ISSN 2783-7599

2025

www.bebras.lt

Bebras®

Informatikos ir informatinio mąstymo uždavinių rinkinys

ŠVIETIMO, MOKSLO IR SPORTO MINISTERIJA

VILNIAUS UNIVERSITETAS · 1579 · UNIVERSITAS VILNENSIS

Bebras
International Challenge on Informatics and Computational Thinking

Šiame rinkinyje pateikiama XXII informatikos ir informatinio mąstymo konkurso (iššūkio) „Bebras“ I etapo, vykusio 2025 metų rudenį, uždaviniai, taip pat aiškinimai, koks informatikos turinys ir konceptai atskleidžiami, kaip ir kuo uždavinys įdomus ugdant informatinį mąstymą.

„Bebro“ konkurso pirmasis etapas rengiamas kasmet lapkričio mėnesį įvairaus amžiaus mokinių – nuo 1 iki 12 klasės – grupėms. Konkurso tikslas – skatinti mokinių domėjimąsi informatika, ugdyti gilesnį technologijų supratimą, mokytis spręsti algoritminius, loginius uždavinius, ugdyti kritinį mąstymą, programavimo ir kompiuterinio raštingumo įgūdžius.

Šis uždavinių rinkinys skiriamas 1–12 klasių mokiniams informatinio mąstymo gebėjimams ugdytis.

Dėkojame Daumilui Ardickui, Valentinai Dagienei, Modestai Dagytei, Daivai Gaučytei, Vaidotui Kinčiui, Audronei Klupšaitei, Alvidai Lozdienei, Vaidai Masiulionytei-Dagienei, Modestui Rimkui, Indrai Sudeikienei, Rimantui Žakauskui, Vaivai Žukauskaitei, talkinusiems verčiant ir adaptuojant uždavinius. Nuoširdžiai dėkojame tarptautinei „Bebro“ konkurso bendruomenei ir uždavinių autoriams.

Knygelės spausdinimą parėmė doc. dr. Gintautas Grigas, nuoširdus ačiū.

Sudarė Vaiva Žukauskaitė
Konsultavo Valentina Dagienė
Redagavo Viktoras Dagys
Viršelį kūrė ir maketavo Vaidotas Kinčius

Įvadas

„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.

Lietuvoje 2025 metais „Bebro“ konkurso I etapo uždavinius sprendė 65 074 mokiniai:

2 444 nykštukai (1–2 klasės),
5 019 mažylių (3–4 klasės),
18 841 bičiulis (5–6 klasės),
19 218 draugų (7–8 klasės),
16 094 jauniai (9–10 klasės),
3 458 kolegos (11–12 klasės).

Kiekviena amžiaus grupė sprendė po 18 uždavinių, išskyrus nykštukus (12 uždavinių) ir mažylius (15 uždavinių). Trečdalis uždavinių buvo lengvesnių, trečdalis – vidutinio sunkumo ir trečdalis – sunkesnių. Uždaviniams spręsti buvo skiriamos 45 minutės.

Taškai skaičiuojami taip:
- Prieš pradėdamas spręsti, kiekvienas dalyvis turi 54 taškus (mažylių grupėje – 45 taškus, nykštukų – 36 taškus);
- 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.

Daugiau informacijos apie „Bebro“ konkursą pateikiama interneto svetainėse:
- Tarptautinė „Bebro“ iššūkio svetainė: www.bebras.org
- Lietuvos „Bebro“ svetainė: bebras.lt
- Konkurso varžybų sistema (Varžybų laukas): lt.bebras.lt

Lentelėje pateikiamas šio rinkinio uždavinių skirstymas pagal amžiaus grupes. Skaičius langelyje prie uždavinio nurodo sudėtingumo lygį:

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

Nr.	Uždavinio pavadinimas	Uždavinio identifikatorius	Nykštukai	Mažyliai	Bičiuliai	Draugai	Jauniai	Kolegos
1	Formų gaminimas	2025-IS-02	6					
2	Stebuklų namas	2025-LT-06	6					
3	Aro lobis	2025-MT-03	6					
4	Kalėdos	2025-UK-02	6					
5	Mėgstamiausias karalienės vaisius	2025-AU-04	9	6				
6	Robotų surinkimas	2025-CA-01	9	6				
7	Pagražinti piešiniai	2025-DE-05	9	6				
8	Bitas neša pagalius	2025-JP-05	9	6				
9	Medos užkandis	2025-ID-01	12	9				
10	Žalias skritulys – kas priešais?	2025-IN-05	12					
11	Persų kilimas	2025-IR-02	12					
12	Lėktuvai	2025-LT-03	12	9	6			
13	Lemputės	2025-CA-03		9	6			
14	Pabaisos piešimas	2025-MX-01		9	6			
15	Žuvų rūšiavimas	2025-NL-02		9	6			
16	Kojūkai	2025-ME-02			6			
17	Safario kelionių reklama	2025-NA-01			6			
18	Salų tyrinėjimo ekspedicijos	2025-AT-01		12	9	6		

19	Plaukimo ratai	2025-CA-04		12	9	6		
20	Statiniai iš blokelių	2025-CH-01a		12	9	6		
21	Lėktuvas tarp debesų	2025-IE-04		12	9	6		
22	Automobilių parkavimas	2025-SI-03		12	9	6		
23	Vietnamietiškas pyragas	2025-VN-01		12	9	6		
24	Giminės medis	2025-DE-07			9	6		
25	Katinų kelionės planas	2025-AZ-03			12	9	6	
26	Bebrų mediena	2025-CH-04			12	9	6	
27	Užtvankos statyba	2025-CY-01			12	9	6	
28	Robotas Kairys	2025-DE-09b			12	9	6	
29	Ekskursijų autobusai	2025-KR-06			12	9	6	
30	Vandens čiuožykla	2025-TW-02			12	9	6	
31	Šviesio žemėlapis	2025-DE-06				12	9	6
32	Advento žvakės	2025-DE-08				12	9	6
33	Dingęs aitvaras	2025-GR-02				12	9	6
34	Robotų labirintas	2025-HU-01b				12	9	6
35	Figūrų logika	2025-IE-03				12	9	6
36	Bebrų salos paštas	2025-TW-04				12	9	6
37	Sprendimo riba	2025-AT-02					12	9
38	Akmuo, popierius, žirklės ir mainai	2025-BR-02					12	9
39	Magiškos salos	2025-IT-04					12	9
40	Laimingų draugų stalas	2025-LT-01					12	9

41	Juoda-balta	2025-PT-01					12	9
42	Gėlynas	2025-SK-03					12	9
43	El. pašto adresų tikrinimas	2025-AT-03						12
44	Grafo papildinys	2025-AT-05						12
45	Paslaptinga piramidė	2025-DE-10						12
46	Biberonai	2025-IT-01						12
47	Viešasis transportas	2025-LT-05b						12
48	Apsauga nuo potvynio	2025-NZ-01						12

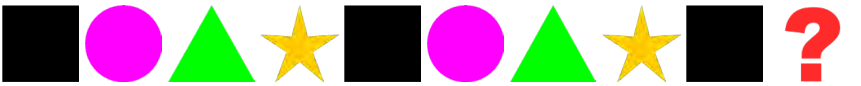
Kiekvieno uždavinio pradžioje nurodoma, kuriai amžiaus grupei jis skiriamas ir jo sudėtingumo lygis. Pabaigoje pateikiamas aiškinimas, kaip uždavinys susijęs su informatika ir kokie informatinio mąstymo gebėjimai ugdomi.

Šio rinkinio uždavinių sprendimų aiškinimo ieškokite

https://bebras.lt/1s/apie-bebra/leidiniai/

1. Formų gaminimas 2025-IS-02

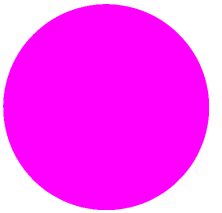
Robotas gamina keturias formas tam tikra seka ir šią seką kartoja daug kartų. Staiga kažkas nutiko ir robotas sustojo.



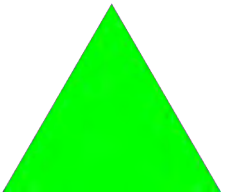
Kokia forma turi būti pagaminta vietoj klaustuko?

A

B



C



D

Teisingas atsakymas: B.

Paaiškinimas

Robotas nuolatos kartoja formas ta pačia tvarka:
kvadratas -> skritulys -> trikampis -> žvaigždė.
Tada vėl pradeda iš pradžių – skritulys ir taip toliau. Kadangi duotoje sekoje paskutinė pagaminta forma yra kvadratas, tai toliau turėtų būti gaminamas skritulys.
Kvadrato pasirinkimas būtų neteisingas, nes jis visada eina po žvaigždės.
Trikampio pasirinkimas būtų neteisingas, nes jis visada eina po skritulio.
Žvaigždės pasirinkimas būtų neteisingas, nes ji visada eina po trikampio.

Tai informatika!

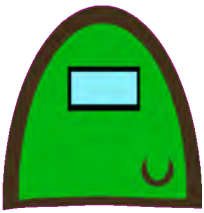
Norint išspręsti šį uždavinį, reikia atpažinti modelį, pagal kurį gaminama figūrų seka. Tai vadinama dėsningumų (šablonų ar modelio) *atpažinimu*. Tai svarbus informatikos uždavinys, kuris būdingas daugeliui taikymų: prognozavime (rinkos, oro ir kt.), vaizdų analizėje, kompiuteriniame vizualizavime, kalbos apdorojime ir pan.
Kai seką gamina robotas, valdomas kompiuterio programa, naudojama kartojimo komanda. Kartojimo komanda programavime dar vadinama ciklu: naudojant ciklą galima kartoti komandų rinkinį. Šiuo atveju ciklas naudojamas kartoti komandų seką, kuria pagaminamos keturios figūros ta pačia tvarka:
( ).

2. Stebuklų namas 2025-LT-06

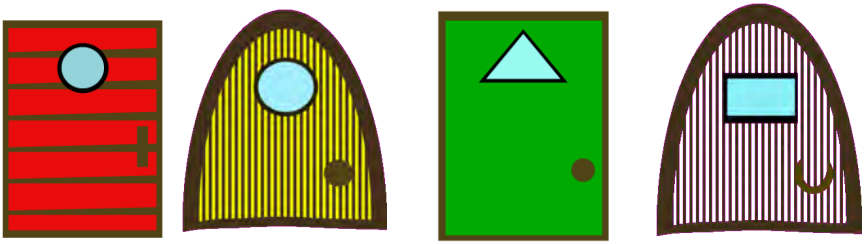
Bebrė Justė įstrigo stebuklų name, iš kurio nori ištrūkti. Ji gali eiti tik pro tas duris, kurios visiškai kitokios nei tos, pro kurias ką tik praėjo. Dvejos durys visiškai kitokios, jei skiriasi:

- durų forma,
- spalva arba raštu,
- lango forma
- rankenos forma.

Praeitos durys:



Kurios durys padės Justei ištrūkti iš stebuklų namo?



Teisingas atsakymas: A.

Paaiškinimas
Pirmųjų geltonų durų forma panaši į ankstesnes duris. Antrųjų žalių durų spalva (raštas) sutampa. Ketvirtosios mėlynos durys panašios savo forma, langu ir rankena. Tačiau trečiosios raudonos durys visiškai skiriasi nuo ankstesniųjų.

Tai informatika!

Šiame uždavinyje kiekvienos durys turi skirtingas savybes – požymius: durų formą, spalvą ar raštą, lango formą ir rankenos formą. Šie požymiai yra informacija, arba kitaip – duomenys, kuriuos galime naudoti apibūdindami kiekvienas duris. Justė gali praeiti pro duris tik tuo atveju, jei visi durų požymiai skiriasi. Tai reiškia, kad naujos durys turi būti visiškai kitokios formos, spalvos, lango formos ir rankenos formos.
Tai panašu į tai, kaip kompiuteris vertina informaciją ir priima sprendimus. Pavyzdžiui, jei ieškote marškinėlių internete, galbūt norite tokių, kurie būtų kitokios spalvos ir stiliaus nei tie, kuriuos jau turite. Kompiuteriai naudoja taisykles, kad filtruotų ir lygintų informaciją, kaip Justė daro su durimis.
Norėdami išspręsti šią problemą, kompiuteriai:
- naudoja sąrašus, kad galėtų nagrinėti visus durų požymius.
- taiko taisykles, kad patikrintų, ar visi požymiai yra skirtingi.

Realioje aplinkoje toks duomenimis pagrįstas sprendimų priėmimas naudojamas dėsningumų atpažinimo, mašininio mokymosi ir kitose sistemose, kur algoritmai turi analizuoti kelis požymius, kad rastų tinkamus atitikmenis ar anomalijas.

3. Aro lobis 2025-MT-03

Mokslininkas Aras rado seną lobio žemėlapį. Žemėlapio tinklelyje pavaizduoti įvairūs simboliai:  (pradinis taškas),  (žolė),  (kalnas),  (vanduo) ir  (lobis). Aras turi iš pradžios taško pasiekti lobį laikydamasis trijų taisyklių:

1. Negali žengti į vandenį.
2. Turi pereiti bent vieną kalną.
3. Gali judėti tik į viršų (↑), žemyn (↓), dešinėn (→), arba kairėn (←), bet ne įstrižai (/ , \ ).

Kuriuo keliu eidamas Aras, laikydamasis visų taisyklių, pasieks lobį?

A

B

C

D

Teisingas atsakymas: B.

Paaiškinimas

Aras turi pasiekti lobį laikydamasis trijų taisyklių. Reikia patikrinti kiekvieną kelią ir įvertinti, ar jis atitinka visas taisykles.

Aras negali žengti į vandenį () (1 taisyklė), todėl variantas A neteisingas.	Aras negali judėti įstrižai (3 taisyklė), todėl variantas C neteisingas.	Aras turi pereiti bent vieną kalną () (2 taisyklė). D variante Aras nepereina jokio kalno (jie paryškinti), todėl šis variantas nėra teisingas.
		

Atmetus netinkamus variantus, teisingas atsakymas yra B. Patikrinę įsitikiname kad laikomasi visų trijų taisyklių.

Tai informatika!

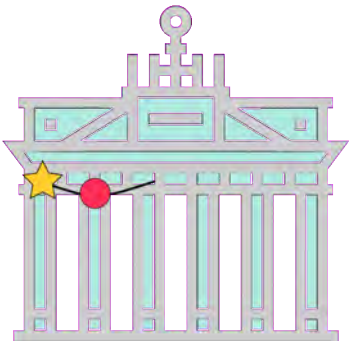
Informatikoje daugelis uždavinių susiję su užduoties atlikimu laikantis tam tikrų taisyklių, sąlygų, apribojimų ar suvaržymų. Šiame uždavinyje turime rasti kelią nuo pradinio taško iki lobio, kad jis atitiktų visas pateiktas taisykles. Yra daug galimų kelių, kurie nėra įtraukti į variantus. Tačiau prašoma įvertinti tik keturis variantus, ar kuris nors iš jų tenkina apribojimus. Tai vadinamoji apribojimų (arba sąlygų) tenkinimo problema.

Problemų sprendimas laikantis tam tikrų sąlygų yra dažnai naudojamas verslo, finansų (pinigų), valdymo (vadovavimo bendruomenėms) ir kitose srityse. Pavyzdžiui, gydytojai turi rasti būdą, kaip paskirstyti darbą kelioms slaugytojoms, kurios turi prižiūrėti daug pacientų (tikslas), bet kartu palikti slaugytojoms laiko poilsiui (apribojimas). Kompiuterinės programos gali būti naudojamos galimiems atsakymams rasti, tačiau daugelį tokių problemų yra tikrai sunku išspręsti, ypač kai norima rasti geriausią sprendimą.

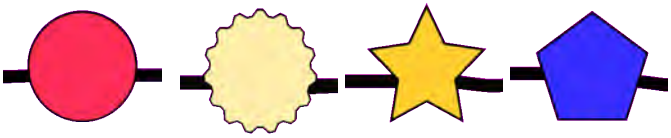
Šis uždavinys taip pat susijęs su kita informatikos problema – kelio paieška, t. y., kelio, kuris atitinka visas taisykles, paieška. Yra daug sprendimų, tad tenka įvertinti, kuris kelias yra tinkamiausias. Pavyzdžiui, picų pristatytojui svarbu rasti kelią pas visus, kurie užsisakė picą, bet tai reikia padaryti taip, kad niekas nelauktų per ilgai ir kiek įmanoma būtų sutrumpintas pristatytojo darbo laikas.

4. Kalėdos 2025-UK-02

Prieš Kalėdas kiekvieną savaitę Bebrų miestelio dekoracijos papildomos naujos formos lempute.

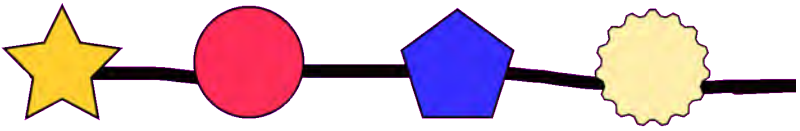


Lemputės laikomos skirtingose dėžėse ir žymimos šitaip:



A dėžė B dėžė C dėžė D dėžė

Po keturių savaičių dekoracijos atrodė taip.

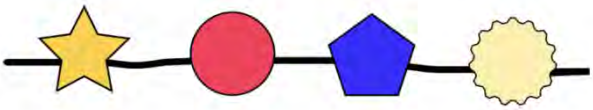


Kokia tvarka buvo atidarytos dėžės?

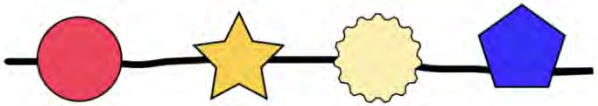
A. ACBD
B. CBDA
C. CADB
D. DACB

Teisingas atsakymas: C.

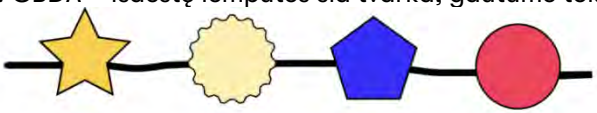
Paaiškinimas
Pirmoji lemputė, kuri turi būti pridėta, yra žvaigždės formos. Ji laikoma C dėžėje.
Antroji lemputė yra skritulio formos. Ji laikoma A dėžėje.
Trečioji lemputė yra penkiakampė. Jis laikoma D dėžėje.
Paskutinioji lemputė yra krumpliaračio formos. Jis laikoma B dėžėje.



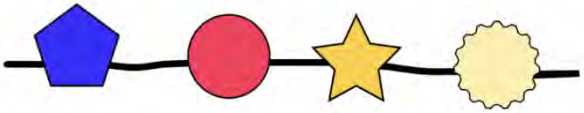
Pasirinkus kitus variantus lemputės būtų išdėstytos neteisinga seka.
1. ACBD – išdėstę lemputes šia tvarka, gautume tokią seką:



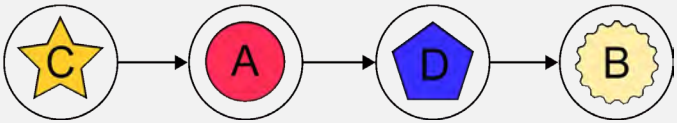
2. CBDA – išdėstę lemputes šia tvarka, gautume tokią seką:



4. DACB – išdėstę lemputes šia tvarka, gautume tokią seką:



Tai informatika!
Informatikoje seka reiškia tvarkingą komandų ar užduočių išdėstymą programoje ar algoritme. Ji apibrėžia konkrečią komandų vykdymo tvarką, kad būtų pasiektas norimas rezultatas. Tai yra pagrindinė programavimo ir algoritmų kūrimo sąvoka.
Šioje **sekoje** parodyta, kad lemputės turi būti surinktos teisinga tvarka iš dėžių, kuriose jos laikomos.



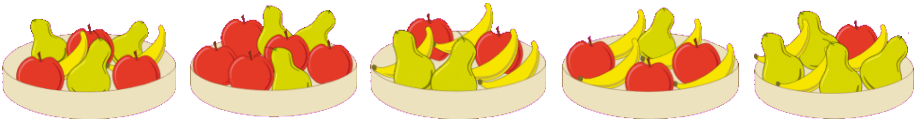
5. Mėgstamiausias karalienės vaisius 2025-AU-04

Penki kaimiečiai atnešė karalienei po krepšelį vaisių: obuolių, bananų ir kriaušių.

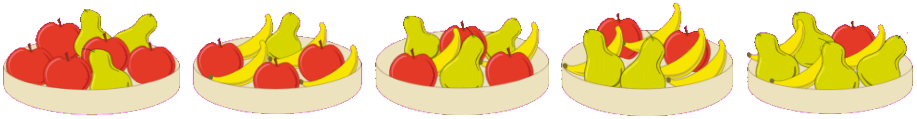
Kiekviename krepšelyje yra po 8 vaisius. Karalienė labiausiai mėgsta obuolius. Ji priims kaimiečius tokia tvarka:

- pirmiausia – tą, kurio krepšelyje daugiausia obuolių,
- paskiausiai – tą, kurio krepšelyje obuolių mažiausiai.
- jei keliuose krepšeliuose yra vienodas obuolių skaičius, tuomet pirmenybė teikiama tam, kuris turi daugiau bananų.

Kokia eile karalienė priims kaimiečius?

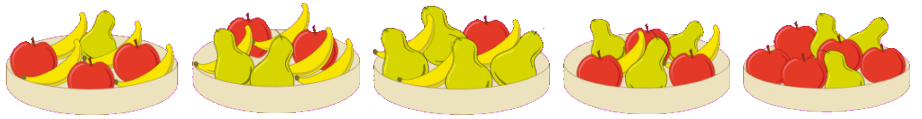


A:



B:

C:



D:

Teisingas atsakymas: A

Paaiškinimas
Pirmiausia pažymime krepšelius raidėmis nuo A iki E – iš kairės į dešinę.
Suskaičiuojame, kiek obuolių yra kiekviename krepšelyje:

- A – 3 obuoliai,
- B – 5 obuoliai,
- C – 2 obuoliai,
- D – 3 obuoliai,
- E – 1 obuolys.

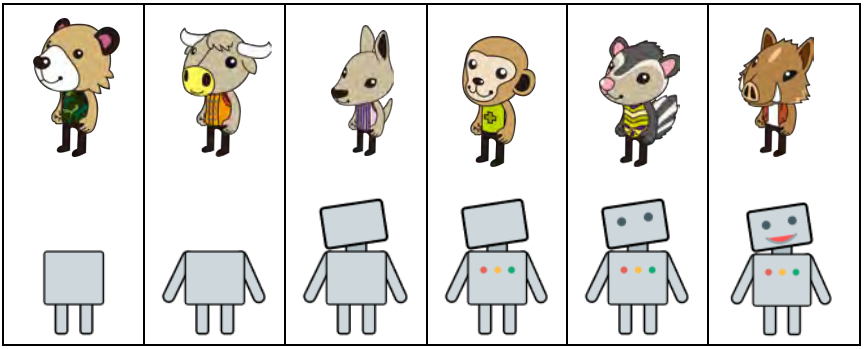
Karalienė priima kaimiečius pagal obuolių skaičių (nuo daugiausiai iki mažiausiai), todėl garbė būti pirmam priimtam tenka kaimiečiui, atnešusiam B krepšelį.
A ir D krepšeliuose yra po 3 obuolius, todėl reikia žiūrėti, kuris iš šių kaimiečių turi daugiau bananų. A krepšelyje yra 2 bananai, o D – 4 bananai, todėl antras bus kaimietis, atnešęs D krepšelį, po jo – A.
Lieka C ir E krepšeliai. Karalienė priims kaimietį, turintį C krepšelį ir paskiausiai – E.
Taigi karalienė priims krepšelius tokia tvarka: B, D, A, C, E.

Tai informatika!

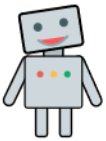
Rikiavimas – tai dažnas veiksmas programose, kai reikia išdėstyti elementus tam tikra tvarka.
Šiuo uždaviniu vaizduojamas ne pavienių skaičių rikiavimas, o rinkinių – šiuo atveju krepšelių. Kai rikiuojame skaičius, jų reikšmės yra aiškios. Tačiau kai rikiuojame rinkinius, turime nuspręsti, pagal ką jie bus vertinami. Šioje situacijoje kiekvieno krepšelio „vertė“ nustatoma pagal obuolių skaičių, o jei jis vienodas – pagal bananų kiekį.
Programavime tai vadinama **rikiavimu pagal raktą**. Raktas nurodo, pagal kokį požymį elementai lyginami tarpusavyje. Šiuo atveju pagrindinis raktas yra obuolių skaičius.
Rikiavimą pagal kelis raktus galima sutikti ir kasdieniame gyvenime. Pavyzdžiui, bibliotekose knygos pirmiausia skirstomos pagal žanrus – kiekviename skyriuje yra tam tikro žanro leidiniai. Tuomet knygos toje pačioje lentynoje surikiuojamos pagal autorių pavardžių abėcėlę. Taigi bibliotekininkas pirmiausia suranda skyrių, o tada – tinkamą vietą lentynoje, kur padėti knygą.

6. Robotų surinkimas 2025-CA-01

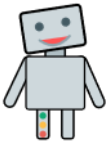
Šeši gyvūnai dirba žaislinių robotų surinkimo linijoje. Kiekvienas jų atlieka skirtingą užduotį tam tikra nustatyta tvarka:



Surinkimo linijos pabaigoje žaisliniai robotai turi atrodyti taip:



Tačiau iš tikrųjų robotai atrodo šitaip:



Kuris gyvūnas netinkamai atlieka savo darbą?

A

B

C

D

Teisingas atsakymas: C.

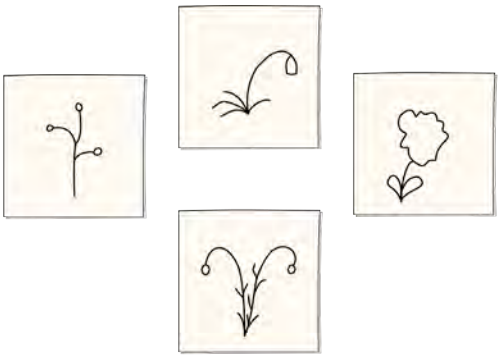
Paaiškinimas
Žaislinis robotas turėtų turėti tris mygtukus ant korpuso, tačiau vietoj to mygtukai yra išdėstyti ant kojos. Tai reiškia, kad mygtukai buvo pritvirtinti neteisingai – tai beždžionėlės darbas.

Tai informatika!
Šiame uždavinyje surinkimo linija nesukūrė žaislų taip, kaip tikėtasi. Tai panašu į situaciją, kai kompiuterinė programa pateikia netikėtą rezultatą.
Kai taip nutinka, programuotojai turi peržiūrėti visus programos vykdomus žingsnius, kad nustatytų, kur įvyko klaida. Šis procesas vadinamas derinimu (angl. *debugging*).

7. Pagražinti piešiniai 2025-DE-05

Joris nupiešė keletą augalų. Jo mažoji sesutė Meda rado piešinius ir bandė juos pagražinti pirštų dažais. Ar vis dar atpažįstate originalius piešinius?

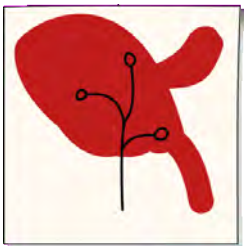
Nutempkite kiekvieną originalų piešinį po jo pagražinta versija.



Teisingas atsakymas

A)

B)



C)

D)

Paaiškinimas

Norėdami rasti teisingą sprendimą, palyginkite (1) „pagražinto“ paveikslėlio juodas linijas su originalaus paveikslėlio linijomis ir (2) „pagražinto“ paveikslėlio baltas sritis su originalaus paveikslėlio baltomis sritimis.

Vaizdas D turi baltą plotą viduryje; yra tik vienas originalus vaizdas, kuriame taip pat yra baltas plotas viduryje. Yra tik vienas originalus piešinys su baltu plotu viršutiniame kairiajame kampe, kaip parodyta A variante. B variante matyti stiebas vaizdo apatinėje pusėje; yra tik vienas originalus piešinys su tokiu stiebu. Tai reiškia, kad variantui C lieka vienas originalus piešinys.

Tai informatika!

Mažoji Meda norėjo pagerinti piešinius, bet iš tikrųjų ji netyčia sugadino savo brolio darbą. Muziejų meno restauratoriai stengiasi sugrąžinti pažeistus arba perdažytus meno kūrinius į jų pradinę būklę. Jų darbas dažnai apima atitikmenų paiešką iš kitų šaltinių, kad būtų galima tiksliai atkurti originalų kūrinį. Šis procesas yra panašus į uždavinyje aprašytus veiksmus.

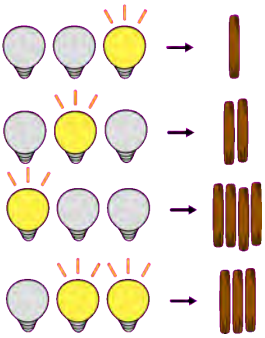
Informatikoje restauravimo technikos naudojamos visiškai naujų vaizdų kūrimui. Šios pažangios sistemos vadinamos difuzijos modeliais. Difuzijos modelis apmokomas naudojant daugybę vaizdų, kad galėtų atkurti sugadintus skaitmeninius paveikslus. Naudodamas šias žinias, jis gali sukurti gražius naujus vaizdus iš paveikslų, sudarytų iš daugybės mažų, išsibarsčiusių taškelių.

Tačiau yra esminis skirtumas nuo pateikto uždavinio. Spręsdami šį uždavinį, naudojotės loginiu mąstymu, tačiau difuzijos modelis nėra skirtas logikai mokyti. Difuzijos modelis sustiprina atkūrimo įgūdžius analizuojant daugybę pavyzdžių – tai naudojama mašininiame mokymesi.

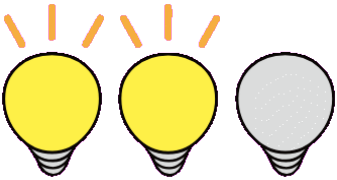
8. Bitas neša pagalius 2025-JP-05

Bebriukas Bitas neša pagalius. Trijų lempučių seka rodo, kiek pagalių Bitas neš.

Kai įjungta dešiniausia lemputė, Bitas neš vieną pagalį, kai įjungta vidurinė lemputė, neš du pagalius, o kai įjungta lemputė kairėje, neš keturis pagalius. Kai įjungta daugiau nei viena lemputė, pagalių skaičius, kurį rodo kiekviena lemputė, sudedamas.



Kai lemputės dega taip, kaip parodyta, kiek pagalių neš Bitas?



A. 2
B. 5
C. 6
D. 7

Teisingas atsakymas: C.

Paaiškinimas
Kai dešinėje dega tik viena lemputė, ji reiškia vieną pagalį, o kai dešinėje ir viduryje dega abi lemputės, tai jos reiškia tris pagalius.
Kai kairėje dega viena lemputė, tai ji reiškia keturis pagalius, todėl, kai dešinėje ir viduryje dega abi lemputės, tai jų rodomus pagalius reikia sudėti: keturis plius du – iš viso bus šeši pagaliai.

Tai informatika!

Dvejetainiai skaičiai vartoja tik du žymenis, 1 ir 0, jais pavaizduoja visus skaičius. Kompiuteriuose skaičiavimas grįstas dvejetainiais skaičiais, arba dvejetainiu kodu.
Šiame uždavinyje lempučių įjungimas bei išjungimas atitinka 1 arba 0 dvejetainį kodą. Klausimas, kiek pagalių reikia nešti, kad įsijungtų kairioji ir vidurinė lemputės, atitinka paklausimą, kam lygus dvejetainis skaičius 110 dešimtainėje sistemoje.

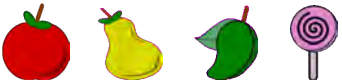
9. Medos užkandis 2025-ID-01

Meda kiekvieną popietę gauna užkandį. Mama visada paruošia keturias užkandžių rūšis: obuolius (), kriaušes (), mangus (), ir saldainį ().
Meda kasdien gali pasirinkti tik vieną užkandžio rūšį ir turi laikytis dviejų taisyklių:

1) Ji negali valgyti mangų dvi dienas iš eilės.
2) Jei šiandien ji valgo saldainį, tai rytoj ji turi valgyti arba obuolius, arba kriaušes.

Užbaik dėlioti užkandžius taip, kad jie atitiktų Medos mamos taisykles.

1 diena	2 diena	3 diena	4 diena	5 diena
				



Paaiškinimas
Galimi šie atsakymai:

2 dieną: obuolys  , kriaušė  arba saldainis  .

3 dieną: obuolys  arba kriaušė  .
Iš viso – 6 galimi variantai.
Dėl pirmosios taisyklės (Meda negali valgyti mangų dvi dienas iš eilės) užkandis negali būti mangas.
Antroji taisyklė sako: „Jei šiandien Meda valgo saldainius, tai rytoj ji turi valgyti obuolius arba kriaušes.“ Jei antrą dieną ji valgys saldainį, tai kitą dieną ji turi rinktis obuolius arba kriaušes. Antrą dieną ji taip pat gali valgyti obuolius arba kriaušes, tokiu atveju obuolius arba kriaušes ji valgys ir sekančią dieną.
Trečią dieną ji negali valgyti saldainių, nes tokiu atveju 4 dieną ji turėtų valgyti kriaušes arba obuolius, bet ji valgys mangus.

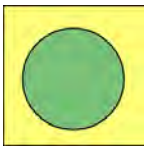
Tai informatika!

Šis uždavinys susijęs su algoritmais ir Būlio logika. Algoritmas – tai žingsnis po žingsnio atliekamos instrukcijos, padedančios mums išspręsti problemą. Šiuo atveju problema yra nuspręsti, kokius užkandžius Meda gali valgyti, laikydamasi duotų taisyklių.
Naudodami „jei“ teiginius, galime apibūdinti užkandžių pasirinkimo taisykles taip:
- Jei vakar valgei mangą, šiandien pasirink užkandį, kuris nėra mangas.
- Jei vakar valgei saldainį, šiandien pasirink obuolį arba kriaušę.

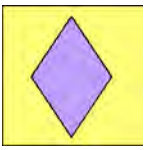
Taip mąstome ir kasdieniame gyvenime. Pavyzdžiui, „jei lyja, pasiimk skėtį“, „jei esi pavargęs, eik miegoti“.
„Jei“ teiginiuose naudojamos sąlygos, kad galima būtų nuspręsti, ką daryti. Jei sąlyga yra teisinga, tuomet vykdoma „tada“ dalis. Jei sąlyga neteisinga, „tada“ dalis nevykdoma. Galite kurti sudėtingas sąlygas, derindami daug teisingų ar neteisingų teiginių. Tai yra vadinamoji Būlio logika, vienas iš pagrindinių informatikos konceptų. Būlio logika grindžiama tik dviem reikšmėmis: teisinga ir neteisinga.
Algoritmas ir jo taisyklės (instrukcijos) turi būti tikslios, kad būtų aišku, ką reikia daryti. Pavyzdžiui, taisyklė „Kartais gali valgyti saldainius dvi dienas iš eilės“ negali būti užrašyta kaip instrukcija, nes neįmanoma žinoti, kada galima valgyti saldainius dvi dienas iš eilės, o kada ne, tai yra dviprasmiška.

10. Žalias skritulys – kas priešais? 2025-IN-05

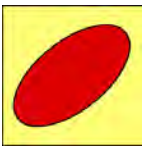
Kiekvienoje bebro kubelio sienelėje pavaizduota skirtinga figūra. Iš viso yra šešios figūros:



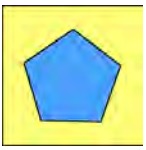
Žalias skritulys



Violetinis rombas

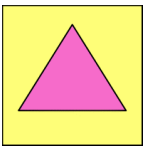


Raudonas ovalas



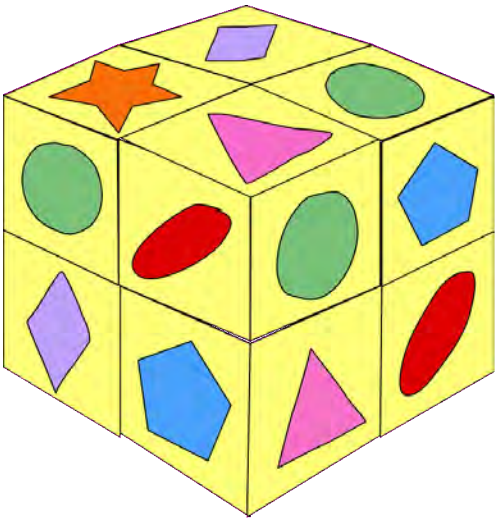
Mėlynas penkiakampis

Oranžinė žvaigždė



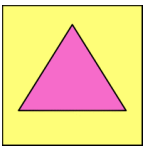
Rausvas trikampis

Aštuoni visiškai vienodi kubeliai sudėti vienas ant kito ir sudaro didelį kubą:

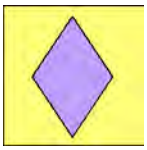


Kokia figūra yra Bebro kubelio sienelėje, priešingoje žaliam skrituliui?

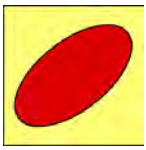
A) Oranžinė žvaigždė



B) Rausvas trikampis



C) Violetinis rombas



D) Raudonas ovalas

Teisingas atsakymas: C.

Paaiškinimas
Iš aštuonių kubelių sudėtame kube šalia žalio skritulio matome šias figūras: oranžinę žvaigždę (viršuje kairėje), raudoną ovalą ir rožinį trikampį (viršuje centre) bei mėlyną rombą (viršuje dešinėje). Nė viena iš šių figūrų negali būti priešais žalią skritulį. Vienintelė figūra, kurios nėra šalia žalio skritulio, yra violetinis rombas.

Tai informatika!

Šį uždavinį galime išspręsti tiesiog logiškai mąstydami.
Galimi penki pasirinkimai – penkios figūros, iš jų keturias reikia atmesti.
Kiekvieną netinkamą pasirinkimą reikia aiškiai argumentuoti, kodėl jis netinkamas. Norėdami suprasti argumentus, turime pasitelkti erdvinį mąstymą, kad atpažintume, kurios iš aštuonių sukrautų kubelių sienelės yra gretimos viena kitai.
Alternatyvų atmetimas siekiant rasti teisingą atsakymą yra vienas pagrindinių kompiuterinio raštingumo įgūdžių. Juo remiamės, pavyzdžiui, taisydami klaidas programose ir nustatydami klaidų priežastis.
Erdvinis mąstymas itin svarbus kompiuterinėje grafikoje, pavyzdžiui, norint atvaizduoti trimačius objektus.

11. Persų kilimas 2025-IR-02

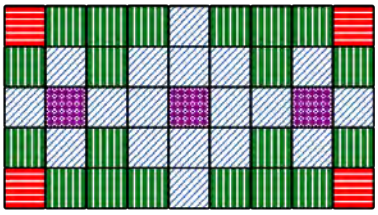
Saulės turimas mezgimo robotas mezga keturių skirtingų raštų kilimus. Kiekvienas raštas robotui nusakomas naudojant dviejų skaitmenų kodą:

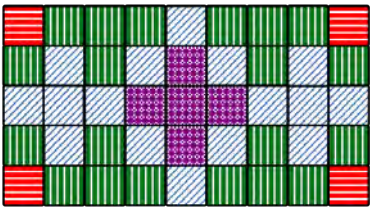
00	
11	
01	
10	

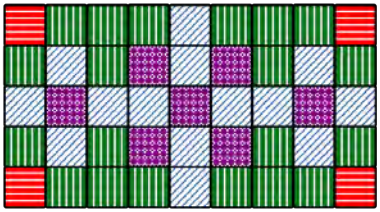
Saulė parašė robotui mezgimo programą. Kiekvienas kodas nurodo, kokį raštą megzti kiekviename kilimo kvadratėlyje. Robotas mezga eilėmis iš kairės į dešinę ir iš viršaus į apačią.

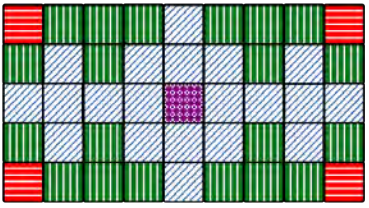
00-10-10-10-01-10-10-10-00
10-01-10-01-01-01-10-01-10
01-11-01-01-11-01-01-11-01
10-01-10-01-01-01-10-01-10
00-10-10-10-01-10-10-10-00

Kurį iš šių kilimų pagamins robotas, naudodamas Saulės parašytą programą?

A) 

C) 

B) 

D) 

Teisingas atsakymas: A.

Paaiškinimas

Norėdami išspręsti šį uždavinį, turėtume iššifruoti visą programą eilutė po eilutės ir tada palyginti ją su pateiktais variantais, kad rastume teisingą, tačiau tai nelengva ir užtruktų daug laiko.

Darykime taip:

1. Skaitykime programą eilutė po eilutės, iš kairės į dešinę, iš viršaus į apačią.
2. Suskirstykime programą į eilutes ir iššifruokime kiekvieną dviejų skaitmenų kodą naudodami kodų lentelę.
3. Sukurkime visą kilimo dizainą.
4. Palyginkime dizainą su keturiais vizualiais variantais ir raskime kvadratą, kuriame dizainai skiriasi.

Efektyvesnis sprendimas yra naudoti variantų skirtumus, kad susiaurintume paieškos sritį ir greičiau rastume atsakymą. Atlikime šiuos veiksmus ir rasime atsakymą:

1. Nustatykime pozicijas, kuriose variantai skiriasi.
2. Raskime programoje besiskiriančio langelio kodą ir suraskime teisingą modelį kodų lentelėje.
3. Atmeskime variantus, kuriuose to langelio modelis yra neteisingas.

Kartokime 1–3 veiksmus, kol liks tik vienas variantas, kuris ir yra teisingas atsakymas.

Pavyzdžiui, A ir B variantuose skiriasi antrosios ir ketvirtosios eilučių ketvirti ir šešti langeliai. Antrosios ir ketvirtosios eilučių ketvirto ir šešto langelių kodas yra 01 (modelis C), todėl B variantą pašaliname.

C ir D variantuose skiriasi trečiosios eilutės ketvirti ir šešti langeliai. Trečiosios eilutės 4-ojo ir 6-ojo langelių kodas yra 01 (modelis C), todėl atmetame C variantą. A ir D variantuose skiriasi trečiosios eilutės antri ir aštunti langeliai. Trečiosios eilutės 2-ojo ir 8-ojo langelių kodas yra 11 (modelis B), todėl atmetame D variantą.

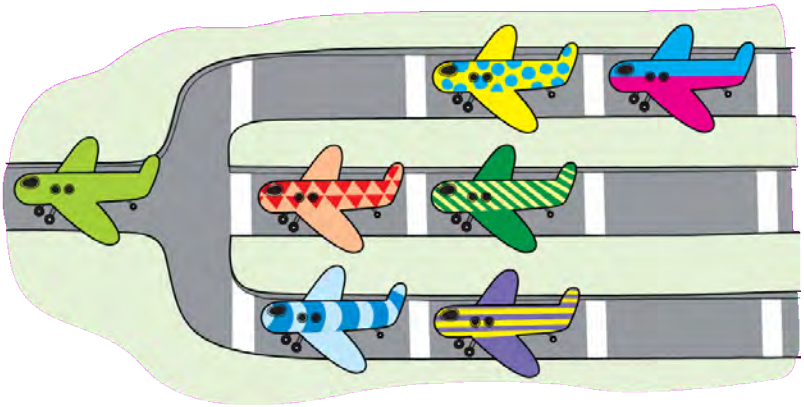
Tai informatika!

Kompiuteriai spalvas atkuria ne pagal spalvos pavadinimą ar žodį, o pagal dvejetainį kodą, sudarytą iš nulių ir vienetų. Šiame uždavinyje naudojame dvejetainius kodus (tik du skaitmenis, 0 arba 1) skirtingiems modeliams žymėti. Tada šių kodų serija (kaip mezgimo programa) perduodama robotui, kad jis žinotų, kaip gaminti kilimus.

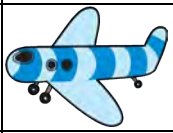
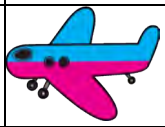
Mūsų kasdieniame gyvenime viskas, kas yra kompiuteriuose, nuo dokumentų ir nuotraukų iki muzikos, vaizdo įrašų ir net skaičių, paverčiama dvejetainiais kodais (0 ir 1), kad būtų galima pateikti informaciją, ją išsaugoti ir persiųsti.

12. Lėktuvai 2025-LT-03

Septyni lėktuvai stovi eilėje, kad galėtų pakilti iš vienintelio kilimo tako, ir jie negali aplenkti eilės!

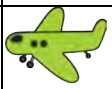
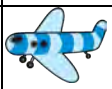
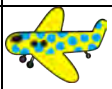
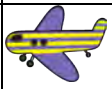


Pateikiamas lėktuvų išvykimo tvarkaraštis (nurodyti būsimi laikai).
Užpildykite tuščius tvarkaraščio langelius, perkeldami atitinkamus lėktuvus.

Laikas	*Lėktuvas*
10:45	
10:52	
10:55	
10:59	
11:00	
11:10	
11:11	

Teisingas atsakymas

Išvykimo tvarkaraštis

Laikas	*Lėktuvas*
10:45	
10:52	
10:55	
10:59	
11:00	
11:10	
11:11	

Paaiškinimas

Lėktuvas gali pakilti tik tuo atveju, jei tuo metu prieš jį nėra kito lėktuvo. Pirmasis lėktuvas  jau yra kilimo take. Antrasis lėktuvas nurodytas lentelėje. Trečiąjį laiko tarpsnį (10:55) gali dalytis 3 lėktuvai:  ,  ir  . Šiuo atveju reikia patikrinti, kuris lėktuvas turi išskristi po jo: matome, kad tai yra lėktuvas  , o tai reiškia, kad jam turi būti atlaisvintas kelias. Taigi, trečiajam skrydžiui turi pakilti lėktuvas  , kuris yra prieš ketvirtąjį skrendančią lėktuvą.
Ta pati logika taikoma penktajai vietai (11:00): yra 2 lėktuvai ( ir ), kurie galėtų pakilti, bet tik vienas iš jų yra tinkamas, nes jis atlaisvina kelią šeštam skrendančiam lėktuvui — tai lėktuvas  . O paskutinis lėktuvas yra tas, kuris liko:  .

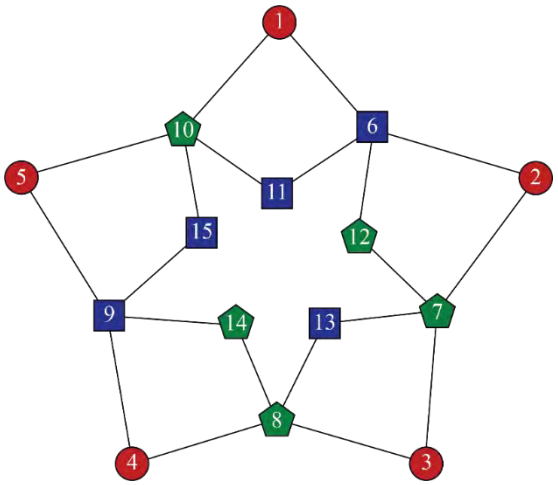
Tai informatika!

Šis uždavinys supažindina su svarbiomis informatikos sąvokomis — duomenimis, sekomis ir laiko planavimu. Lėktuvų pakilimo laikai yra tarytum struktūrizuoti skaitmeniniai duomenys, kuriuos reikia analizuoti ir palyginti. Sekos yra labai svarbios, nes lėktuvai turi būti išdėstyti tam tikra tvarka pagal jų pakilimo laiką, kad būtų užtikrintas teisingas jų išvykimas. Laiko planavimas taip pat svarbus, nes uždavinys sprendžia realią planavimo problemą, kai įvykiai – šiuo atveju lėktuvų pakilimai – turi būti išdėstyti ir valdomi pagal nustatytas laiko taisykles. Šių informatikos sąvokų supratimas padeda sistemingai organizuoti, apdoroti ir tikrinti struktūrizuotą informaciją.

13. Lemputės 2025-CA-03

Eglė turi penkiolika programuojamų lempučių, kurios skirstomos į tris tipus:

Lemputes Eglė sunumeruoja nuo 1 iki 15 ir sujungia žvaigždės forma, kaip parodyta paveikslėlyje.



Eglė suprogramuoja lemputes taip:

- Kiekviena  lemputė valdoma jungikliu.
- Kiekviena  lemputė įsijungia, jei **abi** mažesniais numeriais pažymėtos prie jos prijungtos lemputės yra **įjungtos**.
- Kiekviena  lemputė įsijungia, jei **tik viena** iš dviejų mažesnio numerio prijungtų lempučių yra **įjungta**. Kitaip tariant – viena iš jų turi būti įjungta, o kita – išjungta.

Visos lemputės iš pradžių išjungtos. Eglė įjungia 1-ąją, 2-ąją ir 4-ąją lemputes. Pagal jos programą tai sukels kitų lempučių įsijungimą.
Kai lemputės nustos keistis, kurios iš lempučių nuo 11 iki 15 yra įjungtos?

A. Įjungtos 11, 12, 13, 14 ir 15 lemputės
B. Įjungtos tik 12, 13 ir 15 lemputės
C. Įjungtos tik 11, 12, 13 ir 14 lemputės
D. Įjungtos tik 11, 13 ir 14 lemputės

Teisingas atsakymas: D.
Paaiškinimas

Iš  tipo lempučių įjungtos tik 1, 2 ir 4. Dabar žiūrime link žvaigždės centro ir nustatome, kurios iš likusių lempučių įsijungia.

Kadangi 6-oji lemputė yra  tipo, 1-oji ir 2-oji lemputės yra įjungtos, tai reiškia, kad 6-oji lemputė taip pat įsijungia. .

Kadangi 7-oji lemputė yra  tipo, o 2-oji lemputė yra įjungta, bet 3-oji – išjungta, tai reiškia, kad 7-oji lemputė įsijungia.

Kadangi 8-oji lemputė yra  tipo, o 4-oji lemputė yra įjungta, bet 3-oji – išjungta, tai reiškia, kad 8-oji lemputė įsijungia.

Kadangi 9-oji lemputė yra  tipo, tačiau 4-oji ir 5-oji lemputės nėra įjungtos, tai reiškia, kad 9-oji lemputė lieka išjungta.

Kadangi 10-oji lemputė yra  tipo, o 1-oji lemputė yra įjungta, bet 5-oji – išjungta, tai reiškia, kad 10-oji lemputė įsijungia.

Kadangi 11-oji lemputė yra  tipo, o 6-oji ir 10-oji lemputės yra įjungtos, tai reiškia, kad 11-oji lemputė įsijungia.

Kadangi 12-oji lemputė yra  tipo, tačiau 6-oji ir 7-oji lemputės yra įjungtos, tai reiškia, kad 12-oji lemputė lieka išjungta.

Kadangi 13-oji lemputė yra  tipo, o 7-oji ir 8-oji lemputės yra įjungtos, tai reiškia, kad 13-oji lemputė įsijungia.

Kadangi 14-oji lemputė yra  tipo, o 8-oji lemputė yra įjungta, bet 9-oji – išjungta, tai reiškia, kad 14-oji lemputė įsijungia.

Kadangi 15-oji lemputė yra  tipo, tačiau 9-oji ir 10-oji lemputės nėra įjungtos, tai reiškia, kad 15-oji lemputė lieka išjungta.

Taigi iš penkių lempučių arčiausiai centro įsijungia tik: 11-oji, 13-oji ir 14-oji.

Tai informatika!

Šiame uždavinyje kiekviena lemputė gali būti įjungta arba išjungta, ką informatikai žymi dvejetainiais skaitmenimis – 1 arba 0, arba loginėmis reikšmėmis „true“ (tiesa) arba „false“ (netiesa).
Kompiuterių grandinės, įskaitant įvairius techninius įrenginius, pvz., centrinį procesorių, naudoja logines jungtis, kad keistų loginio (*Boolean*) tipo reikšmes ir atliktų skaičiavimus.

 tipo lemputės šiame uždavinyje yra AND vartų (loginis „ir“) pavyzdys: rezultatas bus „tiesa“ (lemputė įsijungs) tik tada, jei abi įvestys yra tiesa. Priešingu atveju rezultatas bus „netiesa“ (lemputė liks išjungta).

 tipo lemputės atitinka XOR jungtį – tai reiškia „išskirtinis arba“: rezultatas bus „tiesa“ tik tuo atveju, jei tik viena iš dviejų įvesčių yra „tiesa“. Jei abi yra „tiesa“ arba abi „netiesa“, rezultatas bus „netiesa“.
Procesorius naudoja šias logines jungtis atlikdamas visus reikalingus skaičiavimus. Viskas, kas daroma kompiuteryje, kas reikalauja apdorojimo, atliekama naudojant procesorių – o tai reiškia, naudojami milijardai loginių jungčių!

14. Pabaisos piešimas 2025-MX-01

Draugai žaidžia piešimo žaidimą, vadinamą „Išridenk pabaisą“. Jie turi 4 kauliukus, kiekvienas skirtingos spalvos: raudonos, geltonos, baltos ir juodos. Kiekvienas kauliukas reiškia piešiamos pabaisos savybę:

 **Raudonas kauliukas:** pabaisos akių skaičius

 **Geltonas kauliukas:** ragų skaičius

 **Baltas kauliukas:** rankų skaičius

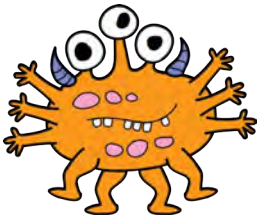
 **Juodas kauliukas:** kojų skaičius

Papildomai, sudėjus du mažiausius skaičius gaunamas pabaisos dėmių skaičius. Draugai gali piešti pabaisas kaip nori, tik turi laikytis išvardytų taisyklių.

Pateikti 4-ių žaidimo kauliukų metimo rezultatai:

Kuris piešinys atitinka visas taisykles?

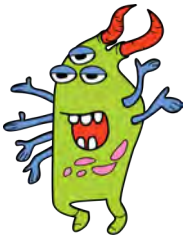
A



B

C

D



Teisingas atsakymas: B.

Paaiškinimas
Išmesti kauliukai mums rodo:
- **Raudonas kauliukas:** 3 akys
- **Geltonas kauliukas:** 2 ragai
- **Baltas kauliukas:** 6 rankos
- **Juodas kauliukas:** 4 kojos

Du mažiausi išmesti skaičiai yra **2** ir **3**, tai reiškia, kad ant pabaisos kūno turi būti **5 dėmės**.
Akys ir ragai: visi variantai turi 3 akis ir 2 ragus.
Rankos: variantai A, B ir D turi po 6 rankas; variantas C rankų neturi.
Kojos: tik variantai A ir B turi po 4 kojas.
Dėmių skaičius: B ir D variantai turi po 5 dėmes, variantas A turi 6 dėmes ir variantas C dėmių iš viso neturi.
Kaip matote, tik pabaisa B atitinka visas taisykles.

Geriau suprasti galima naudojant tokią lentelę:

	A	B	C	D
3 akys	x	x	x	x
2 ragai	x	x	x	x
6 rankos	x	x		x
6 dantys	x	x	x	
4 kojos	x	x		
5 dėmės		x		x

Tai informatika!

Kompiuterinėse programose dažnai pateikiame taisyklių rinkinį, nurodantį kompiuteriui, ką daryti esant įvairioms situacijoms. Pavyzdžiui, „jei matai skaičių 5 (kuris yra nelyginis skaičius), tada naudok taškelius“ yra lyg instrukcija kompiuteriui „jei sąlyga yra teisinga, atlik šį veiksmą“.
Mūsų uždavinys yra apie „jei–tai“ taisyklių naudojimą, norint išsiaiškinti, kurias dalis turi pabaisa ar kokių jos spalvų. Informatikoje šias taisykles vadiname „sąlygomis“ arba „teiginiais“, o kompiuteriai jas naudoja, kad nuspręstų, ką turi toliau vykdyti. Vadovaudamiesi sąlygomis ir matydami, kurios iš jų taikomos, jūs atliekate tokius pat loginius veiksmus, kaip ir kompiuteriai apdorodami informaciją.
Informatikoje taip pat dažnai rūšiuojame, rikiuojame ar filtruojame objektus, vaizdus pagal konkrečias taisykles. Šis uždavinys parodo, kaip sąlygos gali būti naudojamos kažką klasifikuojant arba kuriant žingsnis po žingsnio. Kompiuteriai daro tą patį, kai ieško vaizdų, kurie atitinka tam tikras savybes (pavyzdžiui, spalvą, skaičių arba formą), ir ignoruoja likusias.

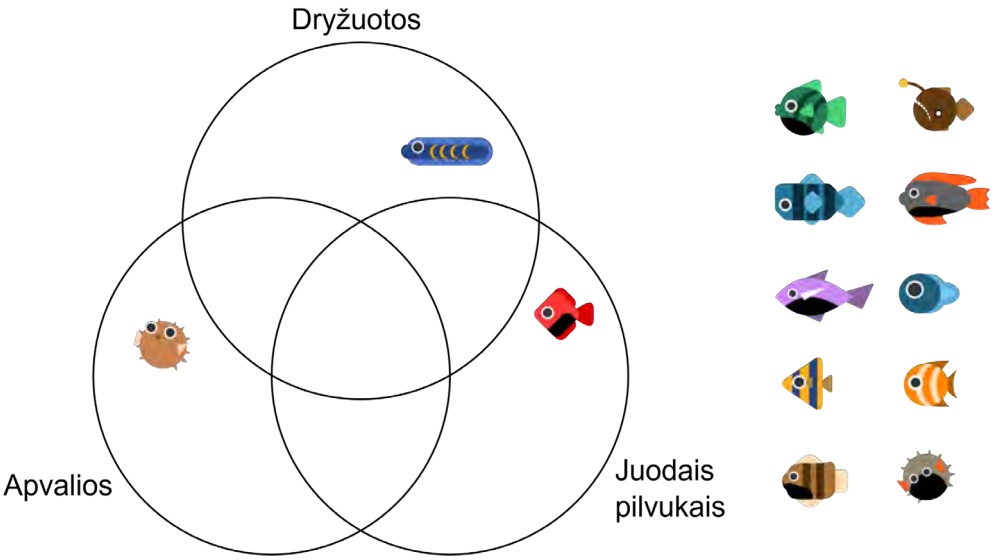
15. Žuvų rūšiavimas 2025-NL-02

Bebras Jūrų muziejuje sudarinėja žuvų sąrašą.
Jis nori surūšiuoti visas žuvis pagal tris požymius:

- apvalios žuvys;
- dryžuotos žuvys;
- žuvys juodais pilvukais.

Kai kurios žuvys patenka daugiau nei į vieną grupę. Padėkite Bebrui užbaigti diagramą.

Nutempkite kiekvieną žuvį į teisingą vietą diagramoje.



Paaiškinimas
Viena žuvis atitinka visus tris požymius:

ji yra apvali, dryžuota ir turi juodą pilvuką  .
Taip pat yra žuvų, kurios atitinka du požymius:

apvali ir dryžuota  ;

dryžuota ir juodu pilvuku  ;

apvali ir juodu pilvuku  .
Likusias žuvis lengva teisingai priskirti vienai grupei.

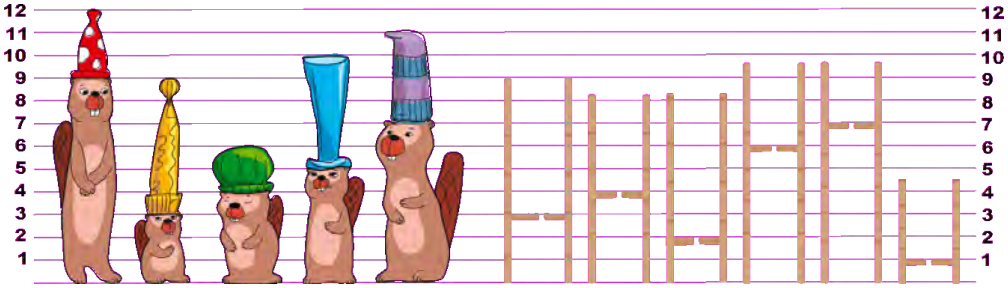
Tai informatika!
Duomenų analizei ir atvaizdavimui naudojama Veno diagrama. Joje galima pamatyti, kur skirtingi duomenų rinkiniai sutampa arba skiriasi, todėl daug lengviau pastebėti dėsningumus ir ryšius. Veno diagramomis galima aiškiai pavaizduoti informatikoje svarbias logines operacijas – IR, ARBA ir NE.
Šia konkrečia diagrama vaizduojama, kaip tam tikri duomenys gali sutapti ir viena žuvis priklausyti dviem ar daugiau grupių.

16. Kojūkai 2025-ME-02

Miestelio šventėje bebrai linksminasi vaikščiodami su kojūkais.

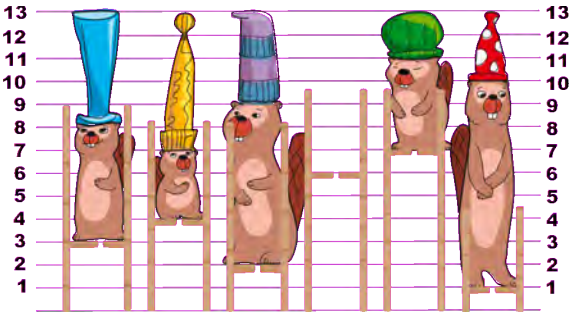
Kad būtų smagiau, bebrai dar užsidėjo įdomias kepures. Jie nori padaryti taip, kad visi bebrai, kartu su kepurėmis ir kojūkais, būtų vienodo ūgio.

Perkelkite bebrus ant kojūkų taip, kad visi būtų vienodo ūgio.



Teisingas atsakymas: A-6, B-2, C-5, D-1 ir E-3
(raidės žymi bebrus, skaičiai – kojūkų poras iš kairės į dešinę.

Paaiškinimas



Pirmiausia išsiaiškinkime norimą bendrą aukštį. Tam aukščiausią bebrą (įskaitant ir jo kepurės aukštį) pastatykime ant kojūkų, kurių pėdų atramos yra žemiausiai. Šiuo atveju aukščiausias bebras yra su raudona kepure (12), o žemiausias pėdų atramų aukštis yra ant dešinėje esančių kojūkų (1). Taigi norimas bendras aukštis yra 12 + 1 = 13. Toliau kiekvienam bebrui su kepure ieškokime kojūkų, kad bebro ir kepurės bei kojūkų pėdų atramų bendras aukštis būtų 13. Pavyzdžiui, dešiniajam bebrui, kurio aukštis su kepure yra 11, reikia kojūkų, kurių pėdų atramų aukštis būtų 2. Jei nebūtų kojūkų su tinkamo aukščio pėdų atramomis, uždavinio nebūtų įmanoma išspręsti.
Yra ir kitas būdas šiam uždaviniui išspręsti. Pirmiausia raskime žemiausią bebrą (įskaitant ir jo kepurę) ir kojūkų porą su aukščiausiai esančiomis pėdų atramomis. Ieškant kiekvieno kito bebro su kepure ir kojūkų derinio reikia tikrinti, ar bendras aukštis yra toks pat, kaip ir pirmojo (arba ankstesnio) derinio. Jei taip nėra, uždavinys neturi sprendimo.

Tai informatika!

Vienas iš šio uždavinio sprendimo būdų – godžiojo algoritmo taikymas. Algoritmas vadinamas godžiuoju, nes kiekviename jo žingsnyje pasirenkama tai, kas tuo metu atrodo geriausia (žemiausias bebras statomas ant kojūkų su aukščiausiai esančiomis pėdų atramomis), nežiūrint į bendrą vaizdą. Kartais godžiuoju algoritmu negalima rasti geriausio sprendimo.
Realiame gyvenime mes nuolat ieškome didžiausio arba mažiausio skaičiaus grupėje. Pavyzdžiui, jei su draugais lyginate, kiek kiekvienas iš jūsų turi lipdukų, greitai atrastumėte, kas jų turi daugiausiai, o kas – mažiausiai. Tai tas pats, kaip rasti žemiausią bebrą (mažiausia reikšmė), kojūkus su aukščiausiomis pėdų atramomis (didžiausia reikšmė) ir surikiuoti juos į eilę.
Pakartotinai trumpiausio (arba mažiausio) elemento grupėje ieškoma atliekant sekos rikiavimą. Sekos rikiavimo algoritmas veikia taip: vis kartojame ieškodami trumpiausio (arba mažiausio) elemento nerikiuotoje sekos dalyje ir perkeldami jį į surikiuotos dalies pabaigą, panašiai kaip rikiavome bebrus pagal ūgį nuo mažiausio iki aukščiausio. Įsivaizduokite, kad turite netvarkingą žaislinių automobilių krūvą ir norite automobilius surikiuoti nuo mažiausio iki didžiausio. Sekos rikiavimas vyktų taip: peržvelgiate visą krūvą, randate patį mažiausią automobilį ir dedate jį į naujos eilės pradžią. Tada peržiūrite likusią krūvą, randate kitą mažiausią automobilį ir dedate jį iškart už pirmojo. Taip tęsiate, kol visi automobiliai bus sudėlioti eilės tvarka!
Norint rasti būdą, kaip surikiuoti bebrus su kepurėmis ant kojūkų, kad bendri jų aukščiai būtų vienodi, reikalinga logika. Tą patį galėtume padaryti tiesiog bandymų būdu, bet teisingą sprendimą rastume greičiau, jei naudotume loginę dedukciją.

17. Safario kelionių reklama 2025-NA-01

Ekskursijų vadovas Elinamas nori reklamuoti safario ekskursijas. Jis turi pasirinkti dvi nuotraukas pagal atsakymus į savo klausimus:

1. Ar aš, Elinamas, turiu būti nuotraukoje? --- TAIP
2. Jei nuotraukoje yra mašina, ar galima rodyti jos numerį? --- NE
3. Jei nuotraukoje yra kitų žmonių, ar galima rodyti jų veidus? --- NE
4. Ar nuotraukoje būtini gyvūnai? --- TAIP

Kurias dvi nuotraukas turėtų pasirinkti Elinamas?

1)

2)

3)

4)

5)

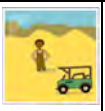
6)

7)

8)

Teisingas atsakymas: Elinamas turėtų pasirinkti 2-ą ir 5-ą nuotraukas.

Paaiškinimas
Atsakykime į kiekvieną Elinamo klausimą taip, kad iškelta sąlyga būtų tenkinama. Elinamas turi pasirinkti tas nuotraukas, kurios tenkina visų keturių klausimų sąlygas.

sąlygos	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
								
Elinamas: rodomas	TAIP	TAIP	TAIP	NE	TAIP	NE	TAIP	TAIP
Mašinos numeris: nerodomas	NE	TAIP	TAIP	NE	TAIP	NE	TAIP	TAIP
Svečių veidai: nerodomi	NE	TAIP	NE	NE	TAIP	NE	TAIP	TAIP
Gyvūnai: matomi	TAIP	TAIP	TAIP	NE	TAIP	TAIP	NE	NE

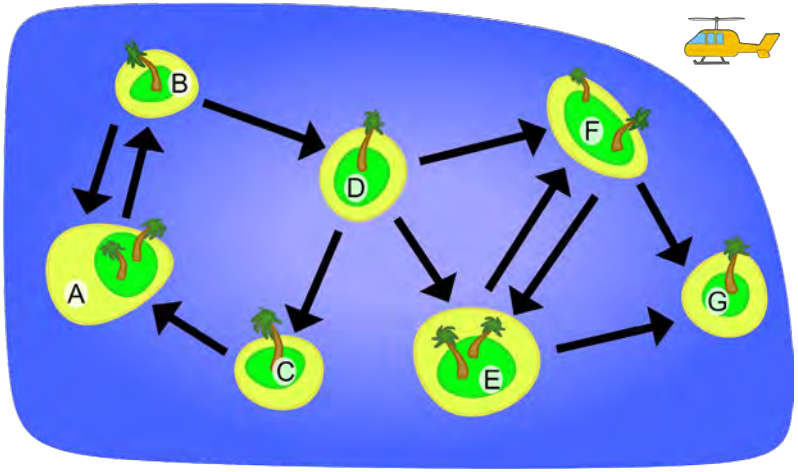
Visas sąlygas atitinka (2) ir (5) nuotraukos.

Tai informatika!

Kiekvieną klausimo ir atsakymo porą galime laikyti taisykle, pagal kurią pasirenkamos nuotraukos. Elinamas gali pasirinkti tik tas nuotraukas, kurios atitinka visas taisykles. Informatikoje tokio tipo uždavinius vadiname apribojimų tenkinimo uždaviniais. Apribojimai yra taisyklės, kurių turime laikytis; taisyklės laikymasis yra vadinamas apribojimo arba sąlygos tenkinimu.
Šiame uždavinyje visi Elinamo klausimai yra tokio tipo, į kuriuos atsakoma „taip“ arba „ne“, todėl kiekvienam klausimui yra tik du atsakymai (dvejetainis kodavimas). Tačiau yra daug skirtingų apribojimų tipų. Kaip uždavinyje gali būti daugiau nei vienas klausimas, taip ir apribojimai gali būti sudaryti iš kelių, kuriuos reikia tenkinti. Kuo daugiau apribojimų reikia tenkinti, tuo sunkiau išspręsti uždavinį.

18. Salų tyrinėjimo ekspedicijos 2025-AT-01

Tyrėjų komanda siekia ištirti visas pavaizduotas salas. Į ekspediciją ji vyksta ir grįžta sraigtasparniu. Tarp salų kursuoja keltai, tačiau tik nurodytomis kryptimis. Ekspedicijos komanda gali nusileisti bet kurioje saloje, keltais keliauti tiek kartų, kiek reikia, tačiau privalo grįžti į tą pačią salą, kurioje buvo paliktas sraigtasparnis.



Kiek mažiausiai ekspedicijų turi surengti komanda, kad aplankytų visas salas?

Teisingas atsakymas: Reikia surengti mažiausiai 3 ekspedicijas.

Paaiškinimas

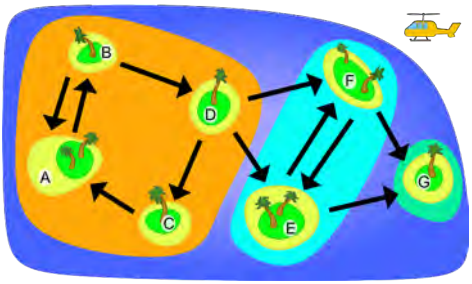
Kadangi baigiantis kiekvienai ekspedicijai komanda turi grįžti prie sraigtasparnio, ji gali aplankyti tik tas salas, iš kurių galima sugrįžti į pradinę salą. Norint išspręsti šį uždavinį, salas suskirstome į grupes taip, kad:

- Iš bet kurios grupei priskirtos salos keltu galima pasiekti visas kitas tos grupės salas.
- Išvykus iš salų grupės į ją grįžti jau nebeįmanoma.

Norint sudaryti tokią grupę, galima pradėti nuo bet kurios dar nepriskirtos salos ir tada įtraukti visas salas, į kurias galima nuplaukti ir iš jų grįžti. Paveiksle salos suskirstytos į tris spalvomis pažymėtas grupes. Kadangi keliauti tarp šių grupių galima tik viena kryptimi, kiekvienai jų reikia atskiros ekspedicijos.

Dažnai daroma klaida, kad štai toks ilgas maršrutas D → C → A → B → D → F → E → G galėtų būti tinkamas. Nors šiuo maršrutu aplankomos visos salos, komanda negalėtų grįžti prie sraigtasparnio, todėl šis sprendimas yra netinkamas.

Taigi komandai reikia trijų atskirų ekspedicijų – po vieną kiekvienai salų grupei.



Tai informatika!

Spręsdami salas suskirstome į grupes, kurių kiekvienoje iš bet kurios salos galima pasiekti bet kurią kitą. Informatikoje toks konceptas vadinamas stipriai susijusiais komponentais.

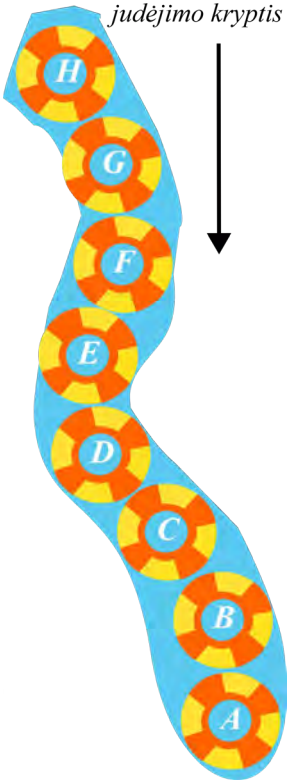
Stipriai susiję komponentai gali jungti ne tik salas, bet ir įvairius kitus objektus. Štai keli pavyzdžiai:

- Interneto naršymas: komponentai gali žymėti tinklalapių grupes, kurios visos yra tarpusavyje susietos.
- Socialiniai tinklai: komponentai gali parodyti vartotojų grupes, kur visi seka vieni kitus.
- Transporto tinklai: komponentai gali apibūdinti regionus, kuriuose galima keliauti tarp miestų abiem kryptimis.

Išskirti stipriai susijusius komponentus naudinga daugelyje sričių, įskaitant tinklų analizę, rekomendavimo sistemas ir optimizavimo uždavinius.

Norėdami išspręsti šį uždavinį, mokiniai turi atpažinti, kad jis susijęs su salų grupių paieška, tarp kurių galima judėti pirmyn ir atgal – tai reikalauja abstrahavimo. Be to, jie turi sukurti sistemingą būdą tokioms grupėms rasti – tam reikia algoritminio mąstymo.

19. Plaukimo ratai 2025-CA-04



Vandens parko nusileidimo kalneliuose panaudoti plaukimo ratai slenka pasroviui trumpa lėtai tekančia upe. Upės gale jie susirenka į eilę, kaip parodyta paveiksle.

Kai kuris nors ratas ištraukiamas iš upelio, kad kas nors galėtų juo pasinaudoti, kiekvienas už jo upelyje esantis plaukimo ratas pasislenka upeliu viena pozicija žemyn ir atsiradusi spraga užsipildo.

Pavyzdžiui, jei kas nors ištraukia iš upelio plaukimo ratą F, tai G ir H ratai kiekvienas pasislenka viena pozicija žemyn. Jei po to kas nors ištraukia A ratą, tada visi šeši likę plaukimo ratai pasislenka po vieną poziciją žemyn. Šiuo atveju iš viso įvyksta aštuoni plaukimo ratų poslinkiai (2 + 6 = 8).

Aštuoni plaukimo ratai upelyje iš pradžių išsidėstė, kaip parodyta paveiksle. Tada penki ratai iš upelio ištraukiami tokia tvarka: B, G, E, D, H. Kiek plaukimo ratų poslinkių įvyko iš viso?

A. 10
B. 11
C. 12
D. 13

Teisingas atsakymas: B.

Paaiškinimas
Pradžioje plaukimo ratų eilė yra tokia: A, B, C, D, E, F, G, H.
Kai ratas B ištraukiamas iš upelio, kiekvienas iš šešių ratų pasislenka viena pozicija žemyn (C, D, E, F, G ir H). Dabar ratų eilė tokia: A, C, D, E, F, G, H.
Kai plaukimo ratas G ištraukiamas iš upelio, tik H ratas pasislenka viena pozicija žemyn. Dabar ratų eilė tokia: A, C, D, E, F, H.
Kai plaukimo ratas E ištraukiamas iš upelio, du plaukimo ratai kameros F ir H pasislenka po vieną poziciją žemyn. Dabar ratų eilė tokia: A, C, D, F, H.
Kai plaukimo ratas D ištraukiamas iš vandens, vėl du plaukimo ratai F ir H pasislenka po vieną poziciją žemyn.. Dabar ratų eilė tokia: A, C, F, H.
Kai plaukimo ratas H ištraukiamas iš vandens, nė vienas ratas nepasislenka žemyn.
Iš viso įvyks 11 poslinkių (6 + 1 + 2 + 2 = 11) po vieną poziciją žemyn.

Tai informatika!

Dažnai duomenys ar informacija savaime sudaro sekas. Kasdieniame gyvenime tai gali būti lėktuvai, laukiantys prie pakilimo tako, dokumentų redagavimo sąrašas (pagal tvarką) ar plaukimo ratai upelyje, kaip šio uždavinio istorijoje. Kai naudojame kompiuterinę programą, kad išspręstume su tokiais scenarijais susijusias problemas, paprastai turime šiuos duomenis saugoti. Vienas iš būdų tai padaryti – saugoti duomenis atmintyje seka. Tai reiškia, kad skirtingi duomenys (pvz., lėktuvai, pakeitimai ar plaukimo ratai) gali būti saugomi iš eilės. Vienas iš šio metodo privalumų yra tai, kad lengva rasti elementą sekoje, žinant jo vietą. Vienas šio metodo trūkumas yra tai, kad pašalinus (arba įterpus) elementus iš sekos, tenka perkilnoti elementus, kad jie būtų saugomi iš eilės atmintinėje. Tai analogiška plaukimo ratams, plaukiantiems pasroviui ir užpildantiems paimto rato paliktą tarpą. Tai užima laiko, o kai tai daroma pakankamai dažnai, gali žymiai sulėtinti programos veikimą.
Atkreipkite dėmesį, kad net jei realiame pasaulyje duomenys nėra seka (pvz., gyventojų skaičius), vis tiek galime nuspręsti juos saugoti atmintyje seka. Priimdami tokį sprendimą, turime įvertinti privalumus ir trūkumus. Tam reikia suprasti, kaip duomenys saugomi atmintinėje ir kaip veikia (vykdomos) kompiuterinės programos.

20. Statiniai iš blokelių

2025-CH-01a

Turi šiuos blokelius:


6 kubelius


1 tiltą

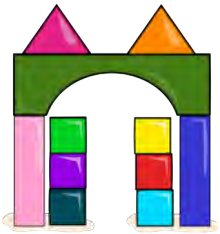

2 piramides

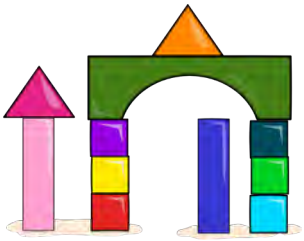

2 stulpelius

Draugas pateikia šias statymo taisykles.

1. Paimk 3 kubelius.
2. Sudėk juos vieną ant kito, kad gautum bokštą.
3. Iš likusių 3 kubelių sudėk dar vieną bokštą.
4. 2 stulpelius padėk šalia pastatytų bokštų.
5. Tiltą uždėk ant savo konstrukcijos.
6. Paimk 2 piramides ir uždėk jas ant savo konstrukcijos.

Pagal šias taisykles gali pastatyti daug įvairių konstrukcijų. Štai du pavyzdžiai:



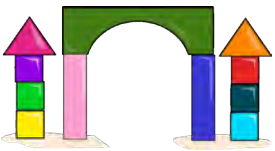


Kurios iš šių konstrukcijų NEĮMANOMA pastatyti pagal pateiktas taisykles?

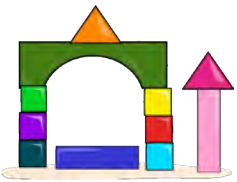
A

B

C



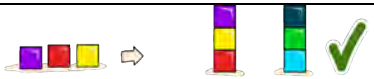
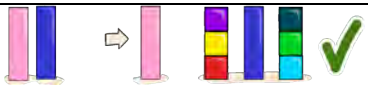
D



Teisingas atsakymas: B.

Paaiškinimas

Matome, kad taisyklės nėra tiksliai aprašytos, todėl galima sudaryti daug skirtingų konstrukcijų. Tačiau yra keletas apribojimų, kurių būtina laikytis. Tik konstrukcija B pažeidžia vieną iš taisyklių. Tikrinsime taisykles žingsnis po žingsnio:

1. Paimk 3 kubelius. 2. Sudėk juos vieną ant kito, kad gautum bokštą.	
3. Iš likusių 3 kubelių sudėk dar vieną bokštą.	
4. 2 stulpelius padėk šalia pastatytų bokštų.	
5. Tiltą uždėk ant savo konstrukcijos.	

Pastebime, kad pirmųjų 4 taisyklių laikomasi. Tačiau laikantis 5-os taisyklės neįmanoma padėti tiltelio PO konstrukcija. Taisyklėje aiškiai nurodyta, kad tiltelis turi būti padėtas ANT konstrukcijos. Reikia bent dviejų vienodo aukščio konstrukcijų, kad ant jų galėtume padėti tiltelį.

Taisyklėse nėra tiksliai apibrėžta:

- ... vertikali blokų padėjimo tvarka. Nurodymas „šalia bokšto“ neapibrėžia, ar stulpelis turi būti padėtas kairėje ar dešinėje bokšto pusėje. Abu variantai yra geri;
- ... ar svarbi blokų spalva;
- ... ar naujas blokas turi būti padėtas ant visų anksčiau pastatytų konstrukcijų, ar tik ant kai kurių iš jų;
- ... blokų kryptis.

Neaiškios taisyklės gali lemti įvairias teisingas konstrukcijas arba nenuspėjamus rezultatus.

Tai informatika!

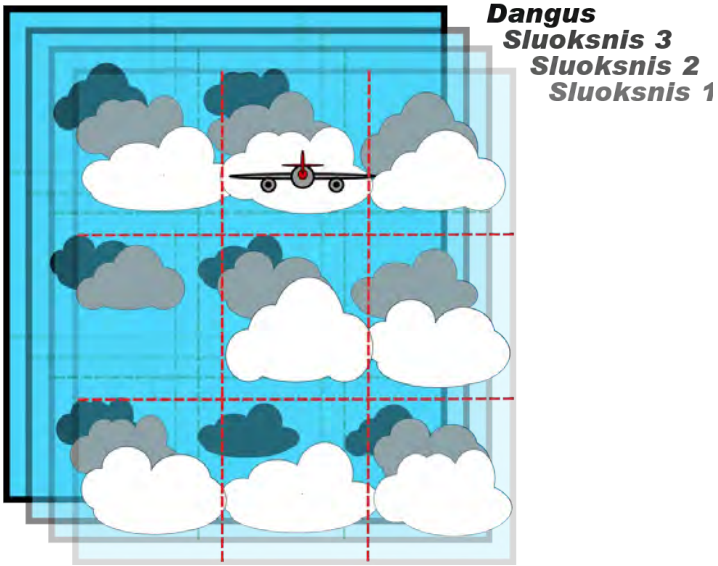
Šis uždavinys parodo, kad aiškios taisyklės yra būtinos ir kasdieniame gyvenime, ir programavime. Jos yra tarsi kepimo receptai: neaiškūs veiksmai gali lemti nenuspėjamus rezultatus. Receptai ir šio uždavinio taisyklės iš tikrųjų yra algoritmai: žingsnis po žingsnio nurodoma, kaip atlikti užduotį arba išspręsti problemą.

Programavime aiškiai apibrėžtos taisyklės yra labai svarbios, nes kompiuteriai negali interpretuoti ar spėlioti. Jiems reikia tikslių taisyklių, kad galėtų atlikti užduotis numatytu būdu. Neaiškios programavimo taisyklės gali sukelti programinės įrangos veikimo sutrikimus arba duoti nenumatytus rezultatus.

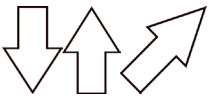
Prieš rašydamas programą, programuotojas turi pagalvoti, kokius apribojimus reikia apibrėžti, kad gautų norimą rezultatą. Šiame uždavinyje taisyklės turėtų nusakyti, pavyzdžiui, blokų vertikalią ir horizontalią tvarką, pačių blokų orientaciją ir formas, kurios turi būti sujungtos, kai blokai dedami vienas ant kito ir pan.

21. Lėktuvas tarp debesų 2025-IE-04

Lėktuvas artėja prie debesų, kurie išsidėstę 3×3×3 tinklelio lauke. Pilotas nori išvengti šių debesų. Kiekvieną kartą, kai pilotas atlieka veiksmą, lėktuvas pakyla, nusileidžia, skrenda į kairę, į dešinę arba įstrižai ir nuskrenda vieną žingsnį į priekį. Balti debesys yra arčiausiai, pilki – toliau, o tamsūs – toliausiai.



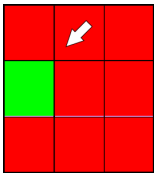
Lėktuvas netrukus įskris į 3×3×3 tinklelio lauką. Jei pilotas nepasuks lėktuvo, jis atsitrenks į baltą debesį viršutinės eilės viduryje.
Kurios komandos padėtų pilotui perskristi 3×3×3 tinklelio lauką neatsitrenkiant į debesis?

A.	B.	C.	D.
			

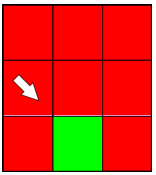
Teisingas atsakymas: 

Paaiškinimas

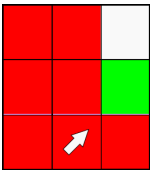
Šis uždavinys skirtas debesų padėčiai atpažinti. Visi debesys išsidėstę trimis sluoksniuose ir žymi sritis, į kurias lėktuvas negali įskristi. Sluoksniai pavaizduoti trimis lentelėmis – taip lengviau suprasti.



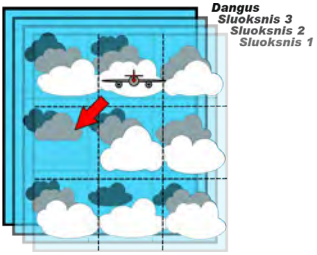
Pirmame sluoksnyje pilotas gali pasirinkti tik šį vieną langelį



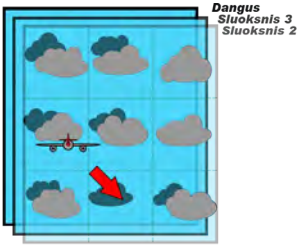
Antrame sluoksnyje pilotas gali pasirinkti tik šį vieną langelį



Pasirinkęs šį langelį pilotas trečiame sluoksnyje, nekliudys debesų



Etapas 1



Etapas 2

Etapas 3

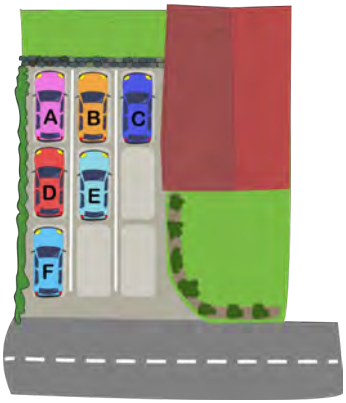
Tai informatika!

Norint išspręsti šį uždavinį reikia gilintis į dvimatį vaizdą tarytum erdvę ir joje orientuotis. Orientavimasis trimatėje erdvėje sutinkamas daugelyje informatikos sričių, pradedant kompiuteriniais žaidimais ir baigiant sudėtingais erdvinės informacijos atvaizdavimais.

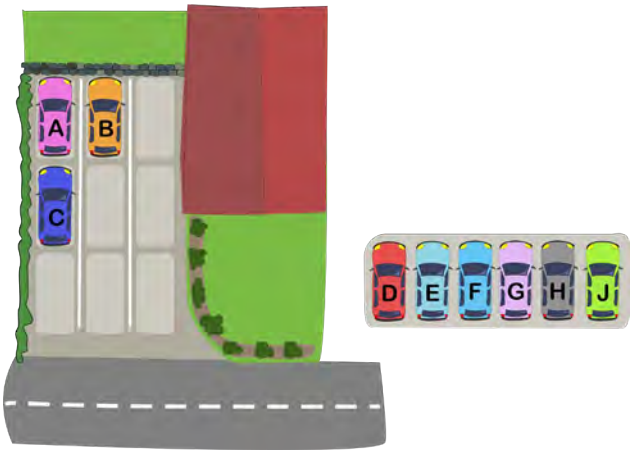
Diskretiška erdvė, pavyzdžiui, šiame uždavinyje langelių tinklelis, atitinka grafo sąvoką. Grafams naršyti sukurta daug algoritmų. Nors šiame uždavinyje yra tik vienas galimas lėktuvo kelias, daugelyje realių informatikos uždavinių tenka pasirinkti vieną iš kelių galimų grafo kelių, pavyzdžiui, rasti trumpiausią kelią. Sukurti algoritmai, kuriais galima naršyti erdvėse, turinčiose daugiau nei tris matmenis, – šitaip sprendžiamos itin sudėtingos problemos.

22. Automobilių parkavimas 2025-SI-03

Jonės šeima rengia vakarėlį ir pakvietė daug svečių. Jie atvyksta automobiliais. Jonės kieme yra vietos 9 automobiliams ir jie turi stovėti trimis eilėmis po 3 automobilius vienas paskui kitą. Kiekvienas atvykstantis prašomas pastatyti automobilį pirmoje laisvoje vietoje bet kurioje eilėje.



Šiame pavyzdyje automobilis A atvyko pirmas, automobilis D – antras, automobilis C – trečias ir pan. Svarbu, kad automobilis F turės išvažiuoti prieš automobilį D.
Svečiai atvyksta tokia tvarka:
Aras, **B**enas, **C**ezaris, **D**alė, **E**lena, **F**austa, **G**aja, **H**erkus, **I**ndra
Jie iš vakarėlio išvyks tokia tvarka:
Gaja, **D**alė, **F**austa, **I**ndra, **H**erkus, **C**ezaris, **B**enas, **A**ras, **E**lena
Sustatyk automobilius pagal atvykimo eilę, kad nė vienas automobilis neužblokuotų automobilio, kuris turi išvykti anksčiau.



Paaiškinimas

Galime išanalizuoti kiekvienos svečių poros situaciją:
Aras atvyksta anksčiau už Cezarį, o Cezarį išvyksta anksčiau už Arą. Taigi jie gali pastatyti automobilius toje pačioje eilėje. Tas pats galioja Arui ir Benui bei Benui ir Cezariui.
Aras ir Elena atvyksta ta pačia tvarka, kuria išvyksta, todėl jie negali pastatyti automobilių toje pačioje eilėje. Tas pats galioja Benui ir Elenai; Cezariui ir Elenai; Dalei, Faustai, Herkui ir Indrai; Faustai, Herkui ir Indrai; Gajai, Herkui ir Indrai.
Taigi Aras, Benas ir Cezaris gali pastatyti automobilius 1-oje eilėje atvykimo tvarka.
Tada atvyksta Dalė, ji pastato automobilį 2-oje eilėje, nes 1 eilė jau užimta.
Atvykusi Elena turi pastatyti automobilį kitoje eilėje nei Dalė, todėl ji turi pasirinkti 3-ią eilę.
Tada atvyksta Fausta, ji taip pat turi pastatyti automobilį kitoje eilėje nei Dalė – 3-oje eilėje, nes ji išvažiuos anksčiau nei Elena, kuri ten pastatė automobilį.
Po jos atvažiavusi Gaja išvyks pirma, todėl ji turi pastatyti automobilį paskutinė eilėje – tai bus trečias automobilis 3-oje eilėje.
Šiuo metu 2-oje eilėje yra dvi vietos. Atvyksta Herkus, po jo – Indra, kuri išvažiuos anksčiau už Herkų, todėl jie gali statyti automobilius 2-oje eilėje atvykimo tvarka.
Po vakarėlio draugai išvažiuoja tokia tvarka: Gaja, Dalė, Fausta, Indra, Herkus, Cezaris, Benas, Aras, Elena. Niekas nėra užblokuotas kito automobilio, nes:

1. Pirmiausia išvyksta Dalė ir Gaja.
2. Kai Gaja išvyksta, Faustos automobilis jau nebebus užstatytas.
3. Dalė atlaisvina Beno automobilį.
4. Fausta atlaisvina Cezario automobilį.
5. Indra atlaisvina Herkaus automobilį.
6. Herkus atlaisvina Elenos automobilį.
7. Cezaris atlaisvina 2 eilę.
8. Benas atlaisvina Aro automobilį.
9. Aras ir Elena gali išvažiuoti, nes tik jų dviejų automobiliai likę eilėje.

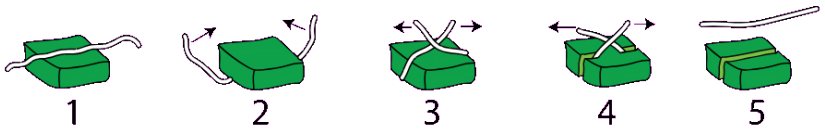
B atsakymas neteisingas, nes Dalė atvyksta prieš Faustą, taigi ji negali parkuotis už jos.
C atsakymas neteisingas, nes Dalė turi išvykti prieš Herkų, bet Herkus jau prisiparkavęs už jos.
D atsakymas neteisingas, nes svečiai parkavosi pagal tai, kaip atvykdavo, todėl Gaja užblokuota, o ji turi išvykti pirmiausiai.

Tai informatika!

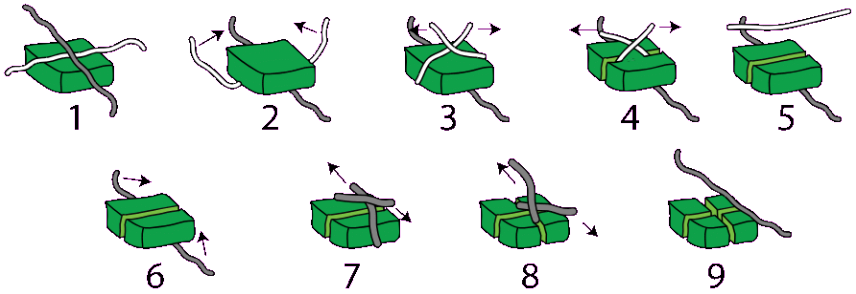
Jonės kieme automobiliai sustatyti eilėmis – tai informatikoje vadinama *dėklu* (arba *steku*). **Dėklas** yra duomenų struktūra, naudojama nuosekliems duomenų ar objektų rinkiniams laikyti. *Duomenų struktūros* yra svarbios informatikoje, nes jos leidžia organizuoti duomenis taip, kad juos būtų galima efektyviai apdoroti, analizuoti ir atlikti įvairius veiksmus.
Dėklas veikia kaip indų krūva: kai reikia įdėti naują objektą, jis dedamas ant dėklo viršaus (šis veiksmas angliškai vadinamas *push*). Visada galima tik pažiūrėti (angl. *peek*) arba pašalinti (angl. *pop*) objektą iš dėklo viršaus. Duomenų struktūra, veikianti tokiu būdu, dar vadinama **LIFO** (angl. *Last-In First-Out* – paskutinis įėjo, pirmas išėjo). Taigi, naudojant dėklą galima tiesiogiai pasiekti ir pašalinti tik paskutinį pridėtą elementą, kaip ir šiame uždavinyje su automobiliais.
Uždavinys, kaip išdėstyti daug automobilių į kelis dėklus (eiles), kad jie netrukdytų vieni kitiems išvažiuoti, ir panašios problemos vadinamos *multisteko padalijimo problemomis*, o jų sprendimo algoritmai taikomi išteklių paskirstymui užduotims, pavyzdžiui, daugiabranduolių procesorių įrenginiuose, kur gali vykti lygiagretus apdorojimas.

23. Vietnamietiškas pyragas 2025-VN-01

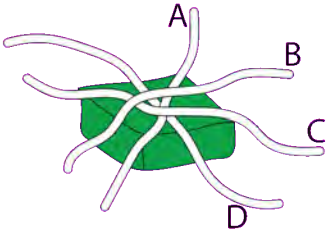
Vietname bebrai švęsdami Mėnulio Naujuosius metus valgo *bánh chưng* pyragą. Tačiau šis ryžių pyragas yra pernelyg lipnus, kad jį būtų galima supjaustyti peiliu, todėl bebrai naudoja virvutes:



Pyragui supjaustyti bebrai naudoja kelias virvutes, bet jie gali susipainioti, jei jas naudos neteisinga tvarka. Pavyzdžiui, toliau pateiktame paveiksle bebras gali pjaustyti pyragą pirmiausia naudodamas baltą virvutę, po to – pilką. Jei bus daroma atvirkščiai, pilka virvutė įsipainios į baltąją:



Kokia tvarka bebras turėtų naudoti keturias virvutes, kad supjaustytų pyragą



nesusipainiodamas?

A. ABDC
B. BCAD
C. CDBA
D. DACB

Teisingas atsakymas: D.

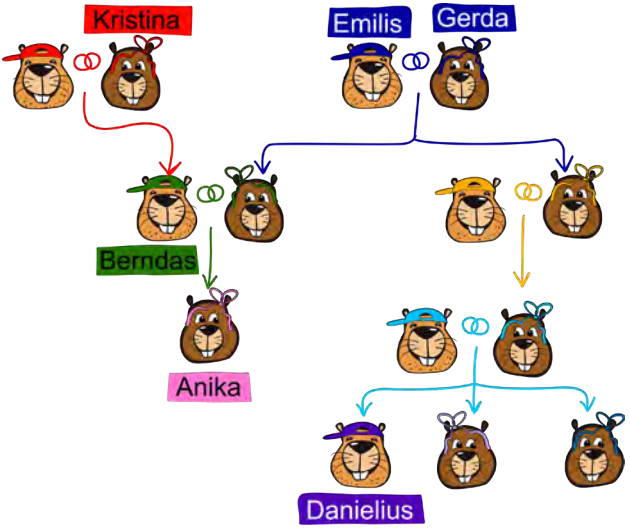
Paaiškinimas
Naudojant virvutes pyragui pjaustyti, reikia laikytis principo, kad pirmoji ant pyrago uždėta virvutė turi būti ir pirma, kuria pjaunama. Iš tiesų pirmoji virvutė yra vienintelė, kuri liečia pyrago paviršių be jokių trukdžių. Kai pirmoji virvutė panaudota, galima naudoti antrąją ir t. t., laikantis tos pačios tvarkos, kuria virvutės buvo uždėtos ant pyrago.
Todėl naudodamas virvutes bebras turi laikytis šios eilės: **DACB**.

Tai informatika!
Uždavinio sprendimo tvarka turi atitikti vadinamąją FIFO (**f**irst **i**n, **f**irst **o**ut – pirmas įdėtas, pirmas išimtas) taisyklę: pirmiausiai pridėtas elementas yra pašalinamas pirmiausia, panašiai kaip eilėje, kurioje niekas neturi praleisti savo vietos. Informatikoje yra daug uždavinių, kuriuose naudinga taikyti FIFO taisyklę, kad duomenys būtų apdorojami taip, kad pirmas įvestas elementas, tai yra, eilės „priekis“, būtų apdorojama pirmiausia. Ši struktūra informatikoje vadinama eile (anglų k. *queue*).
Kita dažnai taikoma taisyklė yra LIFO (**l**ast **i**n, **f**irst **o**ut – paskutinis įdėtas, pirmas išimtas): ji veikia atvirkščiai – paskutinis įvestas elementas yra pašalinamas pirmiausiai, pavyzdžiui, plauti sukrautų lėkščių stirta. Ši struktūra informatikoje vadinama dėklu arba steku (anglų k. *stack*).

24. Giminės medis

2025-DE-07

Bebrai Anika ir Danielius dalyvauja šeimos susitikime. Vienas šeimos narys paklausė: „Kaip jūs esate giminiškai susiję?“ Anika popieriaus lape nupiešė giminės medį. Piešinyje moterys bebrės papuoštos kaspinėliais, o vyrai bebrai – kepuraitėmis.



Anikai patinka apibūdinti giminės ryšį šitaip:

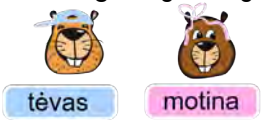
tėvas(X) reiškia „tėvas X bebro“
motina(X) reiškia „motina X bebro“

Pavyzdžiui, Anikos tėvas yra Berndas, o šio motina yra Kristina. Šiuos ryšius mes galime apibūdinti taip:

tėvas(Annika) = Berndas
motina(Berndas) = Kristina
motina(tėvas(Anika)) = Kristina

Staiga Anikai prireikė išeiti. Padėk išsiaiškinti paveiksle pateiktą giminės medį.

Apibūdink Anikos ir Danieliaus giminystės ryšį – užpildyk visus tuščius langelius.



tėvas (motina (Anika)) = ( ) ( ) ( ) (Danielius)))

Teisingas atsakymas: tėvas(motina(Anika)) = tėvas(motina(motina(Danielius))).

Paaiškinimas

Pirmiausiai randame reikšmę kairėje: *tėvas(motina(Anika))*, o tai yra Emilis. Jis yra Anikos motinos tėvas (senelis iš motinos pusės). Kad užpildytume langelius dešinėje pusėje, turime nurodyti, kaip Danielius yra susijęs su Emiliu.

Kad tai išsiaiškintume, turime pradėti nuo Danieliaus ir kilti aukštyn po vieną žingsnį, kol pasieksime Emilį. Pakylame aukštyn viena karta iki Danieliaus motinos – *motina(Danielius)*. Tuomet kylam dar viena karta aukštyn iki Danieliaus motinos motinos – *motina(motina(Danielius))*, tai yra, Danieliaus senelės iš motinos pusės. Galiausiai randame močiutės tėvą – *tėvas(motina(motina(Danielius)))*.

tėvas (motina (Anika)) = tėvas (. motina (. motina (. Danielius)))

Tai informatika!

Funkcijų kompozicija ir įterpimas (angl. *nesting*)

Šis Bebro uždavinys demonstruoja, kaip kreipiniai į funkcijas gali būti komponuojami, sudedami vienas į kitą, kad būtų galima apskaičiuoti sudėtingus ryšius.

Funkcijos yra programų dalys, kurias galima naudoti daug kartų. Funkcija turi pavadinimą ir parametrus. Jei turime funkcijos aprašą ***f(p)***, tai ***f*** yra funkcijos pavadinimas, o ***p*** – įvestis arba parametras. Pavyzdžiui, skaičiaus 3 padvigubinimo procesą galima aprašyti taip: *multiply_by_two(3)*, čia „multiply_by_two“ yra funkcijos pavadinimas, o skaičius „3“ yra parametras. Parametrai kartais vadinami argumentais.

Skaičiavimo požiūriu, „tėvas()“ ir „motina()“ yra funkcijos, į kurias galima kreiptis, kad jos apskaičiuotų ir pateiktų reikšmę, vadinamą **rezultatu**.

Įterptame kreipinyje į funkciją vidinių funkcijų rezultatai tampa išorinės funkcijos įvestimi. Įterptas kreipinys į funkciją nagrinėjamas „iš vidaus į išorę“. Pavyzdžiui, tėvas(motina(motina(Danielius))) pirmiausia randa Danieliaus motiną, tada jo motinos motiną, o galiausiai jo motinos motinos tėvą. Funkcijų kompozicijos mechanizmas galimas daugumoje programavimo kalbų, bet ypač svarbus funkcinėse programavimo kalbose (pvz., *Scheme*).

Hierarchinės duomenų struktūros

Giminės medis yra hierarchinė duomenų struktūra, kurioje kiekvienas asmuo yra organizuotas tam tikru lygiu, pavyzdžiui, tėvai, seneliai ar proseneliai. Uždavinys parodo, kaip tokios struktūros gali būti naudojamos analizuoti ir nustatyti ryšius tarp elementų.

Ši idėja naudojama, pavyzdžiui, duomenų bazėse ir XML struktūrose, įskaitant interneto svetaines. Interneto svetainės yra sudarytos iš hierarchinių elementų. Kiekvienas elementas, kurį matote tinklalapyje – mygtukai, teksto blokai ar paveikslėliai – yra medžio struktūros dalis. Šie elementai turi vadinamuosius tėvų ir vaikų ryšius, leidžiančius naršyklėms tvarkingai ir struktūruotai atvaizduoti sudėtingus išdėstymus. Šią struktūrą aprašanti kalba vadinama **HTML**.

25. Katinų kelionės planas 2025-AZ-03

Bebras turi paimti penkis sergančius katinus ir nunešti juos pas veterinarą. Yra keletas gyvūnų pernešimo dėžių. Kiekviena dėžė turi leistiną svorio ribą (kilogramais).

Dėžė	Leistinas svoris
A	10 kg
B	15 kg
C	20 kg
D	5 kg

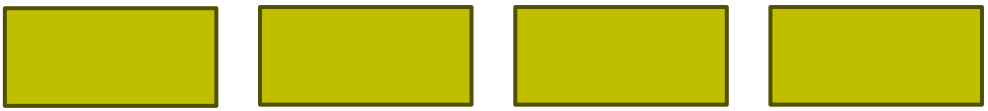
Katinas	Svoris
	3 kg
	7 kg
	10 kg
	5 kg
	6 kg

Bebras nori:

- paimti **visus katinus**,
- panaudoti **kuo mažiau dėžių**, neviršijant jų leistinų svorių.

Sudėkite katinus taip, kad visi jie galėtų būti paimti panaudojant **kuo mažiau dėžių**.

A (10 kg) B (15 kg) C (20 kg) D (5 kg)



Teisingas atsakymas:

Dėžėje B: 

Dėžėje C: 

Paaiškinimas

 (10 kg) +  (5 kg) = 15 kg – telpa į dėžę B (15 kg)

 (3 kg) +  (7 kg) +  (6 kg) = 16 kg – telpa į dėžę C (20 kg)

Norint greitai rasti teisingą atsakymą, galima pradėti nuo to, kiek dėžių naudojama kiekviename variante.
Variantuose a ir e naudojama po 2 dėžes, C ir D – po 3, o B – net 4. Todėl pirmiausia imame tikrinti variantus a ir e.
Paaiškėja, kad a variantas netinka, nes trijų katinų bendra masė būtų 21 kg, o tai viršija dėžės C talpą (20 kg).
Toliau tikrinamas e variantas, jis atitinka visas svorio ribas.

Tai informatika!

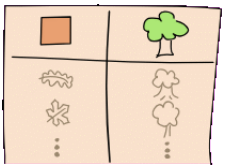
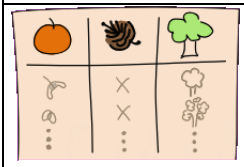
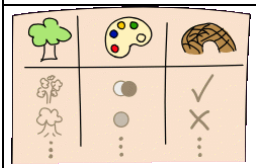
Šiuo uždaviniu modeliuojamas išteklių paskirstymas esant apribojimams – tai vienas pagrindinių informatikos konceptų. Jis primena kuprinės uždavinio (angl. *knapsack problem*) versiją – tai kombinatorinio optimizavimo uždavinys, kai reikia iš duoto skaičiaus daiktų, turinčių svorius, atrinkti tuos, kurie tilps į riboto svorio saugyklą.
Šiuo atveju katinai atitinka daiktus, o saugykla – tai gyvūnų pernešimo dėžės:

- kiekviena dėžė turi svorio ribojimą,
- tikslas – panaudoti kuo mažiau dėžių.

Mokiniai turi įvertinti, kokios daiktų kombinacijos yra tinkamos, kad būtų optimaliai išnaudota saugykla. Tokie sprendimai svarbūs operacinėse sistemose, atminties valdymo ir failų sistemose.

26. Bebrų mediena 2025-CH-04

Aras ir jo draugai mėgsta keliauti pėsčiomis ir rinkti informaciją apie medžius. Informaciją jie užrašo į lenteles.

	**Saulė** renka informaciją apie medžių lapų formas ( ) ir tų medžių rūšis ( ).
	**Gaja** renka informaciją apie medžių vaisius ( ), tų medžių rūšis ( ) ir ar tai spygliuočiai ( ).
	**Meda** renka informaciją apie medžių spalvą ( ), medžių rūšis ( ) ir ar jų mediena tinka bebrų nameliams statyti ( ).

Aras miške rado lapą ir žino jo formą. Jis nori sužinoti, ar šios rūšies medžio mediena tinka bebrų nameliams statyti.

Kurį iš savo draugų jis turi paklausti ir kokia tvarka, kad gautų teisingą atsakymą?

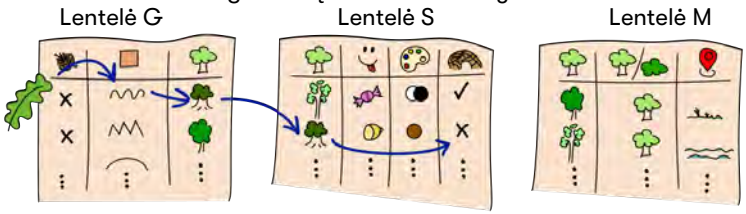
A) Tik Medą.
B) Pirmiausia Saulę, tada Gają.
C) Pirmiausia Saulę, tada Medą.
D) Pirmiausia Gają, tada Saulę ir paskui Medą.

Teisingas atsakymas: C.

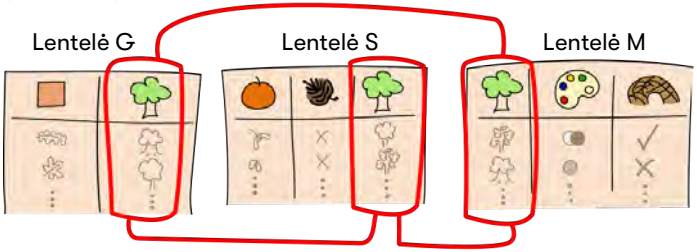
Paaiškinimas

Informacija apie tinkamą bebrams medieną ( ) randama tik Medos lentelėje (M). Tačiau jei Aras turi tik informaciją apie lapą, jis negalės pasirinkti eilutės iš šios lentelės. Jam reikia gauti informacijos apie medžio rūšį ( ) arba medienos spalvą ( ). Gajos lentelėje (G) nėra informacijos apie lapus, bet tai yra Saulės lentelėje (S). Žinodamas lapo formą, Aras gali pasirinkti teisingą Saulės lentelės eilutę ir gauti trūkstamą informaciją apie medžio rūšį ( ). Tada jis gali pasirinkti teisingą Medos lentelės eilutę ir gauti trūkstamą informaciją apie medieną.

Pavyzdžiui, jei lapas yra qžuolo, Aras gali pasirinkti tinkamą eilutę Medos lentelėje ir sužinoti, kad qžuolas nėra tinkama medžiaga bebrų nameliams statyti.



Lentelės yra susietos tarpusavyje vienu bendru duomenų elementu: medžių rūšimi ( ). Jei pasirenkate bet kurį vieną duomenų elementą iš bet kurios lentelės, turėsite prieigą prie visų lentelių duomenų.



Tai informatika!

Šis uždavinys iliustruoja pagrindinius *reliacinių duomenų bazių* konceptus. Emilio draugų sukurtos lentelės atitinka duomenų bazės lenteles, kuriose kiekviena *stulpelis žymi savybę*, o kiekviena *eilutė* yra *įrašas* (angl. *tupla*). Lentelių ryšys naudojant bendrą medžio rūšies savybę ( ) atitinka išorinių *raktų* sąvoką reliacinėse duomenų bazėse ir nustato lentelių tarpusavio ryšius.

Aro informacijos apie tinkamą medieną nameliui statyti paieška pagal medžių lapų formą atspindi tipišką *duomenų bazės užklausą*. Šiai užklausai reikia *sujungti* kelias lenteles, kad būtų gauta norima informacija. Sujungimo operacija susieja skirtingų lentelių eilutes pagal bendrą raktą, taip galima išgauti duomenis, išsiskirsčiusius keliose lentelėse, ir sujungti juos į vieną rezultatų rinkinį – šiuo atveju medžių rūšį ( ), lapo formą ( ) ir medieną ( ).

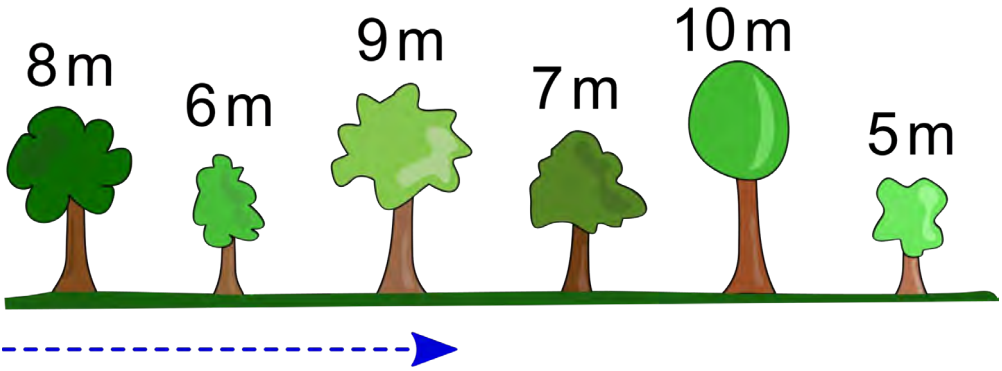
Informatikų sukurta SQL kalba (angl. *Structured Query Language*), naudojama duomenų bazių užklausoms formuluoti ir kitoms operacijoms duomenų bazėje atlikti.

27. Užtvankos statyba 2025-CY-01

Bebrai statomai užtvankai nori nukirsti keletą medžių. Jie pradeda darbuotis iš kairės ir laikosi dviejų paprastų taisyklių, kad išsirinktų kitą medį:

1 taisyklė: kitas medis turi būti toliau nuo pradžios nei ankstesnis.
2 taisyklė: kitas medis turi būti žemesnis nei ankstesnis.

Pavyzdžiui, jei jie nusprendžia pirmiausia nugraužti antrąjį medį (6 m), po jo galės kirsti šeštąjį medį (5 m) ir gaus bendrą 11 m nukirstų rąstų ilgį.



Koks galimas didžiausias bendras aukštis (m) medžių, kuriuos bebrai gali nukirsti?

Teisingas atsakymas: 21.

Paaiškinimas

Reikia rasti mažėjantį medžių aukščių posekį (iš sekos išmetus kažkuriuos narius, gaunama nauja seka, kuri vadinama senosios sekos posekiu), kuris duoda didžiausią medžių aukščių sumą. Taigi posekis yra elementų, kurie pasirodo ta pačia tvarka kaip ir pradinėje sekoje, dalis. Nedidėjantis posekis yra toks, kai kiekvienas elementas yra mažesnis arba lygus ankstesniajam.

Pavyzdžiui, medžių aukščiai yra:

8, 6, 9, 7, 10, 5

Pradėkite nuo medžio, kurio numeris 1.

Pridėkite jo aukštį prie bendros sumos ir laikykite jį ankstesniu medžiu.

Tęskite su medžiu numeris 2.

Jei medis yra aukštesnis už ankstesnį medį, praleiskite jį ir pereikite prie kito medžio.

Jei jis yra lygus arba žemesnis, pridėkite jo aukštį prie sumos ir laikykite jį ankstesniu medžiu.

Tęskite su visais likusiais medžiais.

Kai visi medžiai bus aptarti, pradėkite vėl nuo medžio, kurio numeris 1, bet šį kartą praleiskite medį, kurio numeris 2, ir pakartokite tą pačią procedūrą.

Kai baigsite su visais medžiais, turėsite visus galimus posekius, prasidedančius 8 (pirmasis medis).

Pakartokite visą procesą, pradėdami nuo medžio, kurio numeris 2, tada nuo medžio, kurios numeris 3, kol bus surasti visi posekiai.

Posekio numeris	Posekis	Bendras ilgis (ar aukštis?)
1	8, 6, 5	19
2	8, 7, 5	20
3	8, 5	13
4	6, 5	11
5	9, 7, 5	21
6	9, 5	14
7	7, 5	12
8	10, 5	15

Penktasis posekis (9, 7, 5) užtikrina, kad kiekvieno kito medžio aukštis neviršija ankstesnio medžio aukščio, o bendras aukštis maksimalus (21 metras).

Tai informatika!

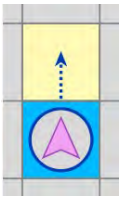
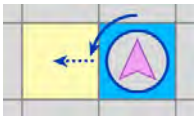
Algoritmai yra tarsi žingsnis po žingsnio vykdomos komandos, kurios sprendžia informatikos uždavinį.

Įsivaizduokite, kad turite mėgstamą sausainių receptą. Recepte nurodyta, kokių ingredientų reikia ir kokius veiksmus reikia atlikti, kad pagamintumėte skanius sausainius. Algoritmas iš esmės daro tą patį.

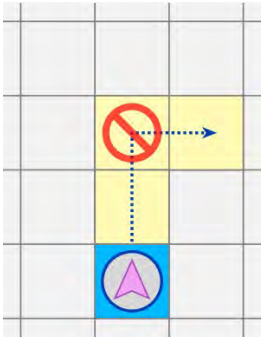
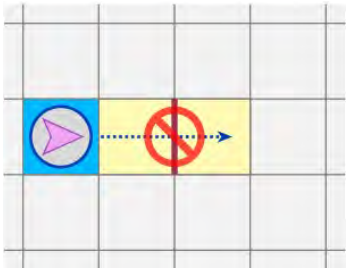
Pagrindinis šio uždavinio tikslas – sukurti ir vykdyti algoritmą, kuris pagal nustatytus apribojimus rastų optimalios medžių kirtimo sekos aukščių sumą.

28. Robotas Kairys 2025-DE-09b

Robotas Kairys  juda tinklelio langeliais. Jis gali atlikti tik tokius veiksmus:

Žengti pirmyn į kitą langelį	Pasisukti kairėn ir žengti pirmyn į kitą langelį
	

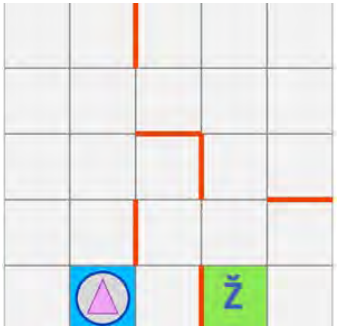
Pavyzdžiuose pavaizduoti atvejai, kokių veiksmų Kairys negali atlikti:

Žengti dešinėn negalima	Jei yra siena, Kairys negali jos peržengti
	

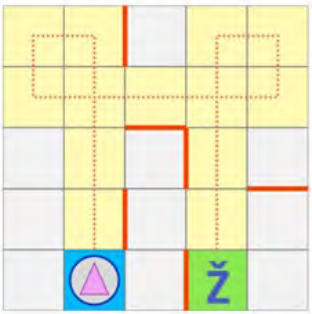
Iš pradžių Kairys yra mėlyname langelyje. Padėkite robotui pasiekti žalią langelį (Ž).

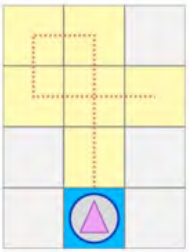
Keliaudamas Kairys turi aplankyti kuo mažiau tinklelio langelių.

Pažymėkite visus tinklelio langelius, kuriuose Kairys apsilankys, kol pasieks žalią langelį.



Paaiškinimas





Kairėje pavaizduota, kaip Kairys pasiekia tikslą pasisukdamas tik kairėn. Dešinėje pavaizduoti langeliai, kuriuos reikia pasirinkti, kad pasikeistų judėjimo kryptis dešinėn.
Sienų padėtis neleidžia pasiekti tikslo kitaip, aplankant mažesnį ar tą patį skaičių langelių.

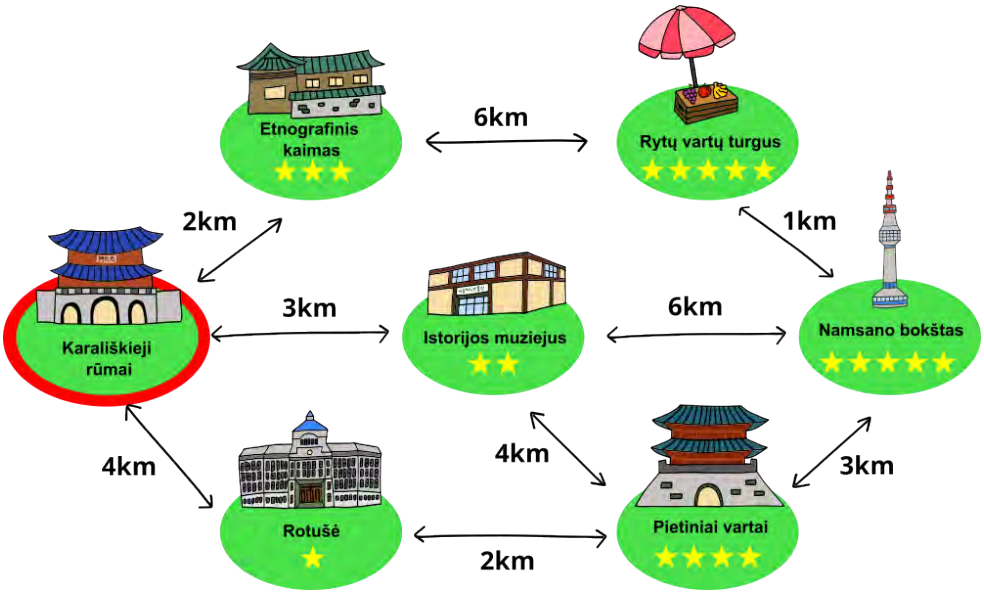
Tai informatika!

Roboto judėjimas atrodo labai ribotas. Jei Kairys galėtų pasukti į dešinę ir galbūt netgi lipti per sienas, jo gyvenimas tinklelyje būtų daug lengvesnis. Tam jam reikėtų sudėtingesnio komandų rinkinio. (Roboto veiksmus valdo atitinkamos jo programinės įrangos komandos.)
Bet ar tai tikrai būtina? Pavyzdžiui, Kairys galėtų pasukti į dešinę, tris kartus iš eilės pasukdamas į kairę. Tiesiog reikia atsikratyti taisyklės, kad robotas po vieno posūkio turi judėti per vieną langelį į priekį. Tada jis galėtų judėti visomis kryptimis. O vietoj to, kad liptų per sieną, jis galėtų tiesiog ją apeiti (jei įmanoma). Tai reiškia, kad ir sumažintas komandų rinkinys gali būti tinkamas. Norėdami įgyvendinti sudėtingesnį, rečiau pasitaikantį atvejį, programuotojai gali sukurti mažesnius programos modulius, kurie sujungia pagrindines komandas į sudėtingesnes.
Informatikoje labiausiai paplitę būtent šie du procesorių komandų rinkinių projektavimo būdai: kai kurie procesoriai yra sudėtingų komandų rinkinio kompiuteriai (angl. *complex instruction set computer*, CISC), kiti – sumažinto komandų rinkinio kompiuteriai (angl. *reduced instruction set computer*, RISC). CISC procesoriai paprastai turi daug įvairių komandų, kurios gali būti labai galingos (pavyzdžiui, perlipti per sieną). RISC procesoriai turi tik tikrai ir dažnai naudojamas komandas, su galimybe naudoti paprogrames rečiau atliekamoms operacijoms. Abiejų tipų architektūros turi savo privalumų ir trūkumų.

29. Ekskursijų autobusai 2025-KR-06

Seule turistams siūloma miesto įžymybes lankyti ekskursijų autobusais. Ekskursijų autobusai jungia pagrindines lankytinas Seulo vietas. Pateiktame paveiksle parodytos pagrindinės lankytinos vietos, atstumai tarp jų, o žvaigždutės rodo jų populiarumą.

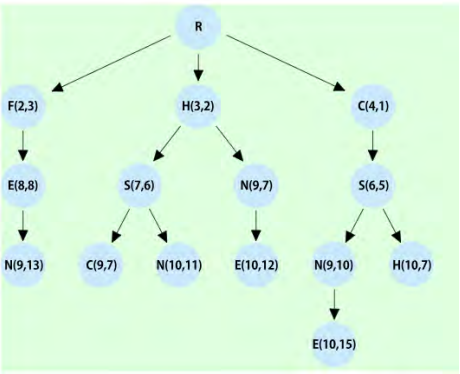
Apsilankiusi Karališkuosiuose rūmuose, Eglė nori važiuoti ekskursijų autobusu, tačiau ji turi transporto kortelę, kuri leidžia nukeliauti tik iki 10 kilometrų. Sudaryk maršrutą turistinėms vietoms aplankyti taip, kad Eglė galėtų surinkti kuo daugiau populiarumo žvaigždučių. Kiekvieną lankytiną vietą galima aplankyti tik vieną kartą.



Spustelėk lankytinas vietas iš eilės, kad suplanuotum geriausią maršrutą. Vieną vietovę galima aplankyti tik vieną kartą.

Teisingas atsakymas:
Karališkieji rūmai (Royal Palace) → Rotušė (City Hall) → Pietiniai vartai (South Gate) → Namsano bokštas (Namsan Tower) → Rytų vartų turgus (East Gate Market)

Paaiškinimas
Vienas iš būdų išspręsti uždavinį – naudoti iš mazgų sudarytą diagramą -- grafą. Diagramos viršuje yra pradžios mazgas, pažymėtas pirmąja raide R, reiškiančia Karališkuosius rūmus (*Royal Palace*). Pradedant nuo R, galima nubraižyti kiekvieną galimą kelią, kurio bendras ilgis neviršija 10 km.
Kiekviena grafo viršūnė reiškia lankytiną vietą, o joje yra informacija:
vietos pirmoji raidė,
bendras atstumas nuo pradžios,
bendras iki tol surinktų žvaigždučių skaičius.
Pavyzdžiui, vienas galimas maršrutas yra aplankyti Etnografinį kaimą (Folk Village), kuris yra 2 km nuo pradžios ir turi 3 žvaigždutes. Šis mazgas pažymėtas F (2, 3). Iš ten galima keliauti į Rytų vartų turgų (East Gate Market E). Tuomet bendras atstumas tampa 8 km, o surinktų žvaigždučių skaičius padidėja iki 8. Šis mazgas pažymėtas E (8, 8). Kitas pasirinkimas būtų Namsano bokštas (Namsan Tower N), kurį aplankius bendras atstumas tampa 9 km, o bendras žvaigždučių skaičius – 13, todėl jis pažymėtas N (9, 13).

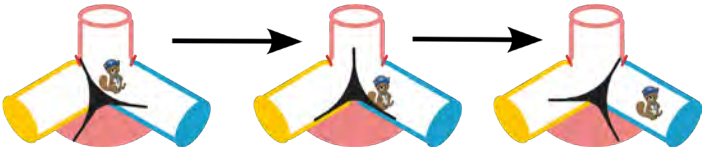


Norint rasti geriausią kelią, reikia išnagrinėti visų galiojančių kelių galinius mazgus ir pasirinkti tą, kuris turi didžiausią žvaigždučių skaičių, neviršijant 10 km ribos. Diagramoje mazgas su didžiausiu žvaigždučių skaičiumi yra E(10, 15).
Jį atitinka kelias: Karališkieji rūmai (Royal Palace) → Rotušė (City Hall) → Pietiniai vartai (South Gate) → Namsano bokštas (Namsan Tower) → Rytų vartų turgus (East Gate Market). Arba, naudojant pirmąsias raides: R → C (4, 1) → S (6, 5) → N (9, 10) → E (10, 15).

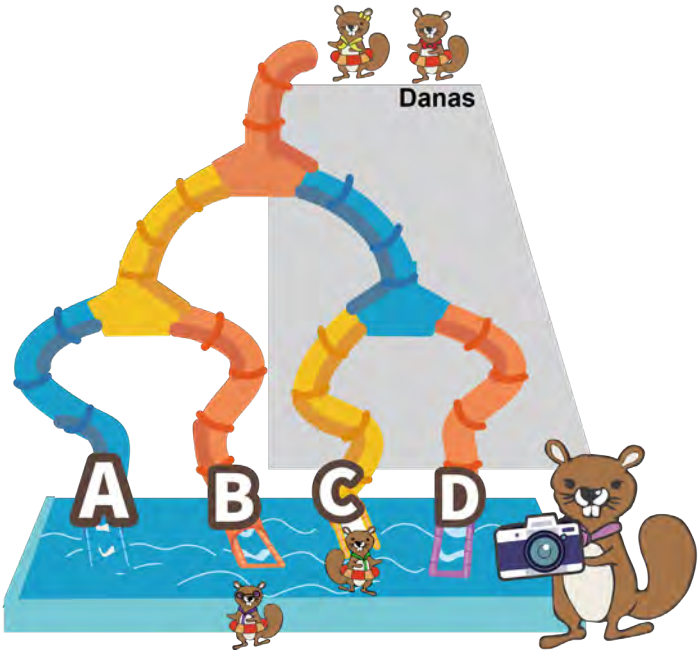
Tai informatika!
Optimizavimo problemos susijusios su sprendimų priėmimu, siekiant padidinti efektyvumą (pvz., rezultatą ar pelną), turint ribotus išteklius (pvz., laiką ar pinigus). Grafų modeliavimas leidžia struktūriškai analizuoti elementus (lankytinas vietas, kelius ir pan.), kad būtų galima sudaryti optimalius maršrutus. Šis metodas taikomas sprendžiant realias problemas – turizmo maršrutų planavimo, logistikos, transporto sistemų.
Uždavinys pristato pagrindinius principus, padedančius suprasti tiek išteklių valdymą, tiek maršrutų optimizavimą.

30. Vandens čiuožykla 2025-TW-02

Bebrų vandens parke yra įspūdinga dinaminė vandens čiuožykla. Diagramoje parodyta, kaip ji veikia: kiekviename išsišakojime čiuožimo kryptis valdoma mechanizmu – kai tik bebriukas pračiuožia pro išsišakojimą, čiuožimo kryptis pasikeičia.



Toliau paveiksle matosi, kaip bebriukas Danas ruošiasi čiuožti. Bebrė mama norėtų žinoti, kur iščiuoš Danas, kad galėtų jį nufotografuoti. Mama jau matė, kad vienas bebriukas iščiuožė pro angą B, tuomet kitas – pro angą C. Dano eilė čiuožti bus po dar vieno bebriuko.



Pro kurią čiuožyklos angą iščiuoš Danas?

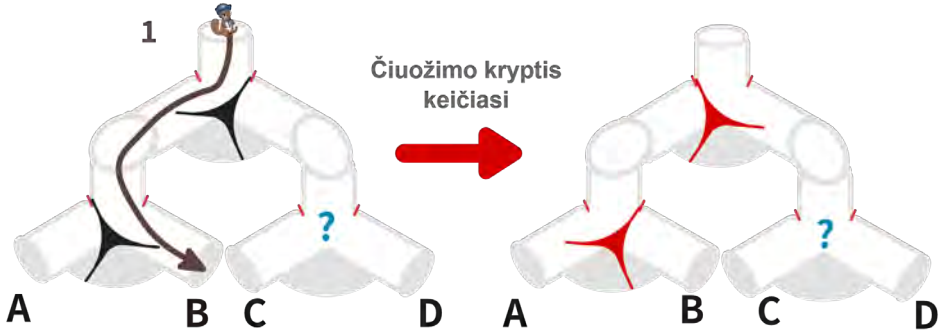
Pro angą A
Pro angą B
Pro angą C
Pro angą D

Teisingas atsakymas: D.

Paaiškinimas

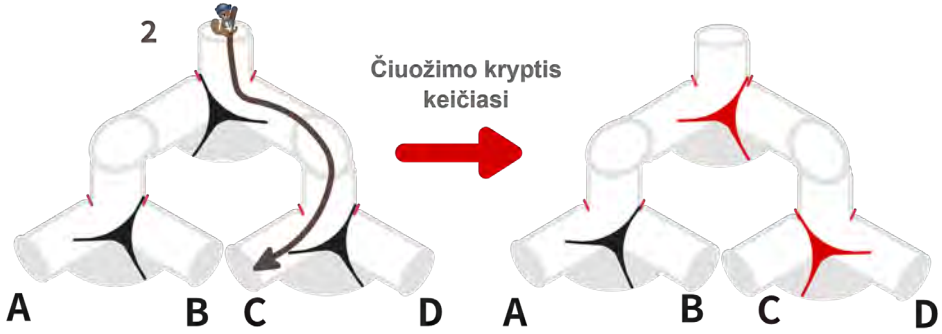
Kadangi čiuožimo kryptis išsišakojime pasikeičia kaskart, kai pro jį pračiuožia bebriukas, galima nustatyti, kokios čiuožimo kryptys yra po to, kai čiuožykla pračiuožė pirmieji du bebriukai.

Kaip parodyta paveikslėlyje, pirmasis bebriukas iščiuožė pro angą B – iš to galima padaryti išvadą, kad pirmame išsišakojime jis čiuožė į kairę, o antrame – į dešinę. Po to čiuožimo kryptis šiuose dviejuose išsišakojimuose pasikeitė: pirmas išsišakojimas kreips į dešinę, o antras (apačioje kairėje) – į kairę. Kol kas dar nežinoma, kokia čiuožimo kryptis yra apatiniame dešiniajame išsišakojime.

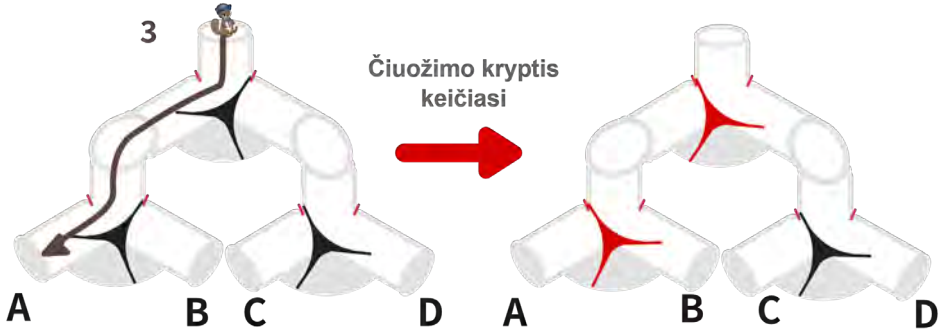


Kitas bebriukas, kaip parodyta kitame paveiksle, iščiuožė pro angą C. Iš to tampa žinoma, kad apatinis dešinysis išsišakojimas bebriuką nukreipė į kairę, o tada čiuožimo kryptis pasikeitė į dešinę.

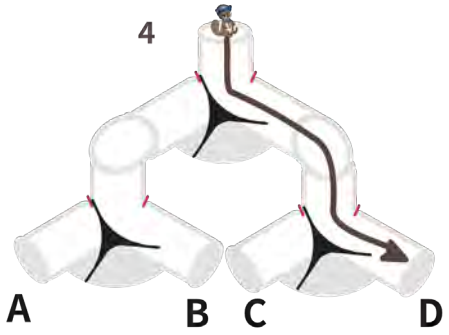
Dabar jau žinomos čiuožimo kryptys visuose išsišakojimuose, tad galima nustatyti,



kaip čiuoš bebriukas, esantis eilėje prieš Daną, o tuomet – ir vėl pasikeitusias kryptis kai kuriuose išsišakojimuose, kaip parodyta paveiksle.



Lygiai taip pat galima išsiaiškinti, kaip čiuoš Danas, ir kad jis iščiuoš pro angą D.



Tai informatika!

Šiame uždavinyje čiuožimo kryptis kiekviename išsišakojime nulemia, pro kurią angą iščiuoš bebriukas. Panašiai veikia sąlyginiai sakiniai programos tekste. Įprasta sąlyginio sakinio struktūra atrodo taip: „*Jei* sąlyga tenkinama, daryk tai; *kitaip* – daryk kažką kita“. Šiame uždavinyje išsišakojimų mechanizmo veikimą galima apibūdinti taip: „*Jei* mechanizmas kreipia į kairę, bebriukas pasuka į kairę, o mechanizmas persijungia į dešinę; *kitaip* – bebriukas pasuka į dešinę, o mechanizmas persijungia į kairę“.

Bebriukui pračiuožiant kiekvieną išsišakojimą, kartojamas tas pats procesas. Bebriukas čiuožia esama išsišakojimo kryptimi, o tada kryptis pasikeičia į priešingą. Reikia įsiminti esamą kiekvieno išsišakojimo kryptį, o tokią informaciją programavime galima aprašyti kaip kintamąjį. Pavyzdžiui, 0 gali reikšti kairę kryptį, o 1 – dešinę. Bebriukui čiuožiant čiuožykla, programa galėtų tikrinti kiekvieno kintamojo reikšmę (išsišakojimo kryptį), nuspręsti, kur nukreipti bebriuką, ir pakeisti kintamojo reikšmę.

Toks mechanizmas panašus ir į skaitmeninės grandinės trigerį – atminties elementą, kuris saugo 1 bitą duomenų ir persijunginėja tarp dviejų būsenų. Kiekvienas išsišakojimas veikia kaip trigeris, keisdamas savo būseną kaskart, kai yra panaudojamas.

Norint išsiaiškinti, kur iščiuoš bebriukas, reikia pažingsniui imituoti kiekvieną sprendimą. Tai panašu į programos teksto sekimą, kuomet programos veiksmai tikrinami po eilutę, norint pamatyti, kokį rezultatą ji duoda.

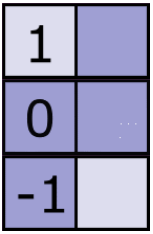
31. Šviesio žemėlapis 2025-DE-06

Sandra domisi vizualiojo meno struktūriniais aspektais. Ji kuria vaizdų šviesio žemėlapius, kurie yra apsupti baltais pikseliais. Šviesio žemėlapyje kiekvienas vaizdo pikselis yra žymimas šviesio skaičiumi.
Pikselio šviesio skaičius nustatomas taip:

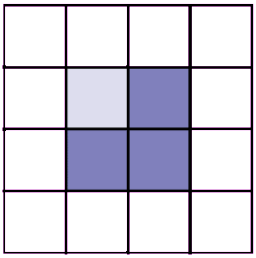
1 – jei pikselis yra šviesesnis už dešinėje esantį pikselį

0 – jei pikselis ir jo kaimynas iš dešinės yra vienodo šviesio

-1 – jei pikselis yra tamsesnis už dešinėje esantį pikselį



Pavyzdys



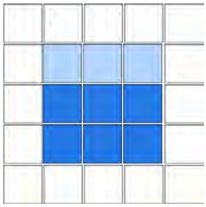
Paveikslėlį juosia balti pikseliai

1	-1
0	-1

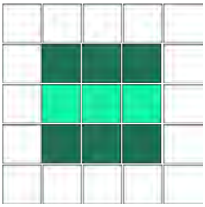
Šviesio žemėlapis

Paveikslėliai gali turėti identiškus šviesio žemėlapius, net jei jie atrodo skirtingi. Kurio paveikslėlio šviesio žemėlapis skiriasi nuo kitų?

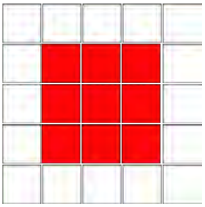
A



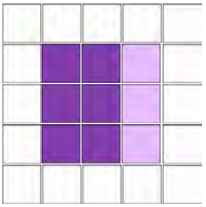
B



C

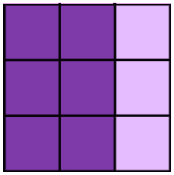


D



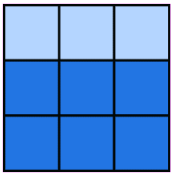
Teisingas atsakymas: D.

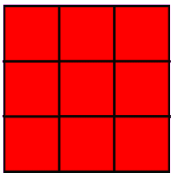
Paaiškinimas
Šio paveikslėlio šviesio žemėlapis skiriasi nuo likusių trijų paveikslėlių:

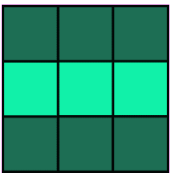


0	-1	-1
0	-1	-1
0	-1	-1

Šių paveikslėlių šviesio žemėlapiai vienodi:







0	0	-1
0	0	-1
0	0	-1

Paveikslėliai yra apjuosti baltais pikseliais, o visi vidiniai pikseliai yra tamsesni už baltus, todėl visų šviesio žemėlapių dešinieji stulpeliai yra vienodi (-1).
Ieškant teisingo sprendimo galima apskaičiuoti visus keturis šviesio žemėlapius ir juos palyginti. Tačiau yra greitesnis būdas. Reikia atkreipti dėmesį, kad šviesio žemėlapis fiksuoja šviesio pokyčius horizontaliąja kryptimi. Trys paveikslai turi vienodo šviesio pikselius kiekvienoje eilutėje, o tai reiškia, kad jie turi turėti tą patį šviesio žemėlapį.

Tai informatika!

Kompiuteriuose vaizdai gali būti pateikiami įvairiais būdais, priklausomai nuo naudojamo spalvų modelio. Dvi dažniausios rūšys yra pilkoji skalė, kai kiekvienas pikselis turi vieną šviesio vertę, ir RGB (raudonos, žalios, mėlynos) spalvų vaizdai, kur kiekvienas pikselis yra trijų verčių, atitinkančių raudonos, žalios ir mėlynos šviesos intensyvumą, sankaupa.
Šiame uždavinyje vaizdai yra spalvoti. Tačiau analizuojant šviesį programa paprastai konvertuoja šiuos RGB vaizdus į pilkosios skalės vaizdus. Šis žingsnis supaprastina duomenis, sumažindamas kiekvieną spalvoto pikselio šviesį iki vienos vertės.
Šviesio žemėlapio kūrimas – vienas iš konvoliucijos taikymo atvejų. Konvoliucija apima mažos matricos (filtro) slinkimą per vaizdą, siekiant sukurti savybių žemėlapį. Šis žemėlapis parodo konkrečias vaizdo struktūrines savybes, pavyzdžiui, kraštus, kampus ar vienodos spalvos sritis, todėl kompiuteriui lengviau interpretuoti vaizdo informaciją.
Konvoliuojamieji neuroniniai tinklai (angl. *convolutional neural networks*, CNN) remiasi šia konvoliuojamojo filtravimo koncepcija. Jie mokosi kurti savo sudėtingesnius filtrus. Gautus išsamius savybių žemėlapius galima naudoti objektams vaizduose rasti (pvz., navikams aptikti medicininiu skeneriu).

32. Advento žvakės 2025-DE-08

Advento tradicija – uždegti žvakes keturis sekmadienius prieš Kalėdas: vieną žvakę pirmą sekmadienį, dvi žvakes antrą sekmadienį ir t. t. Kristijonas labai mėgsta šią tradiciją, jo keturios žvakės yra vienodo ilgio (aukščio). Kristijonas būtų laimingas, jei po paskutinio sekmadienio visos žvakės vis dar būtų vienodo aukščio. Tam jis turėtų kiekvieną žvakę uždegti vienodą skaičių kartų.
Deja, Kristijonas negali rasti būdo, kaip tai padaryti. Jei Adventas apimtų tik tris sekmadienius, tai pavyktų, kiekvieną žvakę uždegus po du kartus:

1-as sekmadienis	
2-as sekmadienis	
3-as sekmadienis	

O jei būtų penki sekmadieniai?
Raskite būdą, kaip padaryti Kristijoną laimingą, kiekvieną sekmadienį uždegdami tinkamas žvakes.
Penktą sekmadienį nėra pasirinkimo, visos penkios žvakės jau dega.

1-as sekmadienis	
2-as sekmadienis	
3-as sekmadienis	
4-as sekmadienis	
5-as sekmadienis	

Paaiškinimas
Štai vienas iš būdų, kaip uždegti žvakes penkis sekmadienius, kad kiekviena žvakė būtų uždegta tris kartus:

1-as sekmadienis	
2-as sekmadienis	
3-as sekmadienis	
4-as sekmadienis	
5-as sekmadienis	

Tai galima pasiekti įvairiais būdais. Visiems teisingiems atsakymams galioja savybė, kurią čia aptarsime. Sugrupuokime sekmadienius poromis, kurių skaičiai sudaro visą sekmadienių skaičių; 5 sekmadieniams tai yra poros 1, 4 ir 2, 3. Kiekvienai porai dvi žvakių grupės, kurios uždegamos kiekvieną šios poros sekmadienį, turi būti nesusijusios. (Jei kiekvieną žvakę laikysime bitu, kur 1 = uždegta, o 0 = neuždegta, dvi bitų sekos poros sekmadieniams turi būti viena kitos bitų papildiniais). Tada per du poros sekmadienius kiekviena žvakė uždegama vieną kartą. Be to, paskutinį sekmadienį kiekviena žvakė uždegama dar kartą. Todėl iš viso visos žvakės uždegamos vienodą skaičių kartų.

Tai informatika!

Iš pradžių šis uždavinys atrodo gana sudėtinga matematikos problema. Egzistuoja matematinis įrodymas, kad Kristijono žvakių įžiebimo problema gali būti išspręsta, kai sekmadienių skaičius yra nelyginis. Jei norite žinoti, kaip tikslingai uždegti žvakes, turite aprašyti algoritmą, pagal kurį uždegamos žvakės, arba išvardyti visus galimus uždegimo būdus.
Tarkime, n yra (nelyginis) sekmadienių skaičius; n-qjį sekmadienį dega visos n žvakės.

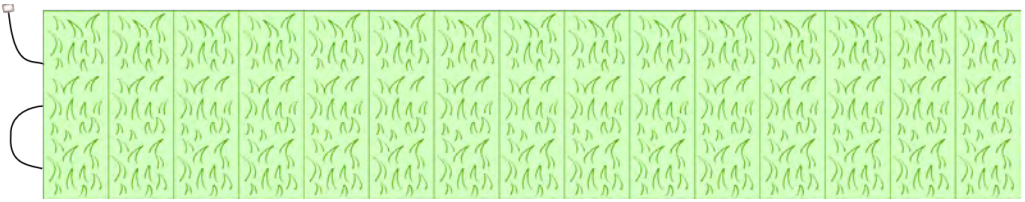
1. n-qjį sekmadieni uždegti visas n žvakių.
2. for $i = 1$ to $(n–1)/2$ atlikti a ir b žingsnius.
   a. uždegti pirmas i žvakes i-qjį sekmadienį.
   b. paskutines $n–i$ žvakes n-i sekmadienį.

Atkreipkite dėmesį, kad šio algoritmo 2a žingsnis gali būti atliktas keliais skirtingais būdais, vis tiek bus išsprendžiamas žvakių įžiebimo uždavinys.
Viena iš svarbiausių užduočių informatikos mokslininkams yra kurti algoritmus problemoms spręsti. Jei algoritmas gali būti pagrįstas patikimu matematiniu modeliu, lengviau patvirtinti norimas algoritmo savybes.
Galima įrodyti, kad čia pateiktas algoritmas veikia esant bet kuriam nelyginiam sekmadienių skaičiui.
Vis tik Advento tradicija apima lyginį sekmadienių skaičių.

33. Dingęs aitvaras 2025-GR-02

Bebras pametė savo aitvarą aukštos žolės pievoje! Ilga aitvaro virvelė susipainiojo, tad dabar sunku aitvarą surasti.
Pieva padalijame į tinklelį iš 3 eilučių ir 15 stulpelių. Bet kurį pievos stulpelį galima apieškoti atskirai ir taip sužinoti, kiek kartų aitvaro virvelė jį kerta.

Bandydami išsiaiškinkite, kaip tokia paieška tinklelyje veikia.



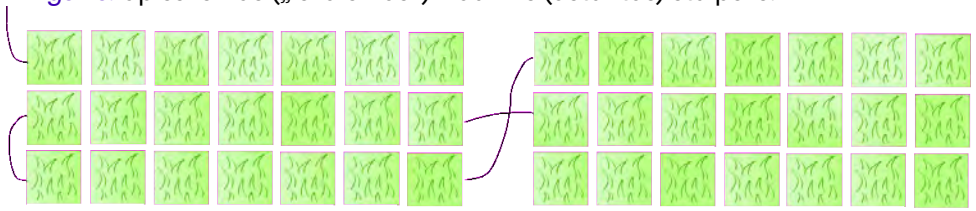
Kiek mažiausiai pievos stulpelių reikės apieškoti, kad tikrai rastumėte aitvarą?

Teisingas atsakymas: 4.

Paaiškinimas

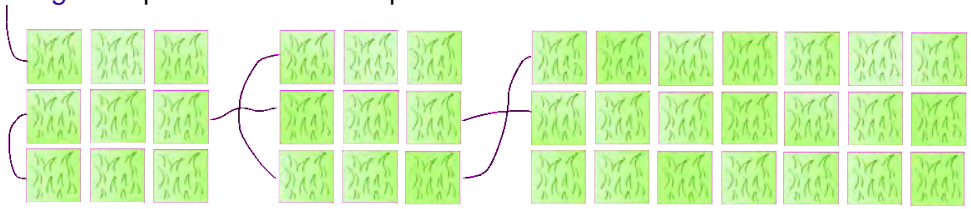
Pasitelkus loginį mąstymą ir strateginę paiešką, galima užtikrintai sakyti, kad pavyks rasti aitvarą apieškojus 4 stulpelius. Apieškojus mažiau nei 4 stulpelius galima neturėti pakankamai duomenų, kad paieška būtų garantuotai sėkminga.
Toliau pateikiamas uždavinio sprendimas pažingsniui.

1 žingsnis: apieškomas („išvalomas“) vidurinis (aštuntas) stulpelis.



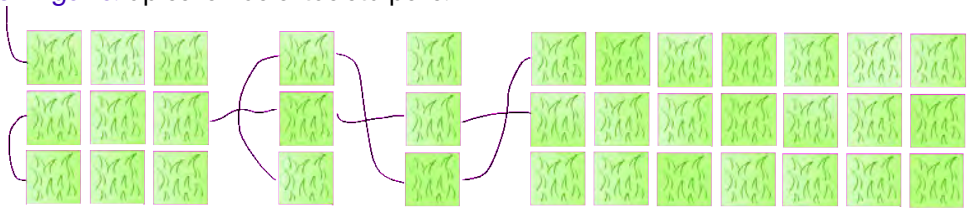
Apieškojus vidurinį stulpelį, kiek įmanoma sumažinamas didžiausias plotas, kurį dar reikia apieškoti, nesvarbu, kurioje pievos pusėje būtų aitvaras. Kitaip tariant, tokiu būdu užtikrinama, kad apieškojus vieną stulpelį likęs paieškos plotas sumažinamas kuo didesne dalimi, t. y., didžiausias likusių stulpelių skaičius sumažinamas perpus. Norint išsiaiškinti, ar aitvaras yra dešinėje, ar kairėje pievos pusėje, reikia atkreipti dėmesį, kiek kartų aitvaro virvelė kerta apieškotą stulpelį. Šiuo atveju paaiškėjo, kad per 8-tą stulpelį virvelė nuėjo į dešinę pusę, bet paskui grįžo į kairę. Vadinasi, virvelės galas ir aitvaras turi būti kairėje pusėje.

2 žingsnis: apieškomas 4-tas stulpelis.

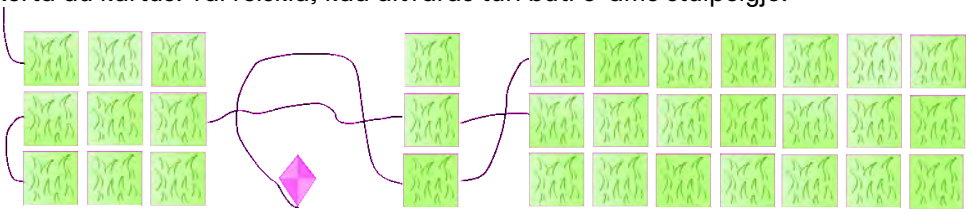


Kaip ir ankstesniame žingsnyje, siekiant optimizuoti paiešką, čia ir vėl apieškomas vidurinis (ketvirtas) likusio paieškos ploto stulpelis. Šįkart nustatoma, jog virvelė kerta stulpelį vieną kartą ir dar sudaro jame kilpą. Kadangi virvelė kerta stulpelį tik kartą, t. y. nueina į dešinę pusę, bet negrįžta į kairę, tai aitvaras turi būti 5–7 stulpeliuose.

3 žingsnis: apieškomas 6-tas stulpelis.



Toliau taikant vidurinio stulpelio apieškojimo metodą paaiškėja, kad virvelė šį stulpelį kerta du kartus. Tai reiškia, kad aitvaras turi būti 5-ame stulpelyje.

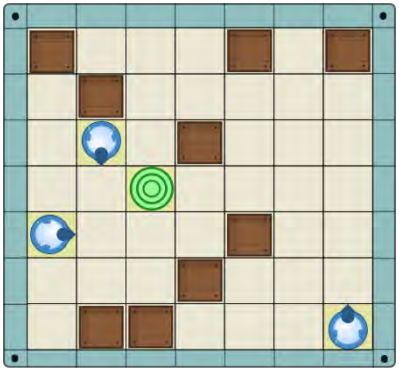


Tai informatika!

Šiuo uždaviniu parodoma, kaip algoritmavimo metodai, pavyzdžiui, dvejetainė paieška, gali palengvinti uždavinių sprendimą. Dvejetainė paieška – tai toks uždavinio sprendimo būdas, kai uždavinys vis dalijamas į dvi lygias dalis, šitaip sumažinant galimų variantų skaičių. Užuot tikrinus kiekvieną variantą, pasirenkamas vidurinysis elementas – šitaip sprendimą galima rasti efektyviau ir su mažiau žingsnių. Toks metodas, įdiegtas kompiuterių programose, leidžia pagreitinti uždavinių sprendimą, nes išvengiama nereikalingų tikrinimų. Informatikoje dažnai tenka ieškoti veiksmingiausio metodo sprendimui, susidedančiam iš kuo mažiau žingsnių, ypač, kai informacija nėra žinoma arba žinoma tik iš dalies.

34. Robotų labirintas 2025-HU-01b

Miglė mėgsta žaisti kompiuterinius labirintų žaidimus. Šiuose žaidimuose yra lenta su kliūtimis  , o tikslas – nuvesti robotą  per labirintą iki tikslo  . Robotui komandas galima duoti paprasta programavimo kalba. Robotas vykdys komandas iš eilės vieną po kitos, iš viršaus į apačią, ir sustos, kai pasieks tikslą.



Komandos:

- eik priekin iki tikslo
- eik priekin iki kliūties
- pasisuk 90 laipsnių į dešinę
- pasisuk 90 laipsnių į kairę

Miglė nori sukurti kuo trumpesnį algoritmą, kurį vykdydami visi trys robotai pasieks tikslą. Robotai startuoja skirtingose vietose ir žiūri skirtingomis kryptimis, bet visi turi pasiekti tą patį tikslą  .

Padėk Miglei! Nutempk ir sudėliok komandas taip, kad vykdydami tą patį algoritmą visi trys robotai pasiektų tikslą. Daryk prielaidą, kad robotai niekada nesusidurs. Kelk komandas į tuščias vietas algoritme. Komandos galima nepanaudoti arba galima naudoti tiek kartų, kiek reikia. Nebūtina užpildyti visų tuščių vietų algoritme.

- eik priekin iki kliūties
- pasisuk 90 laipsnių į kairę
- pasisuk 90 laipsnių į dešinę
- eik priekin iki tikslo

Paaiškinimas
Uždavinį galima išspręsti su šia komandų seka:

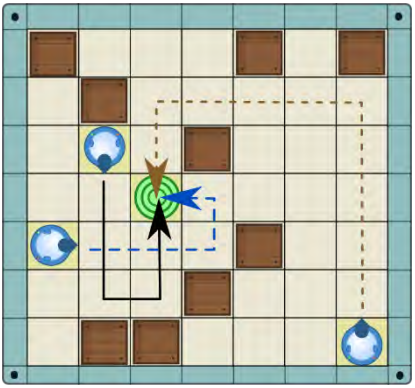
eik priekin iki kliūties

pasisuk 90 laipsnių į kairę

eik priekin iki kliūties

pasisuk 90 laipsnių į kairę

eik priekin iki tikslo



Robotai judės taip, kaip rodo rodyklės paveiksle:

Šiam labirintui nėra kito sprendimo.
Kodėl tai yra trumpiausias teisingas sprendimas?
Atkreipkite dėmesį į šiuos dalykus:

- tikslas visada yra kairėje visų 3 robotų pusėje (vadinasi, sprendime privalo būti posūkis į kairę);
- tikslas išlieka kairėje robotų pusėje ir po pirmos kliūties bei posūkio į kairę.

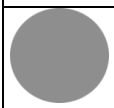
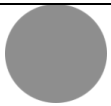
Todėl trumpiausias teisingas sprendimas turi apimti tik dvi komandas judėti pirmyn ir dvi komandas pasukti į kairę, tokia seka, kaip parodyta paveiksle. Likusi komanda judėti pirmyn užbaigs programą, o robotai pasieks tikslą.

Tai informatika!

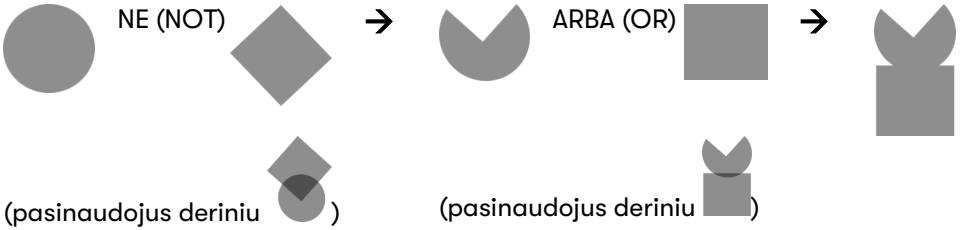
Šiame uždavinyje pateikiama supaprastinta programavimo kalba, kuri turi tik tris komandas. Nepaisant riboto rinkinio, šias komandas galima naudoti tiek kartų, kiek reikia, ir sukurti įvairaus sudėtingumo algoritmus.
Vienas iš pagrindinių programavimo konceptų yra komandų eilės tvarka – nuoseklus algoritmas (angl. *sequencing*). Nuoseklus algoritmas reiškia instrukcijų vykdymą vieną po kitos tam tikra tvarka. Tai paprasčiausias būdas valdyti programos eigą, užtikrinant, kad kiekvienas žingsnis būtų užbaigtas prieš pradedant kitą.
Nuoseklus algoritmas sudaro pagrindą sudėtingesnėms programavimo sąvokoms – ciklams ir sąlygoms. Pavyzdžiui, ciklai (arba kartojimo konstrukcijos) leidžia vykdyti seką kelis kartus, nerašant to paties kodo pakartotinai.
Instrukcija gali apimti reikšmės priskyrimą, įvesties duomenų nuskaitymą, išvesties rašymą arba sudėtingesnio instrukcijų bloko vykdymą.
Realiame gyvenime kūrybiškas sekų kūrimas ir jų derinimas leidžia keliems objektams vienu metu vykdyti algoritmą, kad užduotys būtų atliktos efektyviau ir padidėtų produktyvumas. Pavyzdžiui, kai klientas užsako kelis produktus iš skirtingų tiekėjų, tiekėjai gali paskirstyti skirtingus agentus tiems produktams pristatyti, laikydamiesi to paties darbo srauto, kad klientą pasiektų greičiau.

35. Figūrų logika 2025-IE-03

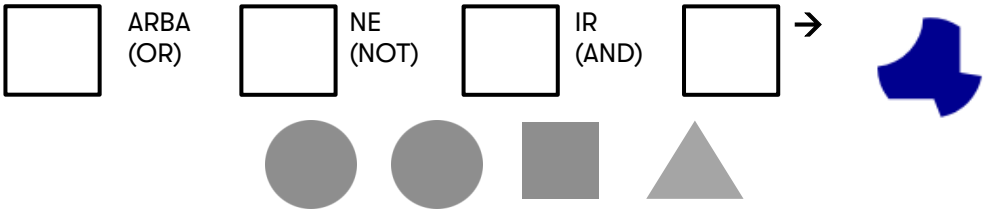
Loginės (Būlio) operacijos naudojamos ir kompiuterinėje grafikoje. Šios operacijos leidžia iš paprastesnių figūrų sukurti sudėtingesnes.
Trijų pagrindinių loginių (Būlio) operacijų – IR (AND), ARBA (OR) bei NE (NOT) – pavyzdžiai:

1-a figūra	2-a figūra	Figūrų derinys	Operacija		
			IR (AND)	ARBA (OR)	NE (NOT)
					
					

Panaudokime šias operacijas sudėtingesnėms figūroms sukurti.

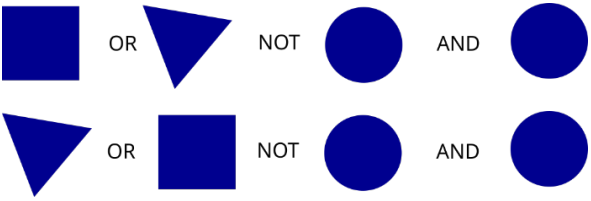


Turime keturias figūras ir tris pateiktas logines (Būlio) operacijas.
Nutempk figūras į teisingas vietas taip, kad pritaikius operacijas rezultatas sutaptų su pateiktu paveikslėliu.



Paaiškinimas

Šie atsakymai yra teisingi:



Norint išspręsti šį uždavinį, reikia galutinėje figūroje atpažinti pagrindinius požymius.

Pirmoji pateikta operacija yra OR, kuri sukurs galutinės figūros detalę sujungiant dvi figūras. Čia reikia naudoti kvadratą ir trikampį. Kadangi naudojama operacija OR, šių dviejų figūrų tvarka nesvarbi. OR rezultatas visais atvejais bus tas pats.

Kita pateikta operacija yra NOT. Tai reiškia, kad pašalinama figūros dalis. Matome, jog reikia atimti dalį iš mūsų figūros viršutinio kairiojo kampo. Kadangi ta dalis yra apvali, jai pašalinti naudojame apskritimą su operacija NOT.

Paskutinė pateikta operacija yra AND, ji palieka tik tas dalis, kurios yra bendros abiem figūroms. Galutiniam rezultatui gauti panaudojamas paskutinis apskritimas.

Tai informatika!

Būlio algebra jau seniai naudojama kompiuterinėje grafikoje. Operacijos OR (dviejų figūrų sąjunga), AND (dviejų figūrų sankirta) ir NOT (vienos figūros atėmimas iš kitos) leidžia iš paprastesnių figūrų kurti sudėtingesnes. Šios operacijos paprastai galimos tiek rastrinėse, tiek vektorinėse piešimo programose. Visgi jos geriau realizuotos vektorinėje grafikoje. Tai savo ruožtu paskatino visos algoritmų klasės, angliškai vadinamos *sweep line algorithms*, naudojimą, kad šios operacijos būtų efektyviai atliekamos su vektoriniais vaizdais.

Loginė reikšmė reiškia kažką, kas gali būti viena iš dviejų: „tiesa“ (TRUE) arba „netiesa“ (FALSE). Būlio logika atlieka svarbų vaidmenį aparatinėje įrangoje ir programavime. Pavyzdžiui, programavime Būlio algebra (loginiai teiginiai) naudojama kuriant sąlygas ciklams ar sąlygoms:

```
if points > 100:
print(“Congratulations”)
```

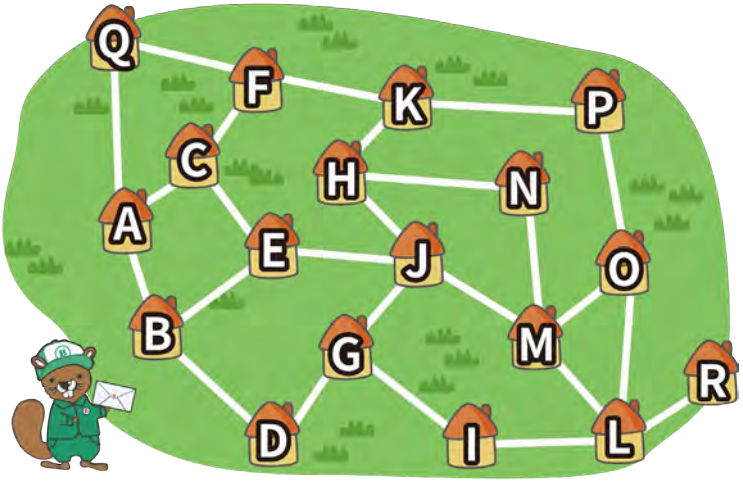
Loginis teiginys visada yra teisingas arba klaidingas; pavyzdžiui, „points > 100“, „Bobas yra vyresnis už Alisą“, „Lyja lietus“.

Loginės (Būlio algebros) operacijos AND, OR, NOT gali būti naudojamos sudėtingesniems teiginiams ar reiškiniams sudaryti, pavyzdžiui:

- „Lyja lietus“ AND „Namas yra geltonas“ – šis teiginys teisingas tik tada, jei vienu metu ir lyja, ir namas yra geltonas.
- „Lyja lietus“ OR „Namas yra geltonas“ – šis teiginys teisingas, jei lyja ir (arba) namas yra geltonas.
- NOT „Namas yra geltonas“ – šis teiginys teisingas, jei namas NĖRA geltonas.

36. Bebrų salos paštas 2025-TW-04

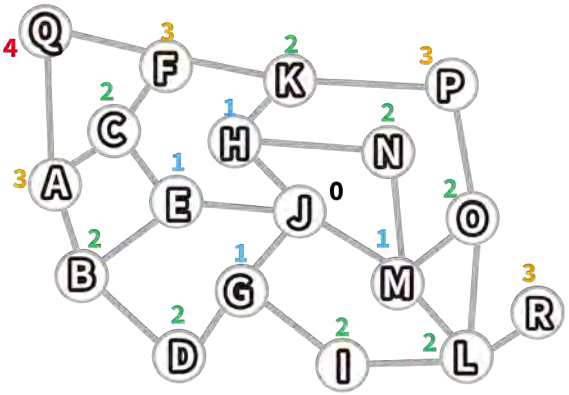
Bebrų saloje yra 18 kaimelių, kaip parodyta paveiksle. Kiekviename kaimelyje yra po kelis paštininkus. Kai iš kaimelio reikia išsiųsti pranešimą, arba kai kaimelis gauna pranešimą, paštininkai pristatys jį į visus tiesioginę jungtį turinčius kaimelius kitą dieną.



Pavyzdžiui, jei pranešimas išsiunčiamas iš kaimelio A, užtruks 1 dieną, kol jis pasieks kaimelius B, C ir Q; 2 dienas, kol pranešimas nukeliaus iki kaimelių D, E ir F; ir taip toliau, kol pranešimas bus pasiekęs visus kaimelius.

Kiek reikės dienų, kad iš kaimelio J išsiųstą pranešimą gautų visi kaimeliai?

Teisingas atsakymas: 4.



Paaiškinimas

Pradėję nuo kaimelio J, galime suskaičiuoti, kiek dienų reikės, kad pranešimas pasiektų kiekvieną kaimelį:

1 diena: pranešimą gauna kaimeliai E, G, H ir M;

2 diena: pranešimą gauna kaimeliai B, C, D, I, K, N, O ir L, o tiksliau:
- kaimeliai B ir C gauna pranešimą iš kaimelio E,
- kaimeliai D ir I gauna pranešimą iš kaimelio G,
- kaimeliai K ir N gauna pranešimą iš kaimelio H,
- kaimeliai N, O ir L gauna pranešimą iš kaimelio M;

3 diena: pranešimą gauna kaimeliai A, F, P ir R; šią dieną tik kaimelis Q dar negaus pranešimo.

4 diena: kaimelis Q gauna pranešimą iš kaimelių F ir A.

Iš viso užtruks 4 dienas, kol visi kaimeliai gaus pranešimą iš kaimelio J.

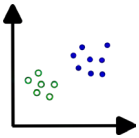
Tai informatika!

Šiame uždavinyje pranešimo pristatymo būdas yra panašus į paieškos į plotį algoritmo taikymą grafams, kai paieška vykdoma sistemiškai einant per grafo viršūnes. Paieškos į plotį algoritmas pradedamas ties tam tikra viršūne ir pirmiausia aplankomos visos jai gretimos viršūnės. Tuomet paieška plečiama į neaplankytas gretimas viršūnes jų išsidėstymo grafe tvarka. Paieškos į plotį esmė yra plėtimasis po vieną žingsnį, užtikrinant, kad pirmiausia aplankomos viršūnės, esančios arčiau pradinės viršūnės, o tuomet paieška plečiama į labiau nutolusias viršūnes.

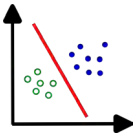
Praktiniai paieškos į plotį taikymo pavyzdžiai galėtų būti paieškos sistemos, žingsnis po žingsnio nuskaitančios tinklalapius per saitus; socialiniai tinklai, analizuojantys, kaip pranešimai plinta iš vieno naudotojo kitiems; medicinos įstaigos, sekančios žmonių kontaktus ir modeliuojančios užkrečiamųjų ligų plitimą.

Į šį panašus yra užpildymo algoritmas, dažniausiai naudojamas susijusių sričių nagrinėjimo ir išplėtimo uždaviniams spręsti. Pavyzdžiui, spalvinimo programoje spustelėjus viename taške, algoritmas nuspalvina visą susijusį plotą. Panašiai šis algoritmas taikomas ir siurblių-robotų programinėje įrangoje, norint, kad robotas aptiktų ir išsiurbtų visus susijusius kambarius, pradėjęs nuo savo stotelės. Užpildymo algoritmo pagrindas dažniausiai yra paieškos į plotį arba paieškos į gylį algoritmai – jie naudojami sistemingai ir išsamiai išnagrinėti visus gretimus susijusius elementus.

37. Sprendimo riba 2025-AT-02

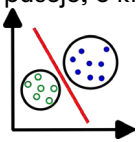


Informatikoje svarbus metodas yra sprendimo ribos radimas.

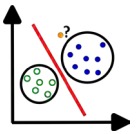


Sprendimo ribos tikslas – kuo geriau atskirti dvi grupes.

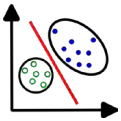
Būtina sąlyga: linija nubrėžiama taip, kad visi vienos grupės taškai būtų vienoje linijos pusėje, o kitos grupės taškai – kitoje pusėje.



Šiame pavyzdyje grupės atskirtos labai gerai.



Naujam taškui (pavyzdžiui, pažymėtam ženklu „?“), galima atlikti klasifikaciją.

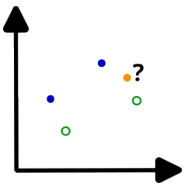


Taškas priskiriamas tai grupei, kuri yra toje pačioje linijos pusėje.

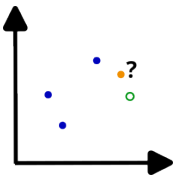
Norime priskirti tašką, pažymėtą „?“ ženklu, vienai iš grupių. Kitų keturių taškų priklausomybė grupėms dar nėra pažymėta spalvomis.

Galima nustatyti, kad, remiantis sprendimo riba, taškas, pažymėtas „?“, gali būti priskirtas vienai iš dviejų grupių tik vienu iš žemiau pateiktų variantų. Kuriuo?

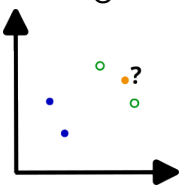
A



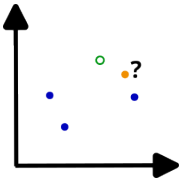
B



C

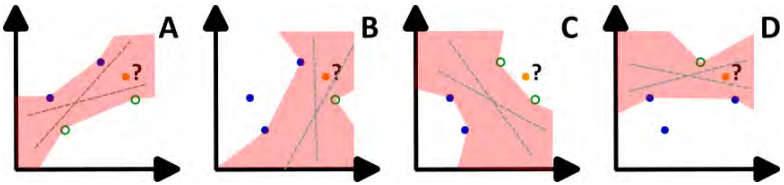


D



Teisingas atsakymas: C

Paaiškinimas



Žinome, kad sprendimo riba turėtų atskirti dvi taškų grupes pagal anksčiau pateiktą sąlygą. Visose keturiuose variantuose pažymėta raudona zona, rodanti, kur turi būti sprendimo riba, kad būtų įvykdyta ši sąlyga.
Kiekvienam atsakymo variantui parodyti du galimi sprendimo ribos variantai. A, B ir D atvejais taškas, pažymėtas „?“, taip pat yra šioje raudonoje zonoje. Tačiau neaišku, kurioje sprendimo ribos pusėje jis atsidurs.
Tik C variante taškas yra už raudonos zonos ribų. Todėl šiuo atveju galime aiškiai pasakyti, kurioje ribos pusėje jis yra.

Tai informatika!

Sprendimo riba yra svarbi informatikos idėja, ypač klasifikavimo užduočių atveju. Ji atskiria skirtingas duomenų rinkinio grupes (arba klases). Modelis išmoksta sprendimo ribą, kad galėtų nuspręsti, kuriai grupei priklauso naujas duomenų taškas.
Jei riba pasirinkta teisingai, modelis gali pateikti tikslias prognozes. Paprastų užduočių atveju riba yra tiesi linija. Sudėtingesnių duomenų atveju ji gali būti kreivė arba dar sudėtingesnė, priklausomai nuo algoritmo.
Tokie metodai – atraminių vektorių klasifikatorius (ang. *support vector machine*, SVM) ar neuroniniai tinklai – naudoja sprendimo ribą duomenų klasifikavimui. Sprendimo riba padeda suprasti, kaip modeliai priima sprendimus ir kaip jie tvarko naujus duomenis.

38. Akmuo, popierius, žirklės ir mainai 2025-BR-02

Aldona, Bronius ir Cezaris žaidžia kortų mainus. Žaidimo taisyklės:
• Akmuo nugali žirkles
• Žirklės nugali popierių
• Popierius nugali akmenį

Aldona, Bronius ir Cezaris atsisėda ir laiko kortas taip, kad visi galėtų jas matyti. Pirmiausia žaidėjai turi susitarti, kiek kartų jie keisis kortomis. Keitimasis kortomis – tai kortų mainai tarp dviejų žaidėjų. Tada jie turi nuspręsti, kurios žaidėjų poros kiekvieną kartą apsikeis kortomis.

Broniaus tikslas – nugalėti Cezarį. Kokią strategiją Bronius turėtų pasirinkti?

A) Bronius turi pasiekti, kad su Cezariu būtų atliktas nelyginis mainų skaičius.

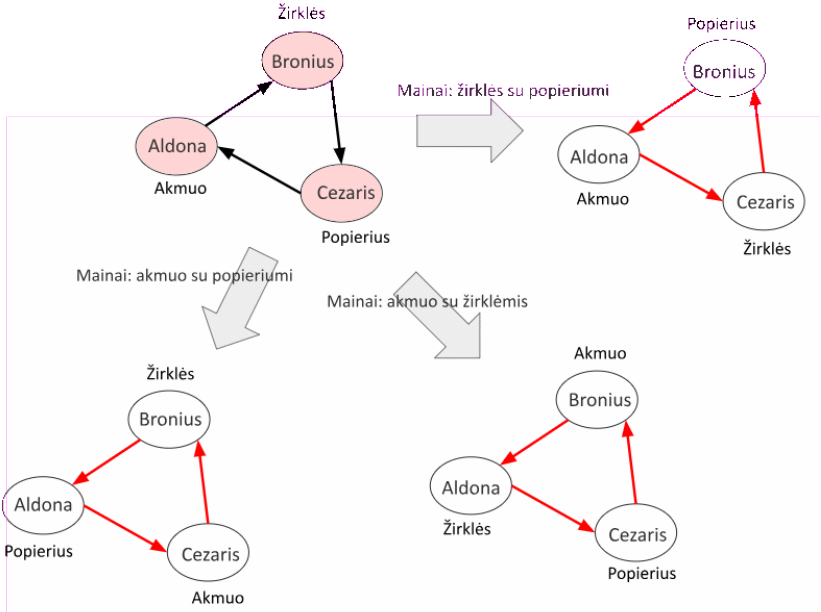
B) Nepriklausomai nuo to, kiek mainų žaidėjai nuspręs atlikti, Bronius niekada neturėtų keistis kortomis su Cezariu.

C) Nepriklausomai nuo to, kiek mainų žaidėjai nuspręs atlikti, Bronius visada turėtų keistis kortomis su Cezariu.

D) Broniui nesvarbu kokie vyks mainai, jam svarbu pasiekti, kad mainų skaičius būtų lyginis.

Teisingas atsakymas: D.

Paaiškinimas
Reikia atkreipti dėmesį į tai, kad atlikus vieną apsikeitimą, nepriklausomai nuo to, kokias kortas turi kiekvienas žaidėjas, visos pergalės yra apverčiamos.

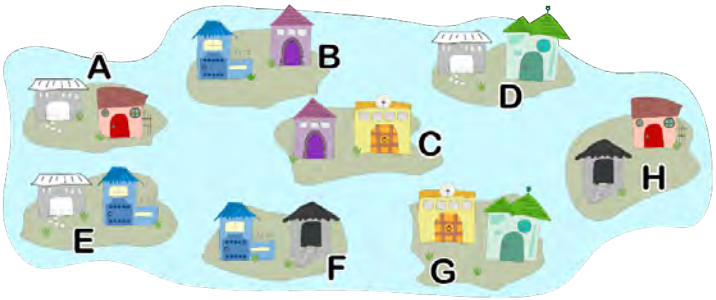


Šiame paveiksle rodyklės rodo, kas ką nugali. Todėl po lyginio skaičiaus mainų pergalės yra tokios pačios kaip ir pradinėje situacijoje. Atvirkščiai, po nelyginio skaičiaus keitimųsi pergalės apsiverčia.
Pradinė situacija tokia, kad Bronius nugali Cezarį. Vadinasi Broniui svarbu užtikrinti lyginį mainų skaičių.

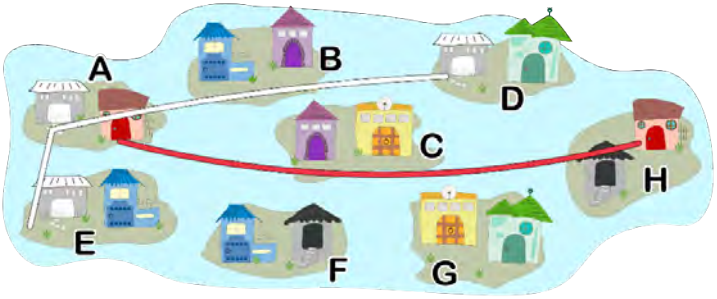
Tai informatika!
Šis uždavinys susijęs su invariantų, t. y., dalykų, kurie keičiant tam tikras sąlygas nesikeičia, išlieka pastovūs, nustatymu.
Invariantai yra labai svarbūs matematikos, fizikos ir informatikos moksluose. Suradus invariantus sumažinamas analizuotinų atvejų skaičius ir uždavinys tampa lengviau valdomas ir sprendžiamas.

39. Magiškos salos 2025-IT-04

Magiškame 8 salų salyne iš salos į salą galima keliauti pereinant per pastatus, kurie yra visiškai vienodi.

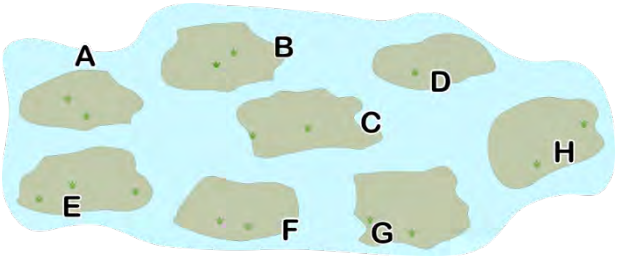


Pavyzdžiui, iš salos A galima keliauti į salas D arba E, įėjus į baltą namą. Taip pat galima keliauti į salą H, įėjus į raudoną namą.



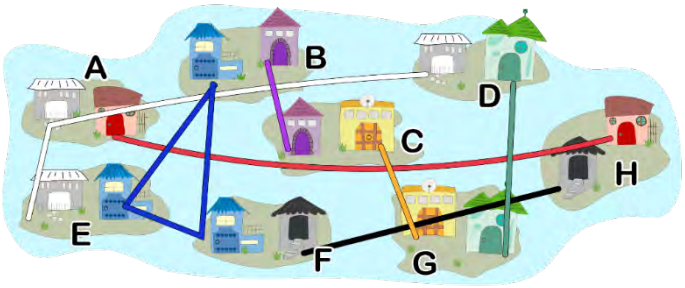
Nora turi žemėlapį, kuriame salos pažymėtos raidėmis. Anksčiau šiame žemėlapyje buvo linijos, jungiančios salas, tarp kurių galima stebuklingai keliauti. Deja, šios linijos išnyko.

Padėk Norai atkurti žemėlapį. Sujunkite visas salų poras atkarpomis taip, kad Nora galėtų stebuklingai keliauti iš vienos salos į kitą.

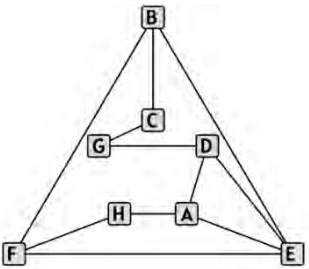


Paaiškinimas

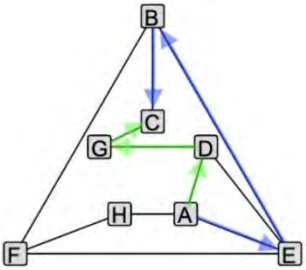
Pirmiausiai sujungiame pastatus, kurie yra vienodi ir turi tos pačios spalvos duris.



Tuomet, remdamiesi pateikta schema, galime atkurti Noros žemėlapį:



Trumpiausias kelias iš A salos į C salą reikalauja išeiti pro 3 duris, kaip parodyta tolesniame paveiksle.
Jame pavaizduoti du skirtingi keliai, trimis žingsniais vedantys iš salos A į salą C.

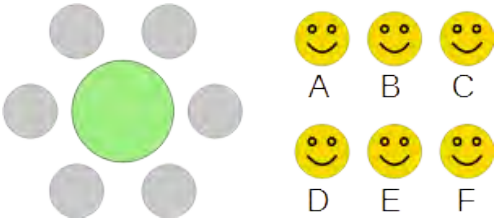


Tai informatika!

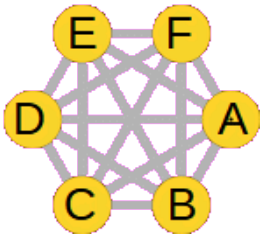
Šio uždavinio diagrama informatikoje vadinama grafu. Grafą sudaro aibė viršūnių (uždavinyje – salų) ir aibė viršūnes jungiančių ryšių, vadinamų briaunomis.
Grafai yra itin dažni informatikoje, nes jie yra natūrali ir naudinga duomenų struktūra daugeliui situacijų modeliuoti, kai egzistuoja dvejetainiai (binariniai) ryšiai tarp skirtingų elementų. Šio uždavinio atveju elementai yra salos, o grafu vaizduojamas dvejetainis ryšys yra „buvimas sujungtam stebuklingomis durimis“. Kiti dvejetainių ryšių pavyzdžiai gali būti draugystė (tarp žmonių porų) arba infrastruktūros jungtys (geležinkelio linijos tarp miestų porų ir pan.).
Uždavinio pabaigoje klausiama, kaip iš A salos pasiekti C salą. Naudojantis grafų terminologija, šią problemą galima performuluoti kaip kelio radimą iš viršūnės A į viršūnę C; mus ypač domina trumpiausias kelias iš visų galimų kelių tarp A ir C. Kai šią situaciją sumodeliuojame grafu ir uždavinį performuluojame grafų terminais, sprendimą rasti paprasta. Sudėtingesniais atvejais pavaizdavimas grafais yra labai naudingas, nes leidžia pritaikyti žinomus (abstrakčių) grafų manipuliavimo algoritmus ir padeda rasti įvairių (konkrečių) uždavinių sprendimus.

40. Laimingų draugų stalas 2025-LT-01

Šeši draugai sėda pietauti prie apvalaus stalo. Kiekvienas nori sėdėti šalia savo geriausio draugo ir labai nuliūsta, jeigu tenka sėdėti šalia mažiau mėgstamo. Susodink draugus aplink stalą, nutempdamas pele veidelius ant kėdžių – pilkų skritulių. Jeigu pasodinsi šalia mažiau mėgstamo draugo, veidelis taps liūdnas. Tavo tikslas – susodinti draugus taip, kad visi veideliai būtų linksmi ir besišypsantys.

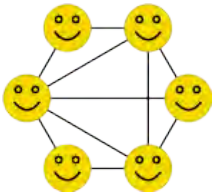


Šis interaktyvus grafas padės išspręsti uždavinį. Jame atsakymo nėra, tiesiog spragsint ant grafo kelio galima keisti jo spalvą.



Paaiškinimas

Pasodinus porą draugų šalia, kurie nemėgsta būti šalia, abiejų veideliai nuliūsta. Taip galime paimti bet kurią porą draugų ir sužinoti, ar jiems patinka būti šalia – tą pasižymėti galime pagalbiniame grafe (jei patinka geltonai, jei ne - raudonai). Vienas iš būdų išspręsti uždavinį – taip pabandyti sodinti visas 15 draugų porų. Tada gautame grafe galima rasti Hamiltono ciklą (t. y., ciklą, kuriame kiekviena viršūnė aplankoma lygiai vieną kartą) ir pagal jį išdėstyti draugus aplink stalą.
Kitas būdas – ieškoti Hamiltono ciklo kartu konstruojant grafą: pradedame pasodinę bet kurį draugą ant bet kurios kėdės, tada kitą šalia jo tol, kol abu taps patenkinti, ir taip toliau. Jei pasiekiame situaciją, kurioje bet kurio draugo pasodinimas daro ką nors nelaimingą, paskutinį pridėtą pašaliname ir tęsiame paiešką.



Išsiaiškinus visą grafą, jis bus toks, kaip pavaizduota paveiksle (sujungti veideliai mėgsta sėdėti šalia), bet raidės gali būti sumaišytos vietomis. Tada vienintelis Hamiltono ciklas gaunamas einant pavaizduoto šešiakampio kraštinėmis.
Kad atsitiktinai sodinant nebūtų galima atsakymo iš karto atspėti (jei atsakymas būtų visiems vienodas, tikimybė atspėti būtų 7/60), kaip raidės, sumaišytos šiame grafe, keičiasi priklausomai nuo to, ką sprendėjas jau žino – pirmenybė teikiama tokiems sumaišymams, kur žinoma daugiau pavaizduoto šešiakampio įstrižainių, ypač tų, kurios yra sujungtos.

Tai informatika!

Šis uždavinys remiasi pagrindinėmis informatikos sąvokomis, susijusiomis su grafų teorija ir algoritminiu problemų sprendimu. Konkrečiai, ji nagrinėja Hamiltono kelius ir ciklus – maršrutus grafu, kai kiekviena viršūnė aplankoma tiksliai vieną kartą (kelio atveju) arba vieną kartą ir grįžtama į pradžios tašką (ciklo atveju). Šios problemos yra itin aktualios realiose srityse, pavyzdžiui, optimizuojant maršrutus transporte ir logistikoje, kur svarbu sumažinti kelionės trukmę, apimant visas paskirties vietas.
Kompiuterių mokslo požiūriu, Hamiltono kelių ar ciklų paieška priklauso NP sudėtingumo (angl. *NP-completeness*) problemų klasei. Tai reiškia, kad šias problemas labai sunku efektyviai išspręsti didėjant grafo viršūnių skaičiui. Šiuo metu nėra žinomo algoritmo, kuris galėtų išspręsti visus uždavinius greitai (polinominio laiko ribose), todėl tai yra puikus pavyzdys, iliustruojantis skaičiavimų požiūriu sudėtingą problemą.
Norint spręsti tokius uždavinius, dažnai naudojamas grįžimo algoritmas (angl. *backtracking*) – strategija, kai sistemingai išbandomos visos galimybės, atmetamos aklavietės ir grįžtama atgal, kad pabandytume alternatyvas. Šis metodas rodo, kad informatika susijusi ne tik su problemų sprendimu, bet ir su skaičiavimo sudėtingumo įvertinimu, efektyvių algoritmų kūrimu bei suvokimu, kokie uždaviniai gali būti išspręsti per priimtiną laiką.
Dirbdami su Hamiltono keliais mokiniai supažindinami su pagrindinėmis informatikos sritimis: duomenų struktūromis (grafais), algoritminiu mąstymu (grįžimo metodas), optimizavimu ir sudėtingumo teorija – viskuo, kas būtina sprendžiant realias problemas kompiuteriu.

41. Juoda-balta 2025-PT-01

Sara žaidžia žaidimą. Ji turi juodų arba baltų kvadratėlių seką ir nori ją užrašyti tam tikru būdu pagal šią taisyklę:

- jei visi sekos kvadratėliai yra **balti**, ji tiesiog rašo „**B**“;
- jei visi sekos kvadratėliai yra **juodi**, ji rašo „**J**“;
- kitais atvejais ji rašo „**X**“ ir po to:
  - taisyklę nuo pradžių taiko **kairiajai** sekos pusei ir rašo jos rezultatą,
  - taisyklę nuo pradžių taiko **dešiniajai** sekos pusei ir rašo jos rezultatą.

Štai keletas pavyzdžių, kaip tokia taisyklė pritaikoma **8 kvadratėlių** sekai:


B


XBJ


XXJBJ


XJXBXJB

Kaip Sara užrašytų tokią kvadratėlių seką?

A) XXJBJXBXBJ

B) XXXJBJXBXBJ

C) XXXBJXBBXXJJXJB

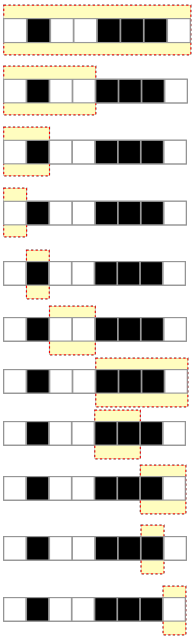
D) XBJBXJJB

E) XXXBJBXJXJB

F) XXBJBXJXJB

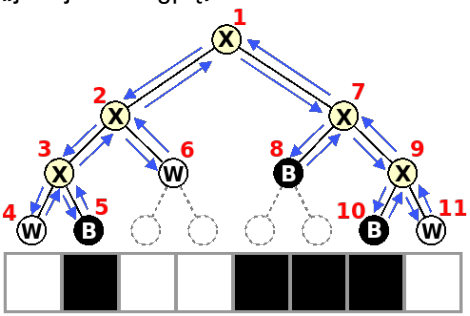
Teisingas atsakymas: E.

Paaiškinimas
Atsakymą galima sudėlioti po vieną raidę štai tokiu būdu:



X	visi 8 kvadratėliai nėra tos pačios spalvos, todėl pradedama nuo X
XX	kairioji pusė nėra tos pačios spalvos, todėl reikia dar vieno X
XXX	kairiosios pusės kairė pusė nėra tos pačios spalvos
XXXB	kairiosios pusės kairės pusės pirmas kvadratėlis yra baltas
XXXBJ	kairiosios pusės kairės pusės antras kvadratėlis yra baltas
XXXBJB	kairiosios pusės dešinė pusė yra balta
XXXBJBX	dešinioji pusė nėra tos pačios spalvos
XXXBJBXJ	dešiniosios pusės kairė pusė yra juoda
XXXBJBXJX	dešiniosios pusės dešinė pusė nėra tos pačios spalvos
XXXBJBXJXJ	dešiniosios pusės dešinės pusės pirmas kvadratėlis yra juodas
XXXBJBXJXJB	dešiniosios pusės dešinės pusės antras kvadratėlis yra baltas

Sprendimą galima pavaizduoti ir tokiu paveikslu, kur mėlynos rodyklės rodo „judėjimo“ kryptį, o raudonas skaičius – kokia tvarka užrašoma seka:



Kaip sudaryti medį?
Pirmiausia seka užrašoma raidėmis kiekvienam kvadratėliui: BJBBJJJB. Tokiu būdu gaunamas apatinis medžio lygmuo – lapai. Tuomet kylama į aukštesnes viršūnes. Jei abu lapai yra vienodi (tik B arba tik J), jie apjungiami aukštesnėje viršūnėje įrašant tą pačią raidę, šioji tampa nauju medžio lapu, o ankstesni lapai pašalinami. Priešingu atveju aukštesnėje viršūnėje įrašome X.

Kaip iš medžio sudaryti teisingą seką?

Pradedama nuo medžio šaknies (paveiksle – viršutinės viršūnės). Medžiu judama kiekviename išsišakojime pirmiausia einant kairėn. Kai pirmą kartą pasiekiama kuri nors viršūnė, į seką užrašoma joje esanti raidė. Kai pasiekiamas lapas, grįžtama į ankstesnę viršūnę ir einama dešinėn. Jei abu keliai jau praeiti, grįžtame į dar ankstesnę viršūnę.

B variantas neteisingas, nes jame pateikiama „papildoma“ eilutė (juoda, kai kvadratas yra baltas, ir atvirkščiai).

C variantas neteisingas, nes jame du vienodi kvadratėliai ne pakeičiami viena raide, o užrašomi dviem vienodomis raidėmis.

A, D ir F variantai, be kita ko, yra neteisingi, nes neprasideda trimis X ženklais, kaip reikia pirmiesiems kvadratams pavaizduoti.

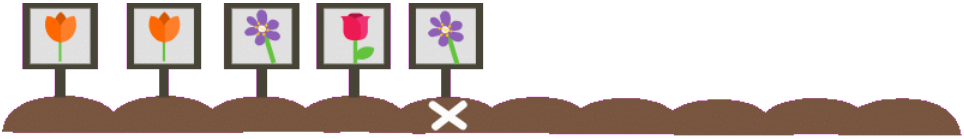
Tai informatika!

Taisyklė, kuri čia naudojama kvadratėlių sekoms užrašyti, vadinama rekursine taisykle.
Rekursija – tai algoritminis metodas, kai funkcija ar procedūra kreipiasi į save pačią ir pakartoja savo pačios veiksmus nuo pradžios, tik su kitais parametrais. Naudojant rekursiją sprendimo procedūrą galima aprašyti pačia savimi: paprastoje situacijoje tiesiog nurodomas rezultatas (šiame uždavinyje tai – „B“ ir „J“), o sudėtingesnėje – uždavinys išskaidomas į smulkesnius uždavinius, kurie sprendžiami tokiu pačiu būdu, tuomet smulkesniųjų uždavinių sprendimai apjungiami ir gaunamas pradinio uždavinio rezultatas (šiame uždavinyje tai – „X“, po kurio eina kairės ir dešinės pusės sprendimai).
Rekursija taikoma daugelyje sričių. Leonardas Fibonačis (Leonardo da Pisa, Fibonacci) analizuodamas triušių populiacijos dydį išvedė savo garsiąją rekurentinę formulę. Naudojant rekursiją galima apskaičiuoti triušių populiacijos dydį po kelerių metų. Kitas pavyzdys – fraktalai – geometrinės figūros, sudarytos naudojant rekursiją.
Kad informacija būtų automatiškai apdorojama, ją reikia atvaizduoti simboliškai; be to, svarbu rasti tokias atvaizdavimo formas, kurios leistų ją efektyviai apdoroti. Todėl atvaizdavimo formų ir duomenų struktūrų tyrimai yra svarbi informatikos sritis. Informacijos konvertavimo iš vienos vaizdavimo formos į kitą procesas vadinamas kodavimu. Jei šiame uždavinyje žinomas tikslus kvadratėlių sekos ilgis, kodą galima konvertuoti atgal į kvadratėlių seką. Tokie kodai vadinami invertuojamaisiais. Jei ilgis yra fiksuotas, toks kodas vadinamas prefiksiniu kodu. Tai reiškia, kad joks kodas nėra kito kodo prefiksas. Tokius kodus galima greitai invertuoti ir jiems nereikia daug skaičiavimo išteklių.

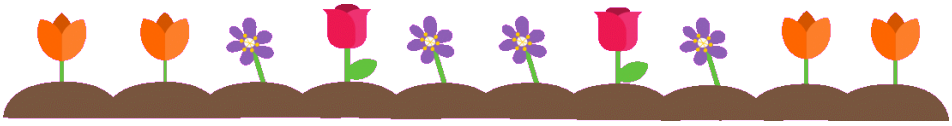
42. Gėlynas 2025-SK-03

Robotas gėlyne pagal ženklus į eilę sodina jau užaugintas gėles. Tuščioje vietoje nėra nei ženklo, nei gėlės. Robotas gėles sodina pagal šias taisykles:

0. Eik į vietą, pažymėtą **x**.
Kartok 1–5 veiksmus, kol neliks vietos su ženklu.
1. Jei tavo vietoje yra ženklas, toje vietoje pasodink ant ženklo pavaizduotą gėlę.
2. Įsimink ką tik pasodintą gėlę.
3. Nuimk ženklą.
4. Eik į dešinę, kol prieisi tuščią vietą, tada ten pasodink ką tik įsimintą gėlę.
5. Eik į kairę, kol atsidursi vietoje su ženklu, arba kol išeisi iš gėlyno.



Kaip atrodys gėlynas, kai robotas baigs sodinti gėles?

A) 

B) 

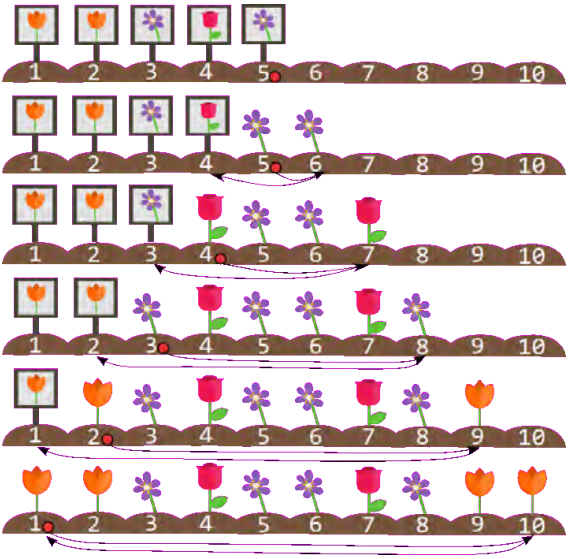
C) 

D) 

Teisingas atsakymas: A.

Paaiškinimas

Sunumeruokime kiekvieną vietą. Robotas pradeda nuo vietos, pažymėtos **x** – tai 5 vieta. Ten jis pasodina našlaitę , įsimena ją ir eina į dešinę. Visai šalia yra tuščia vieta (6), tad robotas pasodina ką tik įsimintą gėlę (našlaitę) ir eina į kairę, kol randa pirmą vietą su ženklu, bet dar be gėlės. Tai ketvirtoji vieta. Joje jis pasodina rožę ir ją įsimena.
Tuomet robotas eina dešinėn iki pirmos tuščios vietos, o tai yra 7-oji vieta, ir pasodina ten įsimintą rožę .
Po to jis vėl eina kairėn iki vietos su ženklu (3). Ten robotas pasodina našlaitę , ją įsimena ir eina dešinėn iki 8-os vietos (pirma tuščia vieta), kur pasodina dar vieną našlaitę .
2-oje ir 9-oje vietose robotas pasodins po tulpę , lygiai kaip ir 1-oje bei 10-oje – dar po vieną tulpę .
Matome, kad į tuščias vietas robotas gėles sodina atvirkštine tvarka.



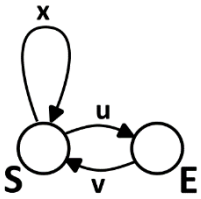
Paveiksle raudonas taškas žymi roboto vietą, o rodyklės rodo, kaip jis judėjo.

Tai informatika!

Šiame uždavinyje instrukcijomis robotui iliustruojamas Tiuringo mašinos veikimo principas. Tiuringo mašina – tai automatas, vykdantis tam tikrą algoritmą. Uždavinyje pritaikyta Tiuringo mašina turi vieną galvutę (robotą), kuri juda išilgai juostos (gėlių eilės) su langeliais (sodinimo vietomis) ir langeliuose gali skaityti (ženklus) ir rašyti (pašalinti ženklą ir pasodinti gėles). Tiuringo mašinos atminties įtaisas yra juosta, o juostų mašina gali turėti daugiau nei vieną. Vykdydamas uždavinio instrukcijas robotas įsimena vėliausiai pasodintą gėlę, o šią informaciją galima įrašyti kitoje juostoje.

43. El. pašto adresų tikrinimas 2025-AT-03

Pristatome sistemą – baigtinį automatą, – kurią sudaro būsenos, pavaizduotos apskritimais, ir pokyčiai (angl. *transitions*) tarp jų, pavaizduoti rodyklėmis. Pokyčius vaizduojančios rodyklės pažymėtos raidėmis.



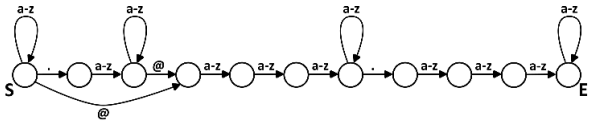
Yra dvi ypatingos būsenos: pradžios būsena (S) ir pabaigos būsena (E).
Paveiksle pavaizduota sistema skaito žodžius, sudarytus iš simbolių „x“, „u“ ir „v“, po vieną simbolį iš kairės į dešinę, pradedant nuo būsenos S.
Šiame pavyzdyje yra tik dvi būsenos S ir E, bet kitos sistemos gali turėti ir daugiau būsenų.

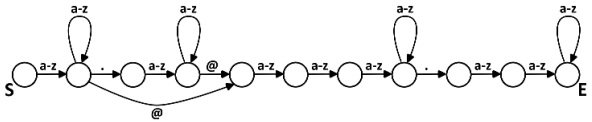
- Jei sistema, perskaičiusi visą žodį, baigia būdama E būsenoje, žodis yra **galiojantis**.
- Priešingu atveju žodis yra **negaliojantis**, t. y., sistema jo nepriima ir nepripažįsta.

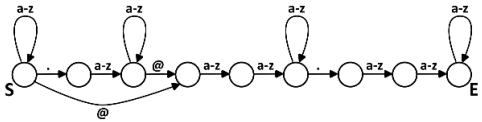
Laikydamiesi šių taisyklių, galime nustatyti, jog:
galioja: xuvu, xuvxxu, u, ...
negalioja: uv, x, xuvyxu, uvuvuv, ...

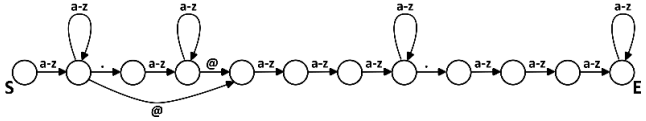
Kuri sistema galės teisingai perskaityti tokius el. laiškų adresus:
U.V@W.X arba **V@W.X**

- U reiškia žodį, sudarytą iš vienos ar daugiau raidžių
- V taip pat reiškia žodį, sudarytą iš vienos ar daugiau raidžių
- W reiškia žodį, sudarytą iš trijų ar daugiau raidžių
- X reiškia žodį, sudarytą iš dviejų ar daugiau raidžių

A


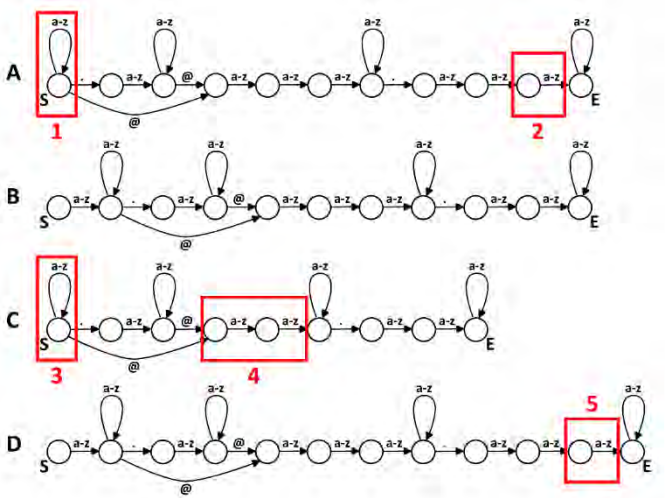
B


C


D


Teisingas atsakymas: B

Paaiškinimas
A, C ir D atvejai turi tokias klaidas:



Skaičiais 1 ir 3 pažymėta klaida: nereikalingas pradinis simbolis.
Skaičiais 2 ir 5 pažymėta klaida: paskutinė dalis po taško turi būti sudaryta iš mažiausiai trijų simbolių, bet apibrėžime reikalaujama tik dviejų.
Skaičiumi 4 pažymėta klaida: po simbolio „@“ esanti dalis gali būti sudaryta ir iš dviejų simbolių, tačiau apibrėžime reikalaujama trijų simbolių.

B yra vienintelis variantas, kuriame nėra nė vienos iš šių klaidų. Ir jis yra teisingas.

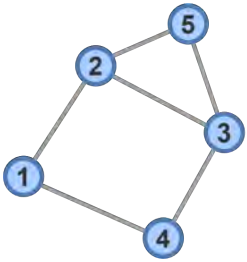
Tai informatika!

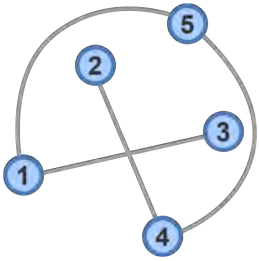
Baigtinis automatas (angl. *finite-state automaton*) – matematinė abstrakcija, modeliuojanti sistemos būsenų kaitą, priklausomai nuo ankstesnės būsenos ir įėjimo signalų, kai būsenų ir galimų įėjimo signalų skaičius yra baigtinis (todėl ir vadinama baigtiniu automatu).
Baigtiniai automatai gali būti naudojami reguliariems modeliams, sudarytiems iš simbolių (raidžių, skaitmenų), tikrinti. Pavyzdžiui, elektroninio pašto adresams, pašto kodams, bankų IBAN kodams, knygų ISBN numeriams, automobilių valstybiniams numeriams, datoms ir pan.
Baigtiniai automatai taip pat padeda valdyti liftus ar šviesoforus. Jie naudojami tinklų protokoluose, kuriuose žingsnis po žingsnio modeliuoja komunikaciją tarp sistemų.
Baigtiniai automatai yra naudingi sprendžiant problemas, kai teoriniais ir praktiniais tikslais reikalingos griežtos taisyklės pokyčiams (angl. *transitions*) tarp sistemos būsenų.

44. Grafo papildinys 2025-AT-05

Grafas parodo, kaip objektai yra susiję. Kiekvienas taškas (viršūnė) kažką reiškia (pavyzdžiui, asmenį, vietą ar įrenginį), o linijos (briaunos) rodo, kurie taškai yra susiję. Grafo papildinys (angl. *complement graph*) – grafas su tais pačiais taškais, bet priešingais ryšiais: jei du taškai yra susiję pradiniame grafe, jie nėra susiję grafo papildinyje, o jei du taškai nėra susiję pradiniame grafe, jie yra susiję grafo papildinyje.

Kairėje esantis grafas yra dešinėje esančio grafo papildinys ir atvirkščiai.

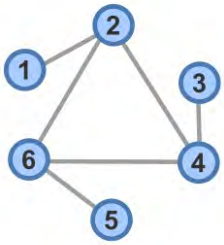




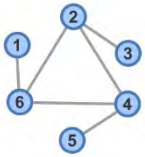
Pavyzdžiui:

- 1 ir 2 taškai yra sujungti kairiajame grafe, bet jie nėra sujungti dešiniajame.
- 1 ir 5 taškai nėra sujungti kairiajame grafe, bet jie yra sujungti dešiniajame.

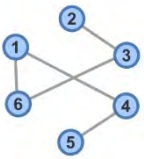
Kuris grafas yra šio grafo papildinys?



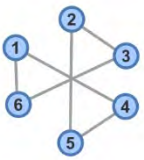
A)



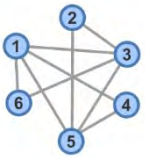
B)



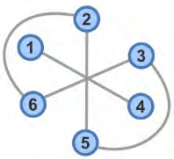
C)



D)



E)

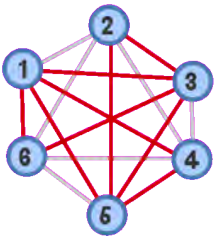


Teisingas atsakymas: D

Paaiškinimas

Pirmame grafe kai kurie taškai (viršūnės) yra sujungti, o kiti – ne. Grafo papildinyje kiekviena taškų pora, kuri anksčiau nebuvo sujungta, dabar sujungta, o kiekviena linija (briauna) iš pirmo grafo yra prarasta.

D variante pateiktas grafas yra vienintelis, kuris atitinka šią taisyklę.



Šiame paveiksle raudonos linijos rodo visus ryšius tarp 6 taškų, kurių nebuvo pradiniame grafe. Jei pašalinsime pilkas linijas (kurios rodo pradinius ryšius), liks ieškomos linijos atitinkančios grafo papildinį.

Tarp 6 taškų gali būti 15 jungčių. Taškas 1 gali būti sujungtas su 5 kitais taškais, taškas 2 – su 4 (neskaitant to, kuris jau sujungtas su tašku 1), taškas 3 – su 3 ir t. t. Tai iš viso duoda 5 + 4 + 3 + 2 + 1 = 15 galimų jungčių.

Pirmas grafas turi šiuos 6 ryšius (pilkosios linijos):
(1-2), (2-4), (2-6), (3-4), (4-6), (5-6)

Tačiau jo papildinys turi šiuos 9 ryšius (raudonosios linijos):
(1-3), (1-4), (1-5), (1-6), (2-3), (2-5), (3-5), (3-6), (4-5)

Nė vienas iš šių ryšių nepasikartoja abiejuose grafuose, bet visi galimi ryšiai yra arba originaliajame grafe, arba jo papildinyje.

Tai informatika!

Grafas yra galingas įrankis, skirtas vaizduoti sąryšius, pavyzdžiui, draugystes, kompiuterių tinklus ar ryšius tarp tinklalapių. Informatikoje analizuoti tai, kas nėra susiję, gali būti taip pat svarbu, kaip analizuoti tai, kas yra susiję. Padėti gali grafo papildinys (angl. *complement graph*, liet. dar vadinamas komplanariu grafu).

Įdomu tai, kad kai kurios problemos, kurias sunku išspręsti originaliame grafe, gali tapti lengvesnės, jei jas sprendžiame naudodami pradinio grafo papildinį. Kai kurie algoritmai gali veikti efektyviau naudojant grafo papildinį. Pavyzdžiui, maksimalios nepriklausomos grupės paieška grafe yra lygiavertė maksimalios klikos paieškai šio grafo papildinyje, o tai kartais gali būti efektyvesnis būdas, priklausomai nuo grafo struktūros. Grafų teorijoje klika (angl. *clique*) yra neorientuoto grafo viršūnių poaibis, kuriame bet kurios dvi skirtingos viršūnės yra gretimos.

Lygiagrečiame programavime užduotys dažnai turi priklausomybes, kurios neleidžia jų vykdyti vienu metu. Šias priklausomybes galima modeliuoti kaip grafą:

- Kiekvienas taškas (šiame kontekste dar vadinamas mazgu) žymi užduotį.
- Linija (šiame kontekste dar vadinama briauna) tarp dviejų mazgų rodo, kad užduotys negali būti vykdomos vienu metu (nes viena priklauso nuo kitos).

Šiuo atveju grafo papildinyje matyti, kurios užduotys gali būti vykdomos vienu metu, t. y., užduotys, kurios nėra sujungtos priklausomybe. Tai nepaprastai naudinga kuriant efektyvius užduočių planuoklius arba nustatant nepriklausomus užduočių rinkinius, kuriuos galima vykdyti lygiagrečiai, o tai pagreitina skaičiavimus.

45. Paslaptinga piramidė 2025-DE-10

Bebras Jonas tyrinėja paslaptingą piramidę. Piramidėje yra daug klastingų koridorių. Kiekvieno koridoriaus gale yra lobis. Jono tikslas – kuo greičiau pasiekti bet kurį lobį.
Kiekvienas koridorius yra apsaugotas blokų eile. Iš pradžių visi blokai yra nuleisti. Priartėjus prie koridoriaus, jo blokai pradeda periodiškai judėti. Blokas, kurio periodas lygus 2, pakyla po 2 minučių, po dar 2 minučių nusileidžia ir taip toliau.

Paveiksle pavaizduotas koridorius, kuriame yra du blokai, pažymėti A ir B, jų periodai lygūs atitinkamai 2 ir 3. Kiekviena paveikslėlio dalis rodo situaciją kas minutę. Norėdamas kuo greičiau pasiekti lobį, Jonas laukia 2 minutes, tada įeina po pakilusiu bloku A, palaukia dar 1 minutę, tada praeina po pakilusiu bloku B ir pasiekia lobį po 3 minučių.
Jono veiksmus galima aprašyti tokia komandų seka:

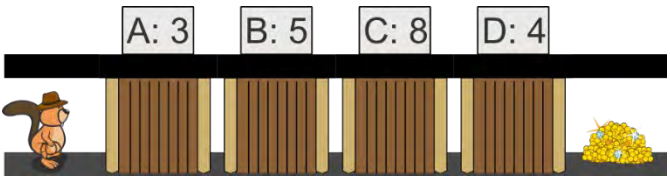
```
laukti(2)
    eiti po bloku(A)
    laukti(1)
    eiti prie lobio
```

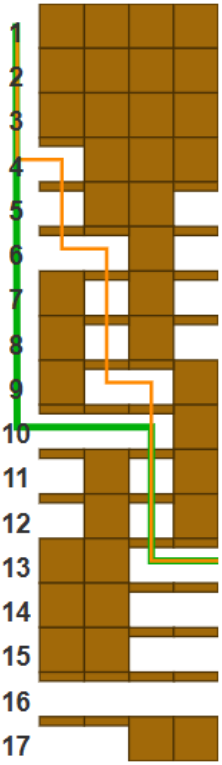
Jono veiksmus galima aprašyti trumpesne komandų seka (lobį jis pasiektų per tą patį laiką):

```
laukti(3)
eiti prie lobio
```

Kitame klastingame koridoriuje yra keturi blokai, kurių periodai yra atitinkamais 3, 5, 8 ir 4 minutės. Jonas nori ir šio koridoriaus lobį pasiekti kuo greičiau.

Kokia trumpiausia komandų seka aprašomas Jono kelias prie lobio?
Atminkite, kad
(a) pereiti po vienu bloku į kitą užtrunka nedaug laiko ir
(b) pereiti po vienu bloku į kitą saugu tik tada, kai abu blokai yra pakelti.





Teisingas atsakymas

Veiksmai turėtų būti pateikti „Blockly“ stiliaus blokais, juose galima keisti parametrus ir juos galima surinkti į seką. Reikalingos šios (daugiausia parametrizuotos) komandos:
„laukti(n)“, „eitį po bloku(1)“, „eiti prie lobio“.
Be to, turėtų būti pateikta interaktyvi priemonė, skirta bloko elgsenai imituoti. Mokinys gali judėti į priekį ir atgal kas minutę ir matyti koridoriaus blokų būsenas. Minutės skaitiklis rodo, kiek minučių praėjo.

Paaiškinimas

Paveikslėlyje pavaizduota, kaip Jonas gali judėti šiuo koridoriumi. Norėdamas judėti kuo greičiau, jis gali atlikinėti šias komandas ir pasiekti lobį per 12 minučių (oranžinė linija):	laukti(3) eiti po bloku(A) laukti(2) eiti po bloku(B) laukti(3) eiti po bloku(C) laukti(4) eiti prie lobio
Tačiau komandų gali būti ir mažiau:	laukti(9) eiti po bloku(C) laukti(3) eiti prie lobio

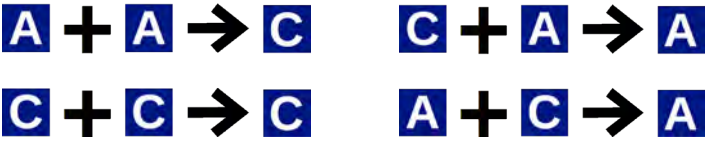
Jonas negali atlikti mažiau veiksmų, neprarasdamas laiko. Jis galėtų atlikti mažiau veiksmų tik tuo atveju, jei iš karto nubėgtų per visą koridorių. Vienintelė galimybė tai padaryti atsiranda palaukus 15 minučių – tada jis gali pasiekti lobį, įvykdęs tik dvi komandas. Tačiau tada bus atlikta veiksmų ir užtrukta daugiau laiko.

Tai informatika!

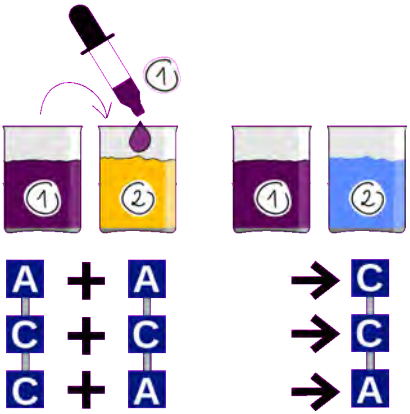
Šis uždavinys apima tris svarbius informatikos aspektus.
1. Algoritmas. Algoritmo sąvoka yra bene svarbiausia informatikos mokslo sąvoka; ji nusako, kaip galima išspręsti problemą ir kaip rasti sprendimą. Algoritmo paieška reikalauja kūrybiškumo, priešingai nei algoritmo vykdymas. Kai algoritmas yra tiksliai apibrėžtas, pavyzdžiui, programavimo kalba užrašyta veiksmų seka, net mašina gali komandas vykdyti be jokio supratimo ar kūrybinio indėlio.
2. Optimizavimas. Šis uždavinys apima įvairių sprendimo būdų svarstymą, jų palyginimą tarpusavyje ir geriausio sprendimo pasirinkimą pagal tam tikrus kokybės kriterijus. Informatikos moksle tokio tipo uždaviniai vadinami optimizavimo problemomis, nes jų tikslas yra sumažinti tam tikrą veiksnį (šiuo atveju – laiką, per kurį Jonas pasiekia lobį, ir veiksmų, kuriuos jis vykdo, skaičių). Vienas iš metodų, kurį galima naudoti optimizavimo problemoms spręsti, yra dinaminis programavimas. Šis metodas apima problemos suskaidymą į dalines problemas ir nuolatinį didesnių pradinės problemos dalių sprendimą naudojant išsaugotus dalinius sprendimus. Mūsų uždavinyje dalinė problema gali būti sudaryta iš to, kad pirmiausia reikia pasiekti tik pirmą, antrą, trečią arba ketvirtą bloką prieš nustatant kelią iki paties lobio.
3. Programavimas. Kartais vienai problemai spręsti gali būti daug skirtingų sprendimų. Norint greičiau ir efektyviau ieškoti sprendimo, paieškos procesas dažnai yra automatizuojamas. Tam naudojamos programavimo kalbos, kurios suteikia pagrindinių operacijų (komandų) rinkinį, pavyzdžiui, komandas „laukti(n)“ arba „eiti po bloku(b)“).

46. Biberonai 2025-IT-01

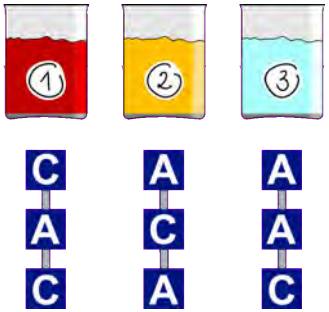
Biberonai yra medžiagos, kurių molekulės susideda iš trijų A ir C tipo sudedamųjų dalių. Įlašinus vos lašelį vieno biberono į kitą biberoną, pastarasis gali visiškai pasikeisti.
Abiejų biberonų molekulių sudedamosios dalys nulemia gauto biberono sudedamąsias dalis pagal šias taisykles:



Paveikslėlyje pateiktas pavyzdys, kai į ACA įlašinamas lašelis ACC: ACA virsta CCA.

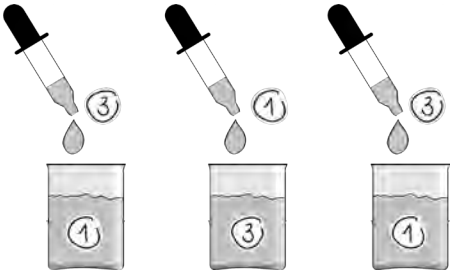


Turime tris rezervuarus – 1, 2 ir 3, – kuriuose yra biberonai CAC, ACA ir AAC. Naudojantis instrukcijomis, galima nurodyti, kad lašas biberono iš vieno rezervuaro (pavyzdžiui, 2-o rezervuaro) būtų perkeltas į kitą rezervuarą (pavyzdžiui, 3-ą rezervuarą).



Sukeiskite 1-o ir 3-o rezervuarų medžiagas: 1-ame rezervuare turėtų tapti AAC, 2-ame rezervuare turėtų likti ACA, o 3-ame rezervuare CAC biberonai. Naudokite kuo mažiau instrukcijų.

Paaiškinimas
Spalvas galima sukeisti vos trimis maišymo veiksmais. Priimtini tokie atsakymai: 3→1,1→3,3→1 arba 1→3,3→1,1→3:



Pirmiausia įrodysime, kad tai galima padaryti panaudojant 3 lašus, pirmuoju atveju: 3→1,1→3,3→1.

Jei geltonų dažų lašas (AAC) iš 3-o rezervuaro sumaišomas su žaliais dažais (CAC) 1-ame rezervuare (3→1), tai gauta spalva 1-ame rezervuare bus raudona: AAC+CAC=ACC. Spalvos rezervuaruose abar yra tokios: .

Jei raudonų dažų lašas (ACC) iš 1-o rezervuaro sumaišomas su geltonais dažais (AAC) 2-ame rezervuare (1→3), tai 3-ame rezervuare gausim žalią: ACC+AAC=CAC. Spalvos rezervuaruose ra tokios: .

Jei žalių dažų lašas (CAC) iš 3-o rezervuaro yra sumaišomas su raudonais dažais (ACC) 1-ame rezervuare (3→1), tai 1-me rezervuare bus gauta geltona spalva: CAC+ACC=AAC. Spalvos rezervuaruose dabar yra tokios: , tai yra norimas derinys, kuriame žalia ir geltona spalvos yra sukeistos vietomis.

Dabar parodysime, kad 3 lašai yra mažiausias kiekis. Vieno lašo akivaizdžiai nepakanka: jis gali pakeisti tik vieną spalvą, o mums reikia pakeisti dvi.
Jeigu užtektų dviejų lašų, pirmasis dažų lašas turėtų paversti 1-ąjį rezervuarą geltonu, o antrasis – 3-iąjį žaliu (arba atvirkščiai). Paieškokime atvejo, kai pirmame žingsnyje 1-asis rezervuaras paverčiamas geltonu:

- (2 → 1) rezultatas yra ACA+CAC=AAA (balta).
- (3 → 1) rezultatas yra AAC+CAC=ACC (raudona).

Taigi neįmanoma pakeisti 1-ojo rezervuaro spalvos į reikiamą tik 1 lašu, o antrasis lašas negali pakeisti dviejų rezervuarų spalvos.
Panagrinėkime atvejį, kai pirmame žingsnyje 3-iasis rezervuaras paverčiamas žaliu:

- (1 → 3) rezultatas yra CAC+AAC=ACC (baltas).
- (2 → 3) rezultatas yra ACA+AAC=CAA (žalsvai mėlyna).

Dėl tos pačios anksčiau nurodytos priežasties neįmanoma panaudojus tik du lašus pakeisti abiejų rezervuarų spalvų į reikiamas. Taigi, mažiausias reikalingas lašų skaičius yra 3.

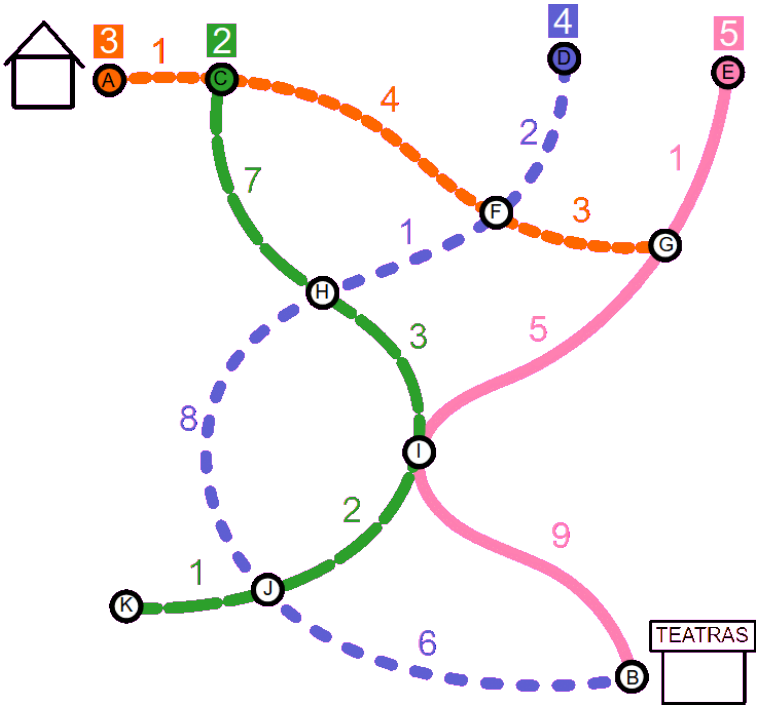
Tai informatika!

Dviejų nukleotidų A ir C sąveikos taisyklė vadinama išskirtinės disjunkcijos (XOR) logine operacija. Jeigu A interpretuosime kaip „teisinga“, o C kaip „neteisinga“, tai rezultatas yra toks: „teisinga“ – kai abu operandai yra skirtingi, „neteisinga“ – kai abu operandai vienodi.
XOR operacija labai dažna informatikoje. Jai būdinga „keitimo vietomis“ savybė naudinga keičiant vietomis dviejų kintamųjų (uždavinyje – rezervuarų) reikšmes nenaudojant trečiojo, laikino, kintamojo.
XOR taip pat yra pagrindinė operacija, naudojama kriptografijoje. Įsivaizduokite, kad turite pranešimą, išreikštą dvejetainiu kodu, pavyzdžiui, 01011. Galite užšifruoti pranešimą pritaikydami jam XOR operaciją su tokio paties ilgio raktu, tarkime, 11001 – rezultatas bus šifruotas tekstas 10010. Bet kas gali atkurti pranešimą, dar kartą šifruotam tekstui pritaikęs XOR su raktu, tačiau nežinant rakto pranešimas yra visiškai slaptas, nes kiekvienas jo simbolis gali būti gautas tiek iš 0, tiek iš 1 pradiniame pranešime.

47. Viešasis transportas 2025-LT-05b

Markas nori iš namų nuvykti į teatrą. Mieste veikia 4 vienakrypčiai autobusų maršrutai. Autobusų stotelės pažymėtos juodai apvestais apskritimais. Autobusų maršrutai pažymėti skirtingų spalvų linijomis. Nuspalvintos stotelės žymi maršrutų pradžias.
Visų maršrutų pirmieji autobusai iš pradžios stotelių išvyksta tuo pačiu metu. Po to kiekvienu maršrutu autobusai vyksta skirtingais laiko intervalais. Skaičiai spalvotuose stačiakampiuose žymi laiko skirtumą tarp autobusų išvykimų minutėmis. Pavyzdžiui, oranžiniu maršrutu autobusai išvyksta kas 3 minutes – 0-inę, 3-ią, 6-tą, 9-tą minutę ir t. t.
Skaičiai šalia maršrutų atkarpų žymi, per kiek minučių autobusas įveikia atkarpą. Laikykime, kad sustoti ir keleiviams įlipti bei išlipti laiko papildomai skirti nereikia. Stotelėse, kuriose stoja kelių maršrutų autobusai, galima perlipti į kito maršruto autobusą. Jei Markas atvyksta į tokią stotelę, jis gali perlipti į autobusą, kuris atvyksta į šią stotelę tuo pačiu metu arba vėliau.

Kurias stoteles (įskaitant pirmą ir paskutinę) Markas aplankys vykdamas į teatrą mažiausiai laiko trunkančiu maršrutu, jei išvyks pirmuoju oranžiniu autobusu?

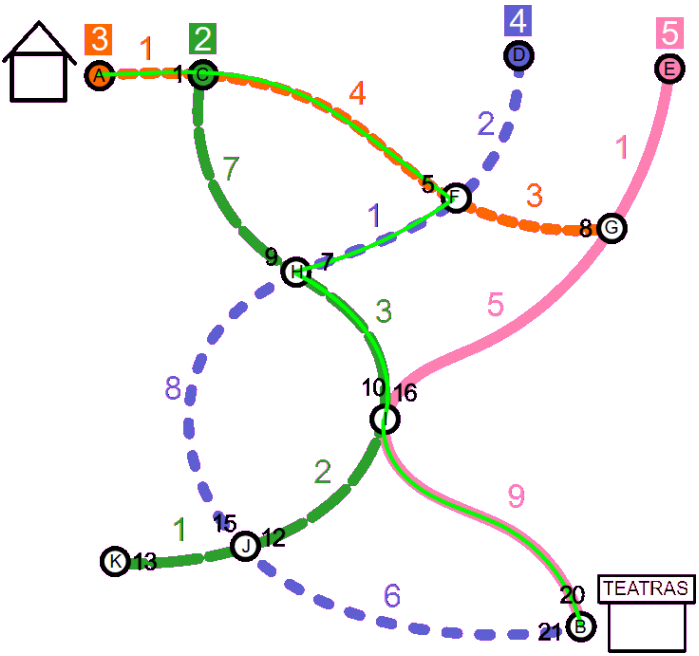


Teisingas atsakymas: A, C, F, H, I ir B stotelės. Šio maršruto trukmė yra 20 minučių.

Paaiškinimas

Teisingas atsakymas: A, C, F, H, I ir B stotelės. Šio maršruto trukmė yra 20 minučių. Atsakymą galima rasti pasirenkant stotelę su jau žinomu greičiausiu atvykimo laiku į ją (pradžioje tai tik A stotelė). Tada galima apskaičiuoti atvykimo į kitas stoteles, pasiekiamas tiesiai iš šitos, laikus. Šis procesas kartojamas tol, kol randamas greičiausias laikas į teatrą.

Stotelės gali turėti kelis įmanomus atvykimo laikus – tuomet reikia atsižvelgti tik į patį greičiausią. Tik tada, kai visi atvykimo į stotelę būdai yra apskaičiuoti ir žinomas pats greičiausias, galima skaičiuoti atvykimo į kitas stoteles, pasiekiamas iš šios, laikus. Toliau aprašomas šio metodo taikymas šiam uždaviniui spręsti.



Teisingas maršrutas yra paryškintas. Skaičiai šalia stotelių nurodo greičiausius laikus, per kuriuos įmanoma jas pasiekti iš kiekvienos krypties.

C stotelė pasiekiama per 1 minutę pirmuoju oranžiniu autobusu.

Tuo pačiu autobusu F stotelė pasiekiama iš viso per 5 minutes, G stotelė – per 8.

I stotelę galima pasiekti iš G stotelės trečiu rožiniu autobusu (palaukus 3 minutes), iš viso per 8 + 3 + 5 = 16 minučių.

H stotelę galima pasiekti iš C stotelės (atvykus 1-ąją minutę) antru žaliu autobusu, palaukus 1 minutę ir juo pavažiavus 7 minutes, iš viso per 1 + 1 + 7 = 9 minutes nuo kelionės pradžios.

H stotelę taip pat galima pasiekti iš F stotelės (atvykus į ją 5-ąją minutę) antru mėlynu autobusu, prieš tai palaukus 1 minutę ir pavažiavus dar 1 minutę, iš viso per 5 + 1 + 1 = 7 minutes. Tai yra greičiau už prieš tai apskaičiuotas 9 minutes, todėl tik trumpesnis 7 minučių laikas bus naudojamas tolesniuose skaičiavimuose. Šis

trumpesnis atvykimo laikas leidžia spėti persėsti į pirmą žalią autobusą ir pasiekti I stotelę per 7 + 3 = 10 minučių nuo pradžios. Naujas atvykimo laikas į I stotelę yra trumpesnis, nei prieš tai apskaičiuotos 16 minučių, todėl 16 minučių tolesniuose skaičiavimuose nebus naudojamos. 10 minučių yra trumpiausias laikas į I stotelę, jis naudojamas skaičiuojant toliau: tuo pačiu autobusu galima pasiekti J stotelę per 10 + 2 = 12 minučių ir K stotelę per 12 + 1 = 13 minučių.
Į J stotelę taip pat galima atvykti iš stotelės H per 7 + 8 = 15 minučių. Tai yra ilgesnis laikas, nei apskaičiuotas anksčiau, ir jis toliau nenaudojamas.
Iš I stotelės, palaukus 1 minutę, antru rožiniu autobusu galima pasiekti B stotelę (kelionės tikslą) per 10 + 1 + 9 = 20 minučių. Dar nežinoma, ar tai galutinis atsakymas, nes į B stotelę galima atvykti ir iš J stotelės. Tai galima padaryti J stotelėje palaukus 3 minutes ir 6 minutes pavažiavus antru mėlynu autobusu, iš viso per 12 + 3 + 6 = 21 minutę. Tai yra ilgiau už 20 minučių, taigi 20 minučių, apskaičiuotų prieš tai, iš tiesų yra greičiausias įmanomas laikas. Atsakant tereikia pasirinkti stoteles, aplankytas vykstant šiuo maršrutu.

Tai informatika!

Autobusų maršrutų planas sudarytas iš stotelių, sujungtų vienakrypčiais autobusų maršrutais. Autobusų maršrutus sudaro atskiros atkarpos tarp dviejų stotelių, kurias nuvažiuoti užtrunka tam tikrą laiką.
Tokia struktūra yra labai panaši į svorinį kryptinį grafą. Stotelės atitinka grafo viršūnes, o maršrutų atkarpos – briaunas. Viršūnių, sujungtų briaunomis, struktūra yra vadinama grafu. Kadangi maršrutai yra vienakrypčiai, taigi ir atkarpos tarp viršūnių (briaunos) turi kryptį, grafas yra kryptinis. Taip pat, kadangi kiekvienai atkarpai priskirtas laikas, per kurį ji įveikiama (briaunos svoris), grafas yra svorinis. Uždavinio struktūra skiriasi nuo įprasto svorinio kryptinio grafo tik tuo, kad keliavimo iš vienos stotelės į kitą laikas priklauso ne tik nuo atstumo tarp jų, bet ir autobusų tvarkaraščio. Svoriniai grafai turi nekintančius briaunų svorius, nepriklausomus nuo atvykimo laiko.
Uždavinyje prašoma rasti trumpiausią kelią tarp dviejų grafo viršūnių. Toks uždavinys labai dažnai pasitaiko įvairiuose informatikos uždaviniuose ir yra tikrų maršruto paieškos algoritmų supaprastinta versija. Yra keletas šio uždavinio sprendimo būdų, o pateiktas sprendimo aprašyme yra labai panašus į dinaminį programavimą. Tai toks uždavinių sprendimo būdas, kai uždavinys padalijamas į mažesnes dalis, kurios išsprendžiamos pirmiausiai, o gauti atsakymai naudojami vis didesnei uždavinio daliai spręsti, kol gaunamas ieškomas atsakymas. Šiuo atveju pradedama nuo atvykimo į arčiausiai nuo kelionės pradžios esančias stoteles laiko radimo ir skaičiavimai plečiami iki tolimiausios stotelės – kelionės tikslo.

48. Apsauga nuo potvynio 2025-NZ-01

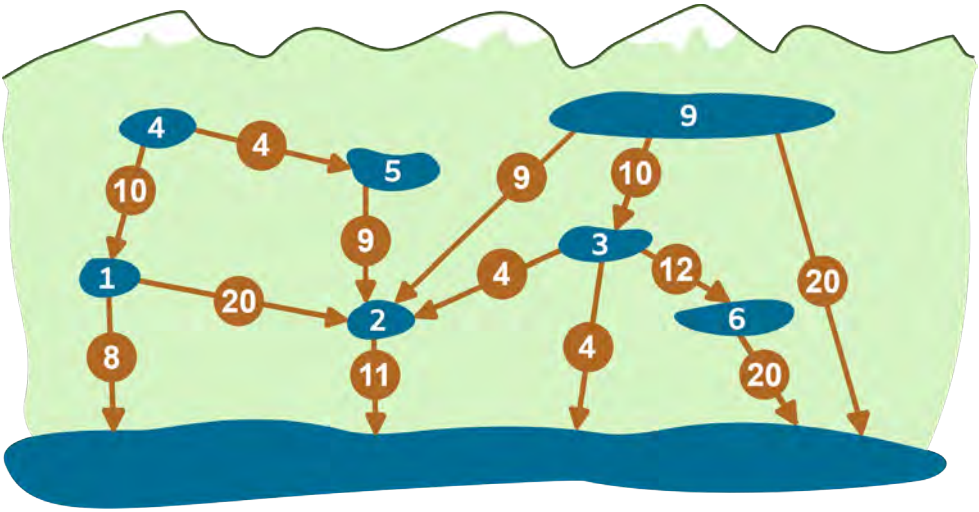
Kalvose šalia Bebrų miesto yra 7 tvenkiniai, kuriuose kaupiasi lietaus vanduo. Per smarkias liūtis šie tvenkiniai gali greitai persipildyti, taip užtvindydami miestą. Kad išvengtų potvynio, miesto valdžia planuoja įrengti naują vamzdyną, kuris sujungtų visus 7 tvenkinius su pagrindiniu miesto rezervuaru. Tikslas – užtikrinti, kad lietaus vanduo iš visų tvenkinių saugiai tekėtų į pagrindinį rezervuarą net ir per smarkią liūtį.

Inžinieriai nustatė galimas vamzdžių trasas tarp tvenkinių porų (ir tarp tvenkinių bei rezervuaro). Kiekvienas siūlomas vamzdis turi tam tikrą pralaidumą – vandens kiekį, kuris gali pratekėti vamzdžiu per valandą – ir kainą, kuri apskaičiuojama vamzdžio pralaidumo reikšmę padauginus iš milijono eurų.

Siekdama sumažinti statybos išlaidas, miesto valdžia nori nutiesti vamzdyną, kuris:

- sujungtų visus 7 tvenkinius su pagrindiniu rezervuaru tiesiogiai arba netiesiogiai;
- susidorotų su tikėtinu vandens srautu per smarkias liūtis;
- kuo mažiau kainuotų nutiesti.

Jūsų užduotis – rasti ekonomiškiausią būdą tokiam vamzdynui nutiesti taip, kad visi tvenkiniai būtų sujungti su rezervuaru ir vamzdžiais galėtų pratekėti pakankamai vandens, kad būtų išvengta potvynių per smarkias liūtis.



Pažymėkite vamzdžius (rodykles), kurie sudarytų pigiausią vamzdyną, užtikrinantį, kad vanduo iš visų tvenkinių saugiai nutekėtų į rezervuarą smarkios liūties metu.

Teisingas atsakymas:

Paaiškinimas

Kadangi dabar tarp tvenkinių nėra jokių vamzdžių, reikia nutiesti visiškai naują vamzdyną, kuris užtikrintų liūčių vandens nutekėjimą į rezervuarą iš visų 7 tvenkinių nesukeliant potvynio. Vandens kiekis, kuris turi nutekėti iš kiekvieno tvenkinio, yra žinomas, o kiekvienas siūlomas vamzdis turi ribotą pralaidumą ir nuo jo priklausomą kainą (kaina = 1 mln. EUR × pralaidumas). Tai iš esmės yra svorinio kryptinio grafo taikymo uždavinys.
Pirmiausia reikia pastebėti, kad 2-as, 5-as ir 6-as tvenkiniai (grafo viršūnės) turi tik po vieną naują išeinančią jungtį, todėl šias jungtis būtina įtraukti, o jų kaina yra 11 + 9 + 20 = 40.
1-as tvenkinys turi dvi išeinančias jungtis. Jei iš 1-o tvenkinio vanduo tekėtų į 2-qjį, tai užblokuotų vandens tekėjimą iš 4-o tvenkinio į 5-qjį (4+5+1+2 = 12, o tai yra daugiau, nei vamzdžio iš 2-ojo tvenkinio į rezervuarą pralaidumas (11)). Vadinasi, vanduo iš 4-o tvenkinio turės tekėti į 1-qjį tvenkinį, o bendra jungčių kaina bus 20 (1→2) + 10 (4→1). Jei iš 1-o tvenkinio vanduo tekėtų tiesiai į rezervuarą, 4-as tvenkinys galėtų būti sujungtas su 5-uoju. Tuomet bendra jungčių kaina būtų 8 (1→rezervuaras) + 4 (4→5) = 12, t. y., mažesnė, nei ankstesnio varianto.
Iš 9-o ir 3-o tvenkinių vanduo galėtų tekėti tiesiai į rezervuarą, o tai kainuotų 20 + 4 = 24. Vis dėlto, iš 9-o tvenkinio vandenį galima leisti į 3-ą tvenkinį, tuomet vanduo bendru srautu, kurio dydis 12, tekėtų per 6-ą tvenkinį į rezervuarą, nes 12 + 6 yra mažiau nei vamzdžio iš 6-o tvenkinio pralaidumas (20). Šios jungtys kainuotų 10 + 12 = 22, t. y., mažiau, nei tiesioginės į rezervuarą, todėl tai yra ekonomiškiausias maršrutas.
Taigi, bendra vamzdyno kaina 40 + 12 + 22 = 74.

Tai informatika!

Tvenkinius ir jungtis galima pavaizduoti kaip grafą su viršūnėmis ir kryptinėmis svorinėmis briaunomis. Lygiai taip pat būtų galima pavaizduoti tinklo pralaidumą ir informacijos srautą kompiuterių tinklų sistemose, pavyzdžiui, internete ar mobiliojo ryšio tinkle.
Didžiausio srauto radimas mažiausiomis sąnaudomis yra tradicinis apribojimų optimizavimo uždavinys, paprastai sprendžiamas grafo kelio paieškos algoritmais. Užuot paeiliui tikrinus visus įmanomus variantus, pagal daugumą šių algoritmų uždavinio sprendimas pradedamas nuo mažiau skaičiavimo sąnaudų reikalaujančio neoptimalaus sprendinio ir bandoma jį nuolat tobulinti siekiant optimalaus sprendinio. Sprendime kaip pirmas žingsnis gali būti naudojamas minimalus jungiamasis medis. Minimalūs jungiamieji medžiai yra naudojami daugeliui grafų analizės uždavinių, susijusių su viršūnių tarpusavio ryšiais, supaprastinti.

ISSN 2783-7599
2025
www.bebras.lt

Afganistanas, Airija, Alžyras, Argentina, Armėnija, Australija, Austrija, Azerbaidžanas, Baltarusija, Belgija, Bolivija, Bosnija ir Hercegovina, Botsvana, Brazilija, Bulgarija, Čekija, Danija, Dominikos Respublika, Egiptas, Ekvadoras, Estija, Filipinai, Gana, Graikija, Honkongas ir Makao, Indija, Indonezija, Iranas, Islandija, Ispanija, Italija, Izraelis, Jamaika, Japonija, Jungtinė Karalystė, Jungtinės Amerikos Valstijos, Juodkalnija, Kamerūnas, Kanada, Kazachstanas, Kinija, Kipras, Kolumbija, Kosovas, Kosta Rika, Kroatija, Kuba, Laosas, Latvija, Lenkija, Libija, Lietuva, Liuksemburgas, Malaizija, Malta, Marokas, Meksika, Mianmaras, Mongolija, Namibija, Naujoji Zelandija, Nyderlandai, Nigeris, Norvegija, Pakistanas, Palestina, Paragvajus, Peru, Pietų Afrikos Respublika, Pietų Korėja, Portugalija, Prancūzija, Puerto Rikas, Rumunija, Saudo Arabija, Senegalas, Serbija, Singapūras, Sirija, Slovakija, Slovėnija, Suomija, Šiaurės Makedonija, Šri Lanka, Švedija, Šveicarija, Tailandas, Taivanas, Tanzanija, Turkija, Ukraina, Urugvajus, Uzbekistanas, Vengrija, Vietnamas, Vokietija