.). Laukiame B taške (30 min.) su įkrovimo lygiu, pakankamu nuvažiuoti dar 100 km. Iš B važiuojame 100 km iki CP-BC2 (1 val.). Iki šiol kelionė truko 4 val. ir visi elektromobiliai eilėje prie CP-BC2 baigė įkrovimo procedūras. Įkrauname elektromobilį stotyje CP-BC2 (1 val.). Važiuojame 100 km iš CP-BC2 į C (1 val.). Iš viso praėjo 6 valandos.

Patikrinkime, ar galima B ir C taškus aplankyti greičiau nei per 6 valandas.

Keliaujant iš A į C per B tektų nuvažiuoti 400 km (4 val.) ir praleisti 30 minučių taške B. Be to, reikia bent vieną kartą įkrauti elektromobilį (1 val.). Taigi, laikas negali būti mažesnis nei 5 val. 30 min. Tai reiškia, kad atsakymas „5 valandos“ – neteisingas.

Keliaujant iš taško A į B per C tektų nuvažiuoti 450 km (4 val. 30 min.) ir praleisti 30 min. taške C. Bent du kartus reikia įkrauti elektromobilį (2 val.), taigi mažiausias laikas būtų 7 valandos, net jei nereikėtų laukti eilėje prie įkrovimo stočių.

Tai informatika!

Trumpiausio kelio tarp taškų aibės paieška – dažnas informatikos uždavinys. Ieškant tokių kelių galima efektyviai organizuoti žmonių ar krovinių pervežimą iš vienos vietos į kitą, taip pat kuo greičiau pristatyti duomenų paketus internetu į nurodytą tinklo mazgą.

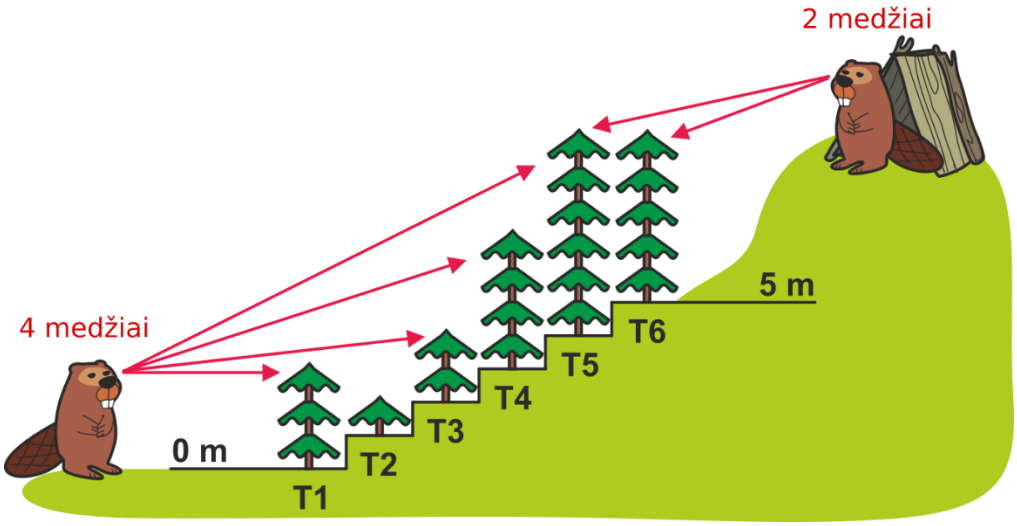
20. Medžiai ant terasų

Kalno šlaite suformuotos šešios terasos T1–T6. Aukščio skirtumas tarp terasų yra 1 metras. Ant kiekvienos terasos auga po vieną medį, kurie šlaite vertikaliai yra išsidėstę viena linija. Visi medžiai yra skirtingo aukščio: 1 m, 2 m, 3 m, 4 m, 5 m ir 6 m.

Absoliutus medžio aukštis yra medžio aukštis + terasos aukštis

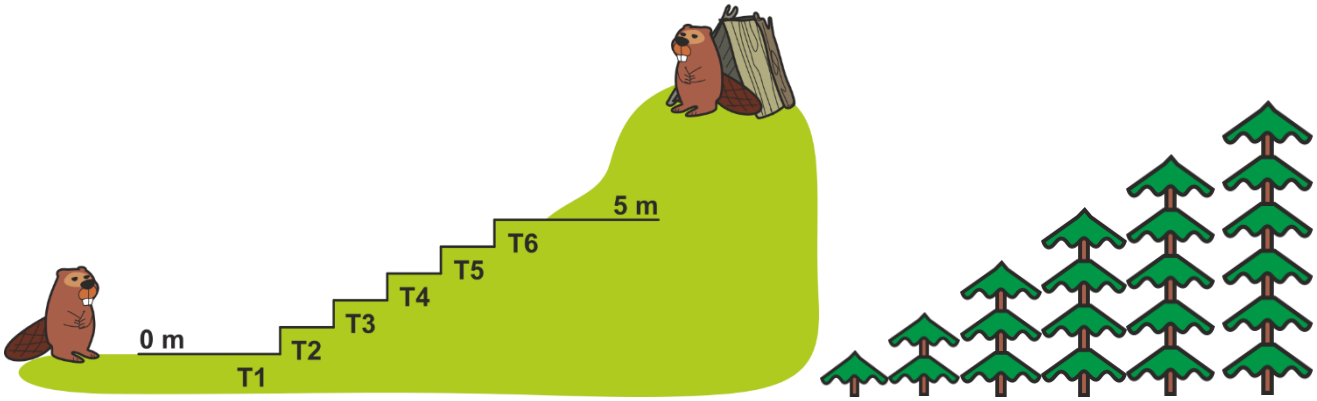
Žiūrint iš kalno papėdės konkretų medį galima pamatyti tik tada, jei prieš jį terasose augančių medžių absoliutus aukštis yra mažesnis negu jo absoliutus aukštis. Žiūrint nuo kalno viršūnės, konkretų medį galima pamatyti tik tada, jei prieš jį terasose augančių medžių aukštis yra mažesnis arba lygus to medžio aukščiui.

Šiame paveiksle kalno papėdėje stovintis bebras mato 4 medžius, o bebras nuo kalno viršūnės mato 2 medžius.



Tarkime, kalno papėdėje augantis medis yra 5 m aukščio. Papėdėje stovintis bebras mato 3 medžius ir bebras nuo kalno viršūnės taip pat mato 3 medžius.

Suraskite vieną iš galimų variantų, kaip šeši skirtingo aukščio medžiai galėtų augti šešiose terasose, kad atitiktų užduotyje pateiktas sąlygas (tempkite medžius į terasas).



Paaiškinimas

Aukščiausias medis turi augti ant ketvirtosios terasos T4, o 4 m aukščio medis – ant trečiosios terasos T3. Yra keletas galimybių, kaip išdėstyti likusius tris žemesnius medžius, tačiau jų išdėstymas nėra visiškai atsitiktinis.

Visi galimi sprendimai yra tokie:

5, 3, 4, 6, 2, 1

5, 2, 4, 6, 3, 1

5, 1, 4, 6, 3, 2

Uždavinį galima išspręsti bandymų metodu lengvai patikrinant, ar siūlomas sprendimas atitinka duotą sąlygą.

Kita vertus, visus galimus sprendimus galima rasti ir kitaip.

Pirmas žingsnis: nustatome, jog aukščiausias medis turi būti ant T4 terasos.

Aukščiausias medis negali būti nei ant T5, nei ant T6 terasos, nes kalno viršūnėje stovintis bebras matytų tik du arba vieną medį (nuo viršūnės nebūtų matomas nė vienas ant T1, T2, T3 ir T4 terasų esantis medis).

4 m aukščio medis negali augti ant T6 terasos, nes tada jo absoliutus aukštis būtų 9 m, taigi, nuo viršūnės būtų daugiausiai matomi du medžiai (tuo atveju, jei aukščiausias medis yra ant T4 terasos, matomi du medžiai, jeigu ne, tai būtų matomas tik 4 m aukščio medis).

Aukščiausias medis negali būti ant T3 terasos, nes jo absoliutus aukštis būtų 8 m, taigi, iš papėdės būtų matomi tik du medžiai (kadangi 5 m aukščio medis yra ant T1 terasos, tai jokio kito medžio ant T2 terasos negalima būtų matyti; taip pat nei vienas likęs medis ant aukštesnių nei T3 terasų negalės pasiekti absoliutaus aukščio, didesnio negu 8 m, nes 4 m aukščio medis negali būti ant T6 terasos).

Aukščiausias medis negali būti ir ant T2 terasos, nes kitaip jo absoliutus aukštis būtų 7 m. Jei norime iš papėdės pamatyti aukščiau nei ant T1 ir T2 terasų augančius medžius, dar vieno medžio absoliutus aukštis turi būti bent 8 m. Kad tai pasiektume, 4 m aukščio medis turėtų būti ant T5 terasos arba 3 m aukščio medis turėtų būti ant T6 terasos. Bet tuomet abiem atvejais nuo viršūnės matytųsi ne daugiau kaip du medžiai.

Išvada: aukščiausias medis turi būti ant T4 terasos.

Antras žingsnis: nustatome, jog 4 m aukščio medis turi būti ant T3 terasos.

Trys medžiai turi būti matomi iš papėdės. Yra matomas 5 m aukščio medis ant T1 terasos ir aukščiausias medis (jo absoliutus aukštis yra 9 m) ant T4 terasos. Neįmanoma pamatyti medžių nei ant T5, nei ant T6 terasų (neįmanoma pasiekti 10 m absoliutaus aukščio). Kad būtų matomas medis, augantis ant T2 arba T3 terasų, turi būti pasiektas bent 6 m absoliutus aukštis, kad jo neužstotų medis, esantis ant T1 terasos. Vienintelis sprendimas yra ant T3 terasos pasodinti 4 m aukščio medį.

Trečias žingsnis: 1 m, 2 m ir 3 m aukščio medžių išdėstymas ant terasų.

Šiuo metu ant kalno viršūnės stovintis bebras mato tik aukščiausiąjį medį, stovintį ant T4 terasos. Jis neturi jokių galimybių pamatyti medžius ant T1, T2 ir T3 terasų. Taigi likę medžiai turi būti išdėstyti taip, kad abudu medžiai, kurie atsidurs ant T5 ir T6 terasų, būtų matomi. Tai reiškia, kad medis, kuris bus ant T6 terasos, turi būti mažesnis už medį, kuris bus ant T5 terasos. Yra lygiai trys galimybės tai padaryti.

Tai informatika!

Ši užduotis yra susijusi su keliais informatikos konceptais.

Visi pirma, reikia atsižvelgti į apribojimus (angl. *constrains*) ir sprendimus konstruoti pagal šiuos apribojimus.

Kitas konceptas – viso objekto ar reiškinio rekonstravimas iš jo dalies.

Abiem atvejais sprendžiantieji turi mokytis tam tikrų sprendimo įgūdžių: konstruoti sprendinius ir juos testuoti, įsitikinti, kad visi galimi sprendimai sukonstruoti, iš dalies pateikto objekto ar reiškinio numatyti visumą, sistemingai ir logiškai atlikti galimus sprendimų variantus.

Kuriame Lietuvos ateitį
2014–2020 metų Europos Sąjungos fondų investicijų veiksmų programa

Vilniaus universitetas

Informatinio mąstymo uždavinių rinkiniai sukurti įgyvendinant projektą „Aukštųjų mokyklų tinklo optimizavimas ir studijų kokybės gerinimas Šiaulių universitetą prijungiant prie Vilniaus universiteto“ (Nr. 09.3.1-ESFA-V-738-03-0001), finansuojamą iš Europos socialinio fondo lėšų pagal 2014–2020 metų Europos Sąjungos fondų investicijų veiksmų programos 9 prioriteto „Visuomenės švietimas ir žmogiškųjų išteklių potencialo didinimas“ įgyvendinimo priemonę Nr. 09.3.1-ESFA-V-738 „Aukštųjų mokyklų tinklo tobulinimas“.

„Bebro“ konkurso uždavinių rinkinys tinka Mokyklos pedagogikos studijų programos moduliui „Informatikos didaktika“. Studentai, būsimi mokytojai, nagrinėdami „Bebro“ uždavinius susipažįsta su įvairiais informatikos konceptais, nagrinėja sudėtingesnes informatikos sąvokas, išmoksta jas paaiškinti.