ISSN 2783-7599

b·e·b·r·a·s

Informatikos ir informatinio mąstymo uždavinių rinkinys

Nr. 13

ŠVIETIMO, MOKSLO IR SPORTO MINISTERIJA

VILNIAUS UNIVERSITETAS · 1579 · UNIVERSITAS VILNENSIS

Bebras
International Challenge on Informatics and Computational Thinking

Šiame rinkinyje pateikiama XX informatikos ir informatinio mąstymo konkurso (iššūkio) „Bebras“ I etapo, vykusio 2023 metų rudenį, užduotys jų atsakymai ir paaiškinimai, koks informatikos turinys ir konceptai atskleidžiami, kaip ir kuo uždavinys įdomus ugdant informatinį mąstymą. Visos užduotys (įskaitant grafiką ir kitą medžiagą) licencijuojamos pagal Kūrybinių bendrijų licenciją – „Creative Commons Attribution-ShareAlike 4.0 International License“. Šis užduočių rinkinys skiriamas 1–12 klasių mokinių informatinio mąstymo gebėjimams ugdyti.

„Bebro“ konkursas rengiamas kasmet lapkričio–kovo mėnesiais įvairaus amžiaus vaikų grupėms nuo 6 iki 19 metų. Konkurso tikslas yra skatinti moksleivių domėjimąsi informatika, ugdyti gilesnį technologijų supratimą, skatinti gebėjimą spręsti algoritminius, loginius uždavinius, ugdyti kritinį mąstymą, programavimo ir kompiuterinio raštingumo įgūdžius, taip pat pritraukti daugiau gabių jaunuolių į informatikos studijas.

Dėkojame Daumilui Ardickui, Daivai Gaučytei, Nojui Gudinavičiui, Karoliui Jasučiui, dr. Tatjanai Jevsikovai, Alvidai Lozdienei, Vaidai Masiulionytei-Dagienei, dr. Broniui Skūpui, dr. Gabrielei Stupurienei, Indrai Sudeikienei, Rimantui Žakauskui, talkinusiems verčiant ir adaptuojant užduotis. Taip pat dėkojame tarptautinei „Bebro“ konkurso bendruomenei ir uždavinių autoriams.

Sudarė Lina Vinikienė

Konsultavo Valentina Dagienė

Redagavo Viktoras Dagys

Viršelį kūrė 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 2023 metų „Bebro“ uždavinius sprendė 55 223 mokiniai:

- 2 358 nykštukas (1–2 klasės),
- 4 536 mažyliai (3–4 klasės),
- 17 887 bičiuliai (5–6 klasės),
- 13 247 draugai (7–8 klasės),
- 13 725 jaunių (9–10 klasės),
- 3 470 kolegos (11–12 klasės).

2023 metais Lietuvoje kiekviena amžiaus grupė sprendė po 18 uždavinių, išskyrus nykštukus (12 uždavinių) ir mažylius (15 uždavinių): trečdalis buvo lengvesnių, trečdalis – vidutinio sunkumo ir trečdalis – sunkesnių. Uždaviniams spręsti skiriamos 45 minutės.

Lietuvoje taškai skaičiuojami taip:

- Prieš pradėdamas spręsti, kiekvienas dalyvis turi 54 taškus (mažylių grupėje – 36 taškus; 18 uždavinių × 3 arba 12 uždavinių × 3);
- Už teisingai išspręstą užduotį skiriama 6, 9 arba 12 taškų (priklausomai nuo užduoties sunkumo lygio);
- Už neišspręstą užduotį – 0 taškų;
- Už klaidingą atsakymą atimamas trečdalis užduočiai 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 sistema: lt.bebras.lt

Lentelėje pateikiamas šio rinkinio uždavinių skirstymas pagal amžiaus grupes.

Nr.	Uždavinio identifikatorius	Uždavinio pavadinimas	Nykštukai	Mažyliai	Bičiuliai	Draugai	Jauniai	Kolegos
1	2023-LT-08	Klounas	6					
2	2023-LT-09	Kelionė per upę	6					
3	2023-LT-10	Slėpynės	6					
4	2020-IE-09_2023	Dubenėliai	6					
5	2023-CZ-03	Obuolių pusės	9	6				
6	2023-DE-06	Vanduo ir žemė	9	6				
7	2023-LT-01	Bebrų rikiuotė	9	6				
8	2023-SK-04	Ūkis	9	6				
9	2023-AT-01	Diena zoologijos sode	12	9	6			
10	2023-CH-01	Skėtis	12	9	6			
11	2023-DE-02	Gėlių puokštė	12	9	6			
12	2023-SA-01	Kamuoliai	12	9	6			
13	2023-IE-05	Amžiaus kodai		6				
14	2023-IN-01	Aplikacija		9	6			
15	2023-CA-01	Stebuklingas medis		12	9	6		
16	2023-DE-04	Karolio svajonių namas		12	9	6		
17	2023-SK-02	Morkų sėjimas		12	9	6		
18	2023-CA-02	Išpūstžandžiai		12	9			
19	2023-US-03	Savaeigis automobilis		12		6		
20	2023-LT-02	Nuotrauka			6			
21	2023-AU-01b	Draugiška kaimynystė			9	6		
22	2023-UY-01	Logikos lobis			9	6		
23	2023-AU-05a	Bebrų nešuliai			12	9	6	
24	2023-CH-05	Rąstų saugykla			12	9	6	
25	2023-CZ-02	Geležinkelio modelis			12	9	6	
26	2023-IN-03b	Traukinio iškrovimas			12	9	6	
27	2023-PE-02	Tomas ir jo kaimynai			12	9	6	
28	2023-SK-07	Arčiau ar toliau?			12	9	6	
29	2023-BR-05	Sulčių vežimėliai				12	9	6
30	2023-CZ-01	Atsitrenkiantys robotai				12	9	6

Nr.	Uždavinio identifikatorius	Uždavinio pavadinimas	Nykštukai	Mažyliai	Bičiuliai	Draugai	Jauniai	Kolegos
31	2023-IE-02b	Ogamo abėcėlė				12	9	6
32	2023-US-01	Maršrutų skaičiavimas				12	9	6
33	2023-ME-03b	Naujas žaislas				12	9	
34	2023-IR-02	Darbų paskirstymas				12		6
35	2023-HU-37	Bebrų statybos kompanija					9	6
36	2023-BE-01	Emos užduotys					12	9
37	2023-DE-08	Zeroboto dilema					12	9
38	2023-NZ-01	Tilto statyba					12	9
39	2023-RO-02	Postfiksinis užrašas					12	9
40	2023-TW-04	Į kalnus					12	9
41	2023-UA-01	Elektroninė spyna					12	9
42	2013-DE-02	Ornamentai						12
43	2016-AT-06	Rekursinis dažymas						12
44	2017-RU-05	Šifro „nulaužimas“						12
45	2019-KR-04	Batų pirkimas						12
46	2023-DE-01	Konfliktų detektorius						12
47	2023-DE-09	Domino						12

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

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

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

1. Klounas

Mikas rado plakatą su klounais. Išsirinko vieną labiausiai patikusį. Surask jį:

Pasirinktas klounas nėra:

- žaliais plaukais;
- liūdnu veidu;
- su kamuoliu.

Paaiškinimas

Teisingas atsakymas:

Kituose piešiniuose esantys klounai atitinka tik vieną arba dvi sąlygas (žaliais plaukais; liūdnu veidu; su kamuoliu).

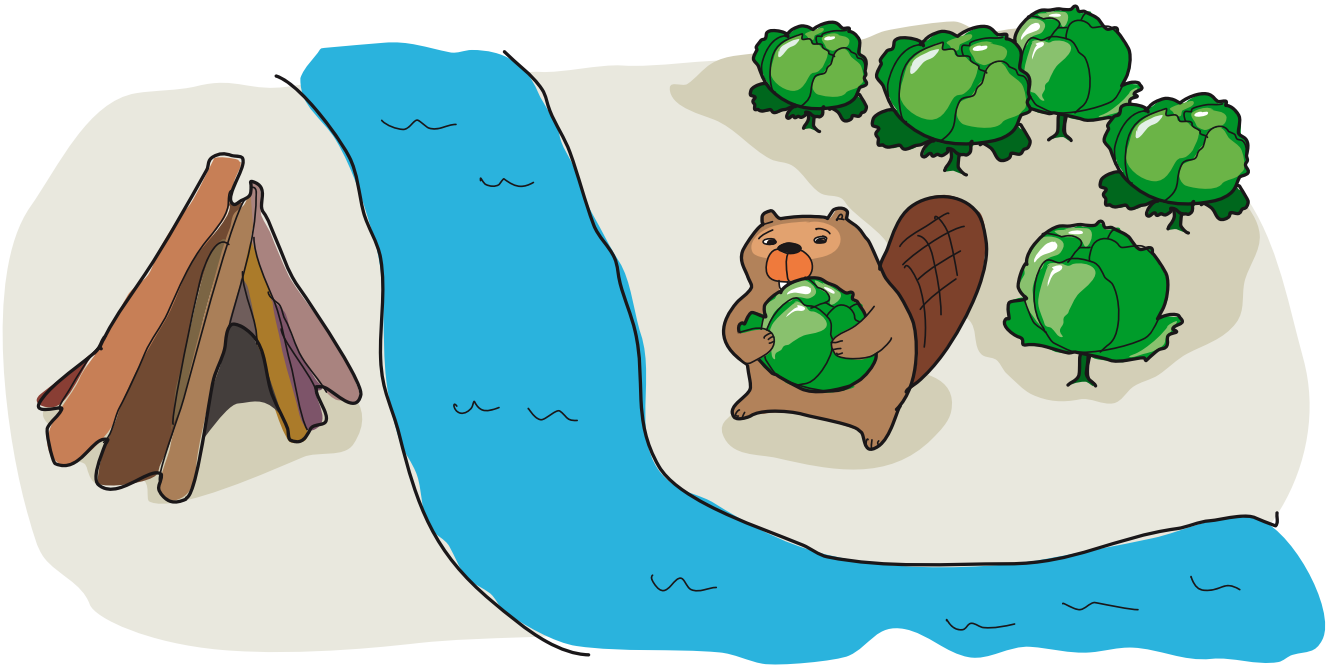
Taigi vienintelis klounas  atitinka visas tris sąlygas.

Tai informatika!

Tai logikos algebros uždavinys. Sprendimui turime pritaikyti vieną iš loginių operacijų – sujungimą (IR). Operacijos sekų sąjunga yra teisinga tik tokiu atveju, jei visos operacijos yra teisingos.

2. Kelionė per upę

Bebriuko darže užaugo kopūstai. Vienu metu jis gali plukdyti tik vieną kopūsto galvą. Kiek kartų bebriukas turi perplaukti upę, kad iš daržo parsigabentų dvi kopūsto galvas?



Paaiškinimas

Teisingas atsakymas: 3.

Kadangi bebras vienu metu gali plukdyti tik vieną kopūsto galvą, jis iš daržo pirmiausia parsigabena vieną kopūstą, grįžta atgal į daržą ir parplukdo antrą kopūstą.

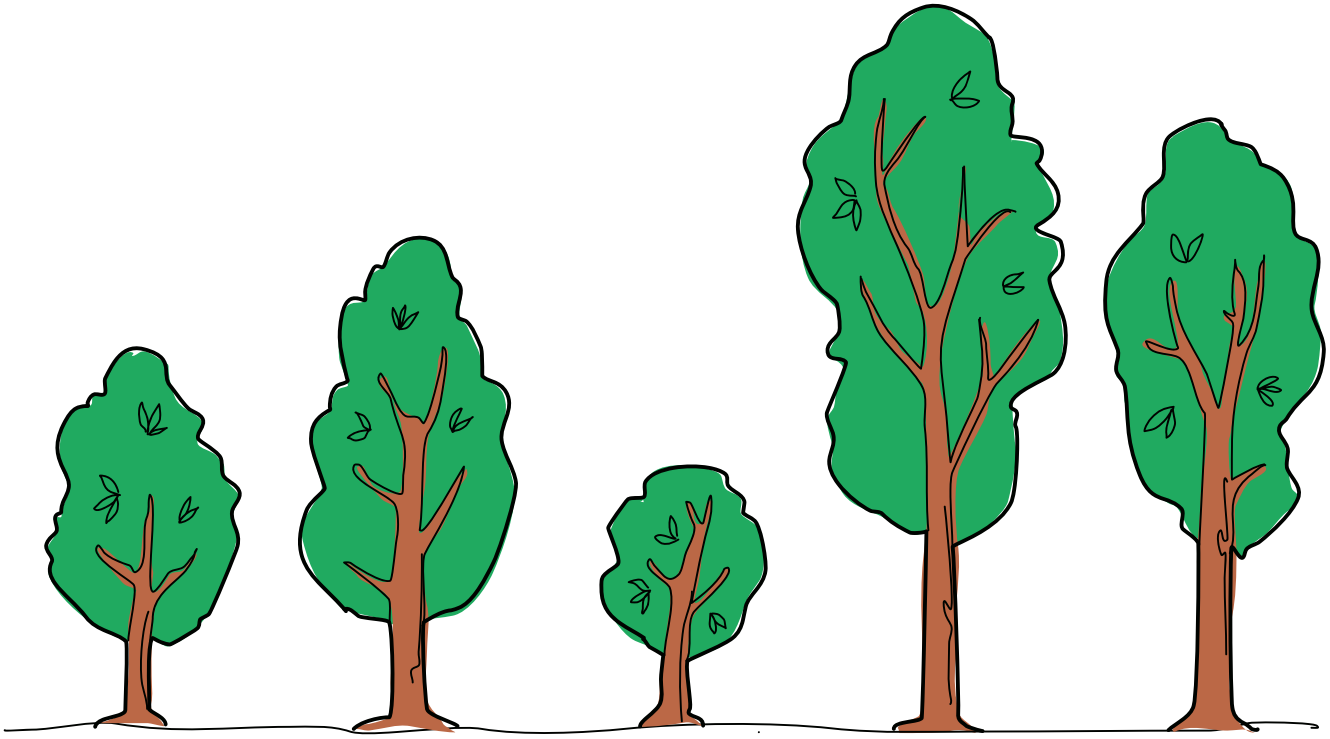
Tai informatika!

Šioje užduotyje atliekamas veiksmo algoritmas, kuris susideda iš kelių žingsnių.

3. Slėpynės

Vaikai miške žaidė slėpynių. Visi pasislėpė už skirtingų medžių. Trumpas vardas – žemas medis. Kuo ilgesnis vardas, tuo aukštesnis medis.

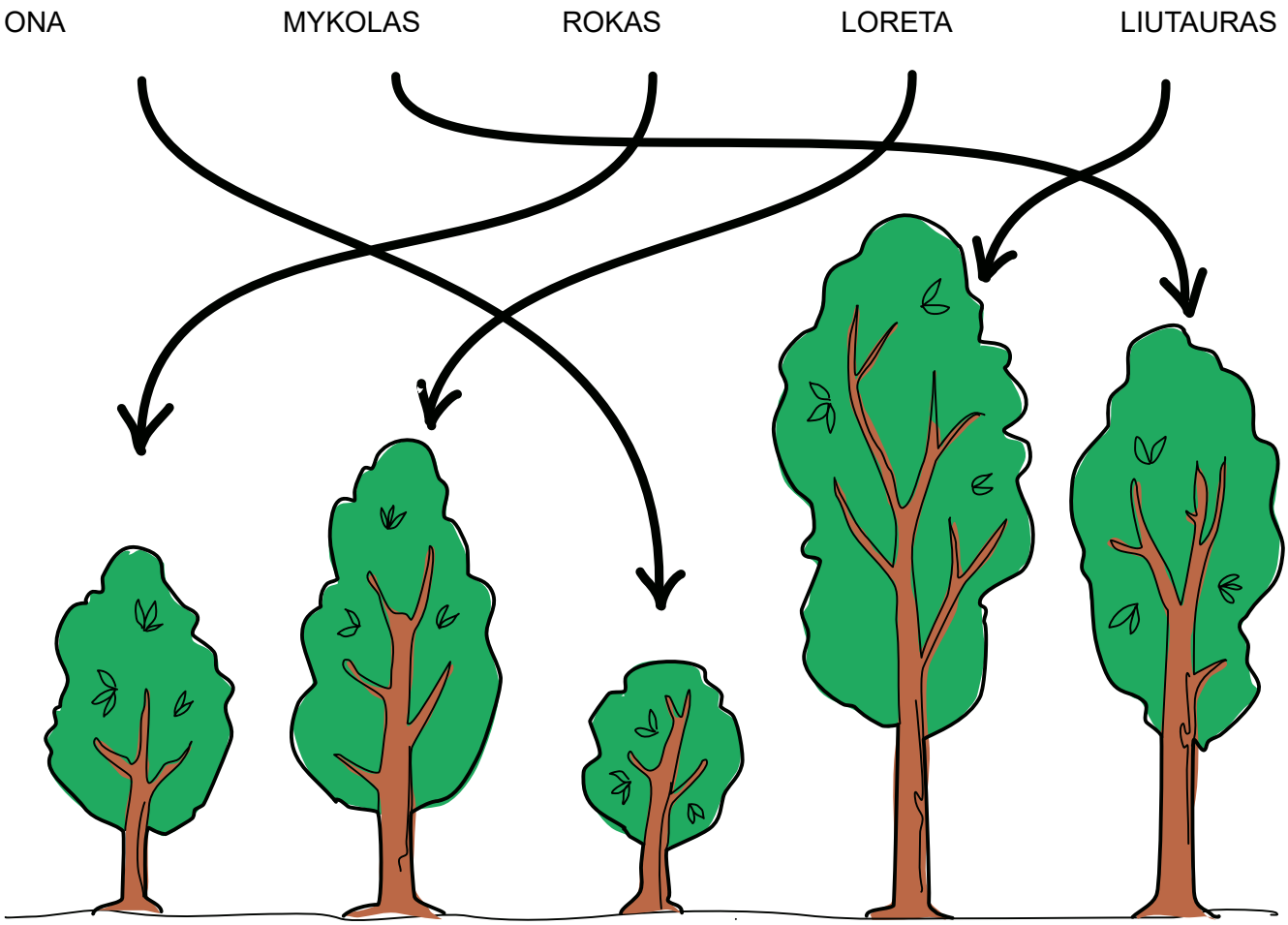
Kur slepiasi vaikai? Nutempk vardus į langelius po medžiais.



ONA MYKOLAS ROKAS LORETA LIUTAURAS

Paaiškinimas

ATSAKYMAS:



Varde ONA yra trys raidės, tai trumpiausias vardas, todėl sujungiame su žemiausiu medžiu. Rokas – 5 raidės, Loreta – 6 raidės, Mykolas – 7 raidės, tad sujungiame su atitinkamais medžiais pagal jų aikštį. Liutauras – 9 raidės: sujungiama su aukščiausiu medžiu.

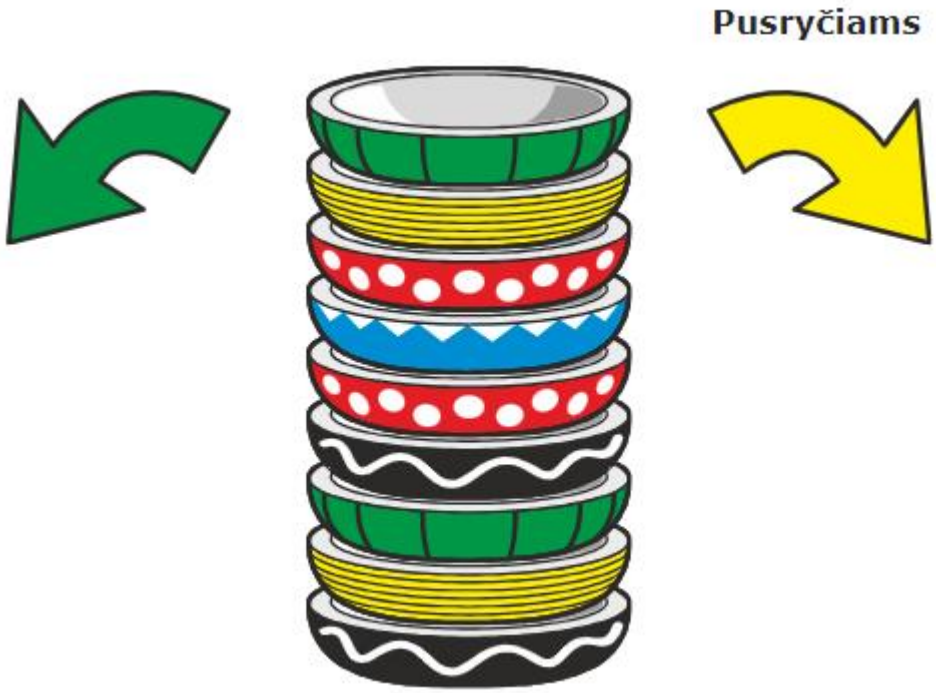
Tai informatika!

Uždavinio esmė – žodžio ilgio ir medžio aukščio susiejimas.

4. Dubenėliai

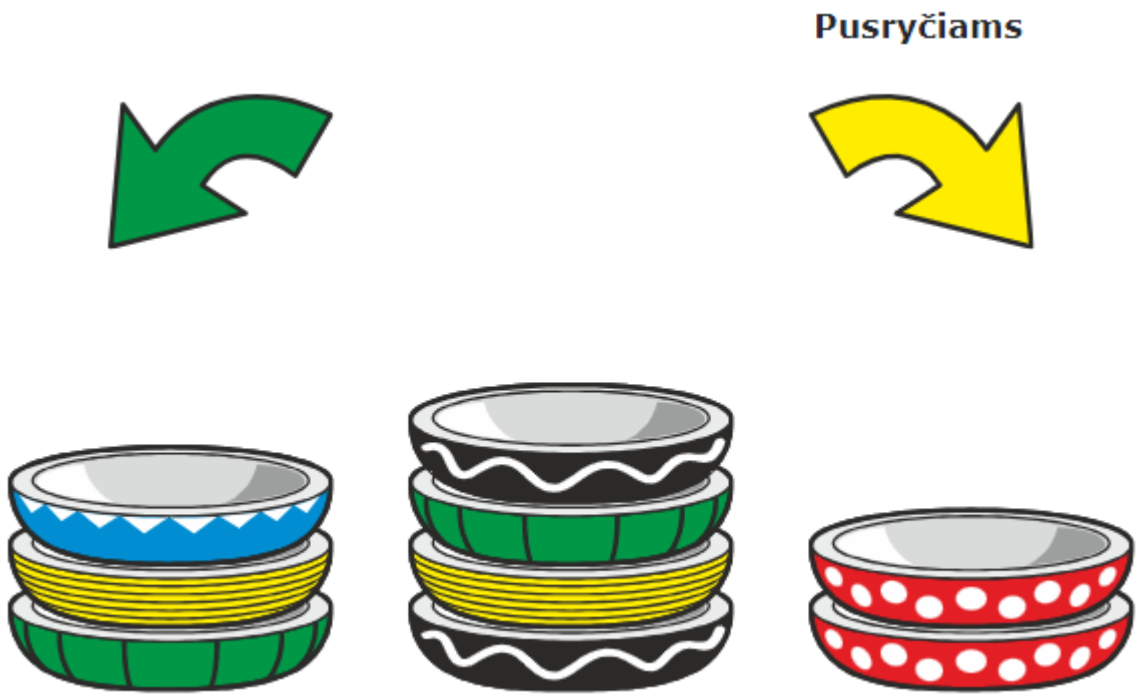
Dvi sesės nori valgyti iš vienodų dubenėlių. Dubenėlį galima imti tik nuo krūvos viršaus.

Spausdami rodykles sudėkite lygiai du vienodus dubenėlius į krūvelę „Pusryčiams“. Nuo pradinės krūvelės imkite kuo mažiau dubenėlių.



Paaiškinimas

Teisingas atsakymas:

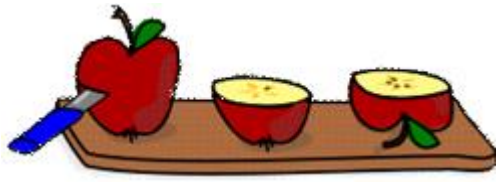


Tai informatika!

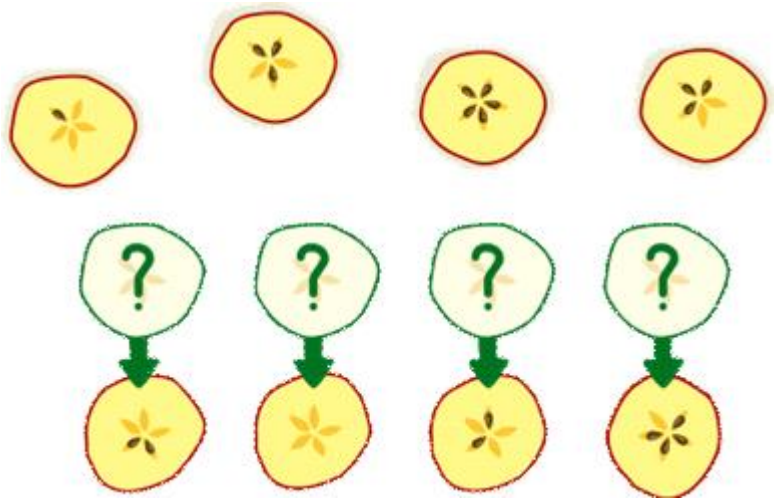
Dėklas (stekas) yra dažnai naudojama duomenų struktūra. Yra taisyklės, kaip dėti daiktus į dėklą ir kaip juos pašalinti iš dėklo. Šiame uždavinyje mums reikia tik paimti daiktus iš dėklo. Taisyklė tokia, kad vienu metu galima paimti tik vieną daiktą nuo dėklo viršaus. Jei norima gauti devintąjį dėklo dubenėlį, vadinasi, turime nuimti devynis viršutinius dubenėlius. Taip pat svarbu, kur sudėti nereikalingus aštuonis dubenėlius – tai taip pat informatikos uždavinys. Jei mes turime antrą dėklą ir dėklai gali būti bet kokio aukščio, tai dubenėlių perkėlimo procesą galėtume suprogramuoti – informatikoje viskam stengiamasi parašyti programas. Taip yra todėl, kad dviem dėklais modeliuojama Tiuringo mašina – labiausiai nagrinėjamas teorinis kompiuteris. Taigi ši paprasta krūvelė – dėklas – iš tiesų yra labai galinga!

5. Obuolių pusės

Kalėdų paprotys – stalą papuošti skersai perpjautų obuolių pusėmis taip, kad pjūvyje matytųsi iš sėklų lovelių sudaryta žvaigždutė. 4 vaikai perpjovė 4 obuolius ir padėjo ant stalo. Dalis sėklų liko vienoje obuolio pusėje, kitos – kitoje. Deja, obuolių pusės ant stalo susimaišė.

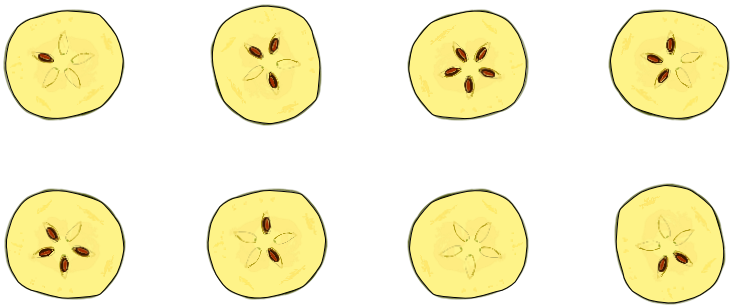


Suraskite tinkamas obuolių puses ir nutempkite ant klaustukų.

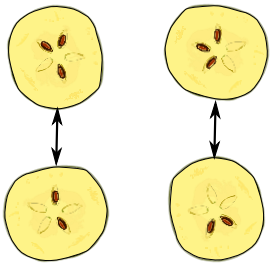


Paaiškinimas

Teisingas atsakymas:



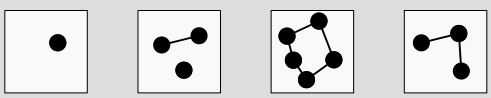
Pastebime, kad viename obuolyje yra 5 sėklos. Tačiau vien suskaičiuoti sėklas nepakanka. Du obuolius su vienodu sėklų skaičiumi vienoje pusėje galima atskirti pagal pilnų ir tuščių sėklų lovelių išsidėstymą. Vienos obuolio pusės pilni sėklų loveliai turi derėti prie kitos pusės tuščių lovelių, pavyzdžiui, jei vienoje obuolio pusėje keli iš eilės einantys loveliai pilni, tai atitinkamai turime rasti obuolio pusę, kurioje tiek pat iš eilės einančių lovelių būtų tušti.



Tai informatika!

Kai kurie realaus gyvenimo objektai turi derėti tarpusavyje, pavyzdžiui, DNR nukleobazių A-T ir C-G poros arba spyna ir jos raktas. Informatikoje kiekvienam slaptažodžiu apsaugotam failui (spynai) yra slaptažodis jam atverti (raktas). Programuojant kai kurioms funkcijoms reikia numatyto skaičiaus parametrų – tam kompiliatorius turi patikrinti, ar funkcijos kreipinyje pateiktų parametrų skaičius (ir jų tipai) sutampa su tuo, kas numatyta funkcijos apraše.

Sprendžiant šią užduotį reikia rasti vieno obuolio dvi puses – jos turi derėti tarpusavyje. Elemento papildymo radimas ir topologinių modelių atpažinimas yra pagrindiniai šios užduoties sprendimo įgūdžiai. Kiekvienai obuolio pusei galime sudaryti grafą, kuriame taškas reiškia sėkla užpildytą lovelį, o briauna – dviejų pilnų lovelių kaimynystę. Štai taip atrodytų 1-osios obuolių pusių eilės grafai:



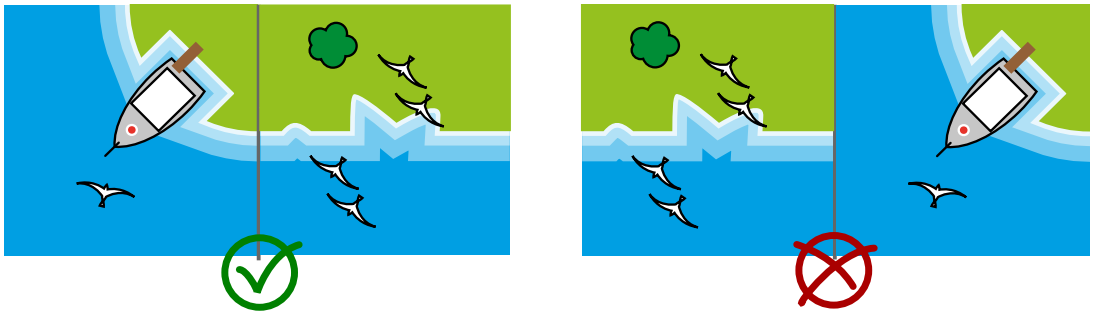
Panašius grafus galime sukurti ir 2-ajai eilei su perpjautais pilnais ir tuščiais sėklų loveliais (taškas reiškia tuščią sėklų lovelį, o briauna – dviejų tuščių sėklų lovelių kaimynystę). Tuomet lengviau palyginti kiekvienos eilės grafus ir rasti kiekvienam jų atitikmenis.

6. Vanduo ir žemė

Ema dėlioja paveikslą iš detalių su vandens ir žemės vaizdais.

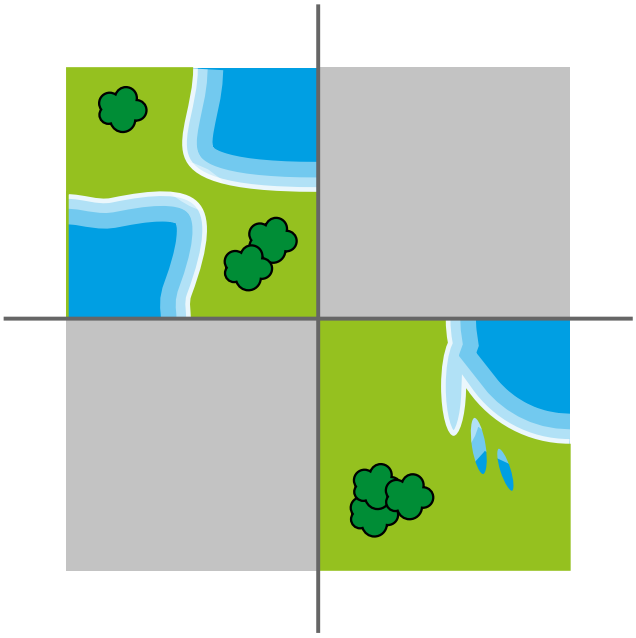
Gretimų detalių žemė turi liestis su žeme, o vanduo – su vandeniu.

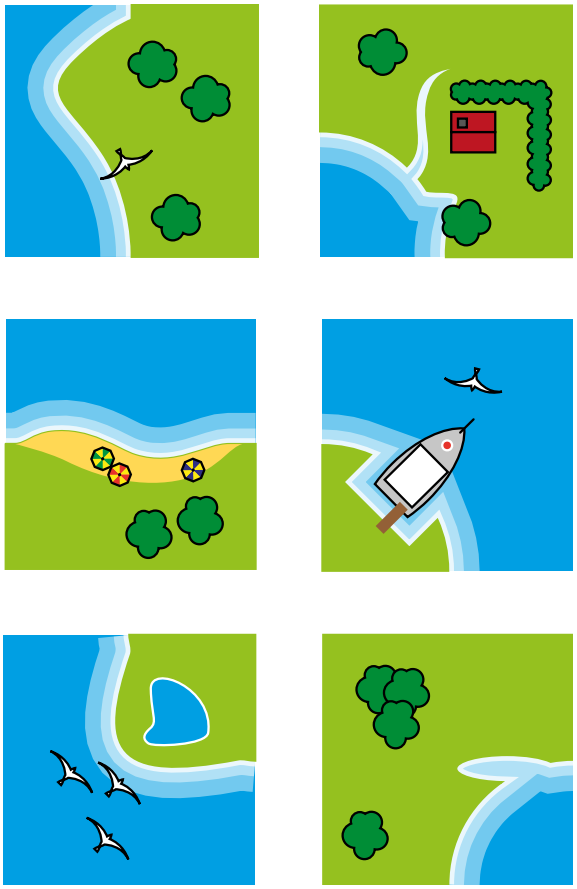
Pateikiame tinkamo ir netinkamo detalių dėliojimo pavyzdžius:



Ema nori sudėti paveikslą iš keturių detalių. Ji jau padėjo dvi detales.

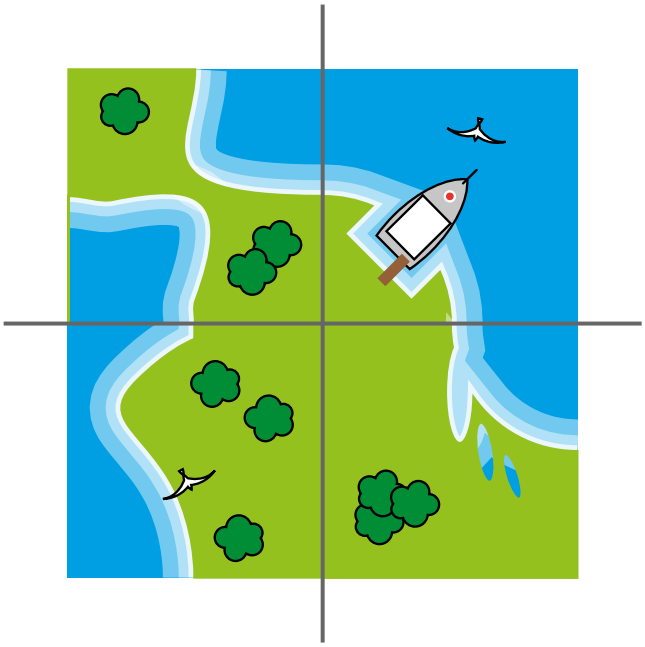
Padėkite Emai sudėlioti paveikslą nutempdami ir įdėdami dvi trūkstamas detales.





Paaiškinimas

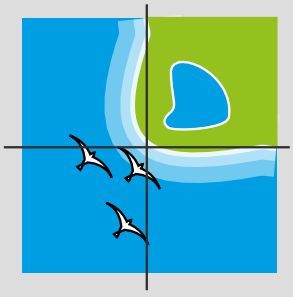
Teisingas atsakymas:



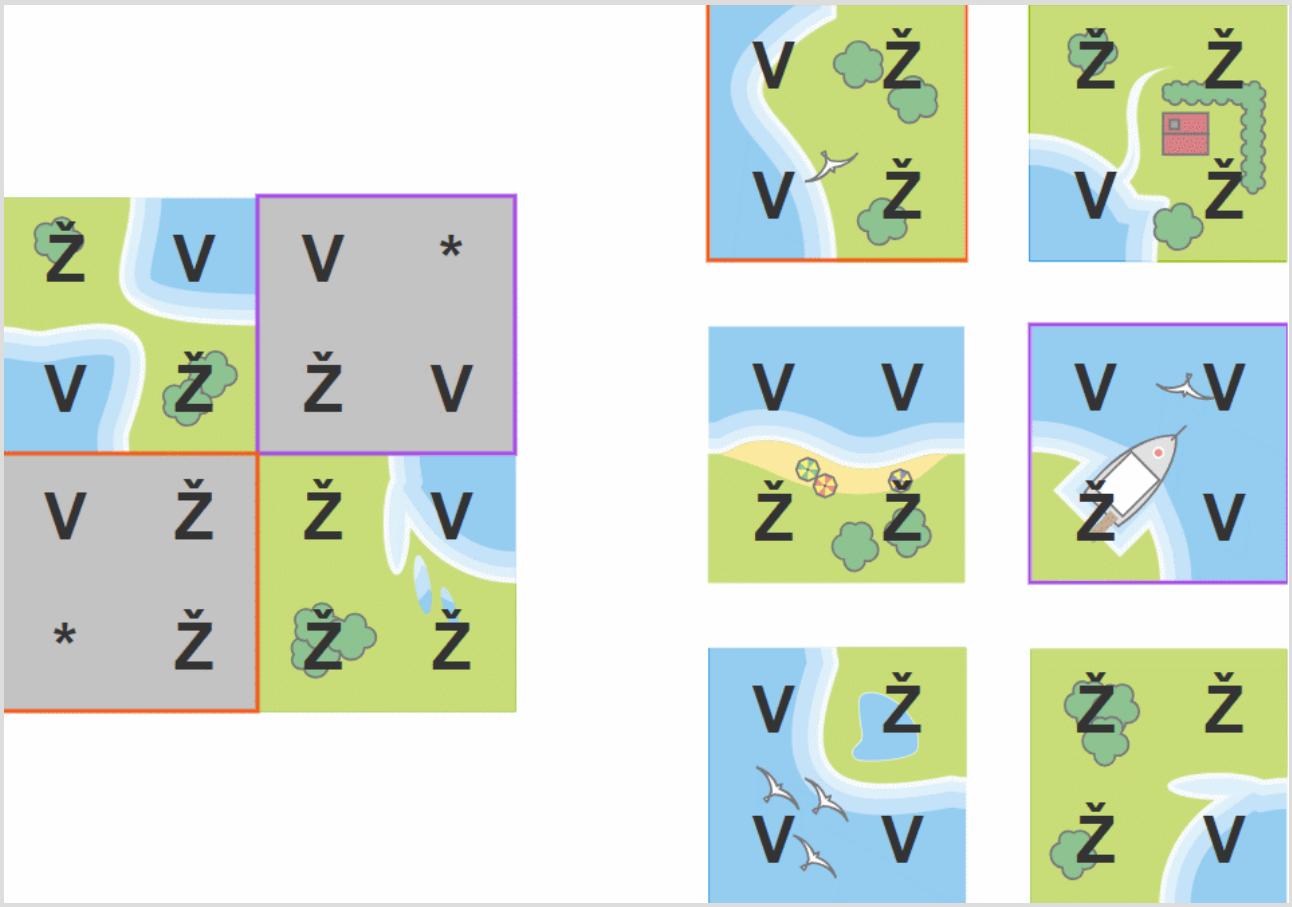
Matome, kad šis atsakymas teisingas – gautas vientisas paveikslas: kiekviena detalė dera su dviem gretimomis detalėmis. Spręsdami šią užduotį turime situacijas nagrinėti sistemingai – į trūkstamą vietą bandyti įdėti kiekvieną iš šešių detalių. Dar reikia įsitikinti, kad galimas tik vienintelis teisingas atsakymas.

Tai informatika!

Pažvelkime atidžiau į Emos turimas detales. Pirmiausia pastebėkime, kad detales galima sužymėti pagal tai, kas vaizduojama prie jų kraštinių – vanduo ar sausuma.



Yra dvi galimybės: raide V žymėsime kraštinių atkarpas, rodančias vandenį, o raide Ž – žemę.



Aišku, kad dvi detalės dera, jei jų gretimų kraštinių atkarpos pažymėtos tomis pačiomis raidėmis. Vadinasi, daugeliu atveju galima automatiškai atrinkti tinkamas detales. Detalių išoriniai kampai mums nerūpi, pažymime žvaigždute (*). Šitaip kiekvienai trūkstamai detalei sudarome pildymo šabloną. Visos detalės turi atitikti šiuos šablonus. Kiekvienam šablonui tinka tik viena detalė.

Apibendrindami galime pasakyti, kad radome ypatingą detalių vaizdų savybę ir tuo pasinaudojome, pakeisdami vaizdus raidėmis V ir Ž. Atlikdami šį žingsnį gerokai sumažinome vaizduose esančią informaciją – sutelkėme dėmesį į tai, ko reikia šiai užduočiai išspręsti. Informatikos mokslininkai vaizdų keitimą simboliais vadina vaizdų modeliais. Modeliavimas grindžiamas abstrakcija, kuri mažina informacijos kiekį. Kompiuteriai apdoroja tikrovę atspindinčius modelius. Kuriant bet kuriuos modelius labai svarbu pasirūpinti, kad nebūtų prarastos svarbios tikrovės savybės.

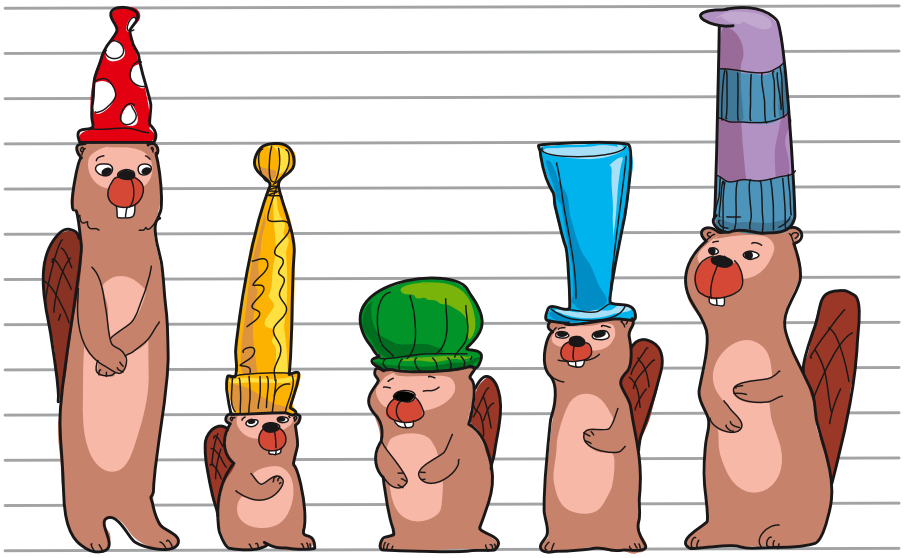
7. Bebrų rikiuotė

Bebrams buvo padovanotos naujos kepurės.

Dešiniau stovintys bebrai turi turėti aukštesnes kepures nei stovintys kairiau.

Išrikiuokite bebrus jų kepurių aukščio didėjimo tvarka. Nutempkite bebrus į reikiamą vietą.

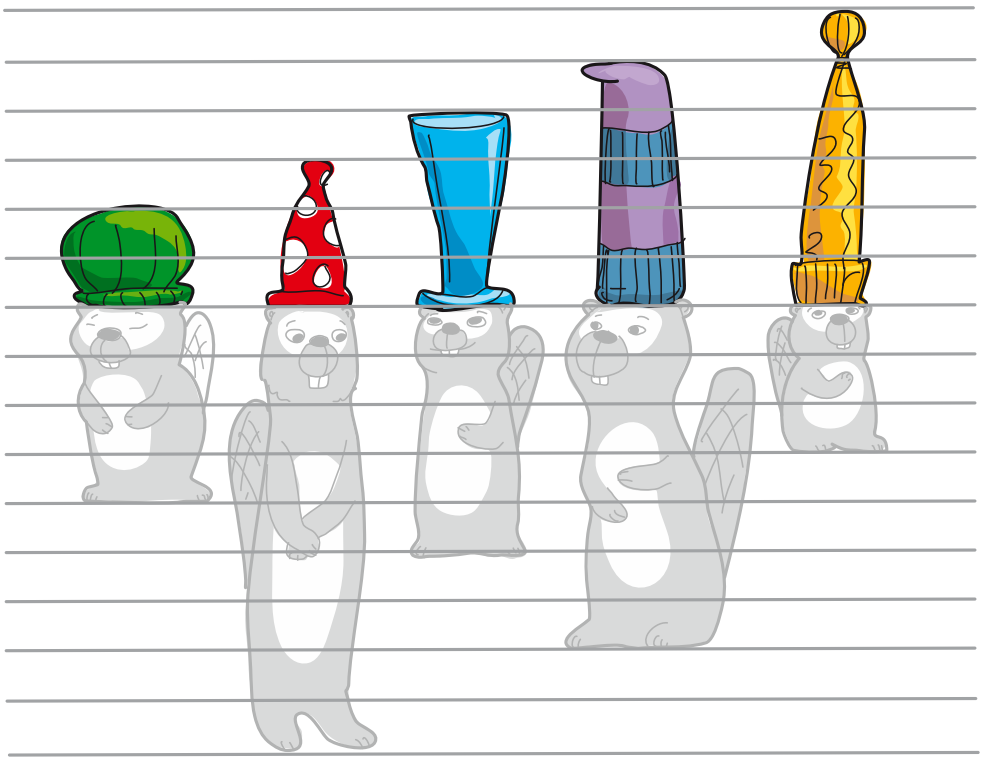
Žemiausia



Aukščiausia

Paaiškinimas

Teisingas atsakymas:



Tai informatika!

Informatikoje duomenų išdėstymas tam tikra sutarta tvarka vadinamas rikiavimu. Tai objektų išdėstymas į eilę pagal kurį nors jų parametrą: skaičių – pagal jų dydį, žodžių – pagal raidžių rikiavimo eilę abėcėlėje ir pan. Rikiuoti galima dvejopai: didėjančiai arba mažėjančiai.

Jeigu objektai skaidomi į kelias grupes, sakoma, kad jie rūšiuojami. Rūšiavimas gali būti panaudotas kaip pagalbinis rikiavimo veiksmas. Objektai surūšiuojami į grupes (rūšis), po to grupės sujungiamos ir gaunama surikiuotų objektų eilė. Atkreipiame dėmesį, kad anglų kalboje neskiriamos rikiavimo ir rūšiavimo sąvokos.

Yra įvairių rikiavimo metodų. Labiau žinomi: burbulinis, spartusis, sąlajinis rikiavimo metodai. Jie skiriasi sudėtingumu ir rikiavimo sparta, kuri taip pat priklauso nuo rikiuojamų duomenų pradinio sutvarkymo. Vieni duomenys gali būti sparčiau surikiuojami vienu metodu, kiti – kitu. Pastebėkime, kiek daug dalykų rūšiuojama ar rikiuojama mūsų kasdieniame gyvenime – spintelės vandens parke, įrašai žodynuose ar telefonų knygose. Nesurikiavus žodyno įrašų sunku būtų rasti reikiamą žodį. Net interneto naršyklės užklausos rezultatai yra rikiuojami pagal jų svarbą. Beveik visų sporto varžybų rezultatai yra surikiuoti

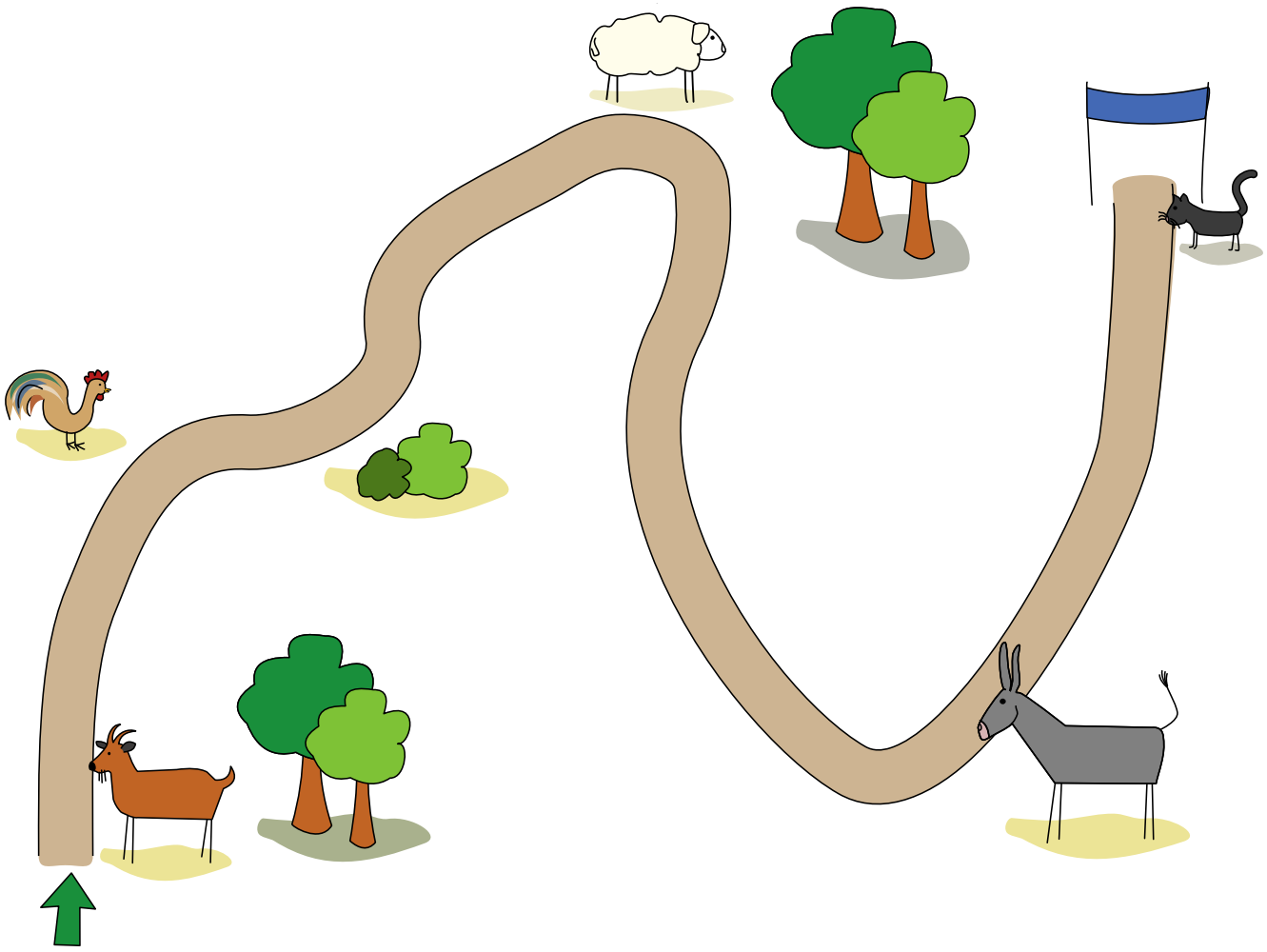
pagal pasiekimus. Visais šiais atvejais surikiuotus elementus lengviau tvarkyti, ypač jei elementų yra daug, rūšiavimo ar rikiavimo pranašumas yra akivaizdus.

Informatinis mąstymas

Pirmiausia taikome informatinio mąstymo vieną iš pagrindinių konceptų – analizę. Reikia išanalizuoti užduotį ir išsiaiškinti, kad turime atskirti bebrų aukštį nuo jų kepurių aukščio. Tada analizuojant duomenis (bebrų kepurių seką) reikia rasti pertvarkymų seką. Idėja, kad kepurių perkėlimas (aukštesnių į dešinę, o žemesnių – į kairę) priartina prie norimo tikslo ir yra pirmas žingsnis kuriant algoritmą, taigi remiamės algoritminiu mąstymu. Analogiškus algoritmo žingsnius kartojame tol, kol visi bebrai bus surikiuoti.

8. Ūkis

Mažasis Marius su seneliais vaikštinėdami po fermą matė įvairių gyvūnų. Senelis nupiešė kelią, kuriuo ėjo.



Kuris teiginys apie pasivaikščiojimą yra neteisingas?

A. Paskutinę jie matė katę.

B. Pirmiausia jie pamatė ožką.

C. Antras gyvūnas buvo asilas.

D. Po gaidžio matė avį.

Paaiškinimas

Teisingas atsakymas: C.

Tik C teiginys (antras gyvūnas, kurį jie matė, buvo asilas) yra klaidingas. Asilas buvo ketvirtas gyvūnas, jie jį matė po avies.

Kiti teiginiai teisingai apibūdina paveikslėlyje pateiktą informaciją.

Tai informatika!

Šioje užduotyje daugiausia dėmesio skiriama darbui su duomenų pateikimu ir teiginiuose pateiktos informacijos teisingumo palyginimui. Paveikslėlyje saugoma informacija parodo atliktų žingsnių seką (pasivaikščiojimo metu matyti gyvūnai). Šią informaciją galime pateikti ir įvairiomis kitomis struktūromis. Klausimuose pateikiami keturi teiginiai, informatikas programuodamas dažnai susiduria su panašiomis situacijomis.

Informatinis mąstymas

Sprendžiant uždavinį svarbu gebėti sutelkti dėmesį į pagrindinį klausimą ir nesigilinti į nesvarbius dalykus. Šiuo atveju turime teiginiuose saugomą tekstinę informaciją palyginti su vaizdu. Turime suprasti teiginius, priskirti objektams (gyvūnams) teisingus pavadinimus, nustatyti jų eiliškumą kelyje ir galiausiai nustatyti klaidingą teiginį.

9. Diena zoologijos sode

Liepa nori praleisti dieną zoologijos sode. Iš skrajutės ji sužino apie daugybę įvairių veiklų, nurodytas kiekvienos veiklos pradžios ir pabaigos laikas.

Liepa nori sudalyvauti kuo daugiau veiklų, tačiau, suprantama, vienu metu gali būti tik vienoje veikloje. Dalyviai negali išeiti viduryje veiklos. Kadangi kai kurie užsiėmimai persidengia, Liepai nepavyks dalyvauti visose veiklose.

Kurias veiklas Liepa turi pasirinkti, kad dalyvautų kiek įmanoma daugiau veiklų?

Paaiškinimas

Per visą dieną Liepa gali sudalyvauti ne daugiau kaip 5-iose veiklose.

Yra du tokių pasirinkimų variantai:

Vienas iš šios užduoties sprendimo būdų – išbandyti visus įmanomus nesikartojančių veiklų derinius. Kiekvienas toks derinys pateikia mums veiklų tvarkaraštį. Tada lieka patikrinti, kuriame tvarkaraštyje yra daugiausia veiklų (kuris tvarkaraštis optimalus). Toks procesas vadinamas išsamiąja (ar visapusiška) paieška.

Išsamioji paieška dažnai yra pernelyg sudėtinga, nes numatoma per daug galimybių, kurias reikia patikrinti. Be to, nelengva įsitikinti, kad visos galimybės (variantai) patikrintos.

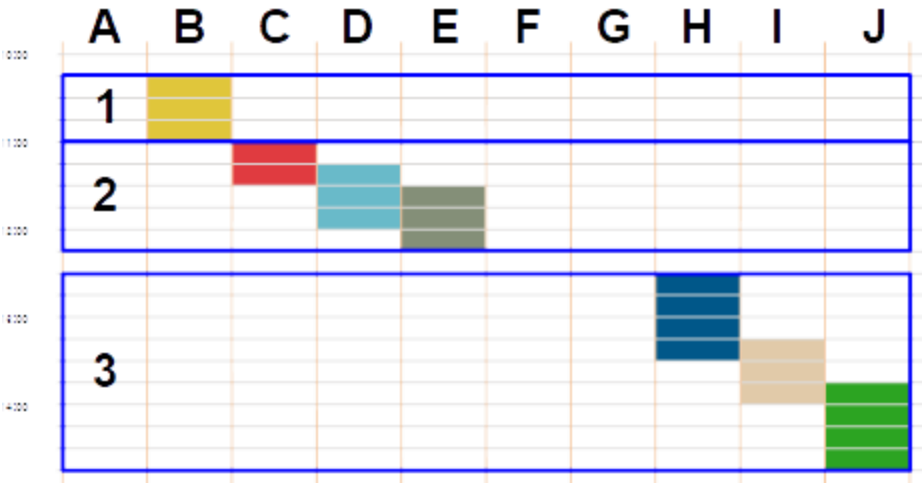
Norėdami supaprastinti užduotį turime panagrinėti veiklų laikus. Jei dvi veiklos sutampa, tai imame tik vieną iš jų. Jei viena veikla visa trunka kitos veiklos metu, tuomet geriau rinktis trumpesnę veiklą.

Pavyzdžiui, B veikla vyksta A veiklos metu ir yra trumpesnė. Todėl geriau nesirinkti A veiklos, nes ji gali sutapti su didesniu skaičiumi kitų veiklų.

Remdamiesi tais pačiais argumentais galime nesirinkti F veiklos, nes E veikla visiškai sutampa su F veikla.

Taip pat galime atmesti G veiklą, nes H veikla visiškai sutampa su G veikla.

Atmetus šias tris veiklas (A, F ir G), galime dekomponuoti (suskaidyti) likusias veiklas į tris nepriklausomas grupes. Galime rasti optimalų visos dienos tvarkaraštį, pirmiausiai surasdami optimalų tvarkaraštį kiekvienai grupei.

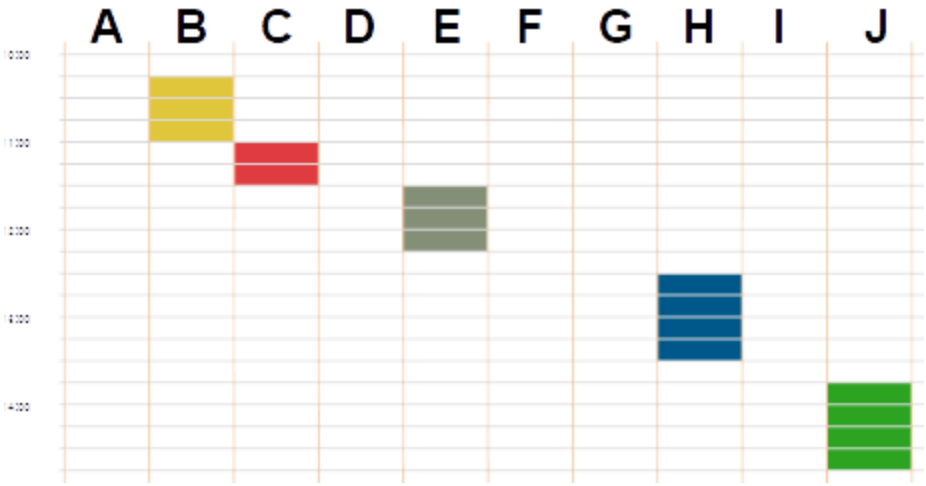


1-oje grupėje yra tik viena veikla, todėl optimalus tvarkaraštis yra pasirinkti ją.

2-oje grupėje yra trys veiklos rūšys. Atlikus išsamią paiešką, šiai grupei galioja tik du tinkami tvarkaraščiai: dalyvauti arba C ir E veiklose, arba tik D veikloje. Optimalus tvarkaraštis yra dalyvauti C ir E veiklose.

Panašiai optimalus 3-ios grupės tvarkaraštis yra dalyvauti H ir J veiklose.

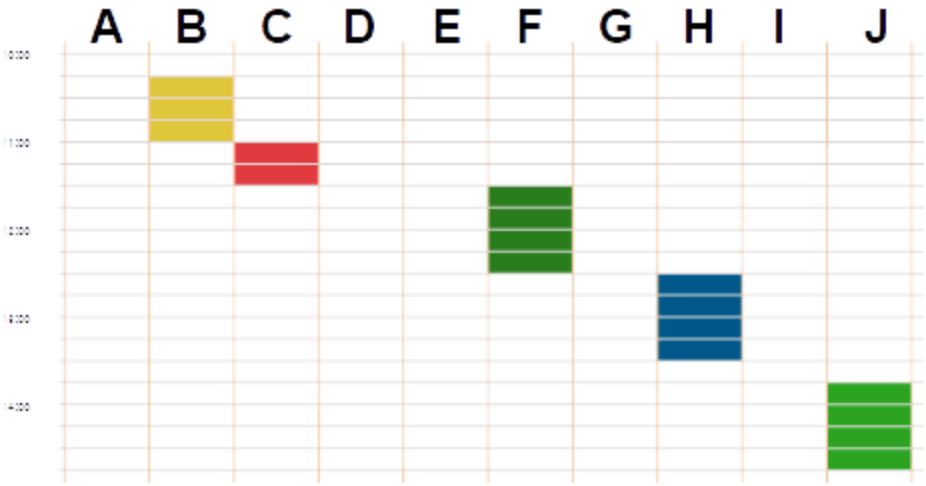
Sujungus optimalius tvarkaraščius, gauname tokį visos dienos tvarkaraštį:



Šiame tvarkaraštyje yra 5 veiklos rūšys ir mes žinome, kad tai yra optimalu. Kitaip tariant, žinome, kad Liepa negali dalyvauti daugiau nei 5 veiklose per vieną dieną.

Pažvelgus į veiklas, kurias atmetėme supaprastindami užduotį, matyti, kad A veikla sutampa su B ir C veiklomis, todėl jos negalima pasirinkti nesumažinus bendro veiklų skaičiaus. Panašiai G veikla sutampa su H veikla, todėl jos taip pat negalima pasirinkti nesumažinus bendro veiklų skaičiaus.

Tačiau F veikla sutampa tik su E veikla, todėl kitą optimalų tvarkaraštį (t. y., tvarkaraštį, kuriame yra 5 veiklos rūšys) galima gauti pasirinkus F veiklą vietoj E veiklos:



Tai informatika!

Dažnai informatikoje kuriant algoritmus uždaviniui spręsti lengviausia taikyti išsamųjį paieškos algoritmą, nes jis gana paprastas ir garantuotai duoda teisingą atsakymą. Pagrindinė problema ta, kad išsamiai paieškai atlikti rankiniu būdu arba kompiuteriu gali

prireikti per daug laiko. Tenka ieškoti būdų, kaip sukurti veiksmingesnius algoritmus, kurie padėtų išspręsti uždavinius per trumpesnį laiką. Kad algoritmas būtų efektyvesnis, reikia atidžiai išnagrinėti uždavinį, kad būtų galima jį supaprastinti. Uždavinys, kuriame reikia maksimizuoti arba minimizuoti rezultatą, vadinamas optimizavimo uždaviniu. Supaprastindami optimizavimo uždavinį turime galvoti, kokius sprendimus galime priimti, kad jie netrukdytų gauti optimalų sprendinį.

Spręsdami zoologijos sodo uždavinį galėjome nepaisyti akvariumo šviesos šou, koalų maitinimo ir dramblių maudynių, nes galime būti tikri, kad bet kurį tvarkaraštį, kuriame yra šių veiklų, galima pagerinti vietoj jų pasirinkus kitą veiklą. Kita strategija, leidžianti padidinti algoritmo efektyvumą, yra uždavinio skaidymas į nepriklausomas dalis, dažnai vadinamas použduviniais. Kompiuteriai tikrai gerai moka atlikti tą pačią užduotį daug kartų su skirtingais duomenimis, todėl sudaryti algoritmą, kurį būtų galima taikyti nepriklausomiems použduviniams, yra perspektyvi strategija. Mūsų spręstas uždavinys – iš kelių veiklų per tam tikrą laikotarpį pasirinkti didžiausią jų skaičių – vadinamas veiklos atrankos uždaviniu.

Informatinis mąstymas

Sprendžiant šią užduotį būtina sukurti maksimalaus sprendinio paieškos strategiją. Tai galima padaryti analizuojant skirtingų veiklų persidengimą ir nustatant taisykles, kurios veiklos turi būti neįtrauktos, o kurios įtrauktos į optimalų sprendimą. Tai tipiškas informatinio mąstymo metodas, kai sudarytos taisyklės taikomos tam tikra tvarka. Jei panagrinėsime sprendimo aprašymą, rasime uždavinio skaidymą į mažesnes dalis, kurios gali būti sprendžiamos savarankiškai – tai labai veiksminga didelių uždavinių sprendimo strategija. Kitas sprendimo būdas gali būti toks: sudaryti visus įmanomus dalyvavimo veiklose derinius ir pasirinkti tuos, kuriuose dalyvaujamų veiklų skaičius yra didžiausias. Šis algoritmavimo metodas apima daugybę kombinacijų, todėl jį sudaro daugiau žingsnių nei sprendžiant anksčiau aprašytu metodu.

10. Skėtis

Štai taip atrodo Onutės skėtis:

Tik viename iš šių paveikslų yra Onutės skėtis. Kuriame?

A

B

C

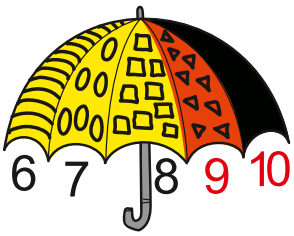
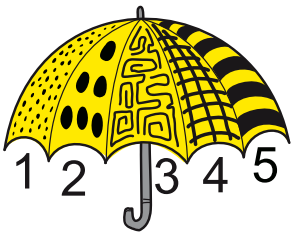
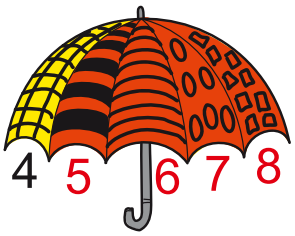
D

Paaiškinimas

Kiekvienas raštas Onutės skėtyje naudojamas tik vieną kartą. Todėl kiekvieną paveikslą galime palyginti su Onutės skėčiu:

- paveiksle pasirenkame kairiausią skėčio raštą ir randame jo vietą Onutės skėtyje;
- patikriname, ar šie raštai atitinka Onutės skėtį.

Kiekviename paveiksle matome tik penkių raštų seką. Taigi negalime nuspręsti, ar paveikslas atitinka visų dešimties Onutės skėčio raštų seką. Tačiau tik C paveiksle matoma penkių raštų seka visiškai atitinka Onutės skėtį. Taigi tik C paveikslas atitinka Onutės skėtį. Visuose kituose paveiksluose raštų sekos tik iš dalies sutampa su Onutės skėčio raštų seka, todėl šie paveikslai negali vaizduoti Onutės skėčio.

	A	B	C	D
Atsakymo variantai				
Onutės skėtis				

Tai informatika!

Šioje užduotyje pateikiama tik dalinė informacija apie atsakymų variantuose nurodytus skėčių modelius. Nepaisant to, galime nustatyti, kuriame paveiksle pavaizduotas Onutės skėtis: galime pažvelgti į keturias raštų sekas ir pastebėti, kad tik viena iš jų sutampa su Onutės skėčiu.

Toks pat principas, kaip ir skėčio paieškos atveju, taikomas ieškant tekstinio dokumento. Kompiuteris, naudodamasis pateikta daline informacija (paieškos seka iš raidžių), ieško

dokumente sutampančių raidžių sekų. Galima pastebėti, kad kuo ilgesnė paieškos seka, tuo mažiau atitikmenų rasite ir tuo didesnė tikimybė rasti ieškomą dokumento vietą. Kuo trumpesnė ieškoma seka, tuo paieška mažiau tiksli, bet greitesnė. Informatikoje paieškos algoritmai buvo sukurti (ir vis dar kuriami nauji, tobulinami) siekiant kuo greičiau gauti tinkamiausius rezultatus. Šie algoritmai gali atlikti paiešką dideliame kiekyje duomenų, pavyzdžiui, naudojant gerai pažįstamus internetinės paieškos įrankius.

Informatinis mąstymas

Šią užduotį galima spręsti naudojant dekompoziciją. Dekompozicija – tai uždavinio skaidymas į mažesnes dalis arba etapus. Norėdami rasti penkių modelių atitikmenų seką, galite ją išskaidyti į žingsnius, aprašytus atsakymo paaiškinime.

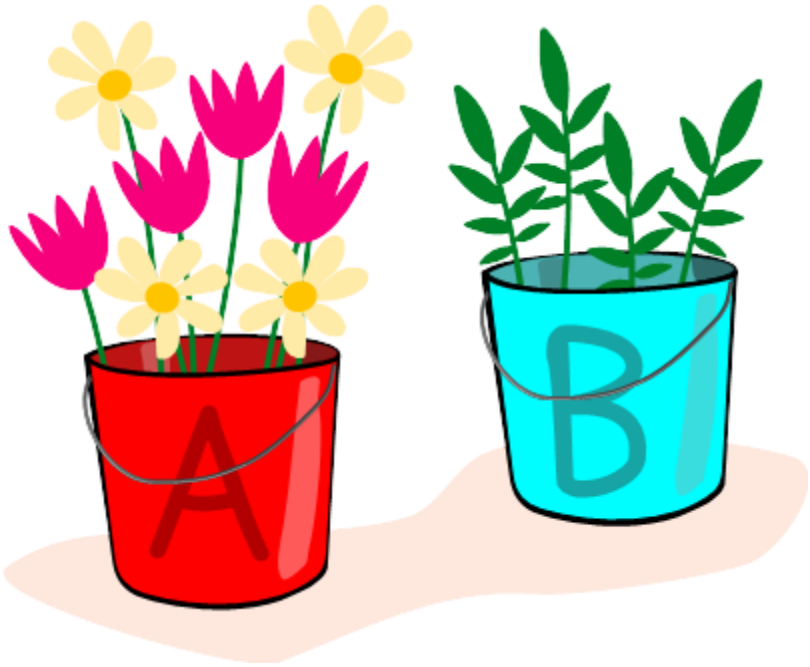
Šį uždavinį taip pat galime laikyti praktišku modelių atpažinimo pratimu. Modelių atpažinimas – tai gebėjimas atpažinti dėsningumus, pasikartojimus ar panašumus duomenų rinkinyje. Norint surasti Onutės skėčio paveikslą, reikia patyrinėti sutampančias raštų sekas. Vėliau, programuojant, dėsningumų atpažinimas padeda taikyti modulinį požiūrį, kai tenka spręsti pasikartojančius dalinius uždavinius.

11. Gėlių puokštė

Florijonas parduoda gėlių puokštes. Puokštę jis sudaro iš eilės taikydamas šias taisykles:

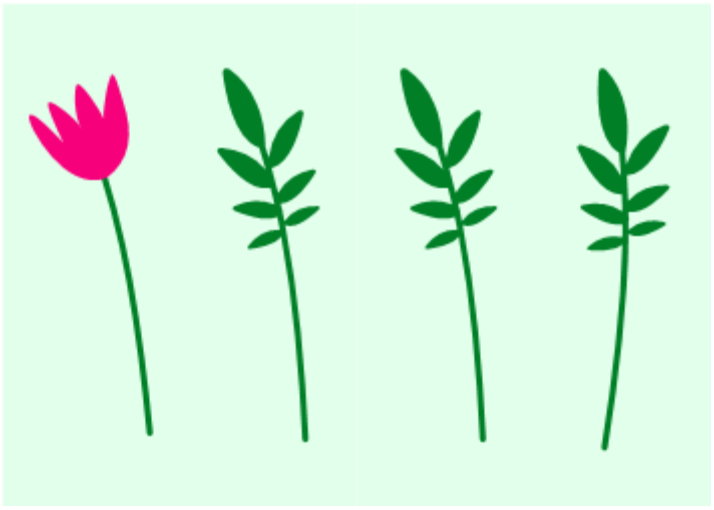
1. Išsirenka gėlę iš A vazos.
2. Jei išsirinko saulutę  , paima dar vieną saulutę  .
3. Paima bent vieną šakelę  iš B vazos ir šią taisyklę kartoja, kol sudaro puokštę iš 4 vienetų.

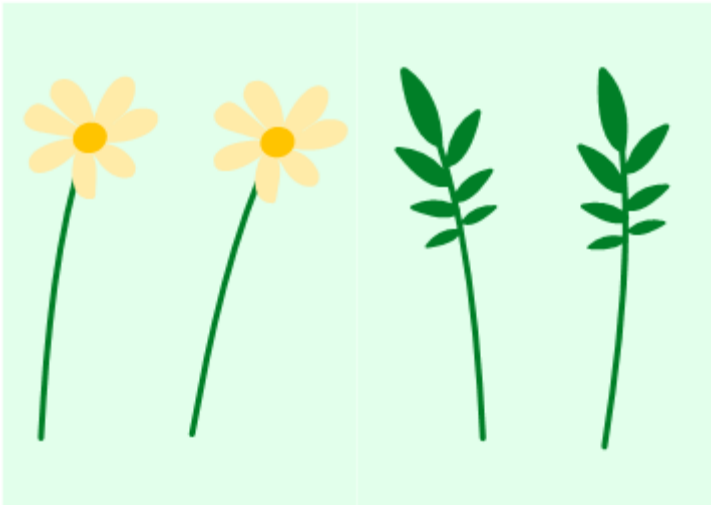
Kaip galėtų atrodyti puokštė, sudaryta iš turimų gėlių ir šakelių?



Paaiškinimas

Yra du teisingi sprendimai:





Kad sukomponuotumėme puokštę teisingai, turime laikytis pateiktų trijų nurodymų (taisyklių).

Vadovaudamiesi pirmuoju nurodymu, galime rinktis saulutę arba tulpę.

Laikydamiesi antrojo nurodymo, renkamės kitą saulutę (jei jau turime saulutę), kad turėtumėme dvi, arba neimame nė vienos gėlės, todėl lieka viena tulpė.

Vykdydami trečiąjį nurodymą, jei turime tulpę, tai pridedame tris šakeles, arba, jei turime dvi saulutes, tai pridedame dvi šakeles, kad gautumėme lygiai 4 vienetus.

Tai informatika!

Nurodymai, kaip sukomponuoti puokštę, yra labai aiškūs ir gali būti pateikti mašinai atlikti. Informatikoje tai vadinama algoritmu. Šioje užduotyje yra keletas tipiškų instrukcijų (komandų), kurios pateikiamos daugelyje kompiuterių programų.

- Pirmoji instrukcija yra atsitiktinis pasirinkimas iš objektų rinkinio.
- Antroji instrukcija vadinama sąlyginiu sakiniu. Komanda „išrink dar vieną saulutę“ vykdoma tik tada, jei tam tikra sąlyga yra teisinga. Ši sąlyga – „pirmoji gėlė yra saulutė“.
- Trečioji instrukcija atrodo paprasta, tačiau naudojama gana sudėtinga sąvoka – ciklas. Čia tam tikras veiksmas, t. y., „paimk šakelę iš antrosios vazos“, kartojamas kelis kartus, kol tenkinama sąlyga. Sąlyga tenkinama, kai puokštėje yra keturi vienetai.

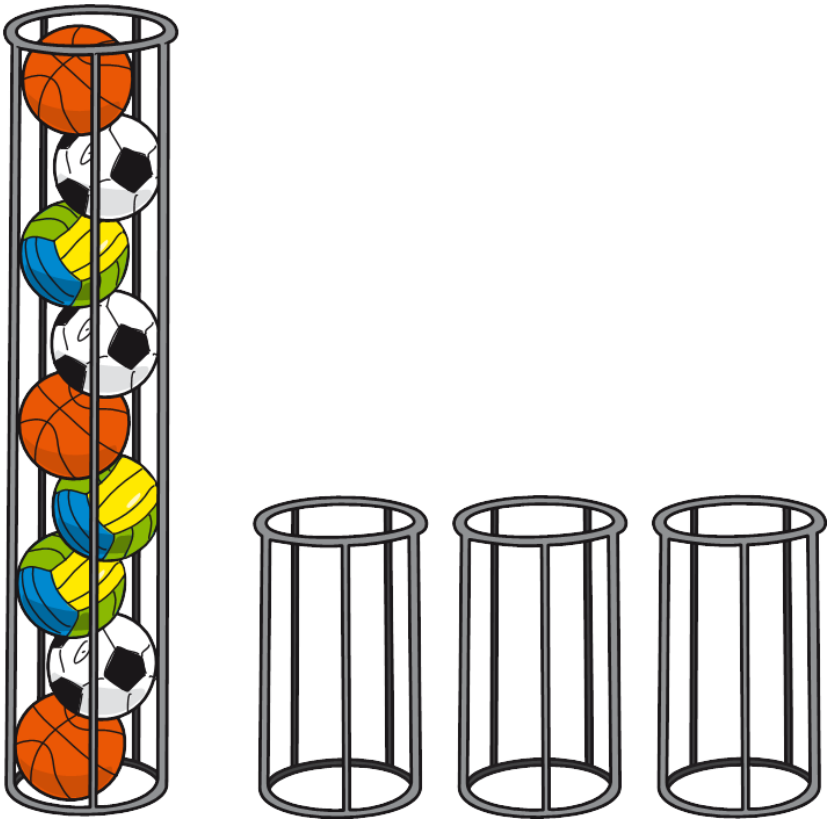
Floristika yra menas ir profesija. Egzistuoja įvairios tradicijos, kuriose yra taisyklių, kaip komponuoti puokštes. Ši užduotis rodo, kad profesinės floristikos žinios apima ir algoritmus.

Informatinis mąstymas

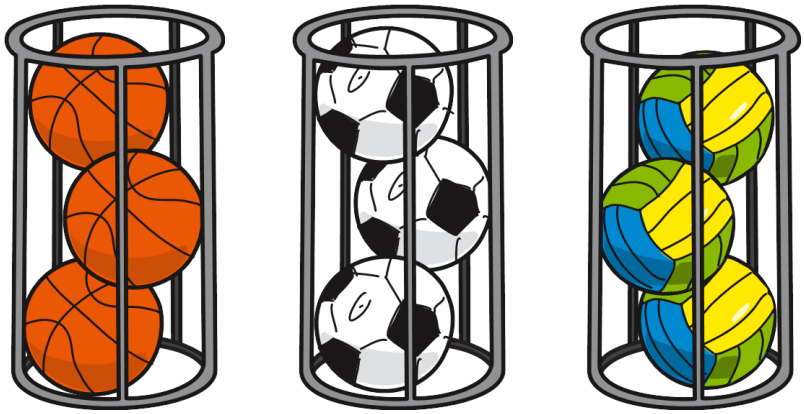
Norint išspręsti šią užduotį reikia suprasti ir vykdyti algoritmą. Be to, ši užduotis parodo, kad algoritmai būdingi ir kultūrinio gyvenimo reiškiniams. Vertinimas taip pat yra šios užduoties dalis, nes pabaigoje turite turėti keturis vienetus puokštėje, ir tai negali būti dvi tulpės ar viena saulutė ir viena tulpė.

12. Kamuoliai

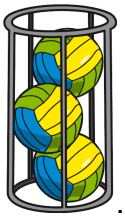
Vienoje didelėje vamzdžio formos talpykloje sudėti 9 kamuoliai. Jie yra trijų rūšių. Ugnė nori kiekvienos rūšies kamuolius sudėti į atskiras talpyklas. Rūšiuodama Ugnė ima po vieną kamuolį iš didžiosios talpyklos ir deda į atskiras mažas talpyklas.



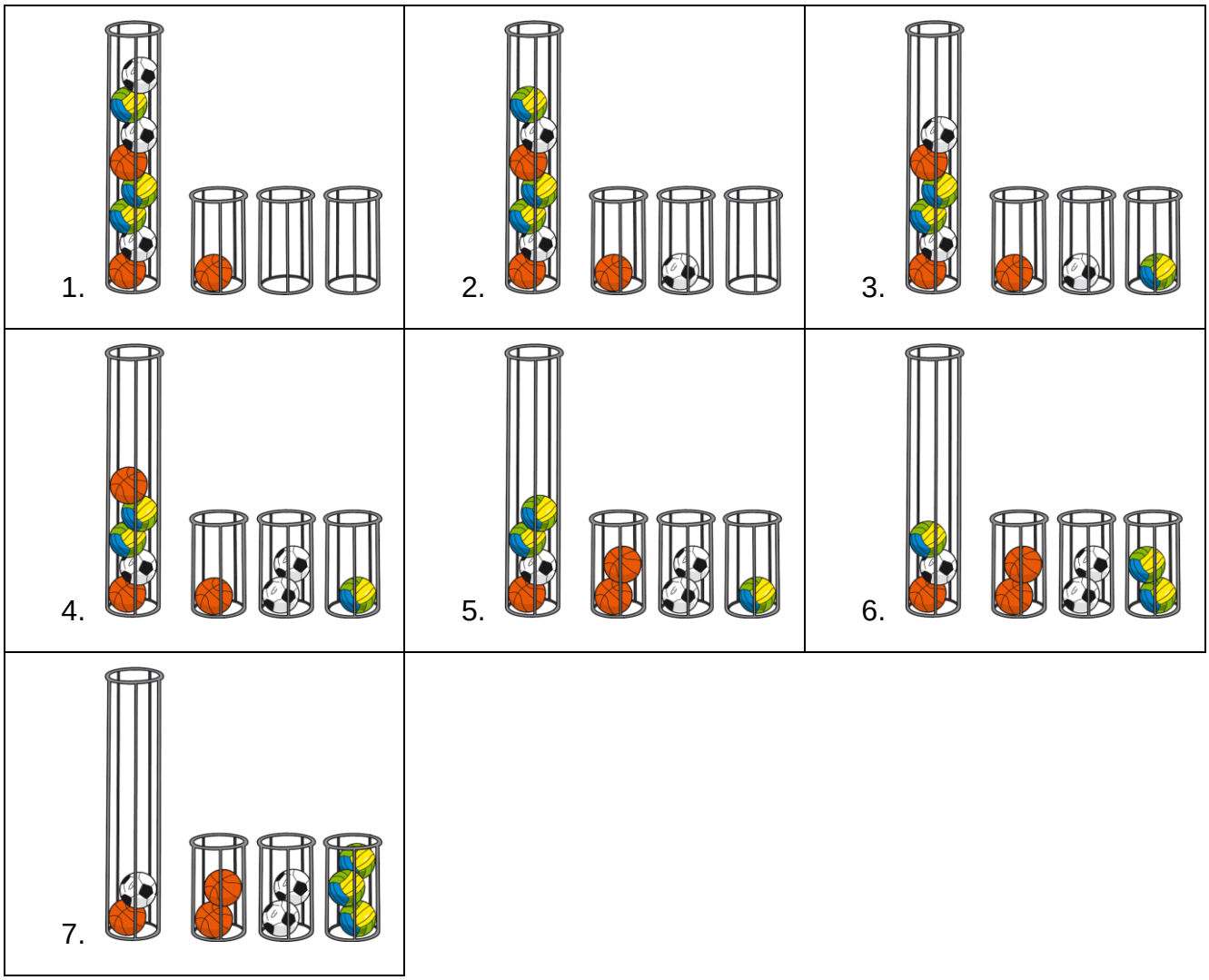
Kurią kamuolių talpyklą Ugnė užpildys pirmiausia?



Paaiškinimas

Teisingas atsakymas:  .

Ugnė ima kamuolius po vieną iš eilės nuo talpyklos viršaus iki apačios. Aišku, kad kažkuri mažoji kamuolių talpykla bus užpildyta tada, kai į ją bus įdėtas trečias tos pačios rūšies kamuolys. Imant kamuolius iš talpyklos po vieną nuo viršaus, trečias tos pačios rūšies kamuolys yra tinklinio kamuolys. Kamuolių dėjimas į talpyklas parodytas žingsniais – rezultatas pasiekiamas septintuoju žingsniu.



Tai informatika!

Iš talpyklos traukiami kamuoliai imant juos nuo viršaus. Informatikoje tokia duomenų struktūra vadinama dėklu (angl. *stack*). Dėklas yra tiesinė duomenų struktūra, naudojama duomenims laikyti įdedant (angl. *push*) arba išimant (angl. *pop*) elementus iš vieno talpyklos galo, vadinamo dėklo viršumi.

Yra daug realių dėklo pavyzdžių: lėkščių ar knygų krūva, iš kurios elementus imame nuo viršaus, taigi apačioje esantis elementas bus paimtas paskutinis.

Informatinis mąstymas

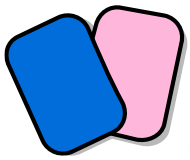
Kamuoliams iš talpyklos išimti reikia naudoti ciklą (kartojimo komandą). Norint rasti trečią kurios nors rūšies kamuolį, reikia naudoti sąlyginį sakinį.

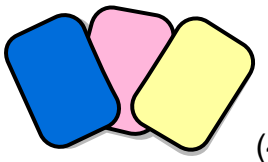
Norint atpažinti ir sudėti drauge tos pačios rūšies objektus (kamuolius) reikia taikyti informatikos metodą – šablonų (modelių) atpažinimą. Šablonų atpažinimas – tai duomenų analizės metodas, kai mašininio mokymosi algoritmai naudojami sistemoms apmokyti pagal mokymo duomenų rinkinį. Tada šios sistemos gali ieškoti panašumų ir sąsajų su kitais duomenimis ir greitai bei tiksliai surūšiuoti objektus.

Šios užduoties sprendimas remiasi analogiškų veiksmų (imti kamuolius) kartojimu, kad būtų surūšiuoti visi kamuoliai. Toks sprendimo procesas priskiriamas algoritminiam mąstymui. Algoritminis mąstymas – tai sprendimo būdas, kai sukuriama sisteminga ir logiška žingsnių seka, kurią būtų galima kartoti ir išspręsti tiek pateiktą, tiek panašius uždavinius.

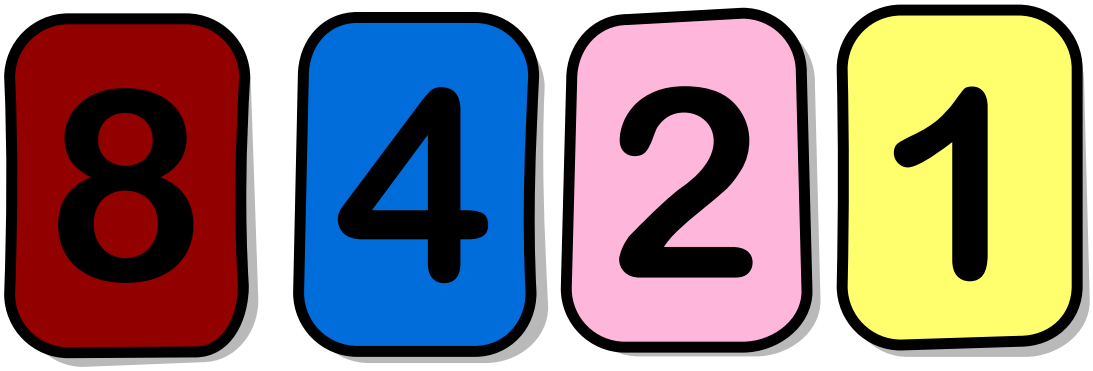
13. Amžiaus kodai

Zita demonstruoja Viliui, kaip parodyti jų amžių naudojant spalvotas korteles.

Zitai yra 6 metai, todėl ji sudėlioja šias korteles:  (4 + 2 = 6).

Viliui yra 7 metai ir jis sudėlioja tokias korteles:  (4 + 2 + 1 = 7).

Sesei Karolinai 9 metai. Kokias korteles ji turi sudėti?



Paaiškinimas

Teisingas atsakymas: 8 + 1 = 9.

Tai informatika!

Informatikoje įprasta naudoti skirtingus skaičių vaizdavimus. Viename kompiuteryje vienu metu gali būti naudojama daug skirtingų skaičių pateikimo būdų. Pavyzdžiui, amžių (sveikieji skaičiai), trupmenas (realiosios reikšmės) ir datas galima vaizduoti naudojant skirtingas taisykles. Šioje užduotyje pateikto skaičių vaizdavimas yra žingsnis į pozicinio skaičių užrašymo (pavyzdžiui, dešimtainės arba dvejetainės sistemos) supratimą, kuris leidžia labai efektyviai atlikti skaičiavimus (pvz., sudėti, dauginti) naudojant pieštuką ir popierių, taip pat ir kompiuterį. Didelės tarptautinės organizacijos (ISO, IEEE ir kitos) deda daug pastangų, siekdamos susitarti dėl tikslaus įvairių tipų skaičių (vadinamųjų skaičių formatų) pateikimo kompiuteriuose. Remiantis dvejetaine skaičiavimo sistema, šioje užduotyje skaičiai ant kortelių yra 2 kartotiniai, kad skaičiams nuo 0 iki 15 pavaizduoti užtektų po vieną skirtingos spalvos kortelę. Tačiau, palyginus su poziciniu užrašu, pavyzdžiui, dvejetaine sistema, šioje užduotyje skaičių vaizdavimas pagal kortelių spalvą yra lankstesnis, nes kortelių eiliškumas neturi reikšmės. Tačiau jis turi du pagrindinius trūkumus, dėl kurių yra nepraktiškas. Pirma, žmonėms ir kompiuteriams bus sunku atskirti skirtingas spalvas, jei bandysime pavaizduoti labai didelius skaičius. Antra, norint atlikti efektyvius skaičiavimus, reikėtų spalvas ir skaičius atvaizduoti kitaip.

Informatinis mąstymas

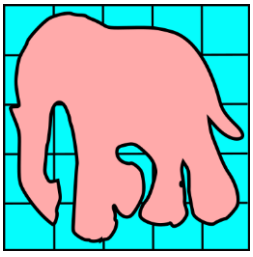
Atliekant šią užduotį naudojama informatinio mąstymo kompetencija „duomenų vaizdavimas“, nes skaičiai koduojami naudojant spalvotas korteles. Šioje užduotyje gali būti naudojama informatinio mąstymo kompetencija „dekomponavimas“, kad būtų galima po vieną (pradedant nuo didžiausios kortelės) nustatyti, kokios yra atvaizdavimo galimybės.

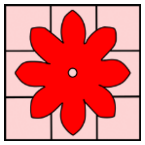
Kaip ir daugelyje klausimų su keliais atsakymų variantais, informatinio mąstymo kompetencija „įvertinimas“ gali būti naudojama norint patikrinti, kuris iš atsakymų variantų yra teisingas. Užduotyje netiesiogiai pasitelkiama informatinio mąstymo kompetencija „algoritminis mąstymas“, nes naudojant Zitos atvaizdavimo sistemą reikalingas algoritmas dešimtainių skaičių kodavimui ir dekodavimui.

14. Aplikacija

Aplikacija – dailės technika, kai prie medžiagos pritvirtinamos figūros – gyvūnai, paukščiai, gėlės.

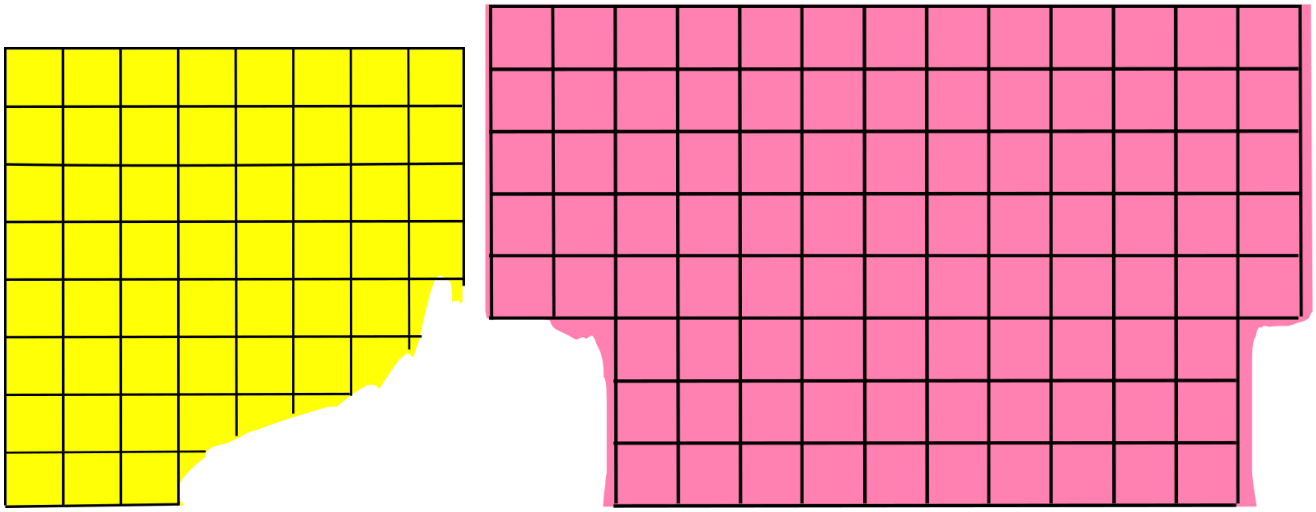
Menininkas aplikacijoms naudoja dvi skirtingas figūras – dramblį ir gėlę. Kad būtų lengviau planuoti, jis padalina medžiagos skiautę į kvadratus. Drambliui pritvirtinti reikia 25 kvadratų ploto (5 x 5 kvadratai), gėlei pritvirtinti reikia 9 kvadratų (3 x 3 kvadratai) ploto.





Abi figūras galima tvirtinti bet kuria kryptimi ir ant bet kokios spalvos medžiagos. Ant kiekvienos medžiagos skiautės pirmiausia reikia pritvirtinti dramblius.

Menininkas nori pritvirtinti kuo daugiau dramblių. Kai jau negalima pritvirtinti daugiau dramblių, ant likusio ploto tvirtinamos gėlės. Figūros negali persidengti.



Kiek iš viso figūrų galima pritvirtinti prie šių dviejų medžiagos skiaučių?

A
8 (3 dramblius ir 5 gėles)

B
10 (4 dramblius ir 6 gėles)

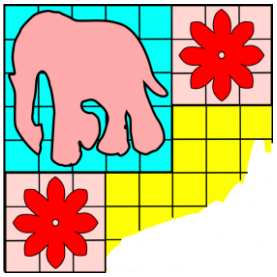
C
9 (3 dramblius ir 6 gėles)

D
10 (3 dramblius ir 7 gėles)

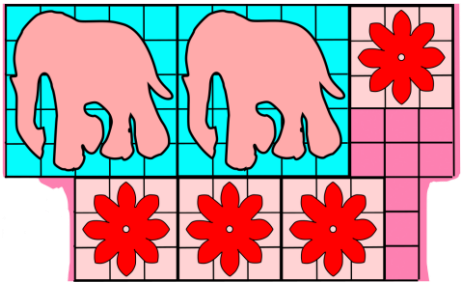
Paaiškinimas

Teisingas atsakymas: C.

Toliau parodytas vienas iš galimų dramblių ir gėlių išdėstymo būdų. Kiti galimi išdėstymo būdai leidžia pritvirtinti tokį patį figūrų skaičių.



Geltona medžiaga



Rausva medžiaga

Jei nereikalaujama pritvirtinti kuo daugiau dramblių, tai rausvai medžiagai yra ir kitas sprendimas, kur toks pat bendras 6 figūrų skaičius, bet tik 1 dramblys ir 5 gėlės.

Tai informatika!

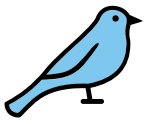
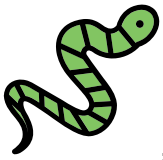
Atminties valdymas yra svarbi operacinės sistemos funkcija. Sukurta daug algoritmų, kurie paskirsto atmintį siekiant užtikrinti maksimalų efektyvumą. Geriausiai tinkantis algoritmas paskiria mažiausią turimą laisvą atminties dalį, kuri atitinka reikalavimus. Pirmas tinkantis algoritmas paskiria pirmą pasitaikiusią gretimos atminties dalį, kurios pakanka reikalavimui patenkinti. Tuo tarpu blogiausias algoritmas paskiria iš didžiausios turimos atminties dalies. Kiekvienas metodas turi savų privalumų ir trūkumų greičio ir atminties panaudojimo atžvilgiu.

Informatinis mąstymas

Šią užduotį galima išspręsti taikant keletą informatinio mąstymo įgūdžių. Norint naudoti tik nurodytų dydžių plotus, reikia abstrakcijos. Algoritminio mąstymo reikia norint taikyti taisykles: pirmiausia pritvirtinti drambliukus, o tada likusią medžiagos dalį užpildyti gėlėmis. Dekomponuojant užduotis suskaidoma į du atskirus uždavinius: pirmiausia reikia pritvirtinti kuo daugiau dramblių, o tada nustatyti vietas, kuriose galima įdėti gėles. Užduotį išspręsti gali padėti abstrakcijos naudojimas, dramblius ir gėles traktuojant kaip paprastus tinklelio plotus.

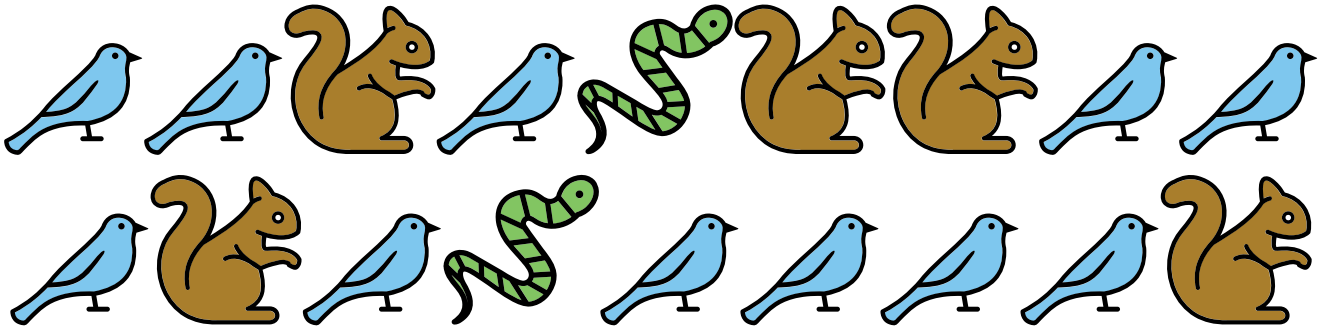
15. Stebuklingas medis

Prie bebro Barto namo auga stebuklingas medis.

Kai tik ant jo nutupia paukštis  , medis išaugina 2 obuolius. Kai tik į jį įšoka voverė  , medis numeta 1 obuolį (jei tik turi). Bartas taip pat pastebėjo, kad kai medį aplanko gyvatė  , visi obuoliai bemat išnyksta.

Vieną rytą Bartas suskaičiavo stebuklingo medžio obuolius – jų buvo 25. Bartas visą dieną stebėjo medį piešdamas gyvūnus, kurie apsilankė.

Gyvūnų piešiniai pateikti jų apsilankymo eile:



Kiek obuolių bus ant stebuklingo medžio dienos pabaigoje?

Paaiškinimas

Dienos pabaigoje ant medžio bus 7 obuoliai.

Kadangi visi obuoliai akimirksniu dingsta, kai tik medį aplanko gyvatė, galime nekreipti dėmesio į viską, kas vyksta iki gyvatės pasirodymo. Po paskutinio gyvatės apsilankymo ant medžio nutupia keturi paukščiai, o tai reiškia, kad medis išaugins 4 × 2 = 8 obuolius. Bet dar į medį įšoka voverė ir dėl to nukrenta vienas obuolys, taigi lieka 8 - 1 = 7 obuoliai.

Tai informatika!

Ši užduotis supažindina su dviem pagrindinėmis programavimo sąvokomis. Pirmoji yra kintamojo sąvoka. Kintamasis naudojamas kompiuterių programose informacijai laikyti. Kintamojo reikšmė gali keistis priklausomai nuo programos komandų. Šioje užduotyje obuolių skaičius ant medžio yra kintamasis, jo reikšmė gali didėti (), mažėti () arba būti nustatyta iš naujo ().

Kad kompiuterio programa keistų kintamojo reikšmę, kompiuteris turi priimti sprendimus. Tai yra antroji esminė programavimo sąvoka, ji vadinama pasirinkimu. Sprendimai priimami naudojant specialias komandas, vadinamas sąlyginiais sakiniais, kurie leidžia pasirinkti vieną iš galimų sprendimų. Paprastai pasirinkimo komandos būna tokios formos: „jei tas, tai anas“. Šiame uždavinyje vienas iš sąlyginių sakinių būtų „jei ant medžio nutūps paukštis, tai obuolių skaičių reikia padidinti 2“.

Ar šiame uždavinyje galite rasti kitų sąlyginių sakinių?

Informatinis mąstymas

Ši užduotis priklauso abstrakcijos sričiai, nes sprendėjai turi išskirti skirtingus uždavinio atvejus ir sutelkti dėmesį tik į būtiną informaciją. Norint efektyviai išspręsti uždavinį, pakanka nagrinėti duomenis po paskutinio gyvatės apsilankymo. Ankstesni įvykiai nėra svarbūs, nes, pasirodžius gyvatei, obuolių skaičius visada tampa lygus 0.

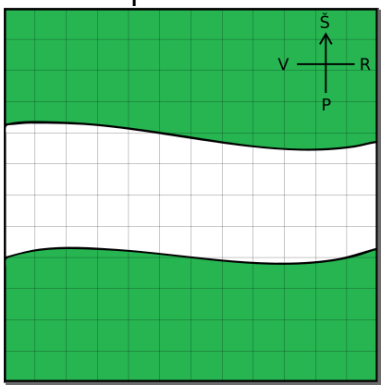
Užduočiai spręsti reikalingas algoritminis mąstymas arba algoritminis projektavimas, nes mokiniai turi ne tik išspręsti uždavinį žingsnis po žingsnio, bet ir pasirinkti sprendimą priklausomai nuo situacijos, tai yra taikant minėtą sąlyginį sakinį (jei… tai...).

16. Karolio svajonių namas

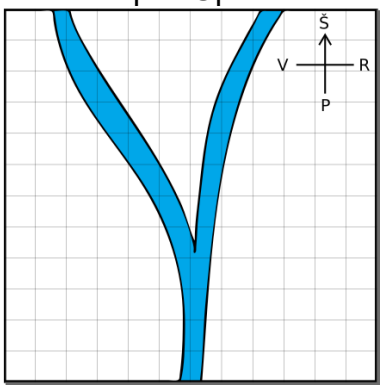
Karolis turi tris žemėlapius, kuriuose vaizduojama ta pati vietovė. Viename žemėlapyje pavaizduoti miškai, kitame – upės, o trečiame – namai.

Karolio svajonių namas turi būti miške ir netoli upės.

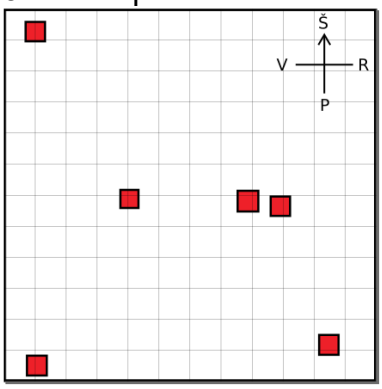
1 žemėlapis. Miškai



2 žemėlapis. Upės



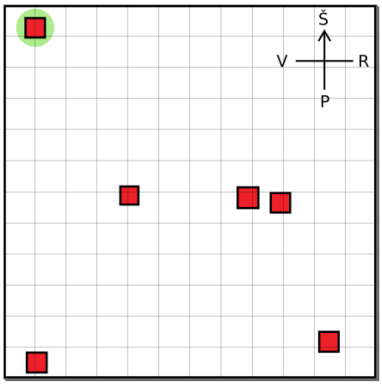
3 žemėlapis. Namai

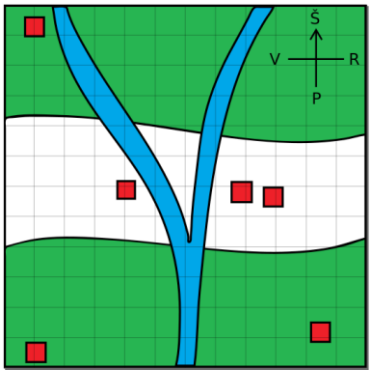


Kuris yra Karolio svajonių namas?

Paaiškinimas

Teisingas atsakymas:





Norėdami nustatyti Karolio svajonių namo vietą, turime įvertinti visų trijų žemėlapių informaciją. Svajonių namas turi būti miško teritorijoje ir netoli upės – tai tinka tik kairėje viršuje esančiam namui. Tai lengvai pastebėtume, jei žemėlapius uždėtume vieną ant kito.

Tai informatika!

Jei visa informacija apie miškus, upes ir namus būtų pateikta viename žemėlapyje, būtų lengva nustatyti ieškomą namą. Geografinė informacinė sistema (GIS) sujungia įvairią erdvinę informaciją ir ją pavaizduoja žemėlapyje. Taigi GIS padeda vizualizuoti ir analizuoti geografinius duomenis. Naudodami GIS nelaimių kontrolės pareigūnai gali kaupti informaciją, pavyzdžiui, evakuacijos planams.

Kelių sluoksnių naudojimas pateikiant skirtingą vaizdinę informaciją žinomas ir iš grafinių programų. Visada kyla svarbus klausimas: kuris sluoksnis ar kurie objektai rodomi pirmame plane? Pateiktame pavyzdyje namų žemėlapis turėtų tapti viršutiniu sluoksniu, kad namų nepaslėptų miško plotai.

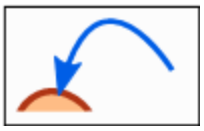
Informatinis mąstymas

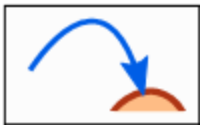
Sprendžiant šią užduotį reikia sistemingai ir logiškai sujungti kelis žemėlapių sluoksnius ir nustatyti tam tikrą vietą. Sprendžiant atliekama geometrinių duomenų struktūros analizė. Pagrindinis dėmesys skiriamas trijų žemėlapių vaizdavimo abstrakcijai, kad būtų sukurtas nuoseklus ir informatyvus sudėtinis žemėlapis.

17. Morkų sėjimas

Robotas-triušis sėja morkas į šias keturias lysves.

Jis moka tokias komandas:

 peršokti į kairę ant kitos lysvės

 peršokti į dešinę ant kitos lysvės

 pasėti morkos sėklą į lysvę, ant kurios stovi

Parengėme robotui skirtą komandų seką:

Nežinome, nuo kurios lysvės robotas pradėjo, bet žinome, kad atlikdamas šią seką jis kiekvieną iš trijų morkų sėklų pasėjo į skirtingas lysves.

Kuris paveikslas vaizduoja morkų sėjimo eigą, atitinkančią pateiktą komandų seką?

A 

B 

C 

D 

Paaiškinimas

Teisingas atsakymas:  .

Robotas triušis turi pradėti nuo trečios iš kairės lysvės, nes kitaip jis negalėtų šokti taip, kaip nurodyta pateiktoje komandų sekoje, t. y., vieną kartą į dešinę ir tris kartus į kairę:

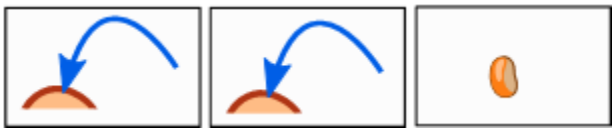
Atlikęs pirmąsias dvi komandas

robotas pasėja morkos sėklą į dešiniausią lysvę:

Tada jis atlieka kitas dvi komandas

ir pasėja sėklą:

Tada jis padaro du šuolius į kairę ir pasėja paskutinę morkos sėklą:

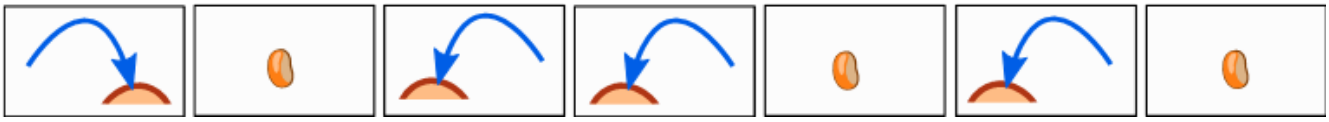


Morkų sėklos pasėtos į lysves taip, kaip ir nurodyta D variante:

Atliekant A variantą

robotas galėtų pradėti nuo trečios lysvės

ir programa atrodytų šitaip:



Atliekant B variantą

robotas galėtų pradėti nuo ketvirtos lysvės

ir programa atrodytų šitaip:

Atliekant C variantą

robotas galėtų pradėti nuo pirmos lysvės

ir programa atrodytų šitaip:

Tai informatika!

Kompiuteriai programuojami panašiai, kaip valdomi aprašytieji robotai-triušiai, tačiau kompiuteriai „supranta“ žymiai daugiau komandų. Šioje užduotyje roboto komandų seka pateikiama paveikslėlių blokais. Šį užrašą (programavimo kalbą) vaikai gali nesunkiai suprasti ir interpretuoti. Vaikai taip pat pradės suprasti, kad komandos pateikiamos tam tikra tvarka, nuo kurios priklauso rezultatas – morkų sėklų sėjimas į lysves. Norint rasti pradinę roboto-triušio vietą, reikia sekti duotą komandų seką ir laikytis sutartos sąlygos (triušis kiekvieną sėklą sėja į skirtingas lysves). Derinimas yra svarbi programavimo dalis. Atlikdami šį procesą ne tik ieškome klaidų programoje, bet ir galime nustatyti tinkamus pradinius (įvedimo) duomenis, kad gautume norimus rezultatus.

Informatinis mąstymas

Šis uždavinys skatina algoritminį mąstymą. Tai įgūdis sekti sekas ir laikytis taisyklių.

18. Išpūstžandžiai

Jonė nori išsiaiškinti, kaip atrodo išpūstžandžiai. Ji atidžiai apžiūri penkias išpūstžandžių nuotraukas ir pasižymi keletą teiginių, kurie nusako tai, ką ji mato.

Pamačiusi šeštąją išpūstžandžio nuotrauką, Jonė supranta, kad vienas iš jos užsirašytų teiginių yra neabejotinai klaidingas.

Kuris iš Jonės užsirašytų teiginių apie išpūstžandžius yra neabejotinai klaidingas?

A. Išpūstžandžiai visada turi dantis.

B. Kai kurie išpūstžandžiai turi sparnus.

C. Išpūstžandžiai turi arba ragus, arba tris akis, bet ne juos abu kartu.

D. Jei išpūstžandžiai turi lygiai dvi rankas, tai jie turi lygiai dvi kojas.

Paaiškinimas

Teisingas atsakymas: D.

Netiesa, kad jei išpūstžandis turi lygiai dvi rankas, tai jis turi ir lygiai dvi kojas. Šeštojoje nuotraukoje išpūstžandis turi dvi rankas ir penkias kojas.

A varianto teiginys visais atvejais teisingas. Visose šešiose nuotraukose išpūstžandžiai turi dantis.

B varianto teiginys teisingas. Antroje nuotraukoje išpūstžandis su sparnais.

C varianto visi teiginiai teisingi. Pirmoje, antroje, penktoje ir šeštoje nuotraukose išpūstžandžiai turi ragus ir tik dvi akis. Likusiose nuotraukose (trečioje ir ketvirtoje) išpūstžandžiai turi tris akis ir neturi ragų.

Tai informatika!

Jonė mokosi atpažinti išpūstžandžius nagrinėdama jų nuotraukas ir ieškodama panašių bruožų. To galima išmokyti ir kompiuterius, naudojant vadinamąjį lyginimo procesą. Šiais laikais galime taikyti mašininį mokymąsi, kad automatiškai ir per trumpą laiką palygintume daugybę daiktų. Taikant mašininį mokymąsi, kompiuteriui pateikiamas duomenų rinkinys, tada jis šiuose duomenyse ieško dėsningumų ir daro išvadas. Kaip ir Jonė, kompiuteris gali padaryti neteisingą išvadą. Pateikus daugiau duomenų, tikslesnių ir išsamesnių, kompiuteris gali ištaisyti savo klaidas ir atnaujinti išvadas.

Pasikartojančių šablonų ir dėsningumų aptikimas duotuose duomenyse yra svarbus informatikos įgūdis, paprastai vadinamas šablonų atpažinimu. Programuojant dėsningumų atpažinimas leidžia taikyti modulinį metodą, kai pasikartojantį dalinį uždavinį reikia spręsti tik vieną kartą. Tai yra sėkmingo algoritmų ir programinės įrangos projektavimo pagrindas, nes taip sumažinamas programos teksto (kodo) pertekliškumas ir padidinamas jo palaikomumas.

Šablonų atpažinimas taip pat labai svarbus mašininiam mokymuisi ir dirbtiniam intelektui, nes kompiuteriams suteikiama galimybė mokytis iš duomenų ir priimti prognozes ar sprendimus. Pavyzdžiui, vaizdų atpažinimo srityje šablonų atpažinimo metodai padeda nustatyti objektus vaizduose, klasifikuoti juos į skirtingas kategorijas, net atpažinti veidus.

Natūralios kalbos apdorojimo srityje šablonų atpažinimas leidžia kompiuteriams suprasti sakinių struktūrą, nustatyti temas ir net kurti panašius į žmogaus tekstus.

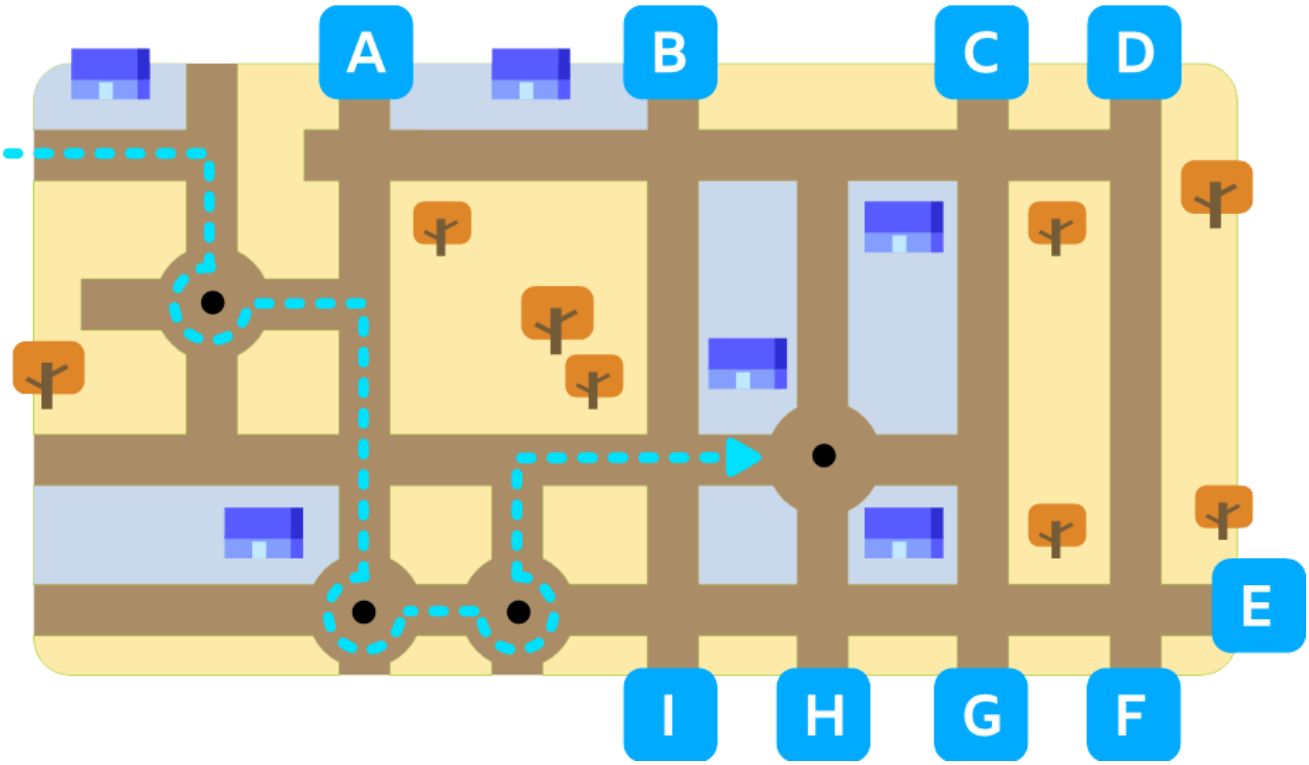
Jei duomenys mašininiam mokymuisi neatidžiai atrenkami, į kompiuterinę sistemą gali patekti klaidų ir šališkumo. Pagalvokite: kokį poveikį visuomenei turėtų kalbos atpažinimo sistema, jei jai apmokyti naudotume tik vyrų, kuriems anglų kalba yra gimtoji, duomenis?

Informatinis mąstymas

Šablonų atpažinimas yra susijęs su duomenų tyryba ir duomenų analize. Sprendžiant uždavinius, susijusius su šiomis sąvokomis, be kitų įgūdžių, labai svarbus algoritminis mąstymas. Mokiniai jį taiko, kai, naudodamiesi savo supratimu apie šablonus, posūkius ir erdvinius ryšius, kuria nuoseklią, žingsnis po žingsnio išdėstytą užduoties sprendimo procedūrą. Duomenų analizė, apimanti simbolių nustatymą ir jų padėties supratimą, leidžia mokiniams daryti išvadas, remiantis pateikta informacija ir atpažįstamais dėsningumais. Galiausiai, vertinimas yra labai svarbus siekiant įvertinti savo sprendimų veiksmingumą ir nustatyti tobulintinas sritis.

19. Savaeigis automobilis

Tomas keliauja savaeigiu automobiliu. Ką tik atnaujintoje automobilio programinėje įrangoje įsivėlė rimta klaida. Dėl šios klaidos automobilis kiekvieną kartą, pasiekęs to paties tipo sankryžą, daro tą patį posūkį. Pavyzdžiui, kiekvienoje žiedinėje sankryžoje automobilis suka į trečiąjį išvažiavimą (žr. paveikslą).



Kuria raide pažymėtą vietą pasieks Tomas, baigęs savo kelionę?

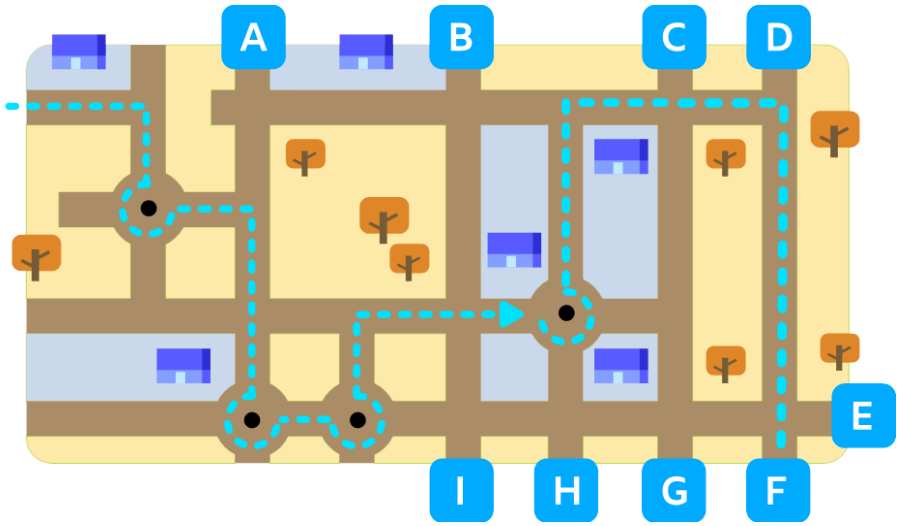
Paaiškinimas

Teisingas atsakymas: F.

Iš paveikslo matome, kad navigacijos sistema kiekvieno tipo sankryžoms taiko šias taisykles:

- T formos sankryžoje automobilis visada suka į dešinę;
- Žiedinėje sankryžoje automobilis visada suka į trečiąjį išvažiavimą dešinėn;
- + formos sankryžoje automobilis visada važiuoja tiesiai.

Pratęsdami Tomo kelionę matome, jog kita sankryža yra žiedinė, todėl automobilis suka į trečiąjį išvažiavimą viršun. Tolesnė sankryža yra T formos, todėl automobilis suka į dešinę. Dar kitoje + formos sankryžoje automobilis važiuoja tiesiai, o viršuje esančioje T formos sankryžoje pasuka į dešinę. Galiausiai automobilis važiuoja tiesiai per + formos sankryžą ir sustoja ties F. Paveiksle parodytas Tomo kelias:



Tai informatika!

Informatikoje įprasta modeliuoti dinamišką sistemą, kurią valdo tam tikromis taisyklėmis grindžiamas algoritmas. Būsimą sistemos elgseną galima numatyti tik atidžiai išanalizavus algoritmą ir nustačius tikslias taisykles, kuriomis jis turi vadovautis. Tiriama sistema gali būti ir reali sistema, ir virtuali programinės įrangos sistema, pavyzdžiui, skirta programos testavimui.

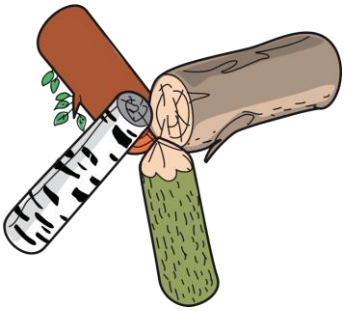
Informatinis mąstymas

Šiam uždaviniui išspręsti reikia šablonų (modelių) atpažinimo įgūdžių ir algoritminio mąstymo.

20. Nuotrauka

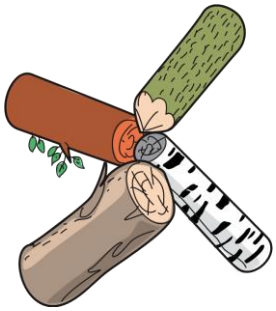
Bebras ką tik padarė nuotrauką.

Kuri iš pateiktų nuotraukų yra ką tik Bebro padarytoji?

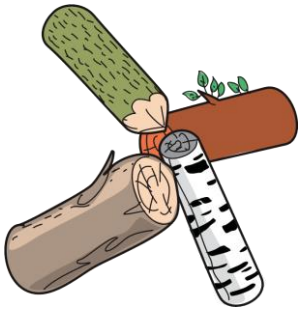


A

B



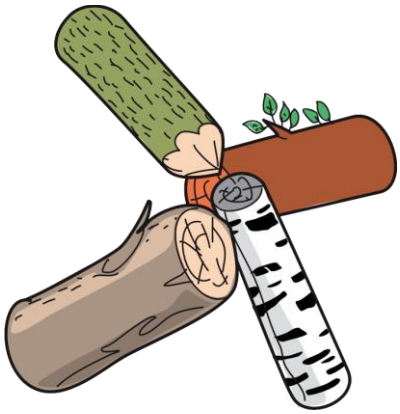
C



D

Paaiškinimas

Norint nustatyti, kaip rąstai atrodo nuotraukoje, svarbu išanalizuoti, kurie rąstai yra kairėje ir kurie dešinėje vienas kito atžvilgiu. Šiuo atveju žalias rąstas smailu galu (panašus į pieštuką) yra tarp dviejų (pilkšvo ir rudo) rąstų, todėl A ir C variantai netinka. Dar pastebėkime, kad bebras, darydamas nuotrauką, „pieštukinį“ rąstą mato kairiau pilkšvo rąsto (be lapelių). Taip yra tik D varianto atveju.



Tai informatika!

Šioje užduotyje reikėjo išnagrinėti rąstų eilės tvarką, kokia jų padėtis vienas kito atžvilgiu. Nagrinėdami atsižvelgėme į ryšius tarp rąstų: kiekvieną rąstą susiejome su dešinėje esančiu rąstu ir kairėje esančiu rąstu.

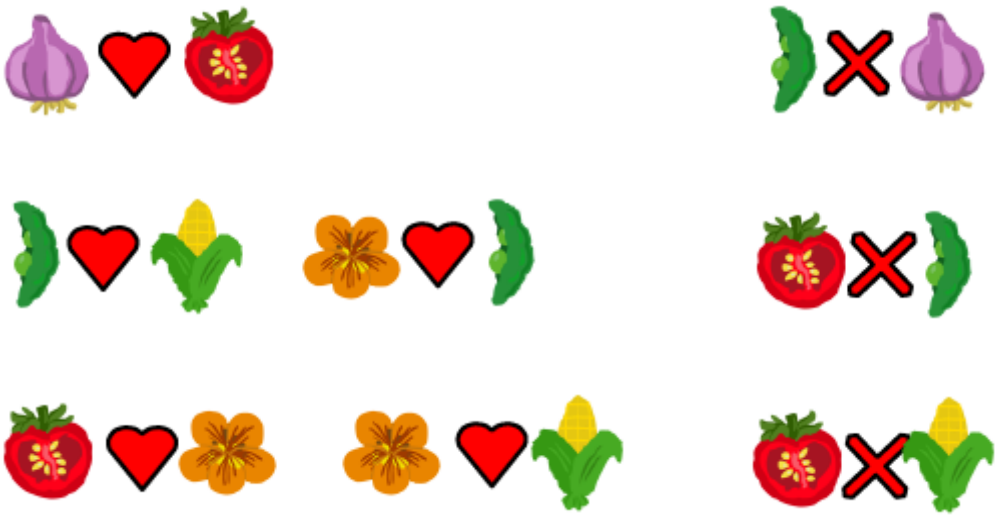
Kompiuteriuose gali būti naudinga duomenis saugoti panašiu būdu. Kiekvienas duomenų elementas (pavyzdžiui, skaičius ar raidė) gali turėti sąsajas su dviem kaimynais. Informatikoje tokia duomenų struktūra vadinama dvigubai susietu sąrašu. Ji yra labai lanksti: galima saugoti tiek duomenų, kiek reikia, duomenis į tokį sąrašą galima lengvai įtraukti ir pašalinti.

Informatinis mąstymas

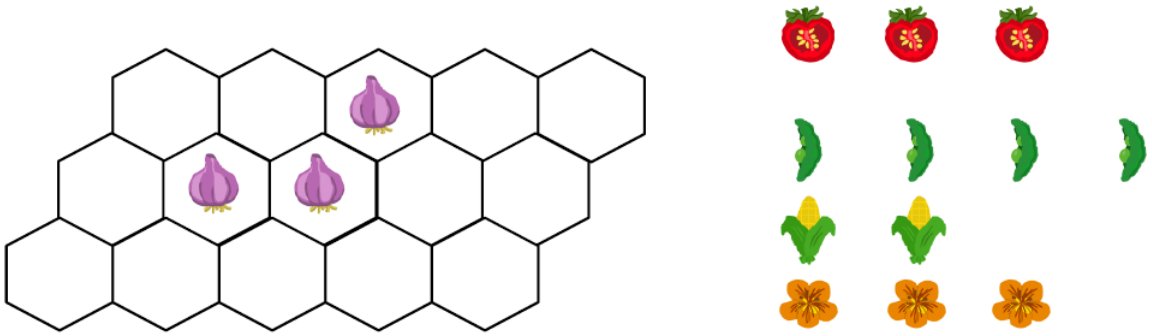
Šiai užduočiai išspręsti reikia gebėjimo pasinaudoti abstrakcija, t. y., ištraukti svarbią informaciją (kas būdinga rąstų išdėstymui) ir erdvinio mąstymo įgūdžių (pertvarkant objektų atvaizdavimą jų ryšiai turi išlikti).

21. Draugiška kaimynystė

Dalia darže sodina įvairių rūšių augalus. Kai kurie augalai yra draugiški ir padeda vieni kitiems. Paveiksle parodyta, kurie augalai yra geri draugai ( ), o kurie ne ( ).



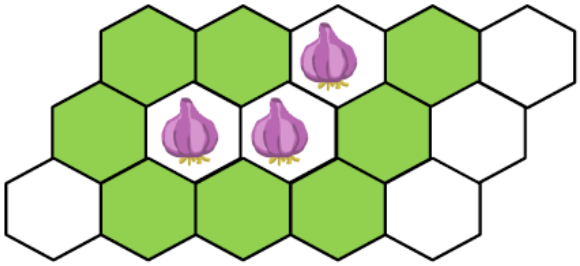
Dalia daržą padalino į šešiakampes lysves. Kiekvienoje lysvėje ji nori pasodinti lygiai po vieną augalą. Dalia vadovaujasi taisykle: lysvėse, kurios tiesiogiai ribojasi viena su kita, negali būti nedraugiškų augalų. Dalia į tris lysves jau pasodino česnakus.

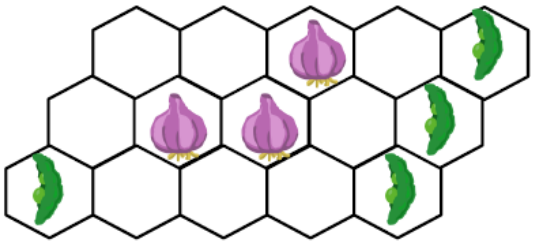


Remiantis nurodyta taisykle padėkite Daliai pasodinti likusius augalus.

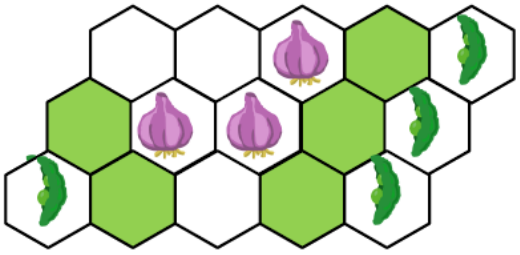
Paaiškinimas

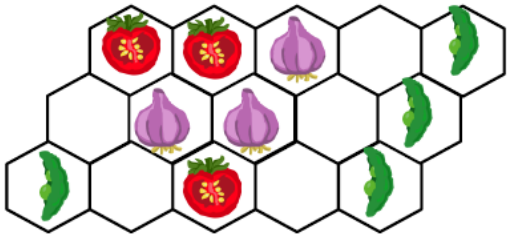
Pirmiausia imamės lysvių, kurios supa jau pasodintus tris česnakus (jos pažymėtos žaliai) ir bandome jose pasodinti draugiškus augalus. Į šias lysves negalime sodinti pupelių. Taigi darosi aišku, kur reikia sodinti pupeles:



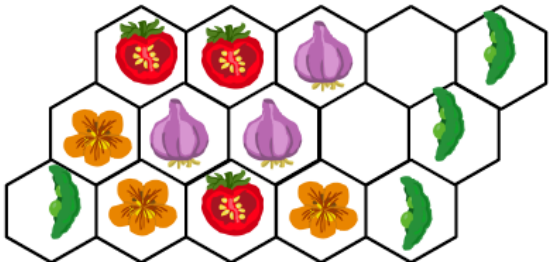


Dabar atsiranda kitas ribojimas ir darome tą patį: lysves, kuriose turi būti sodinami draugiški pupelėms augalai, pažymime žaliai. Šiose lysvėse negalime sodinti pomidorų. Taigi labai aišku, kur reikia sodinti pomidorus:





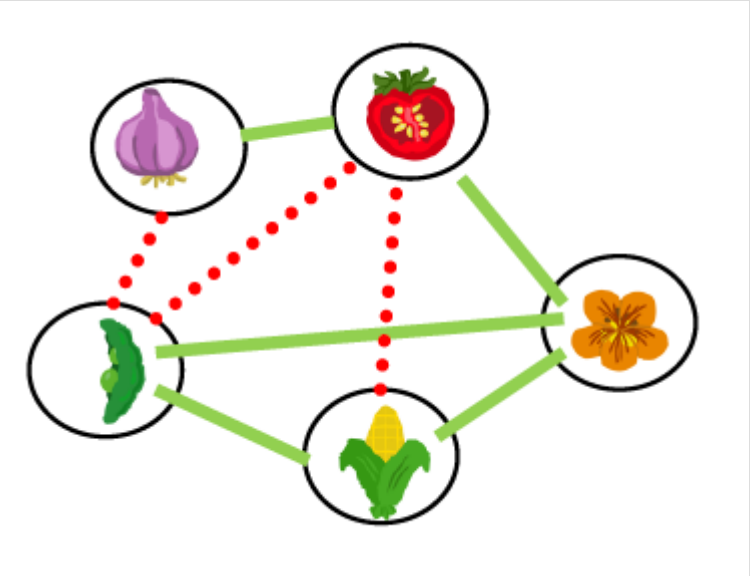
Liko trys gėlės ir du kukurūzai. Šie abu augalai yra draugiški pupelėms, tačiau pomidoras naudingesnis gėlėms (pritraukia bites), o kukurūzams jis blogas kaimynas. Taigi gėles sodiname šalia pomidorų. Likusiose dviejose lysvėse pasodinsime kukurūzus, jie draugiški ir pupelėms, ir gėlėms.



Tai informatika!

Šią užduotį galima spręsti įvairiais būdais. Galima tiesiog naudoti brutalios jėgos metodą (angl. *brute-force approach*), kai kiekvieną augalą bandome sodinti į kiekvieną lysvę, kol bus tenkinamos visos sąlygos, tačiau tai užims daug laiko. Vietoj to geriau nagrinėti uždavinį ir ieškoti ribojimų, kurie leistų sumažinti pasirinkimų skaičių.

Santykius tarp augalų galima vaizduoti neorientuotų ribojimų grafu: dvi viršūnės yra sujungtos, jei jas atitinkantys augalai yra draugiški vienas kitam (žalios ištisinės briaunos); viršūnėms taikomi ribojimai, kai jas atitinkantys augalai negali augti vienas šalia kito (raudonos punktyrinės briaunos).



Ši užduotis gali būti laikoma ir kombinatorikos uždaviniu, tiksliau, sprendimo priėmimo uždaviniu. Tokiems uždaviniams konkretaus atvejo sprendiniai nurodomi loginių sąlygų rinkiniu. Daugelį verslo uždavinių galima modeliuoti grafais ar panašiais daliniais uždaviniais. Šitaip sprendžiami optimizavimo, simuliavimo „kas būtų, jei“, ribojimų tenkinimo, grafų klasifikavimo, grafų braižymo ir vizualizavimo uždaviniai.

Ūkininkai ir daržininkai, formuodami daržus, naudoja tokio tipo sąlygų logiką. Ši informacija padeda jiems priimti pagrįstus sprendimus ir padidinti savo daržų produkciją. Pavyzdžiui, pupelės naudingos kukurūzams, aprūpina juos azotu, o kukurūzai gali paremti pupeles. Kompiuterių moksle sąlygos naudojamos siekiant padėti programoms priimti sprendimus priklausomai nuo teikiamos informacijos teisingumo ar klaidingumo.

22. Logikos lobis

Saloje paslėptos trys lobių skrynios: viena – netoli kalnų, kita – po palme, trečia – pajūry. Dienos pradžioje visos trys lobių skrynios tuščios  . Kažkuriuo metu dieną stebukladaris piratas vieną iš skrynių pripildo aukso  . Trys keliautojai tyrinėjo salą ir fotografavo lobių skrynias. Vienas keliautojas padarė nuotraukas prieš piratui pripildant skrynią auksu. Kiti du fotografavo po to, kai piratas jau buvo pripildęs skrynią aukso.

Alisa:
Skrynia pajūry buvo tuščia.

Benas:
Skrynia pajūry buvo tuščia ir skrynia po palme buvo tuščia.

Ema:
Skrynia netoli kalnų buvo tuščia ir skrynia po palme buvo tuščia.

Pirato laimei, aukso nerado nė vienas keleivis. Kurioje skrynioje buvo auksas?

A. Skrynioje po palme
B. Skrynioje pajūry
C. Skrynioje netoli kalnų
D. Skrynios su auksu negalima nustatyti

Paaiškinimas

Auksas buvo skrynioje netoli kalnų.

Kadangi galimos trys aukso buvimo vietos, atidžiai išnagrinėkime kiekvieną jų ir išsiaiškinkime, kuri iš jų atitinka legendą.

- Jei auksas buvo skrynioje po palme, tuomet Beno ir Emos nuotraukos turėjo būti padarytos prieš piratui užpildant skrynias auksu. Legendoje sakoma, kad prieš tai fotografavo tik vienas turistas, taigi atsiranda prieštaravimas. Skrynioje po palme negali būti aukso.
- Jei auksas buvo skrynioje pajūry, tai Alisos ir Beno nuotraukos turėjo būti padarytos prieš piratui užpildant skrynias auksu. Tai irgi prieštarauja legendai. Skrynioje pajūry negali būti aukso.
- Jei auksas buvo skrynioje netoli kalnų, tuomet tik Emos nuotrauka padaryta, kol skrynia tuščia. Tai atitinka legendą.

Darome išvadą, kad piratas pripildė auksu skrynią netoli kalnų.

Tai informatika!

Ši užduotis susijusi su loginiu samprotavimu. Naudojantis pateiktomis nuotraukomis ir atsižvelgus į tris vietas, kuriose galėtų būti auksas, reikia logiškai mąstyti ir pagrįsti, kodėl kažkas turi arba neturi būti tiesa. Argumentuojant ypatingai reikia remtis prieštaravimais. Jei loginiai veiksmai veda prie dviejų teiginių, kurie negali būti teisingi tuo pačiu metu, tada galime drąsiai teigti, kad pradinė prielaida yra klaidinga.

Logika atlieka svarbų vaidmenį įvairiose informatikos srityse: duomenų bazėse, programavimo kalbose, dirbtiniame intelekte, aparatinės ir programinės įrangos projektavime ir testavime.

Informatinis mąstymas

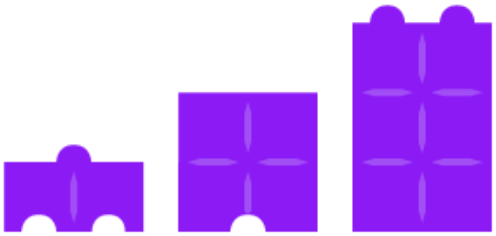
Vienas iš šios užduoties sprendimo būdų – išnagrinėti kiekvieną galimą atsakymą darant prielaidą, kad jis teisingas ir sekant šios prielaidos loginėmis pasekmėmis nustatyti, ar atsakymas tinka, ar jis prieštarauja kitiems užduotyje pateiktiems duomenims. Šis procesas vadinamas modeliavimu ir vertinimu. Taigi uždavinys yra susijęs su situacijų (proceso) modeliavimu ir proceso rezultatų vertinimu.

23. Bebrų nešuliai

Žaislų konstravimo rinkinio detalėms būdingos keturios savybės:

1) plotis (reikšmės: siaura, vidutinė arba plati);
2) aukštis (reikšmės: žema, vidutinė arba aukšta);
3) pusapskritimių skaičius viršuje (reikšmės: nulis, vienas arba du);
4) iš apačios išpjautų pusapskritimių skaičius (reikšmės: nulis, vienas arba du).

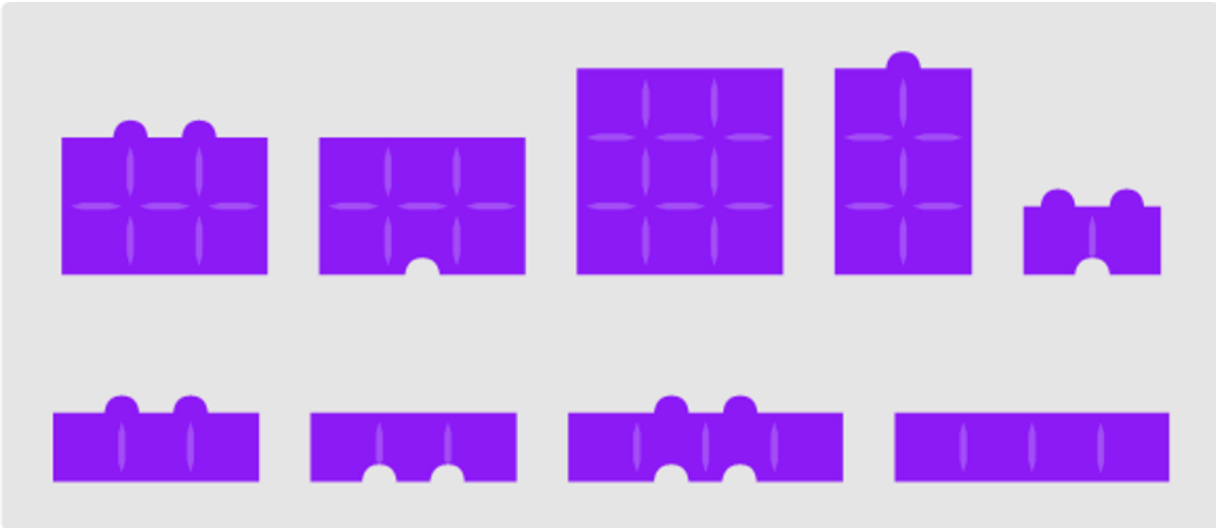
Bebras nori suskirstyti detales į tris grupes ir tikrina kiekvieną savybę, vadovaudamasis tokia taisykle: kiekviena grupės detalė arba turi tą pačią savybės reikšmę, arba visos tos savybės reikšmės yra skirtingos.



Šiame pavyzdyje detalių suskirstymas atitinka taisyklę, nes:

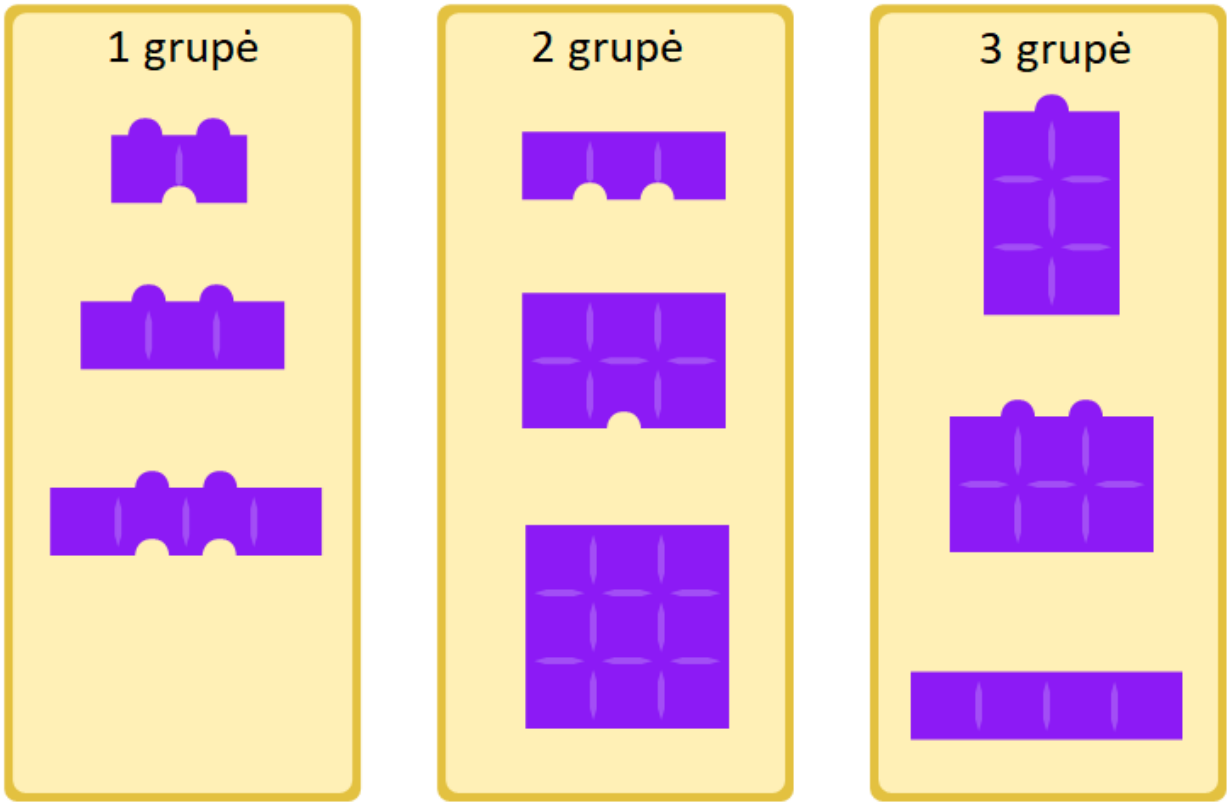
- kiekviena detalė yra to paties pločio;
- kiekviena detalė skirtingo aukščio;
- kiekviena detalė turi skirtingą pusapskritimių skaičių viršuje;
- kiekviena detalė turi skirtingą skaičių pusapskritimių, išpjautų iš apačios.

Devynias detales reikia suskirstyti į tris grupes, kad kiekviena grupė atitiktų taisyklę.



Paaiškinimas

Teisingas atsakymas:



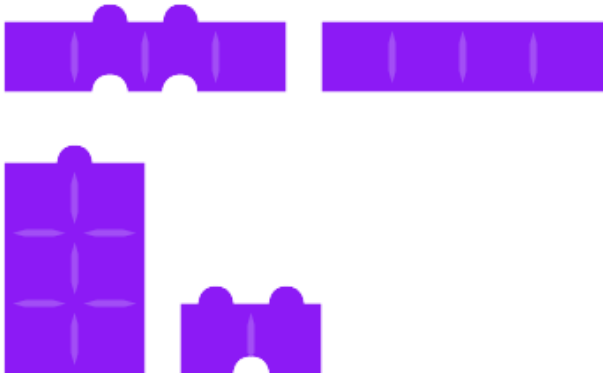
1-oje grupėje detalių pločiai skirtingi, aukštis visų vienodas, kiekvienos detalės viršuje yra po du pusapskritimius, o apačioje – skirtingas išpjautų pusapskritimių skaičius.

2-oje grupėje visos detalės vienodo pločio, skirtingo aukščio, visos trys neturi viršutinių pusapskritimių, o apačioje turi skirtingą išpjautų pusapskritimių skaičių.

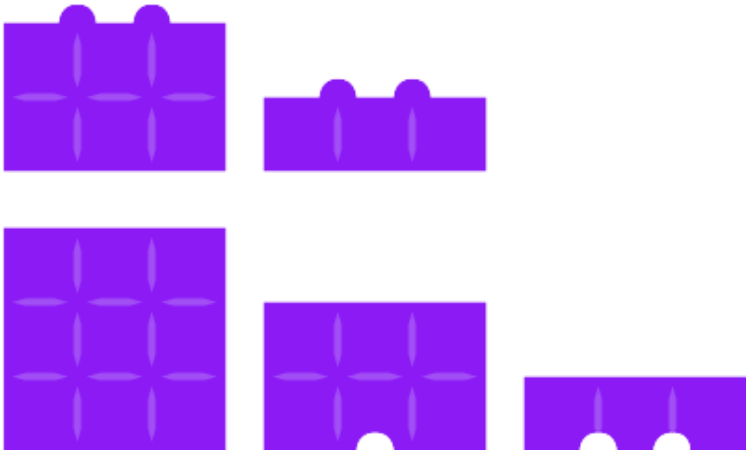
3-oje grupėje visos detalės neturi apačioje išpjautų pusapskritimių, yra skirtingo pločio ir aukščio, viršuje turi skirtingą pusapskritimių skaičių.

Yra keletas būdų, kaip rasti teisingą atsakymą ir patikrinti, ar tai vienintelis teisingas atsakymas. Paprasčiausia – išbandyti visus būdus. Tai gali užtrukti ilgai! Toliau aprašomas kitoks būdas, kai naudojamas kruopštus samprotavimas.

Yra tik dvi detalės, kurių pločio reikšmė yra „plati“, ir dvi detalės, kurių pločio reikšmė yra „siaura“:



Tai reiškia, kad turi būti grupė, kurios visų detalių pločio reikšmė yra „vidutinė“. Iš penkių detalių, kurių pločio reikšmė yra „vidutinė“, dviejų detalių viršuje yra du pusapskritimiai, o trijų – pusapskritimių nėra.



Todėl šios trys detalės be pusapskritimių viršuje turi sudaryti grupę (atsakymo paveikslėlyje tai 2-a grupė).

Turime padalyti likusias šešias detales į dvi grupes, kuriose visos kiekvienos grupės detalės turi skirtingas pločio reikšmes. Atkreipkite dėmesį, kad turi būti grupė detalių, kurių visų aukščio reikšmė „trumpa“, ir grupė – kurių aukščio visos trys reikšmės skirtingos. Pažvelgę į šių detalių apačią, galime nustatyti, kad turi būti sudaryta 3-a grupė.

Likusios trys detalės sudaro 1-ą grupę.

Tai informatika!

Ši užduotis iliustruoja praktinį Ramzio (Ramsey) teoremos atvejį, kai siekiama rasti tvarkingus šablonus didelėje sistemoje ar struktūroje. Teorema teigia, kad bet kokiam duotam šablonui ar konfigūracijai egzistuoja mažiausias sistemos ar struktūros dydis, kuriam šablonas garantuotai atsiras.

Šablonus galima apibrėžti įvairiai, atsižvelgus į kontekstą ir nagrinėjamą uždavinį. Pirminėje teoremos formuluotėje skirtingoms kategorijoms žymėti buvo naudojamos spalvos, tačiau pačiai teoremai spalvos sąvoka nėra esminė. Šioje užduotyje ji buvo pritaikyta detalėms (figūroms).

Ramzio teorema plačiai taikoma matematikoje (kombinatorikoje, grafų teorijoje), informatikoje, fizikoje ir kitose srityse. Ji leidžia suprasti, kad iš pažiūros chaotiškose sistemose egzistuoja tvarka ir struktūra, ir yra svarbi tinklų analizei, optimizavimo uždaviniams ir kriptografijai.

Informatinis mąstymas

Ši užduotis iliustruoja problemų sprendimo situaciją, kai naudojame loginį mąstymą, šablonų atpažinimą ir dekomponavimą.

Loginės išvados naudojamos norint rasti sprendimą, kuris atitiktų taisyklių rinkinį. Pavyzdžiui, taisyklę dėl pločio galima išreikšti taip: „visos vienos grupės detalės yra vienodo pločio ARBA visos yra skirtingo pločio“.

Šioje užduotyje galima atpažinti daug įvairių šablonų (modelių). Pavyzdžiui, turint dvi figūras, galima pridėti lygiai kitą figūrą ir sudaryti taisyklę atitinkančią grupę.

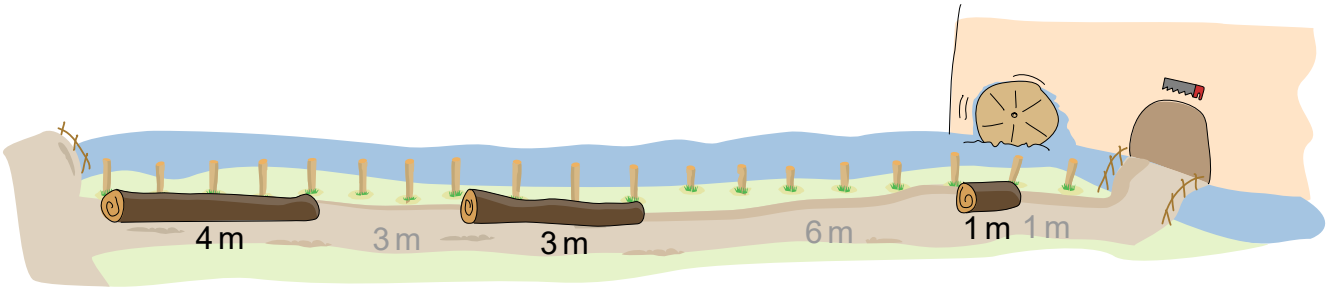
Šią užduotį taip pat galima laikyti dekomponavimo pavyzdžiu. Kiekviena grupė turi keturias taisykles, kurias turi atitikti kiekviena detalė. Šios keturios taisyklės gali būti tikrinamos nepriklausomai, o galutinis kiekvienos grupės rezultatas yra šių keturių patikrinimų rezultatas.

24. Rąstų saugykla

Bebrė Jonė pjauna įvairaus ilgio rąstus ir juos parduoda. Rąstus ji krauna ant 18 metrų ilgio siauro kelio vieną paskui kitą, nes rąstai netelpa šalia vienas kito.

Dėliodama rąstus, Jonė laikosi tokios taisyklės: kiekvieną rąstą ji deda į kairiausią tuščią kelio vietą, kurioje rąstas telpa. Kai ji parduoda rąstą, jis tiesiog paimamas, nejudinant kitų rąstų.

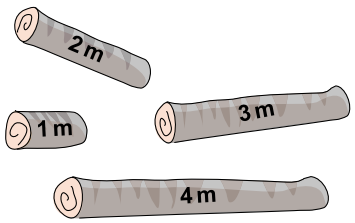
Jonei nupjovus ir pardavus keletą rąstų, kelias atrodo taip:



Jonei reikia supjaustyti rąstus, kad jie būtų 1, 2, 3 ir 4 metrų ilgio.

Kokia tvarka Jonė turėtų rąstus supjaustyti, o po to pagal minėtą taisyklę sudėlioti taip, kad jie visi tilptų?

1. 2. 3. 4.



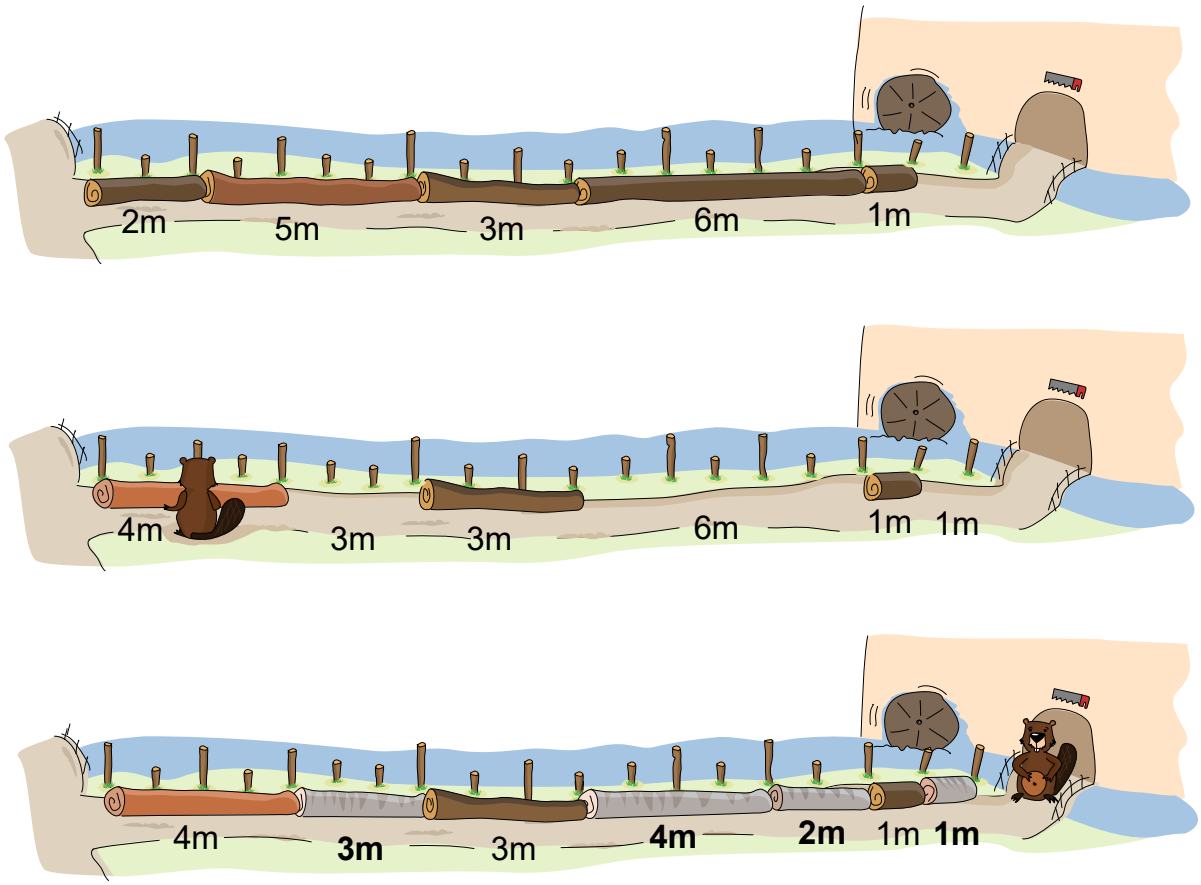
Paaiškinimas

Galimi trys teisingi atsakymai

- pirmas rąstas 4 m, toliau 3 m, toliau 2 m, toliau 1 m;
- 3 m, 4 m, 2 m, 1 m (galutinis sudėliojimas bus toks pat);
- 3 m, 2 m, 4 m, 1 m (šiuo atveju sukeistos vietomis 2 m ir 4 m rąstų padėtys).

Paveiksle matome, kad prieš Jonei išpjaunant naujus rąstus yra trys laisvos vietos: kairėje esančios vietos ilgis yra 3 metrai, vidurinės vietos ilgis yra 6 metrai, o dešinėje esančios vietos ilgis yra 1 metras.

Pirmuoju iš trijų galimų sprendimų pirmasis rąstas dedamas į vidurinę vietą, nes jis netelpa į kairiausiąją vietą; tada antrasis rąstas užpildo kairiausiąją vietą; trečiasis rąstas baigia užpildyti vidurinę vietą, o paskutinis rąstas užpildo dešiniausiąją vietą. Taip gaunamas čia parodytas išdėstymas:



Nesudėtinga patikrinti, kad kiti du galimi sprendimai taip pat teisingi. Tačiau iš kur žinome, kad daugiau sprendinių nėra?

Atkreipkite dėmesį, kad bendras keturių rąstų ilgis yra toks pat, kaip ir bendras turimos erdvės ilgis. Tai reiškia, kad 1 m ilgio rąstas turės atsidurti pačioje dešiniausioje vietoje, nes ši erdvė netinkama jokiam kitam rąstui. Tai savo ruožtu reiškia, kad 1 m ilgio rąstas turi būti nupjautas paskutinis, nes priešingu atveju jis atsidurtų vienoje iš kitų vietų, o tai užkirstų kelią visam sprendimui. Lieka trys rąstai. 2 m ilgio rąstas turi būti nupjautas po 3 m ilgio rąsto, nes priešingu atveju 2 m ilgio rąstas atsidurtų kairiausioje vietoje, o toje vietoje likusi 1 m atkarpa netiktų nei 3 m, nei 4 m ilgio rąstui, o vidurinė yra 6 m ilgio. Šie apribojimai reiškia, kad 3 m, 2 m ir 1 m rąstai gali būti pjaunami ir dedami tik tokia tvarka. Tuomet pjaunant 4 m rąstą prieš 3 m rąstą arba tarp 3 m ir 2 m rąstų, arba tarp 2 m ir 1 m rąstų, gauname tris galimus sprendimus. Todėl jokių kitų sprendimų negali būti.

Tai informatika!

Jonės kelią galima įsivaizduoti kaip kompiuterio atmintį, o rąstus – kaip kompiuterio procesus, kuriems reikia tam tikro atminties kiekio, kad galėtų veikti. Šiame pavyzdyje, nors teoriškai yra pakankamai vietos, kad būtų galima sudėlioti visus rąstus, dėl jų padėjimo ir išėmimo tvarkos (ir strategijos) kartais neįmanoma visų sudėti. Panašiai gali nutikti ir kompiuteriuose: priklausomai nuo to, kokia tvarka skiriamos ir atlaisvinamos atminties sritys, gali būti neįmanoma skirti naujos tam tikro dydžio atminties, net jei bendras laisvos atminties kiekis yra didesnis už tą dydį. Ši situacija vadinama atminties fragmentacija. Fragmentacijos problemą padeda sumažinti protingesnės nei Jonės paskirstymo strategijos arba atminties modeliai, leidžiantys, pavyzdžiui, gabalėlius perkelti. Atminties fragmentacija yra ne tik operatyviosios atminties problema: skiriant vietą diske įvairaus dydžio failams įrašyti, kyla labai panašių problemų.

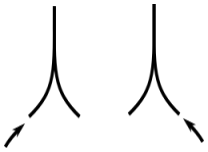
Informatinis mąstymas

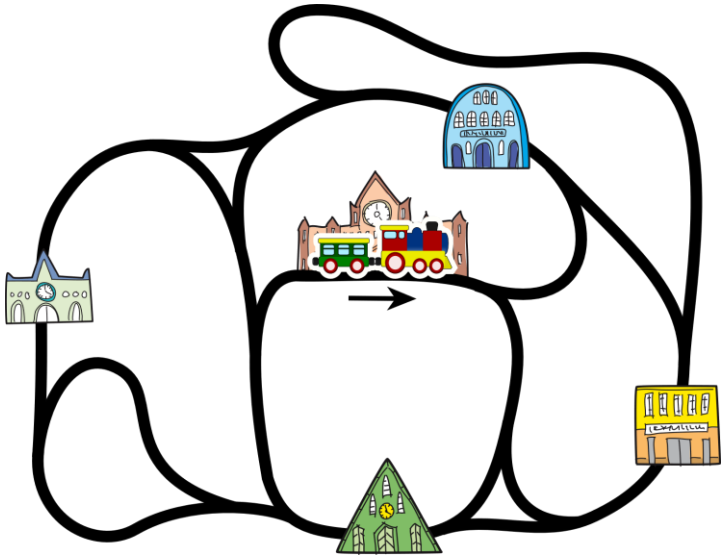
Norint išspręsti šią užduotį reikia įvertinti skirtingus siūlomus variantus pagal aprašytą strategiją ir išsiaiškinti, kurie iš jų veda prie galimo sprendimo, o kurie – ne. Galima išvesti taisykles, padedančias suformuluoti kriterijus, pagal kuriuos tam tikra tvarka (arba tvarkos pradžia) būtų perspektyvi arba vestų į keblią situaciją (pavyzdžiui, sukurtų kelis nedidelius atminties fragmentus, kurie vėliau greičiausiai nebus panaudoti).

25. Geležinkelio modelis

Indrė sukūrė geležinkelio modelį, kuriame yra pradžios stotis  ir 4 kitos stotys. Traukinys programuojamas: vykdo komandų seką ir pagal komandą daro posūkį geležinkelio iešmuose. Įvykdęs visas komandas, traukinys sustoja artimiausioje stotyje.

Komandos pateikiamos tik raidėmis K ir D. Traukiniui pasiekus geležinkelio iešmą, tolesnis kelias pasirenkamas remiantis lentele:

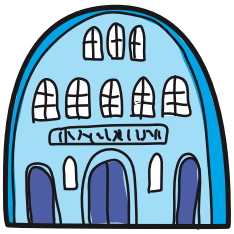
Geležinkelio iešmas	Kita komanda	Kur važiuoti
	**K**	
	**D**	
	Nenaudoti kitos komandos	

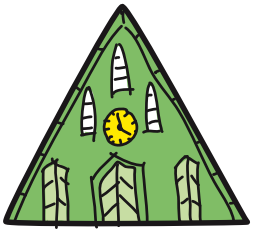
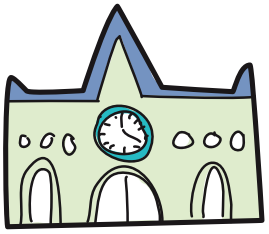


Indrė padeda traukinį pradžios stotyje  . Traukinys juda rodyklės kryptimi. Jei Indrė pateikia komandas DK, traukinys sustos stotyje  .

Jei Indrė grąžina traukinį į pradžios stotį  judėjimui ta pačia kryptimi ir pateikia komandas KD, traukinys vėl sustos stotyje  .

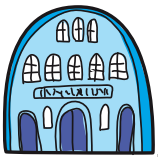
Kurioje stotyje sustos traukinys, pradėjęs kelią nuo pradžios stoties  rodyklės kryptimi ir įvykdęs komandas KKDD?

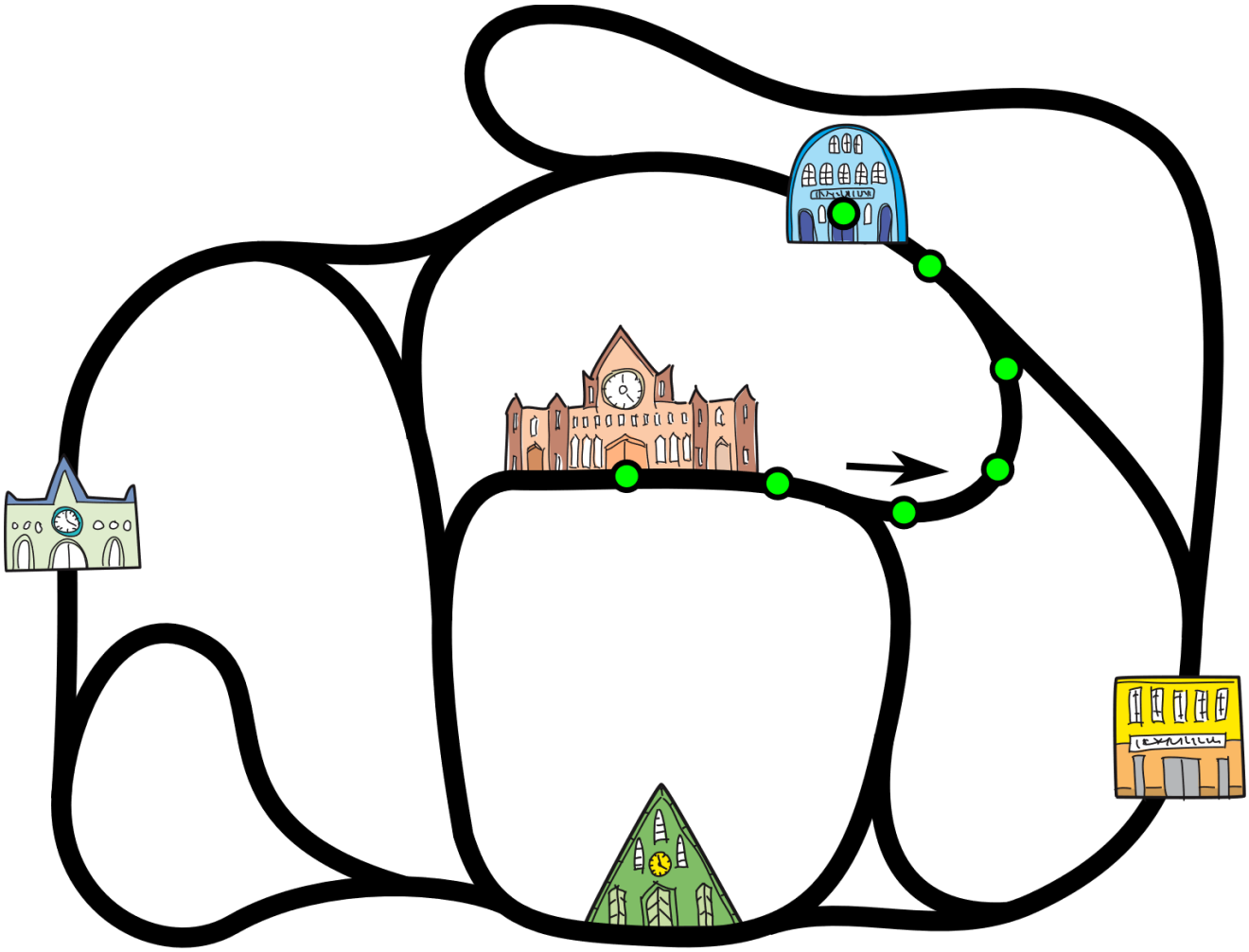


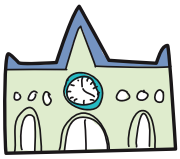
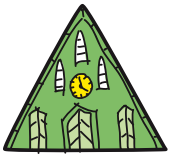



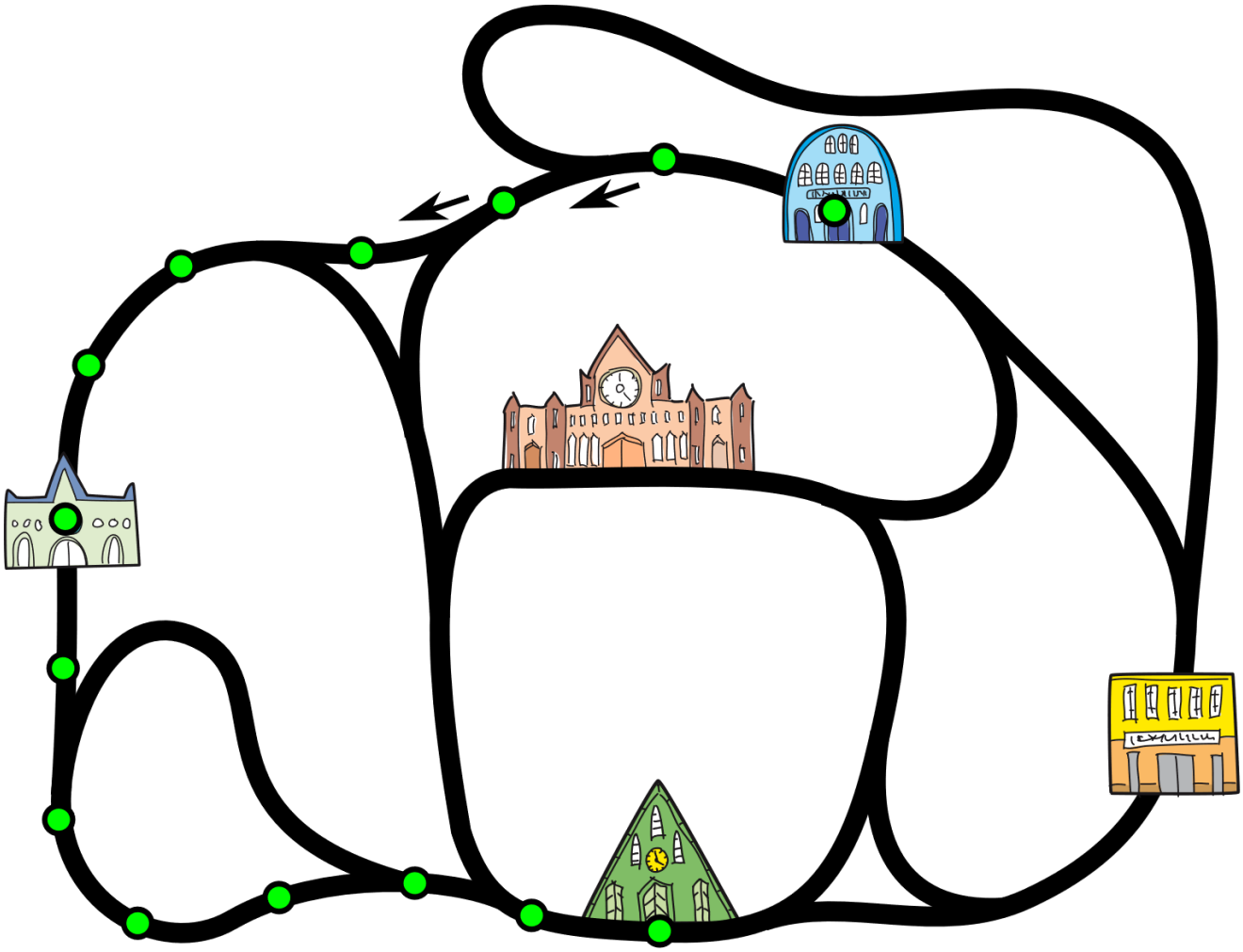
Paaiškinimas

Teisingas atsakymas:  .

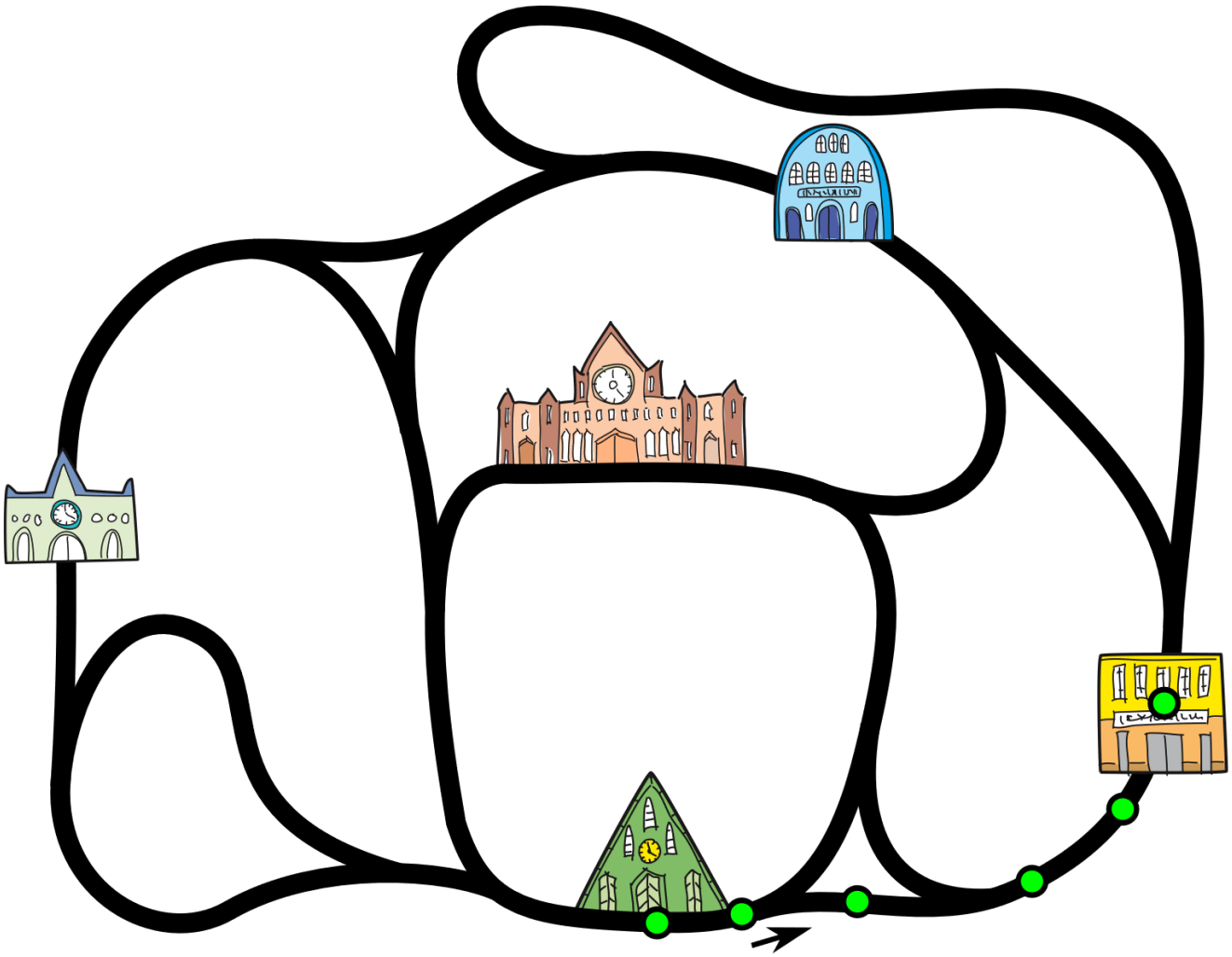
Įvykdęs K komandą, traukinys važiuoja per stotį  .



Įvykdęs komandas KD, traukinys pravažiuoja stotį  . Kadangi komanda nurodo judėjimo kryptį į kairę, traukinys toliau juda ir pravažiuoja stotį  be jokių komandų.



Įvykdęs D komandą, traukinys atvyksta į stotį  . Kadangi tolesnių komandų nėra, traukinys sustoja šioje stotyje.



Tai informatika!

Programa – tai komandų seka. Komandos kartu su tam tikromis sąlygomis (traukinio judėjimo kryptimis geležinkelio iešmuose) apibrėžia traukinio kelią. Siekiant išvengti klaidų, tokios komandos turi būti aiškiai nusakomos ir suprantamos komandas vykdančiam agentui. Automatinis geležinkelio valdymas reiškia, kad komandos yra saugomos, o kelias gali būti išanalizuotas ar pakartotas vėliau.

Informatinis mąstymas

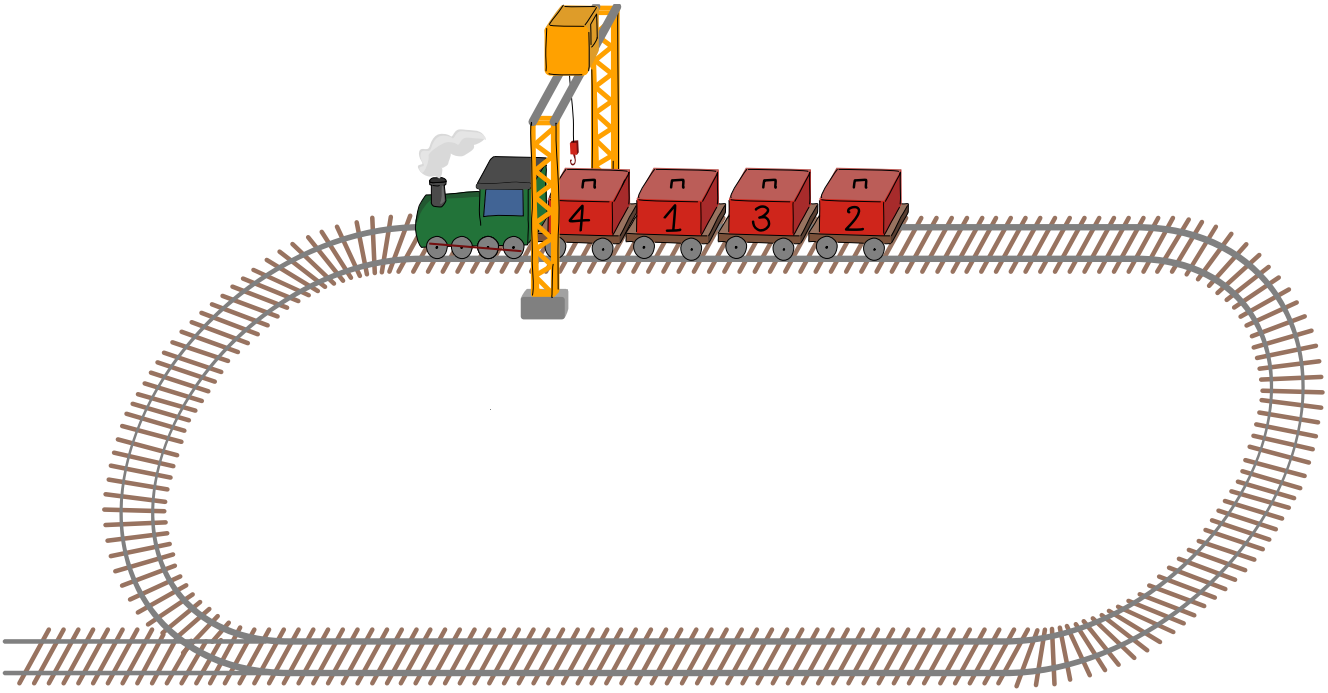
Šiai užduočiai išspręsti reikia suprasti grafo modelį ir traukinio judėjimo programavimo būdus. Abstrahavimo įgūdžiai reikalingi siekiant išskirti uždaviniui spręsti reikalingus esminius dalykus.

Uždaviniui išspręsti taip pat reikia algoritminio mąstymo. Turime procesą aprašančią programą ir nagrinėjame šios programos vykdymo rezultatą.

26. Traukinio iškrovimas

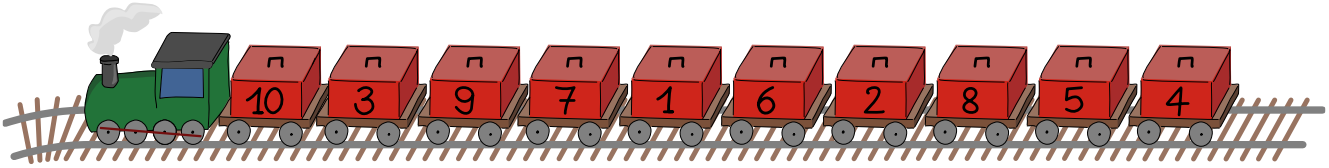
Kiekviename krovininio traukino vagone yra po konteinerį su numeriu. Konteineriams iškrauti naudojamas vienas stacionarus kranas, stovintis konkrečioje vietoje. Norint iškrauti konteinerį reikia jį pastatyti tiesiai po kranu.

Konteinerius norima iškrauti jų numerių eilės tvarka, pradedant nuo 1-o. Traukinys juda ratu ir tik į priekį. Apvažiavęs ratą, traukinys gali vėl pastatyti norimą iškrauti konteinerį po kranu.



Konteinerius iškrauti reikia tokia tvarka: 1, 2, 3, 4. Važiuodamas pirmą ratą traukinys praveža 4-ą konteinerį, iškraunamas 1-as konteineris, toliau praveža 3-ią konteinerį, iškraunamas 2-as konteineris. Darydamas antrą ratą traukinys praveža 4-ą konteinerį, iškraunamas 3-ias konteineris. Traukinys turi apvažiuoti dar vieną ratą, kad būtų iškrautas paskutinis, 4-as konteineris.

Kiek ratų turės apvažiuoti traukinys, kad būtų iškrauti visi šio sąstato konteineriai?



Paaiškinimas

Teisingas atsakymas: 7.

Reikiama konteinerių iškrovimo tvarka yra 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Jei laikysimės anksčiau aprašytos tvarkos, tai važiuojant pirmą ratą bus iškrauti 1-as ir 2-as konteineriai, antrą ratą – 3-as ir 4-as konteineriai, trečią ratą – 5-as konteineris, ketvirtą – 6-as, penktą – 7-as ir 8-as konteineriai kartu, šeštą ratą – 9-as konteineris ir galiausiai važiuojant septintą ratą – 10-as konteineris. Taigi traukiniui teks apvažiuoti 7 ratus.

Patyrinėkime dėžių numerių seką. Pastebėsime taisyklę, kad kiekvienam skaičiui 1, 2, ..., jei į kairę nuo jo eina didesnis skaičius, reikia atlikti papildomą ratą. Pavyzdžiui, jei 3 yra į kairę nuo 2 (yra prieš), tai 3 bus praleistas (nes reikia iškrauti 2-ą konteinerį), todėl reikės papildomo rato, kad 3 patektų po kranu. Pateiktoje užduotyje poros, kurios eina ne eilės tvarka, yra šios: (2, 3), (4, 5), (5, 6), (6, 7), (8, 9) ir (9, 10), todėl reikės 6-ių papildomų ratų, taigi iš viso traukinys apsuks 7-us ratus.

Tai informatika!

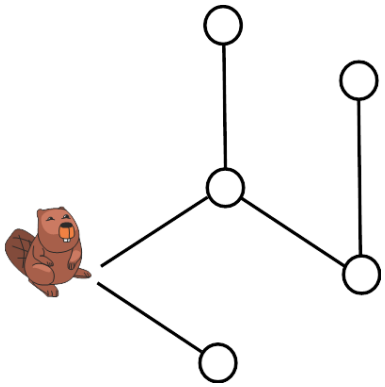
Jei sveikųjų skaičių sekoje didesnis skaičius eina prieš mažesnį, tai vadinama inversija. Kiekvienai inversijai reikia papildomo traukinio rato. Jei suskaičiuosime inversijų skaičių, tai ir bus šios užduoties atsakymas. Inversijų skaičiavimas pageidaujamos sekos atžvilgiu turi daug taikymų. Kai kuriems rikiavimo algoritmams, pavyzdžiui, rikiavimui burbulu, inversijų skaičius parodo, kiek apsikeitimų reikia tam tikrai sekai surikiuoti. Jei du klientai pagal pageidavimus reitinguoja tą patį prekių rinkinį, inversijų skaičius jų reitinguose parodo, kiek sutampa jų skoniai. Tai naudojama internetinėse parduotuvėse „panašaus skonio“ klientams nustatyti, kad būtų galima rekomenduoti produktus.

Informatinis mąstymas

Šiai užduočiai spręsti galima pasitelkti algoritminį mąstymą. Iškrovimo procedūrą galima apibūdinti žingsnių seka, kurią turime atidžiai sekti, kad nustatytume, kuriuos konteinerius galima iškrauti. Šią užduotį taip pat galima spręsti pasitelkus dekompoziciją drauge su logika: nagrinėjame poras (1, 2), (2, 3), (2, 3), (3, 4), … ir žiūrime, kuriose didesnis skaičius eina prieš mažesnį – tai ir lemia iškrovimo ratų skaičių.

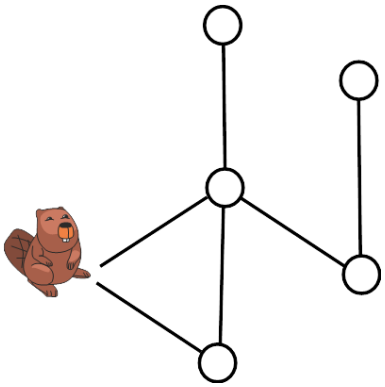
27. Tomas ir jo kaimynai

Bebrų kaime yra šeši namai. Namus jungia keliai, kiekvienu jų keliauti užtrunka tiek pat laiko. Bebras Tomas sudarė kelių, jungiančių jo namą su kitais namais, planą, kad iš savo namo į bet kurį kitą namą galėtų nuvykti trumpiausiu keliu. Tomo sudarytame plane nebūtinai yra visi kaimo keliai, nes jis įtraukė tik trumpiausius kelius.

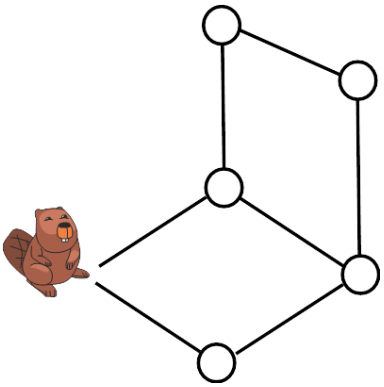


Yra daugybė būdų, kaip galėtų atrodyti kaimo planas su visais keliais. Kuris iš šių variantų, atsižvelgiant į Tomo sudarytą planą, negali būti tinkamas kaimo planas?

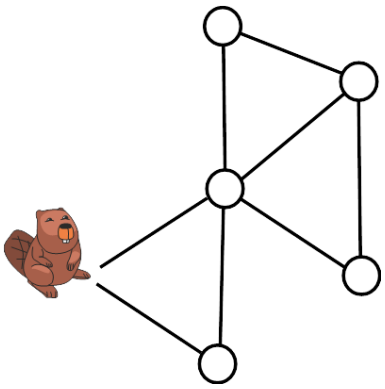
A.



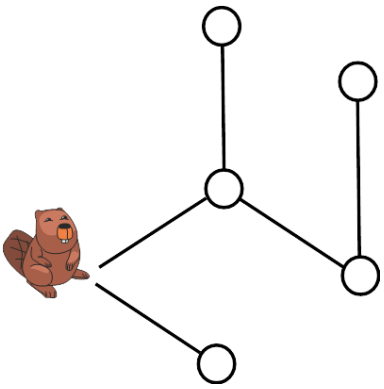
B.



C.



D.



Paaiškinimas

Teisingas atsakymas: C.

Tomo plane toliausiai į dešinę esančiam namui pasiekti reikia įveikti tris kelius. Jei C yra teisingas kaimo planas, šį namą galima pasiekti greičiau, t. y., reikia įveikti tik du kelius. Tai reiškia, kad arba Tomo planas neteisingas, arba C negali būti kaimo planas.

Visi kiti variantai gali būti kaimo planai. Kiekviename iš jų nuo Tomo namų iki bet kurio iš jo draugų namų galime nueiti per tiek pat kelių, kiek parodyta Tomo plane.

Tai informatika!

Grafų teorija yra pagrindinė informatikos mokslo sritis. Grafas – tai viršūnių ir briaunų aibė, o briauna – nesutvarkyta viršūnių pora. Grafai tinka modeliuoti tikrovę, pavyzdžiui, taškus, kuriuos jungia miesto keliai.

Klasikinis grafų algoritmų uždavinys yra rasti trumpiausius kelius iš vienos specialios viršūnės, vadinamos šaknimi, į bet kurią kitą grafo viršūnę. Tai galima padaryti pasinaudojus standartiniu paieškos į plotį (angl. *breadth first search*, BFS) algoritmu.

Informatinis mąstymas

Labai plati yra algoritmų sritis, kurią aprėpia optimizavimas. Norime gauti atsakymą, kuris būtų optimalus kitų atsakymų atžvilgiu. Pavyzdžiui, galime sakyti, kad medis, gautas pasinaudojus paieška į plotį, yra optimalus, jei atitinkami keliai iš šaknies yra trumpiausi keliai.

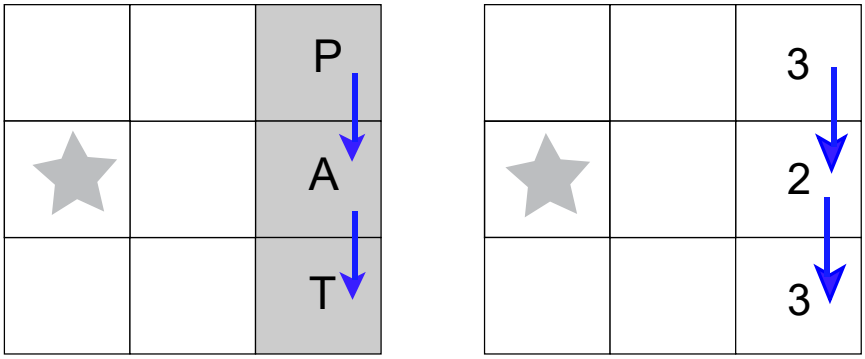
Optimalumo supratimas labai svarbus informatiniam mąstymui. Šiuo konkrečiu atveju matome, kad sprendinio optimalumas prarandamas, jei tam tikra briauna sujungia viršūnes optimaliau nei pradinis grafas.

28. Arčiau ar toliau?

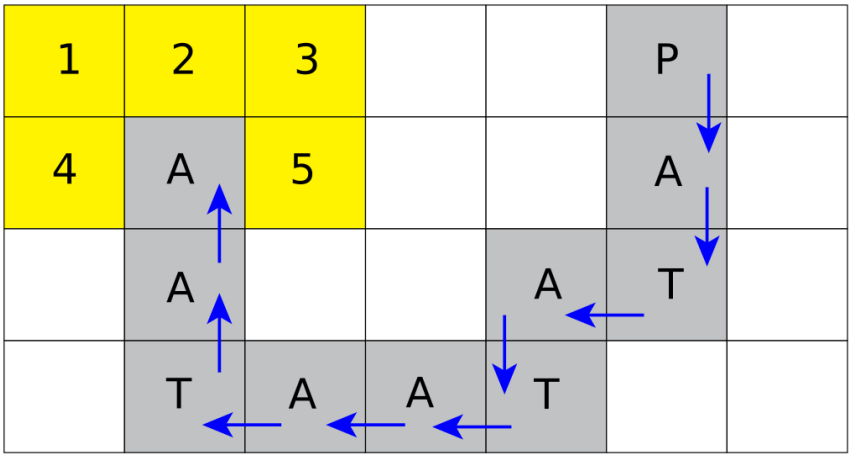
Danas žaidžia žaidimą: kvadratėlių tinklelyje aiškinasi, kur užkastas lobis. Danas pradeda iš pradinio kvadratėlio, pažymėto **P**, ir gali žengti **vieną žingsnį** tik horizontaliai arba vertikaliai į kaimyninius kvadratėlius. Po kiekvieno žingsnio Danas gauna informuojantį signalą, ar jis yra arčiau (A), ar toliau (T) nuo lobio. **Atstumas** iki lobio matuojamas mažiausiu žingsnių skaičiumi, kuriuo jis pasiekiamas.

Pavyzdžiui, pateiktame 3x3 tinklelyje lobis yra užkastas po kvadratėliu, pažymėtu  .

Danas žengia du žingsnius į priekį – jie parodyti rodyklėmis. Skaičiai greta rodo atstumą iki lobio. Po kiekvieno žingsnio Danas gauna signalus A ir T atitinkamai.



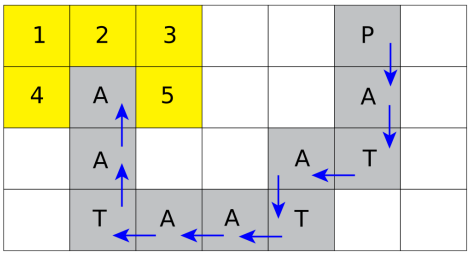
Danas ieško lobio 4x7 tinklelyje, rodyklėmis parodytas jo kelias ir gauti signalai. Danas taip pat gavo užuominą, kad lobis užkastas po vienu iš penkių numeruotų kvadratėlių.



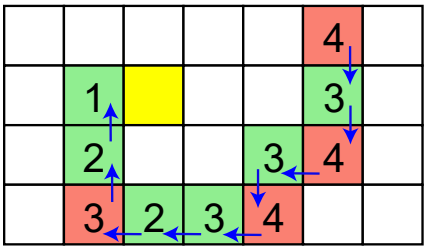
Po kuriuo iš kvadratėlių užkastas lobis?

Paaiškinimas

Teisingas atsakymas – kvadratėlis, kurio numeris 5.

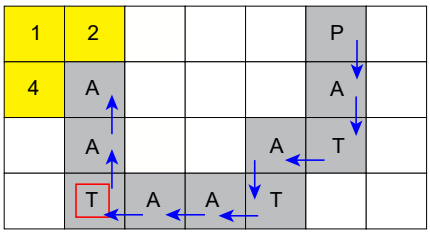


Patikriname, ar teisingai tinklelyje sužymėti A ir T signalai – tam kiekviename kvadratėlyje surašome atstumus iki 5-ojo kvadratėlio. Įsitikiname, kad jei judant rodyklės kryptimi atstumas iki 5-ojo kvadratėlio padidėjo, tai pažymėta raide T, o jei šis atstumas sumažėjo, tuomet pažymėta A. Matome, kad pateikta signalų seka atitinka mūsų nurodytą rezultatą. Taigi, lobis tikrai padėtas po 5-uoju kvadratėliu.

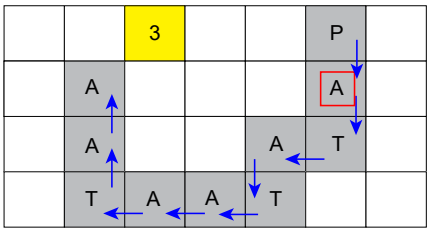


Patyrinėję kitas galimas lobio vietas signalų sekoje rastume neteisingą raidę.

Jei lobis būtų užkastas po 1-u, 2-u arba 4-u kvadratėliu, tuomet 4-os eilutės ir 2-o stulpelio kvadratėlyje esanti raidė turėtų būti A, nes šis kvadratėlis yra arčiau lobio (visoms trims vietoms), nei kvadratėlis, iš kurio buvo ateita (t. y., 4 eilutė, 3 stulpelis).



Jei lobis būtų užkastas po 3-iu kvadratėliu, tai 2-os eilutės ir 6-o stulpelio kvadratėlio raidė turėtų būti T, nes ši kvadratėlis yra toliau nuo lobio, nei kvadratėlis, iš kurio Danas pradėjo žaidimą (t. y., 1 eilutė, 6 stulpelis).



Tai informatika!

Sustiprintas mokymasis (angl. *reinforcement learning*) – tai mašininio mokymosi metodas, kuriuo siekiama, kad protingi agentai turėtų atlikti veiksmus aplinkoje taip, kad maksimaliai padidėtų bendras atlygis. Pagrindiniai sustiprinto mokymosi sistemos komponentai yra agentas arba besimokantysis, aplinka, su kuria jis sąveikauja, gairės ar taisyklės, kurių laikosi agentas, priimdamas sprendimus, ir atlygio signalas, kurį jis gauna atlikęs veiksmus. Norint įvertinti atlygio signalą, vertės funkcija matuojamas tam tikros būsenos „gerumas“.

Šioje užduotyje aplinka yra nauja vieta po ėjimo žaidimo tinklelyje, o signalas, gautas iš nustatyto atstumo, yra atlygio signalas (A reiškia teigiamą, o T – neigiamą atlygį). Žaidėjas Danas, kuris renka atlygio signalus, priimdamas optimalų sprendimą kitam žingsniui, yra agentas. Jis turi įvertinti galimybę, kad kvadratėlyje yra lobis, o tai panašu į vertės funkciją.

Verta paminėti, kad šioje užduotyje atstumas tarp kvadratėlių matuojamas Manheteno atstumu (dar vadinama taksometrija), o žaidėjas gali judėti tik horizontaliai arba vertikaliai.

Autonominis vairavimas – tai pavyzdys, kaip galima taikyti sustiprintą mokymąsi. Kad sėkmingai veiktų nenuspėjamoje aplinkoje, autonominio vairavimo sistema turi atlikti daugybę stebėjimo ir planavimo užduočių (transporto priemonės kelio planavimą, judėjimo prognozavimą). Transporto priemonės kelio planavimas susijęs su įvairiomis žemo ir aukšto lygio strategijomis, naudojamomis priimant sprendimus, kurie įvertina skirtingus laiko ir erdvės mastelius. Kita vertus, judėjimo prognozavimas apima pėsčiųjų ir kitų transporto priemonių judėjimo prognozavimą, pateikiant įžvalgas apie tai, kaip situacija gali pasikeisti priklausomai nuo aplinkos būklės.

Informatinis mąstymas

Sprendžiant šią užduotį ir kuriant strategiją, kaip judėti į priekį, būtina laikytis sisteminio požiūrio, apimančio algoritmų kūrimo principus. Taip sukuriamas gerai struktūrizuotas planas kaip judėti tinkleliu ir optimizuoti lobio paieškos procesą, t. y., plėtojamas algoritmų projektavimo gebėjimas.

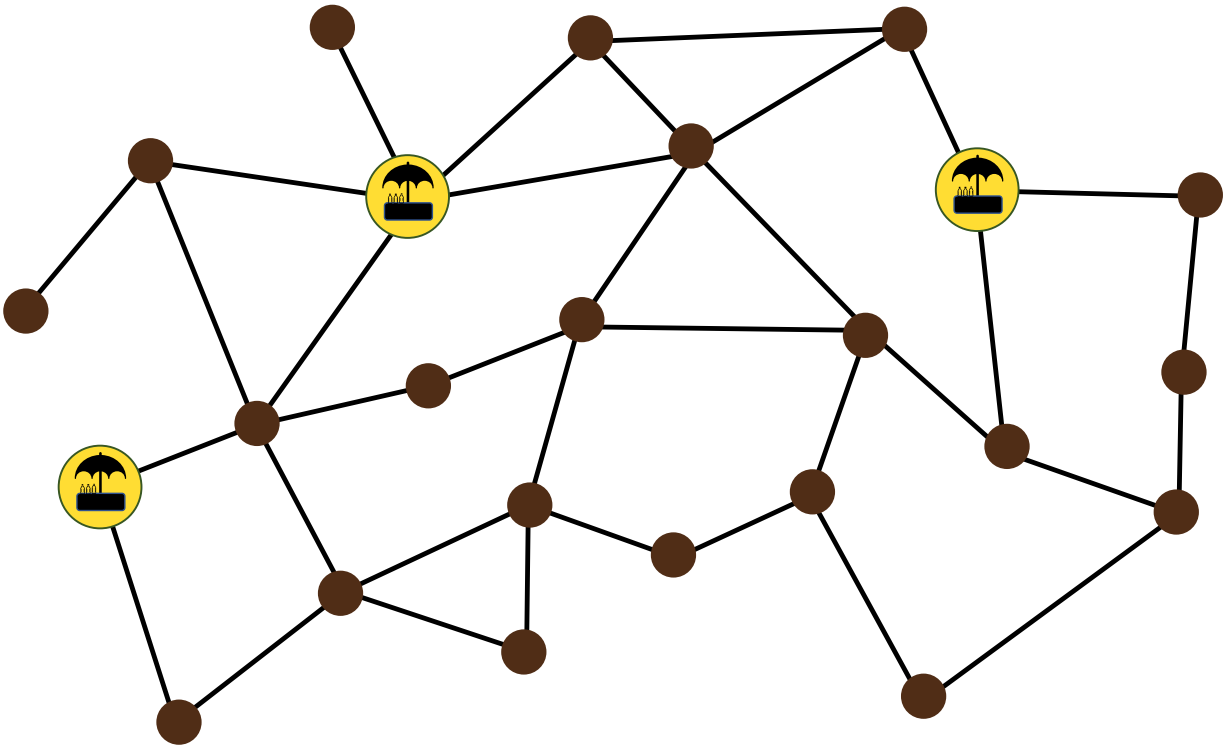
Šioje užduotyje pagal pateiktą signalą, žymimą A arba T, galima taikyti „skaldyk ir valdyk“ strategiją, kad padalijus tinklelį būtų nustatytos galimos užkasto lobio vietos. Kai judant į priekį susiduriama su signalu T, tai reiškia, kad nė vienas iš priekyje esančių kvadratėlių negali būti laikomas potencialiu sprendimu. Gera algoritmo kūrimo strategija pasitarnauja siaurinant sprendinio paieškos erdvę.

29. Sulčių vežimėliai

Vasaros Bebrijoje labai karštos, tad meras kai kuriose miesto sankryžose nusprendė pastatyti sulčių vežimėlius. Sulčių vežimėlių vietos turi būti parinktos taip, kad kiekvienas miestietis galėtų pasiekti sulčių vežimėlį iš bet kurios miesto sankryžos praeidamas ne daugiau kaip dvi gatves.

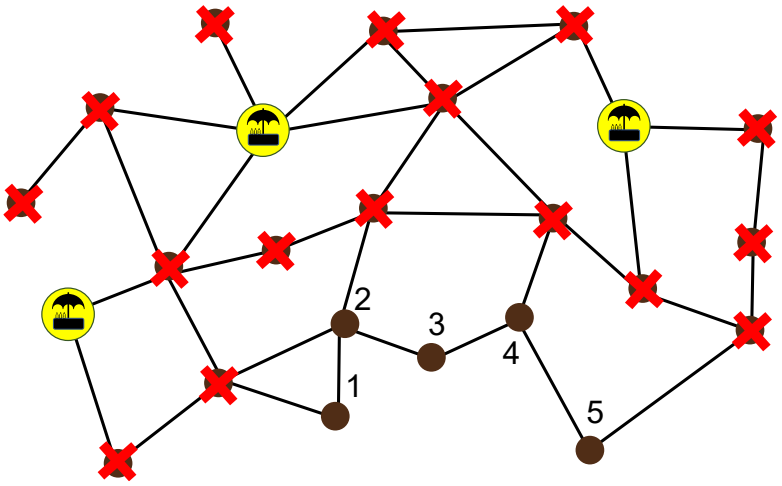
Paveikslėlyje pateiktas miesto planas. Tamsiais skrituliukais pažymėtos tuščios sankryžos, o sulčių vežimėlio piktogramos  žymi sankryžas, kuriose jau yra pastatyti sulčių vežimėliai. Gatvės vaizduojamos atkarpomis, jungiančiomis skrituliukus arba piktogramas.

Merui paskambino bebras ir pasiskundė, kad iš vienos sankryžos (deja, jis nepamena, iš kurios) neįmanoma pasiekti sulčių vežimėlio praeinant ne daugiau kaip dvi gatves. Meras paprašė pastatyti papildomą sulčių vežimėlį.



Pažymėkite vieną sankryžą, kurioje turėtų stovėti papildomas sulčių vežimėlis, kad meras būtų patenkintas.

Paaiškinimas



Norėdami rasti sankryžą papildomam sulčių vežimėliui turime patikrinti visas sankryžas, kurios yra daugiau kaip už dviejų gatvių nuo jau pastatytų sulčių vežimėlių.

Jei sulčių vežimėlį pastatysime 4-oje sankryžoje, tai užduotis nebus išspręsta, nes iš 1 pažymėtos kairės apatinės sankryžos neįmanoma pasiekti sulčių vežimėlio perėjus tik dvi gatves. Lieka 3-a sankryža. Šioje sankryžoje pastačius vežimėlį, uždavinio sąlyga bus tenkinama.

Tai informatika!

Ši užduotis susijusi su grafais – galinga ir universalia struktūra (matematine abstrakcija), naudojama skaitmeniniams duomenims apdoroti. Grafai gali būti taikomi įvairiais atvejais, pavyzdžiui, gatvėms žemėlapiuose, žmonių ryšiams, uždaros sistemos būsenoms, hierarchiškai sutvarkytiems žodžiams. Nors grafas vizualiai paprastas, patikrinti, ar jis pasižymi tam tikromis savybėmis, gali būti gana sudėtinga. Šiuo atveju kalbama apie dominuojančias aibes. Šią idėją galima pritaikyti sprendžiant realius uždavinius, pavyzdžiui, sukurti viešojo transporto tinklą, kuris aptarnautų visus gyventojus ir vietoves.

Informatinis mąstymas

Ši užduotis reikalauja iš mokinio vertinimo įgūdžių ir tam tikro lygio sisteminio mąstymo nustatyti, kuris atsakymas atitinka pateiktą sąlygą. Norint rasti atsakymą, naudotina abstrakcija (pašalinamos detalės).

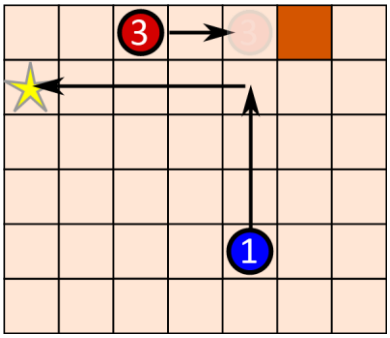
Užduoties autorius dar siūlo pabandyti mokinių paklausti, kurios sankryžos atitinka sąlygą. Arba – ar pateiktame miesto plane įmanoma taip išdėstyti sulčių vežimėlius, kad užtektų mažiau nei keturių vežimėlių? Šis klausimas sudėtingas net profesionalams.

30. Atsitrenkiantys robotai

Žaidimo „Atsitrenkiantys robotai“ tikslas – nurodyti robotui, kaip iš jo pradinės pozicijos pasiekti žvaigždę ir sustoti. Žaidime taikomos šios taisyklės:

1. Rudi kvadratai yra kliūtys.
2. Kiti robotai taip pat yra kliūtys.
3. Robotas suprogramuotas taip, kad gali judėti tik keturiomis kryptimis: aukštyn (↑), žemyn (↓), dešinėn (⟶), kairėn (⟵).
4. Pradėjęs judėti tam tikra kryptimi, robotas gali sustoti tik atsitrenkęs į kliūtį arba lentos kraštą.
5. Robotui pradėjus judėti, kiti robotai nejuda ir laukia, kol šis robotas sustos.

Paveikslėlyje parodytas atvejis, kai žaidėjas su 1 robotu trimis komandomis pasiekia žvaigždę.

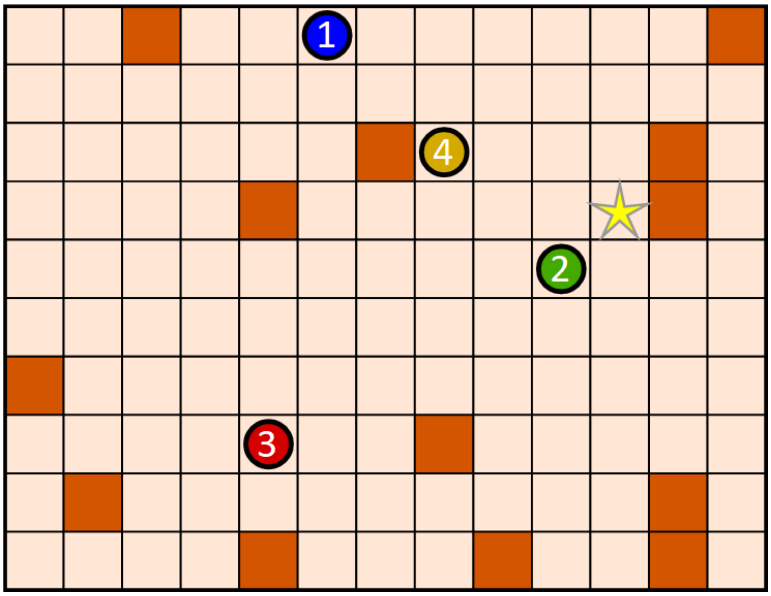


Pirmiausia žaidėjas nurodo 3 robotui judėti dešinėn. Robotas sustoja prie rudu kvadratu pažymėtos kliūties.

Po to žaidėjas nurodo 1 robotui judėti aukštyn iki trečiojo roboto.

Vykdydamas trečiąją komandą 1 robotas juda kairėn ir pasiekia žvaigždę.

Atsižvelgdami į tolesniame paveiksle pateiktą situaciją ir atlikdami mažiausiai ėjimų su 1 robotu pasiekite žvaigždę .



Paaiškinimas

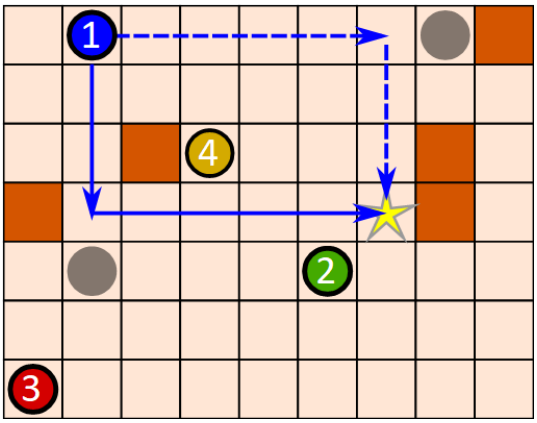
Nagrinėjant 1-o roboto galimus judėjimo variantus, matome, kad yra labai daug kombinacijų. Galima pradėti nuo galo ir ištirti paskutinį 1-o roboto ėjimą. Vienas iš galimų sprendimų yra toks, kad 1-as robotas eina iš kairės į taikinį (žvaigždę). Jei 4-as robotas pajudėtų į dešinę iki kliūties, tai 1-as robotas galėtų pasiekti žvaigždę iš apačios. Tačiau norint pasiekti iš apačios, reikėtų daugiau duoti komandų tiek 1-am robotui, tiek kitiems robotams, kurie taptų kliūtimis.

Jei 1-o roboto judėjimą į žvaigždę iš kairės laikysime paskutiniu veiksmu, galėsime patikrinti, kaip 1-as robotas gali sustoti paskutinio ėjimo pradžios langelyje. Žinome, kad žemiau to kvadrato reikia kliūties. Šis kvadratas gali būti tinkama vieta kitiems robotams sustoti. Šis procesas kartojamas tolesniuose žingsniuose, kol pasiekiame sprendimą.

Kita strategija – rasti trumpiausią kelią iki galutinio tikslo, neatsižvelgiant į tai, ar tose vietose, kur robotas pasuktų, yra kliūtis. Jei kliūties nėra, galima bandyti į šį langelį perkelti kokį nors kitą robotą.

Kodėl neužtenka trijų komandų?

Pirmiausia, nėra jokių galimybių robotui pasiekti žvaigždę atliekant tik vieną komandą.



Paveikslėlyje parodytos dvi galimybės, kai 1-am robotui užtenka vykdyti dvi skirtingas komandas. Norint įgyvendinti tokią strategiją, pirma reikia pastatyti kažkurį kitą robotą į vieną iš pilku skrituliuku pažymėtų langelių, kad 1-as robotas galėtų sustoti atsitrenkęs į kliūtį. Tačiau neįmanoma nei vienam robotui pateikti tik vienos komandos, kad šis, ją atlikęs, sustotų bet kuriame langelyje su pilku skrituliuku. Todėl keturios skirtingos komandos yra pats mažiausias skaičius tikslui pasiekti. Informatikoje jis vadinamas apatiniu rėžiu.

Tai informatika!

Sąveikavimas atlieka svarbų vaidmenį užtikrinant veiksmingą robotų ir informacinių sistemų veikimą. Nesąveikaudami tarpusavyje robotai gali pakenkti patys sau arba pakenkti žmonėms. Programuojant robotų rinkinį labai svarbu užtikrinti, kad jie sklandžiai sąveikautų.

Kurdami informacines sistemas, kuriomis naudojasi daug klientų, programuotojai turi atsižvelgti į įvairius procesus, kurie tarpusavyje turi veikti darniai. Tinkamos tvarkos nustatymas, vaidmenų apibrėžimas ir naudotojų bei sistemos komponentų hierarchinės struktūros įgyvendinimas yra labai svarbūs veiksniai, skatinantys veiksmingą sąveikavimą sistemoje.

Kelių robotų sąveikavimas gali būti sudėtingas optimizavimo uždavinys. Ieškant geriausio sprendimo kelių robotų darbo erdvėje gali kilti kombinatorinis sprogimas. Mokslininkai ir inžinieriai gali taikyti stochastinio optimizavimo algoritmus (į algoritmą įvesti atsitiktinumą), kad per tam tikrą laiką rastų geriausią, nors ir neoptimalų, sprendimą.

Informatinis mąstymas

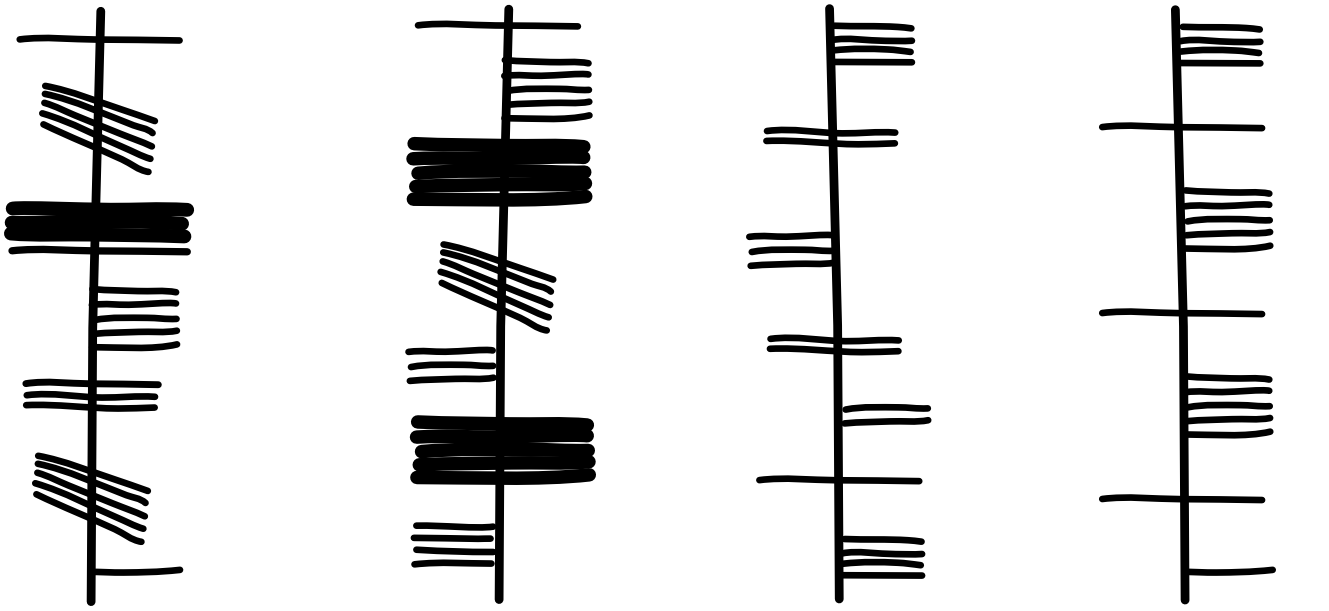
Sprendžiant šį uždavinį labai svarbu tinkamai koordinuoti kelis objektus ir nustatyti optimalią jų judėjimo tvarką ir kryptį. Tam reikia įvertinti bendrą situaciją ir išskaidyti uždavinį, kad būtų atsižvelgta į visus robotus ir kliūtis. Svarbu atsižvelgti į robotų sąveikavimą viso proceso metu, kad būtų pasiektas norimas rezultatas.

31. Ogamo abėcėlė

Mokykloje Rūta ir Vaidas susipažino su senovės keltų abėcėle, vadinama Ogamu. Joje raidės sudarytos iš įvairaus skaičiaus ir įvairiai orientuotų brūkšnių grupelių. Skirtingas raides skiria didesni tarpai tarp brūkšnių grupelių.

Rūta Vaidui parodė Ogamo abėcėle jos užrašytus žodžius: BANANAS, CITRINA, BRUNERA, SALOTOS.

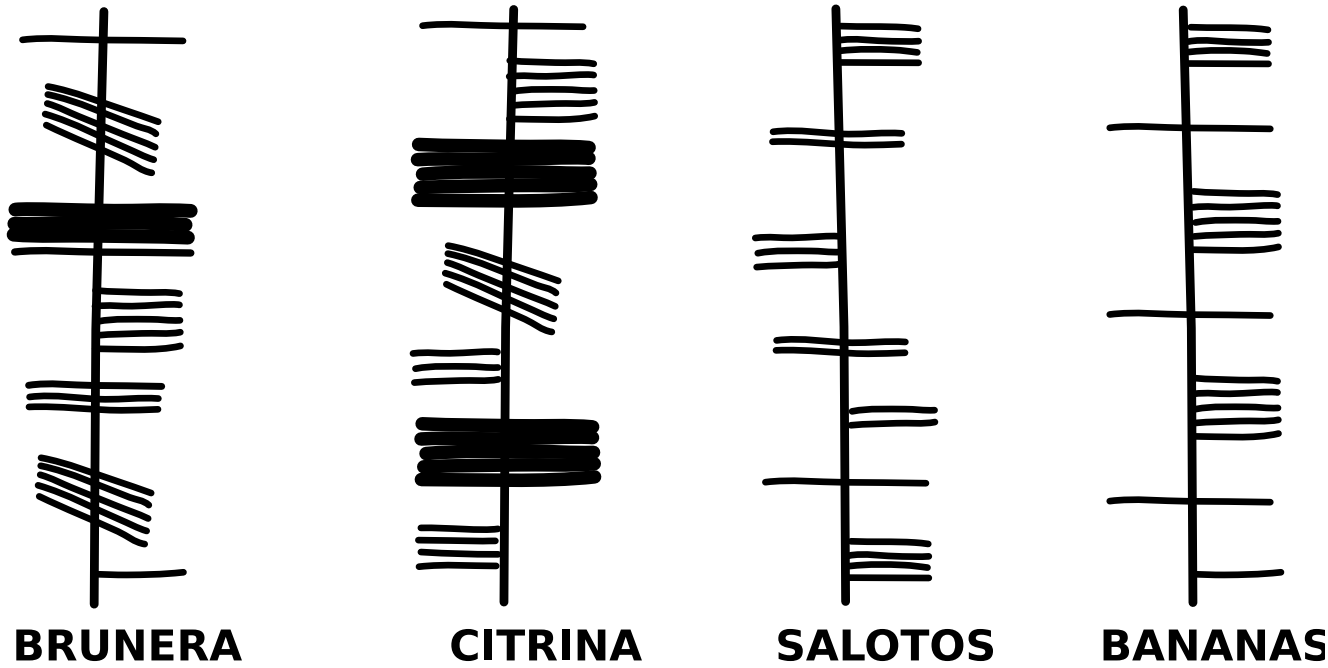
Padėkite Vaidui ir nuvilkite žodžius į tinkamus langelius.



BANANAS CITRINA BRUNERA SALOTOS

Paaiškinimas

Teisingas atsakymas:



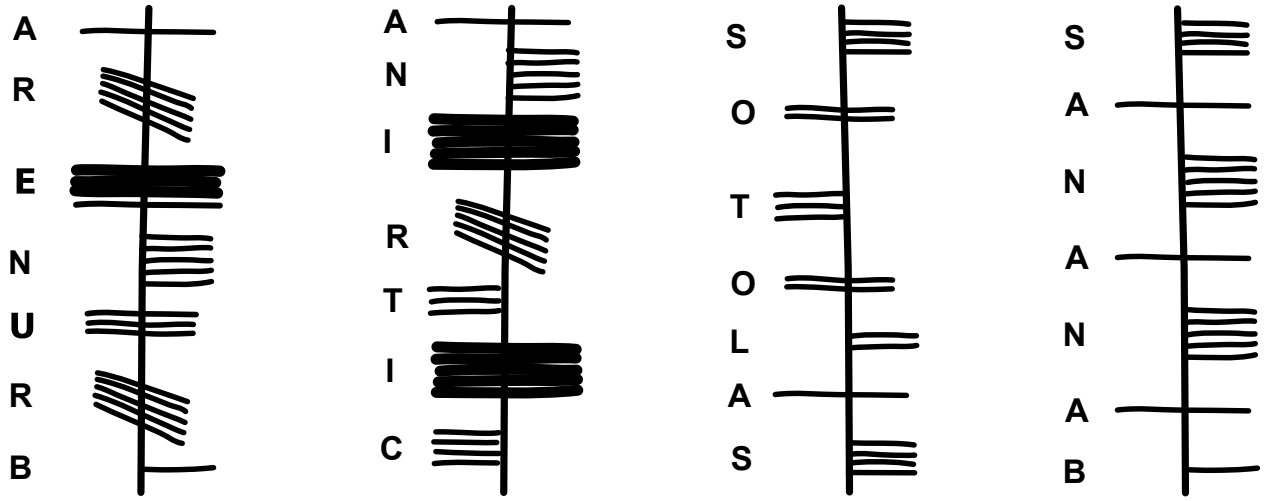
Žodžių atitikimą galima nustatyti įvairiais būdais. Pirmiausia reikia išsiaiškinti, kokia kryptimi reikia skaityti žodžius. Ne iš kairės į dešinę (arba iš dešinės į kairę), nes tokiu atveju žodžiai turėtų po 4 raides ir žodžių turėtų būti septyni. Be to, nebūtų prasminga ir langelių vieta. Dabar galima palyginti keturias viršutines ir apatines raides su pirmąja ir paskutine raidėmis žodžiuose. Kadangi du žodžiai baigiasi ta pačia raide „S“, o kiti du – raide „A“, tai jie turi baigtis viršuje, taigi skaityti reikia iš apačios į viršų.

Tai atlikę, galime analizuoti kiekvieną žodį atskirai.

BANANAS yra vienintelis žodis, kuriame yra trys raidės A, o ketvirtas paveikslėlis yra vienintelis, kuriame yra trys vienodos raidės, todėl tai turi būti BANANAS. Dabar jau žinome, kad raidė A yra vienas ilgas horizontalus brūkšnys, B – brūkšnys iš dešinės, N – penki brūkšniai iš dešinės ir t. t.

SALOTOS baigiasi S raide, tai joms tinka trečias paveikslėlis.

Dabar jau aišku, kad pirmas paveikslėlis tinka žodžiui BRUNERA, o antras – žodžiui CITRINA.



Tai informatika!

Ši užduotis – paprastas kriptoanalizės pavyzdys. Kriptoanalizė tyrinėja, kaip „nulaužti“ užšifruotą informaciją. Ji naudojama siekiant išsiaiškinti užšifruotų pranešimų turinį neturint prieigos prie rakto, kuris paprastai reikalingas šiam tikslui. Ekspertai, vadinami kriptoanalitikais, bando iššifruoti pranešimus. Tai darydami jie naudojasi savo žiniomis apie žodžius, kurie galbūt yra paslėptame pranešime.

Informatinis mąstymas

Šablono atpažinimas: sprendžiant šią užduotį svarbu rasti dėsningumus (pvz., pasikartojančias tas pačias raides) tarp žodžių ir juos palyginti su paveikslėliais.

Dekomponavimas: šią užduotį galima išspręsti išskaidant ją į išsprendžiamas dalis (pvz., išsiaiškinti po vieną raidę). Sujungus dalinius sprendimus, galima išspręsti iš pradžių sudėtingą uždavinį.

Atvaizdavimas: ši užduotis reikalauja suprasti, kaip raidės vaizduojamos įvairiais simboliais ar paveikslėliais.

32. Maršrutų skaičiavimas

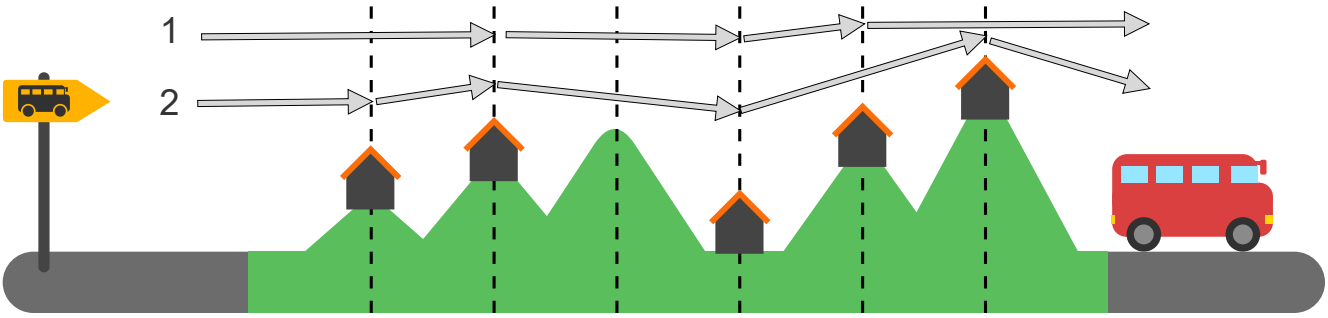
Meda mėgsta ilgus žygius pėsčiomis, kiekvieną naktį nakvodama vis kitoje vietoje. Meda planuoja žygio maršrutą ir pavaizduoja jį schema. Per dieną Meda gali nueiti tik vieną ar dvi atkarpas (pažymėtas pilkomis linijomis).

Schemoje parodyta maršruto pradžia  , pabaiga  ir visos nakvynės vietos  .

Pateikti du galimi maršrutai:

1-e maršrute yra 3 sustojimai nakvynei;

2-e maršrute yra 4 sustojimai nakvynei.



Kiek skirtingų maršrutų (įskaitant du parodytus) Meda gali pasirinkti savo žygiui pėsčiomis?

Paaiškinimas

Teisingas atsakymas: 6.

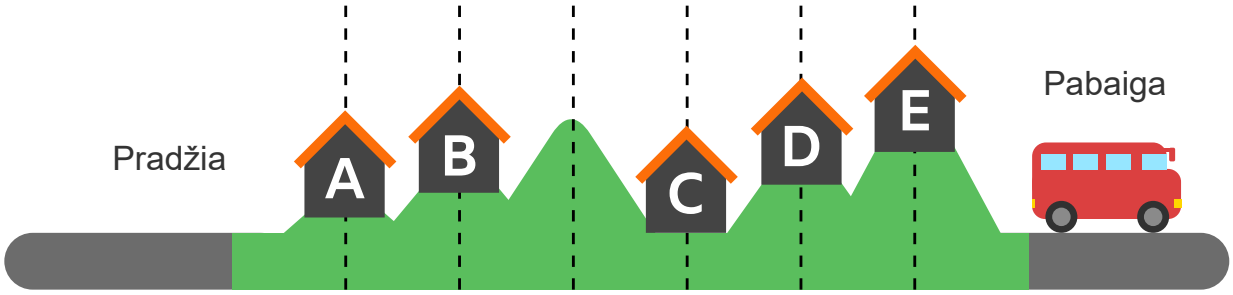
Uždavinį galime spręsti naudodami dekompoziciją. Pirmiausia pastebime, kad Meda turi apsistoti B ir C vietose, nes atstumas tarp šių dviejų vietų yra didžiausias, kurį ji gali nueiti per vieną dieną. Tai leidžia mums dvi prieš ir po atkarpas laikyti dviem mažesniais uždaviniais. Tada galima pastebėti, kad yra tik du būdai iš pradžios taško nuvykti į B (kurie parodyti 1-e ir 2-e maršrute). Todėl yra tik du galimi maršrutai į C. Toliau reikia nustatyti, kiek yra būdų patekti iš C į žygio pabaigą. Spręsdami šį mažesnį uždavinį išsiaiškiname, kad yra tik trys galimybės:

C → D → E → pabaiga

C → E → pabaiga

C → D → pabaiga

Tai reiškia, kad iš viso yra 2 x 3 galimi maršrutai.



Tai informatika!

Šį uždavinį verta spręsti išskaidant jį į du mažesnius uždavinius. Sprendžiant panašius, tik sudėtingesnius, uždavinius naudojamas dinaminio programavimo metodas. Tai strategija, kai algoritmas suskaido uždavinį į labai mažus uždavinius ir mažesnių uždavinių sprendinius naudoja viso uždavinio sprendiniui gauti.

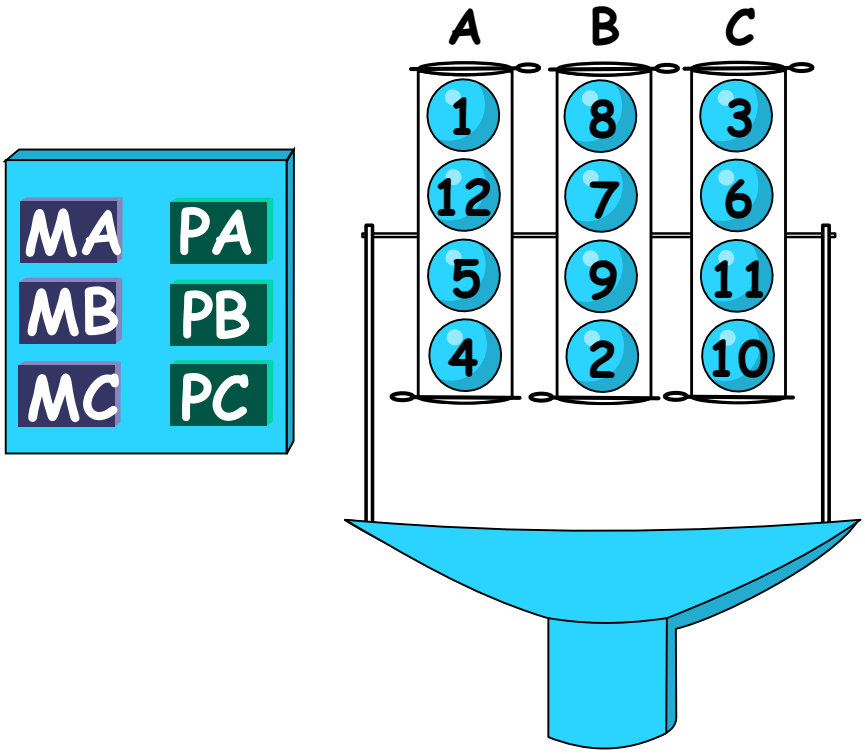
Medos apsistojimo vietas pažymėkime: pradžia, A, B, C, D, E, pabaiga. Tada sudarome lentelę, skaičiuodami, kiek yra būdų pasiekti kiekvieną vietą.

Vieta	*Keliais būdais galima į ją patekti*
Pradžia	1
A	1 (tiesiai iš pradžios)
B	1 + 1 = 2 (arba iš pradžios, arba iš A)
C	2 (reikia atvykti iš B)
D	2 (reikia atvykti iš C)
E	2 + 2 = 4 (arba iš B, arba iš C)
Pabaiga	2 + 4 = 6 (arba iš D, arba iš E)

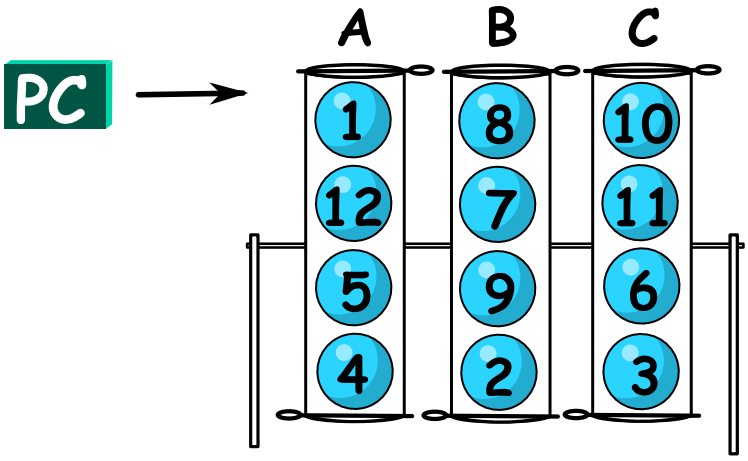
Jau aiškinant atsakymą matėme, kad uždavinio sprendimas iškelia keletą svarbių informatikos konceptų. Pirmiausia, taikome abstrakciją: namus, kalnus ir autobusus pakeičiame taškais, pažymėdami juos raidėmis. Padarius tai paprasčiau taikyti algoritmą mažesniems uždaviniams spręsti – kiek yra būdų pasiekti kiekvieną tašką, pradedant nuo pradinio taško. Toliau remiamės strategija, jog žinant atsakymus iki taško B, nesunku apskaičiuoti ir taškui C. Toks pateiktos lentelės užpildymas vadinamas dinaminiu programavimu. Tai išmanus algoritmas, padedantis rasti sprendimą tokiais atvejais kaip šis, taip pat tinka sudėtingiems uždaviniams, kuriems išspręsti reikia ypač daug laiko.

33. Naujas žaislas

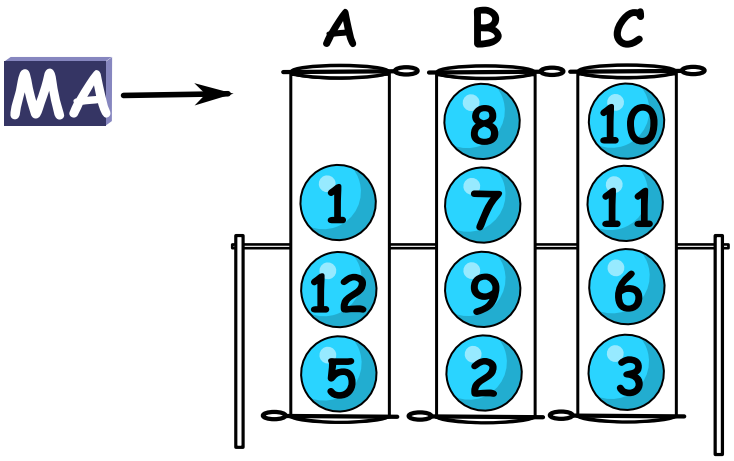
Jonas gimtadienio proga gavo dovaną – naują žaislą. Šis žaislas sudarytas iš trijų vamzdelių, kurių kiekviename yra po 4 kamuoliukus. Jonas gali paspausti vieną iš 6 mygtukų, esančių valdymo skydelyje. Mygtukai PA, PB ir PC pasuks atitinkamai vamzdelį A, B ir C 180 laipsnių (t. y., apvers žemyn). Mygtukai MA, MB ir MC mes vieną kamuoliuką iš atitinkamo vamzdelio į saugyklą (žr. pavyzdį paveiksle).



Pavyzdžiui, paspaudus mygtuką PC, C vamzdelis apsivers:



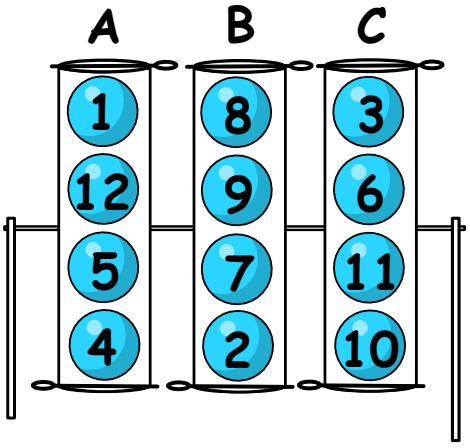
Tada paspaudus MA, kamuoliukas iškris iš vamzdelio A į saugyklą:



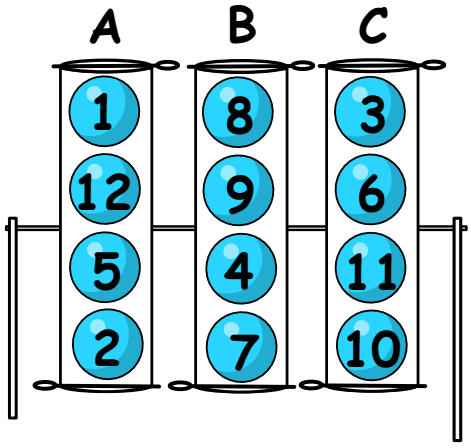
Jonas nori visus kamuoliukus perkelti iš vamzdelių į saugyklą kamuoliukų numerių didėjimo tvarka. Tačiau kartais kamuoliuko vieta vamzdelyje neleidžia to įgyvendinti.

Kuriai iš šių situacijų esant Jonui to padaryti nepavyks?

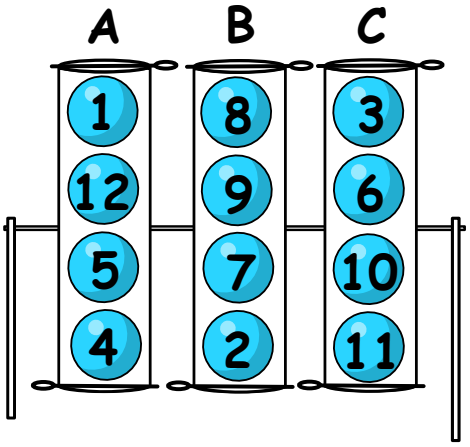
A.



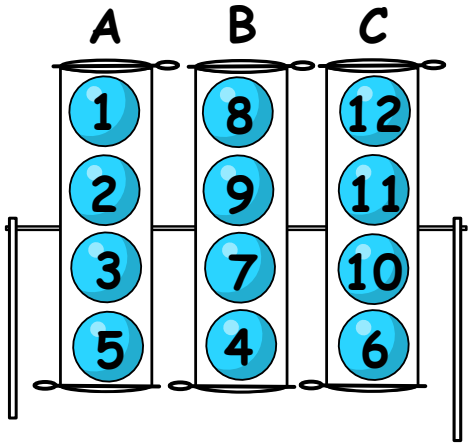
B.



C.

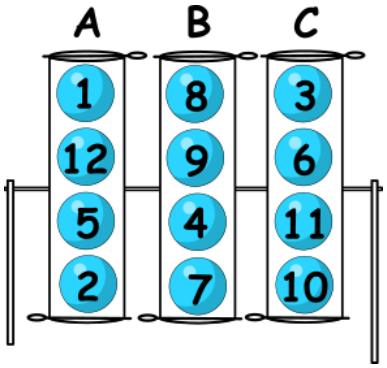


D.



Paaiškinimas

Teisingas atsakymas: B.



Žinome, kad teisinga kamuoliukų seka turėtų būti 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12. Tačiau nėra galimybės mesti į saugyklą 4-ą kamuoliuką prieš 7-ą ir 9-ą, kadangi 4-as kamuoliukas yra tarp kamuoliukų, pažymėtų numeriais 7 ir 9.

Visi kiti variantai tinka kamuoliukams surikiuoti.

Tam, kad Jonas kamuoliukus į saugyklą sumestų didėjimo tvarka, kamuoliukai kiekviename vamzdelyje nuo apačios iki viršaus turėtų būti išdėstyti vienu iš šių būdų:

1) didėjimo tvarka;
2) mažėjimo tvarka;
3) didėjimo tvarka, po kurios eina seka, surikiuota mažėjimo tvarka.

A variantas – neteisingas. Kamuoliukai A vamzdelyje išdėstyti didėjimo tvarka [4, 5, 12], po kurių eina seka mažėjimo tvarka [12, 1]. Kamuoliukai B vamzdelyje išdėstyti didėjimo tvarka [2, 7, 9], po jų – mažėjimo tvarka [9, 8]. C vamzdelyje kamuoliukai išdėstyti didėjimo tvarka [10, 11] ir mažėjimo tvarka[11, 6, 3].

C variantas – neteisingas. Kamuoliukai A vamzdelyje išdėstyti didėjimo tvarka [4, 5, 12], po kurių eina seka mažėjimo tvarka [12, 1]. Kamuoliukai B vamzdelyje išdėstyti didėjimo tvarka [2, 7, 9], po jų – mažėjimo tvarka [9, 8]. C vamzdelyje kamuoliukai išdėstyti didėjimo tvarka [11, 10, 6, 3].

D variantas – neteisingas. Kamuoliukai A vamzdelyje išdėstyti didėjimo tvarka [5, 3, 2, 1]. Kamuoliukai B vamzdelyje išdėstyti didėjimo tvarka [4, 7, 9], po jų – mažėjimo tvarka [9, 8]. C vamzdelyje kamuoliukai išdėstyti didėjimo tvarka [6, 10, 11, 12].

Tai informatika!

Ši užduotis priklauso rikiavimo uždavinių kategorijai. Rikiavimo algoritmai – vieni pagrindinių informatikos algoritmų. Rikiavimo algoritmų daug, vienas iš jų – sąlajinis rikiavimas. Sąlajinis rikiavimas yra efektyvus palyginimu grįstas rikiavimo algoritmas. Tai metodas, kai rikiuojamos sekos elementai skirstomi į grupes (rūšiuojami), o po to grupės paeiliui suliejamos į vieną seką. Procesas kartojamas keičiant skirstymo į grupes parametrą, kol visa seka tampa surikiuota. Sekos elementai gali būti rekursyviai dalijami į dvi ar tris dalis – kaip šioje užduotyje, kai iš trijų vamzdelių išrenkamas mažiausias elementas ir perkeliamas į saugyklą. Tik skirtingai nuo sąlajinio rikiavimo, kamuoliukai vamzdeliuose yra nerikiuoti, todėl turime apversti vamzdelį.

Vamzdeliai yra abipusių eilių (angl. *double-ended queue*, *deque*) pavyzdžiai. Abipusė eilė yra abstraktusis duomenų tipas, leidžiantis įterpti ir šalinti elementus iš abiejų eilės galų.

Vamzdeliuose esančius kamuoliukus galime laikyti programos įvesties duomenimis. Pateiktos konfigūracijos atitinka tris bandymų pavyzdžius, kuriems mūsų programa pateikia sprendinius, ir vieną bandymų pavyzdį, kuriam sprendinio nėra.

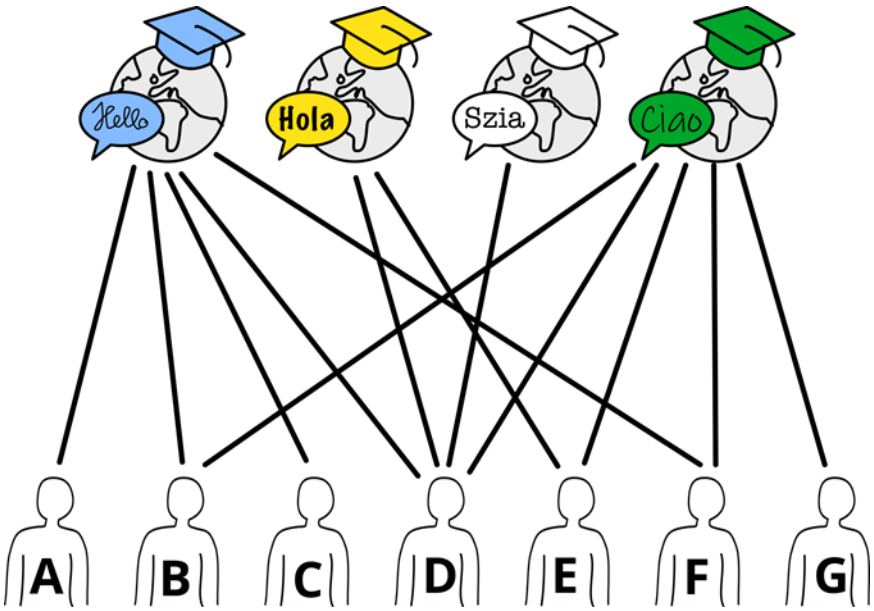
Informatinis mąstymas

Algoritmai. Sprendžiant šią užduotį reikia suprasti, kaip veikia algoritmas. Sudarome operacijų seką arba programą (šioje užduotyje – posūkis ir metimas), kad pasiektume tikslą (kamuoliukai saugykloje būtų surikiuoti).

Abstrakcija. Vamzdeliams vaizduoti galime naudoti masyvą arba abipusę eilę ir patikrinti, ar jie tinka pradiniams duomenims mūsų programai pateikti. Sukuriamos bendrosios sąlygos, kad patikrintume, ar kamuoliukai vamzdeliuose gali būti išdėstyti didėjimo tvarka.

34. Darbų paskirstymas

Užsienio kalbų mokykla planuoja keturis vasaros kursus. Esami mokytojai gali mokyti šių kalbų (žr. schemą):



Kiekvienas kursas turi turėti mokytoją ir kiekvienas mokytojas gali mokyti tik vieną kursą. Mokytojai, kurie nemoko jokio kurso, paskiriami dirbti kitur.

Kuris iš šių teiginių yra neabejotinai teisingas?

a) Jei B, F ir G nėra, vienas kursas turi būti atšauktas.

b) D mokys ispanų kalbos 

kursą.

c) E mokys italų kalbos 

kursą.

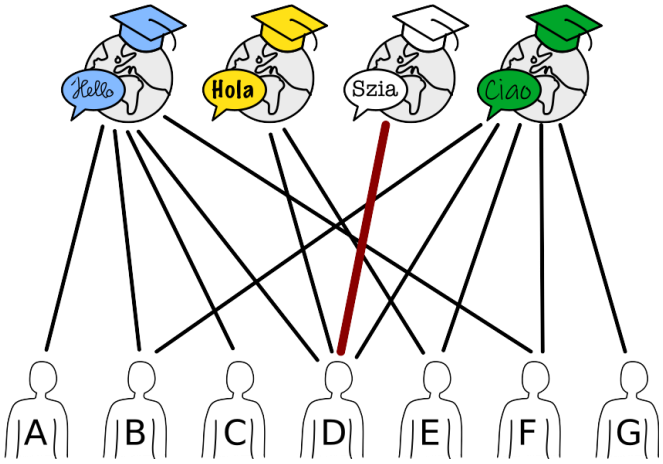
d) 4 mokytojai neturi kurso, kurį galėtų mokyti.

Paaiškinimas

Teisingas atsakymas: A.

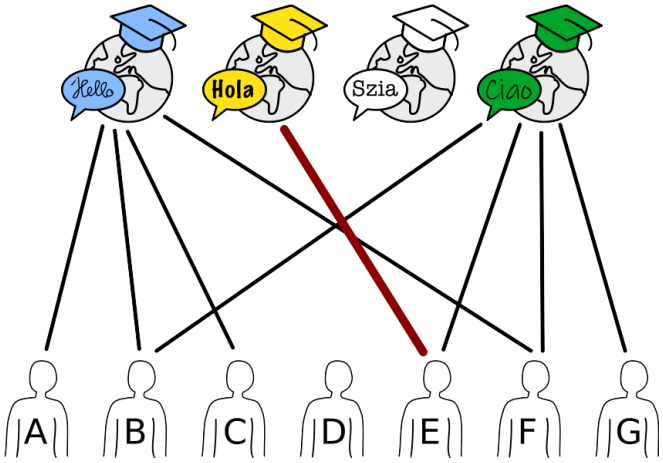
D yra vienintelis mokytojas, kuris gali mokyti vengrų 

kalbos kursą. Taigi šis kursas užimtas ir D negali mokyti kitų kalbų kursų. (Atsakymas B yra klaidingas)

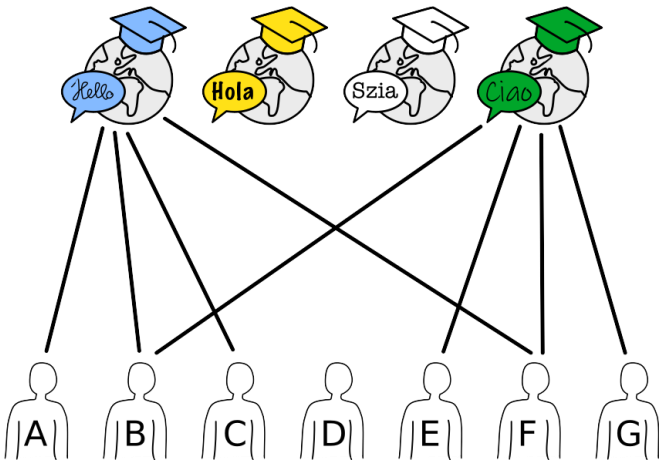


E yra vienintelis mokytojas, kuris gali mokyti ispanų 

kalbos kursą. Taigi šis kursas užimtas ir E negali mokyti kitų kalbų kursų. (Atsakymas C yra klaidingas)



Likusius du kursus galima laisvai rinktis. Pavyzdžiui, A galėtų mokyti anglų ir B – italų. Tai reiškia, kad yra 4 mokytojai, kurie moko 4 kursus, taigi lieka 7–4 = 3 mokytojai, kurie neturi jokio kurso. (Atsakymas D yra klaidingas).



Dabar panagrinėkime variantą A. Dar kartą pažvelgę į ankstesnį paveikslėlį pastebėsime, kad D turi mokyti vengrų kalbos, o E – ispanų kalbos, todėl likę italų kalbos mokytojai yra B, F ir G. Jei „pašalinsime“ B, F ir G, neliks italų kalbos mokytojo, taigi A variantas yra teisingas.

Tai informatika!

Kalbų kursų ir mokytojų duomenų struktūrai pavaizduoti šioje užduotyje panaudotas dvidalis grafas (angl. *bipartite graph*, *bigraph*). Tai yra ribojimų tenkinimo uždavinys, kai kintamiesiems priskiriant reikšmes atmetamos kitos galimybės. Uždavinį galima pavaizduoti naudojant tradicinį aibės užrašą, pavyzdžiui, ispanų = {D, E}.

Ribojimų tenkinimo uždaviniai būdingi daugeliui realaus pasaulio išteklių paskirstymo uždavinių ir galvosūkių (pvz., sudoku).

Šios užduoties esmė – mokytojams paskirstant kursus naudoti dvidalius grafus. Dvidaliai grafai naudoja dvi viršūnių kategorijas, kurios leidžia polinominio laiko algoritmu spręsti viršūnių atitikimo uždavinį.

Informatinis mąstymas

Užduotyje naudojamas informatinio mąstymo konceptas – dekomponavimas nagrinėjant kiekvieno mokytojo darbą atskirai ir nepriklausomai nuo kitų darbų.

Sprendžiant užduotį naudojamas loginis mąstymas, teigiant „jei kurį nors kursą moko tik vienas mokytojas, jis turi mokyti šį kursą, net jei jis gali mokyti ir kitus kursus“. Tuo remiantis randamas algoritmas šiam uždaviniui išspręsti, o tai yra informatinio mąstymo ugdymas.

35. Bebrų statybos kompanija

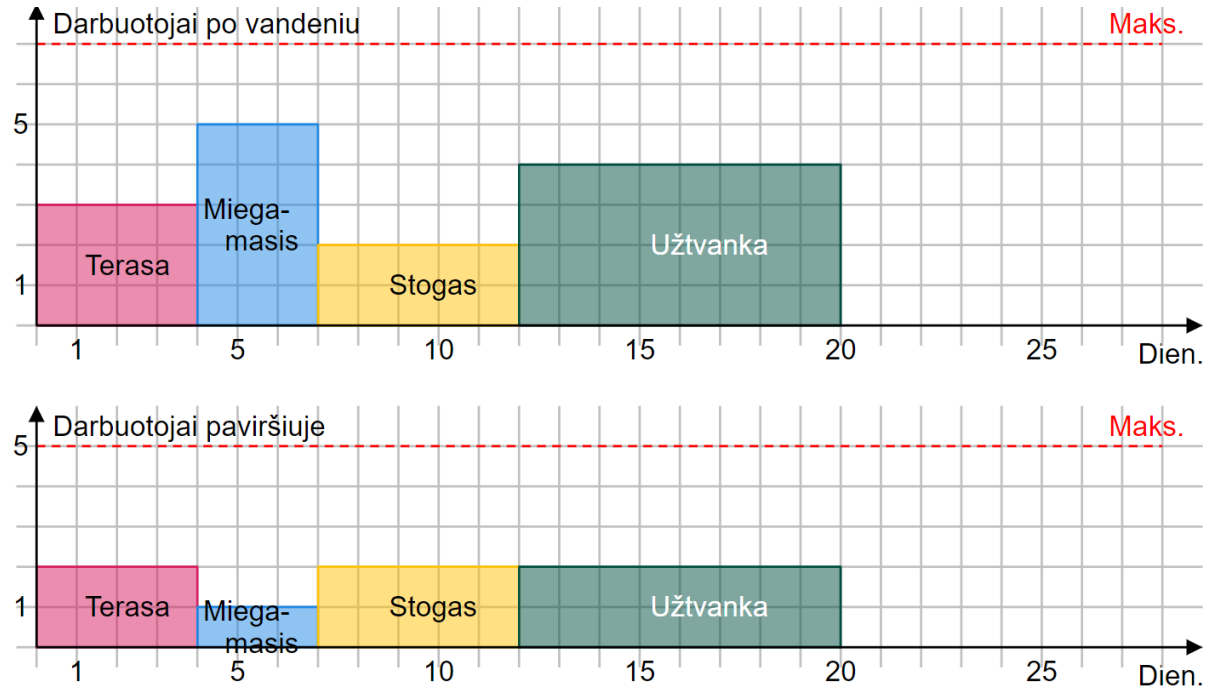
Bebrų statybos kompanija – tai bebrų specialistų komanda, kuri stato namelius bebrų šeimynoms. Bebrai gali dirbti po vandeniu arba paviršiuje. Jie turi pastatyti tokias keturias konstrukcijas:

Konstrukcija	**Terasa**	**Miegamasis**	**Stogas**	**Užtvanka**
Reikalingas laikas	4 dienos	3 dienos	5 dienos	8 dienos
Darbuotojai darbams po vandeniu	3	5	2	4
Darbuotojai darbams paviršiuje	2	1	2	2

Stogas gali būti statomas tik tada, kai užbaigtas įrengti miegamasis.

Sukurkite bebrams tvarkaraštį, kuris leistų darbus atlikti kiek galima sparčiau, kai turimi 7 darbuotojai darbui po vandeniu ir 5 – darbui paviršiuje.

Vilkdami atitinkamas konstrukcijas žyminčius stačiakampius sudarykite optimalų tvarkaraštį.



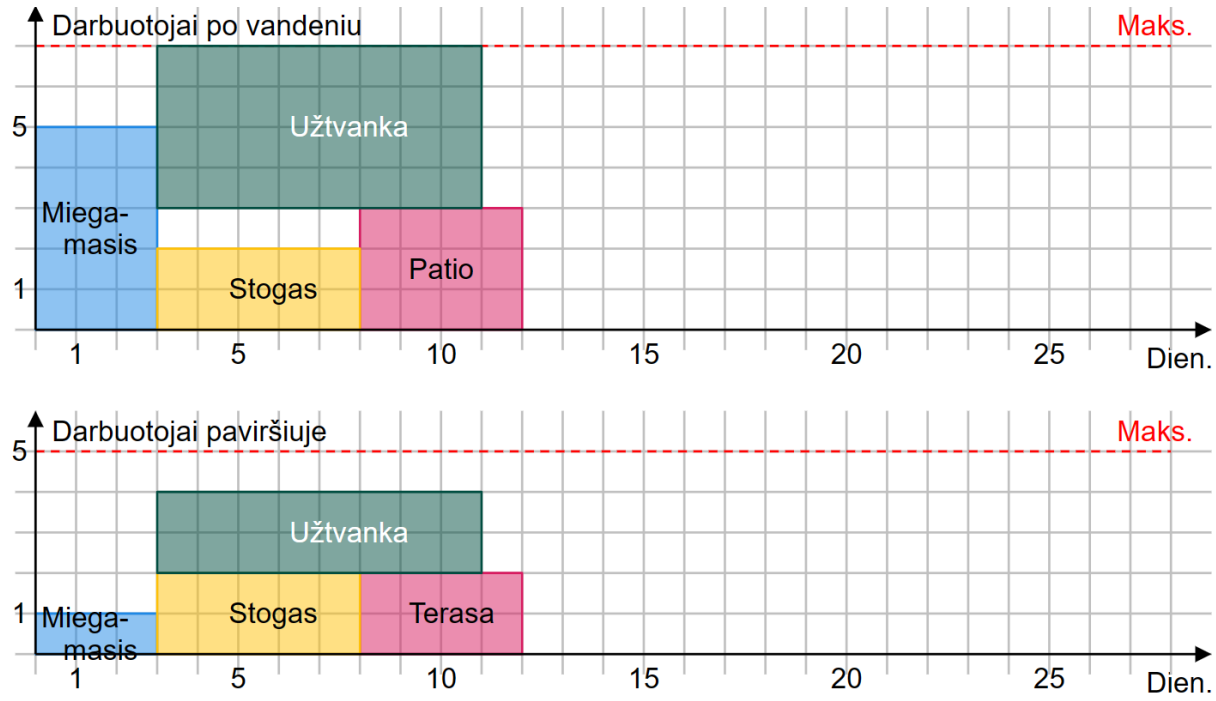
Paaiškinimas

Ieškant optimalaus sprendimo reikia atlikti du žingsnius.

Pirma, stogas turi būti planuojamas statyti vėliau negu miegamasis. Kadangi užtvankai ir terasai pastatyti reikia nuo 3 iki 4 darbuotojų darbams po vandeniu, o miegamajam reikia 5 tokių darbuotojų, tai 7 darbuotojai, dirbantys po vandeniu, negali statyti miegamojo ir vienos iš kitų dviejų konstrukcijų vienu metu. Tai reiškia, kad miegamasis statomas neatliekant jokių kitų darbų tuo pačiu metu, o stogas statomas tam tikru metu po miegamojo.

Antra, turi būti pastatytos likusios dvi konstrukcijos (užtvanka ir terasa). Vieną iš jų galima statyti lygiagrečiai su stogo darbais, t. y., abi tuo pačiu metu. Sparčiausia kombinacija stogo ir terasos darbams realizuoti – statyti juos vieną paskui kitą, statant daugiausia laiko reikalaujančią konstrukciją (užtvanką) lygiagrečiai.

Taigi optimalus tvarkaraštis būtų 12 dienų. Vienas iš galimų planų yra toks:



Tai informatika!

Sudėtingų tvarkaraščių sudarymo uždavinių sprendimas – viena iš informatikos temų, turinti įvairių taikymo galimybių. Projektų planų sudarymas dažnai remiasi priklausomybėmis tarp jų elementų ir tokie planai turi būti įgyvendinti per gana trumpą laiką. Šio uždavinio pavyzdžio planas grafiškai vaizduojamas Ganto diagrama, kurią 1910–1915 m. sukūrė Henris Gantas (1861–1919 m.). Tokios diagramos rodo visų išteklių (šiuo atveju – dviejų tipų darbuotojų) naudojimą per tam tikrą laiką.

Problema ta, kad rasti optimalų sprendimą yra labai sudėtinga. Didesniems projektams negu šiame pavyzdyje neįmanoma per priimtiną laiką išbandyti visų galimų kombinacijų. Todėl imamas darbinis sprendimas ir bandoma optimizuoti lokaliai. Taip gaunamas sprendimas, gana artimas optimaliam.

Šis principas taikomas ne tik dideliems įmonių projektams, bet ir kompiuteriuose konkuruojantiems procesams paskirstyti remiantis turimais ištekliais (procesoriaus, prieigos prie atmintinės, prieigos prie išorinių įrenginių, pavyzdžiui, atmintinės įrenginių, spausdintuvų ar tinklo sąsajų). Išteklių planavimas vyksta realiu laiku, kad būtų kuo efektyvesnis.

Informatinis mąstymas

Sprendžiant šią užduotį reikia optimizuoti planą ir tuo pačiu laikytis dviejų tipų apribojimų: priklausomybės miegamasis → stogas ir dviejų tipų išteklių – povandeninių ir paviršiaus darbuotojų – naudojimo. Nors interaktyvumas padeda mokiniams nesudaryti neįmanomų planų, mokinių siekis pagerinti planą paskatins juos bandyti įvairius variantus. Taip mokiniai susikuria optimizavimo strategijų laikymosi metodus.

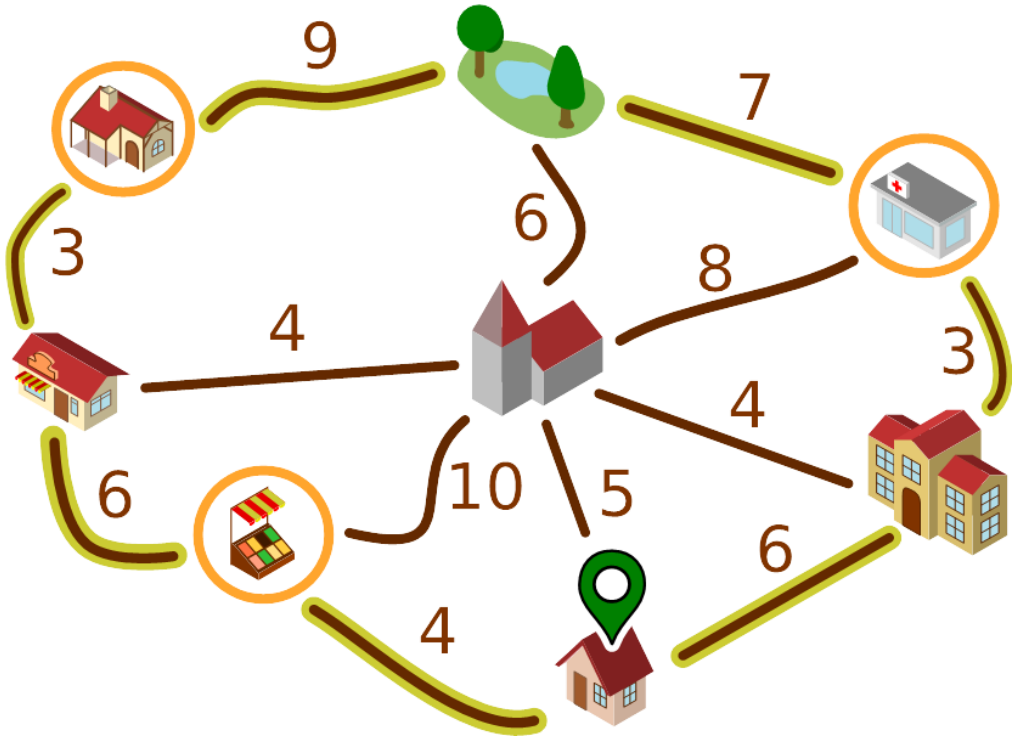
36. Emos užduotys

Emos namas pažymėtas  . Ema turi atlikti tris užduotis ir grįžti namo:

- Paimti arkliui pasagą iš kalvės  ;
- Parnešti vaistų iš vaistinės  ;
- Nupirkti vaisių turguje  .

Ema nežino, kiek užtruks atlikti visas užduotis. Ema taip pat nežino, kiek užtruks kiekvienoje miesto vietoje. Tačiau ji nori, kad ėjimo laikas būtų kuo trumpesnis.

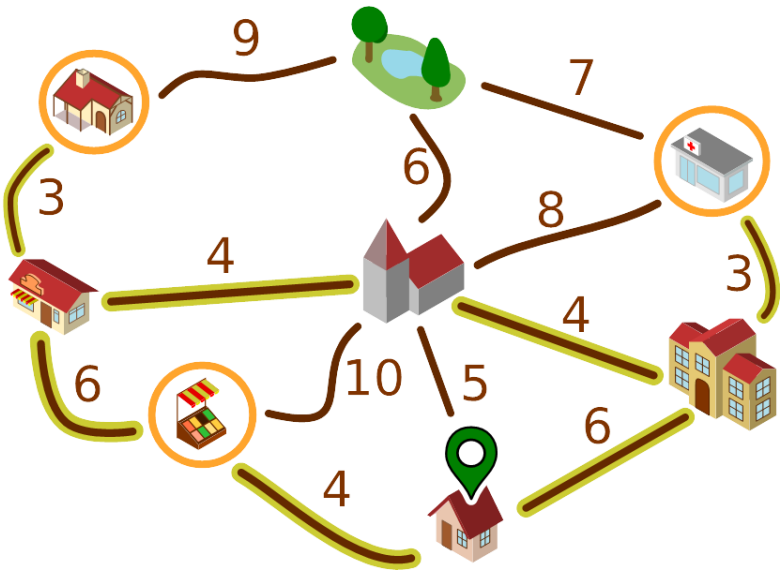
Žemėlapyje Ema surašė, kiek minučių užtrunka eidama kiekvienu keliu. Pavyzdžiui, eidama pažymėtais keliais, Ema sugaištų 4 + 6 + 3 + 9 + 7 + 3 + 6 = 38 minutes. Ema svarsto, ar galėtų atlikti kelionę dar greičiau. Gal verta kai kuriais keliais eiti ten ir atgal (suprantama, tai užtruktų dvigubai daugiau laiko)?



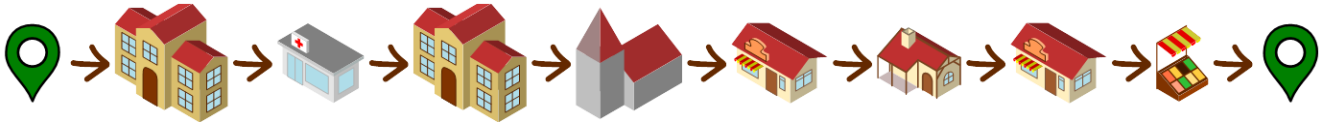
Koks būtų greičiausias kelias, kurį Ema galėtų pasirinkti norėdama atlikti visas tris užduotis?

Paaiškinimas

Teisingas atsakymas: 36 minutės.



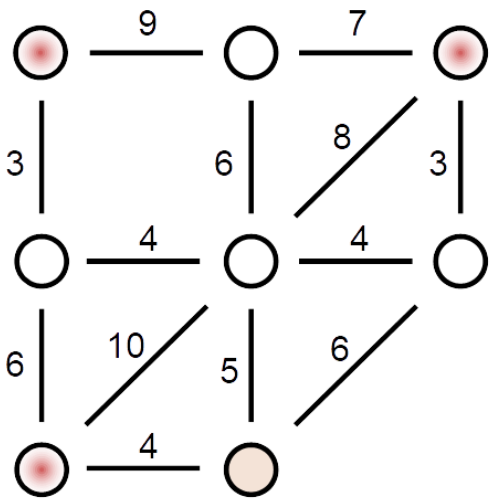
Teisingo atsakymo maršrutas:

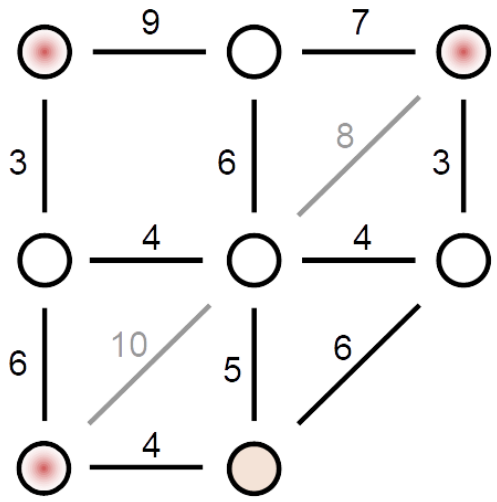
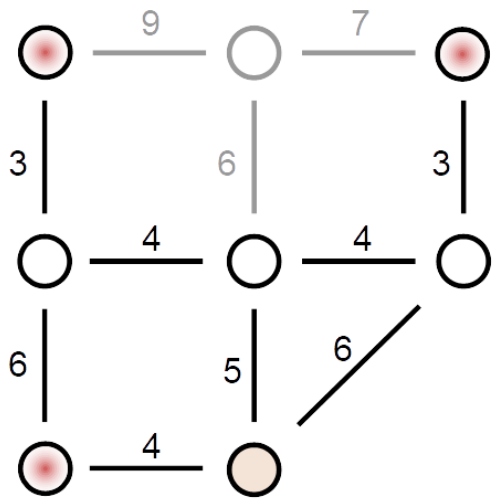
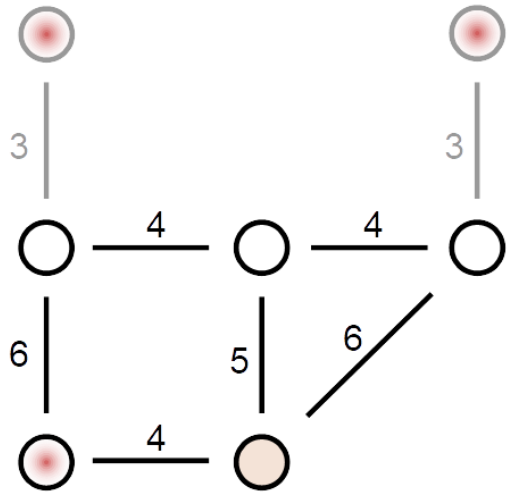


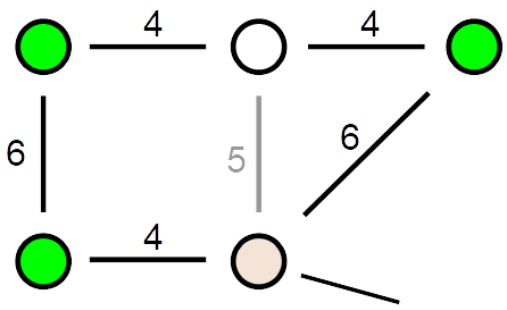
Emos namai → Mokykla → Vaistinė → Mokykla → Bažnyčia → Kepykla → Kalvė → Kepykla → Turgus → Emos namai

Skaičiuojame laiką: 6 + 3 + 3 + 4 + 4 + 3 + 3 + 6 + 4 = 36 minutės.

Ema taip pat gali eiti priešinga kryptimi ir užtruks tiek pat laiko.

Įrodymui naudojame supaprastintą schemos versiją. Apskritimai (toliau vadinami viršūnėmis) žymi skirtingus kaimo objektus, o atkarpos (toliau vadinamos briaunomis) – kelius tarp jų. Kaip ir schemoje, šalia kiekvienos briaunos esantis skaičius rodo laiką (minutėmis), per kurį galima nueiti šį kelią.	

Pilkos spalvos briaunas ignoruojame, nes yra greitesnių kelių per tarpinę viršūnę.	
Taip pat galime nepaisyti viršūnės „Parkas“. Ema neturi užduoties, kurią reikia atlikti parke, o kiekvienam maršrutui, kuris eina per parką, yra greitesnė alternatyva.	
Ema turi aplankyti šiame paveikslėlyje viršuje esančias dvi viršūnes (kalvę ir vaistinę). Tai visada vyks per žemiau esančias viršūnes ir abiem atvejais užtruks 3 minutes pirmyn ir 3 minutes atgal. Pašaliname šias viršūnes ir nepamirštame, kad prie galutinio rezultato reikės pridėti 12 minučių.	

Lieka tik toks paveikslėlis. Turime pradėti ir baigti nuo apatinės viršūnės dešinėje (su rodykle) ir aplankyti visas tris spalvotas viršūnes. Trumpiausias maršrutas eina per visas penkias viršūnes, ignoruojant pilkai nuspalvintą briauną. Šis maršrutas trunka 4 + 6 + 4 + 4 + 6 = 24 minutes. Pridėjus 12 minučių, gautų ankstesniame etape, gauname bendrą laiką, lygų 36 minutėms.	

Tai informatika!

Supaprastintas paveikslas, kurį naudojome trumpiausiam maršrutui rasti, vadinamas grafu. Daugelį informatikos uždavinių galima redukuoti į grafus. Čia ieškomas trumpiausias maršrutas, einantis per tam tikras grafo viršūnes. Kompiuteriai naudojami trumpiausio kelio uždaviniams su daug daugiau viršūnių ir briaunų. Pavyzdžiui, sprendžiant transporto ir paskirstymo uždavinius, kai reikia apskaičiuoti trumpiausią maršrutą, kuriuo važiuoja šiukšliavežė, paštininkas ar pienovežis.

Informatinis mąstymas

Grafo naudojimas šiai užduočiai modeliuoti yra tam tikra abstrakcija: ignoruojame įvairių objektų pavadinimus ir kelių tarp jų formas, bet sutelkiame dėmesį į tai, kaip objektai yra susiję, koks tarp jų atstumas (išreikštas minutėmis). Ši abstrakcija padeda išspręsti užduotį. Pati kaimo schema (žemėlapis) jau yra abstrakcija. Žinojimas, kaip interpretuoti tokią abstrakciją, taip pat yra informatinio mąstymo dalis.

37. Zeroboto dilema

Žaisdamas Zerobotas naudoja baterijas, įkrautas pavojinga medžiaga. Tai ir yra Zeroboto dilema: jam reikia šios medžiagos, kad galėtų judėti, tačiau jis privalo visą žaidime esančią pavojingą medžiagą sunaudoti, kad išgelbėtų pasaulį.

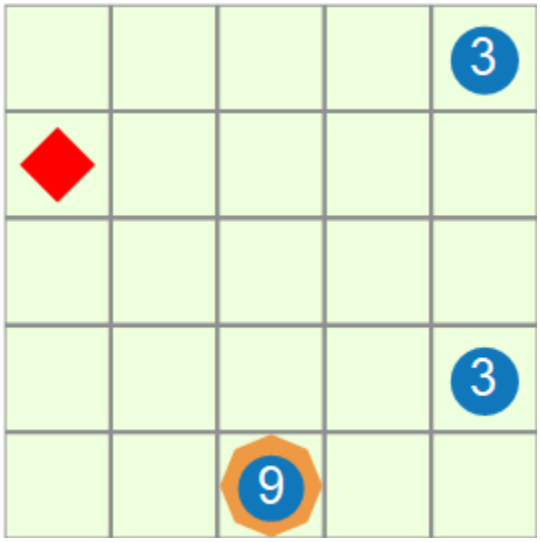
Kiekvieną kartą, Zerobotui pereinant iš vieno langelio į kitą, baterijos įkrova sumažėja vienetu.

Kai kuriuose langeliuose yra po vieną atsarginę bateriją  . Atvykęs į tokį langelį, Zerobotas būtinai privalo pasikeisti savo bateriją atsargine. Tik tada jis gali judėti toliau.

Zerobotas turi pasiekti savo namus  , kai visos žaidimo baterijos yra išsikrovusios iki 0.

Pradinis Zeroboto langelis ir jo baterijos įkrovos lygis pavaizduotas taip:  .

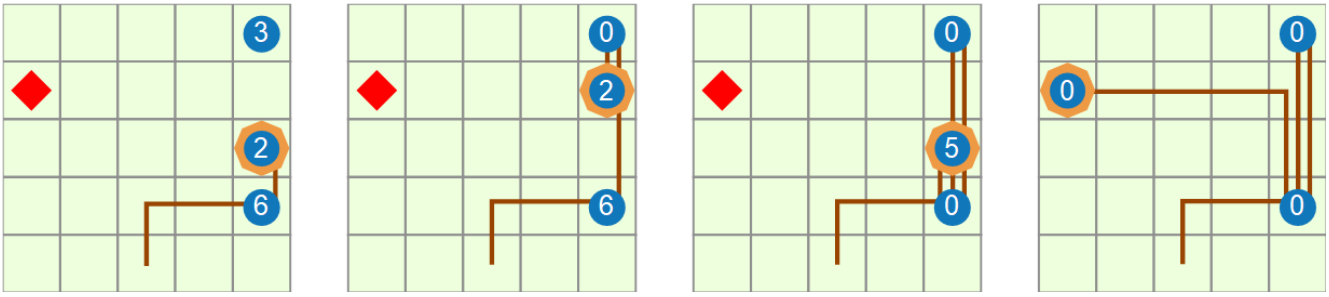
Kaip Zerobotas turi judėti, kad išgelbėtų pasaulį?



Spusteldami atsarginių baterijų langelius ir paskutinį namų langelį nurodysite Zerobotui, kur jam eiti. Zerobotas visada atliks mažiausią ėjimų skaičių, kad pasiektų nurodytą langelį.

Paaiškinimas

Zerobotas aplanko 15 langelių ir paskutinis yra namų langelis, o visos baterijos išsikrovusios iki 0:



Zerobotas turi aplankyti 15 langelių, nes bendra visų baterijų įkrova yra 9 + 3 + 3 = 15.

Tai vienintelis galimas langelių skaičius, nes kitaip arba liks nepanaudota įkrova, arba turimos įkrovos neužteks pasiekti namus.

Pavyzdžiui, jei Zerobotas pirmiausia aplankytų viršutiniame dešiniajame kampe esančią bateriją, jis turėtų pereiti per 17 langelių, kad pasiektų namus, o tai neįmanoma.

Tai informatika!

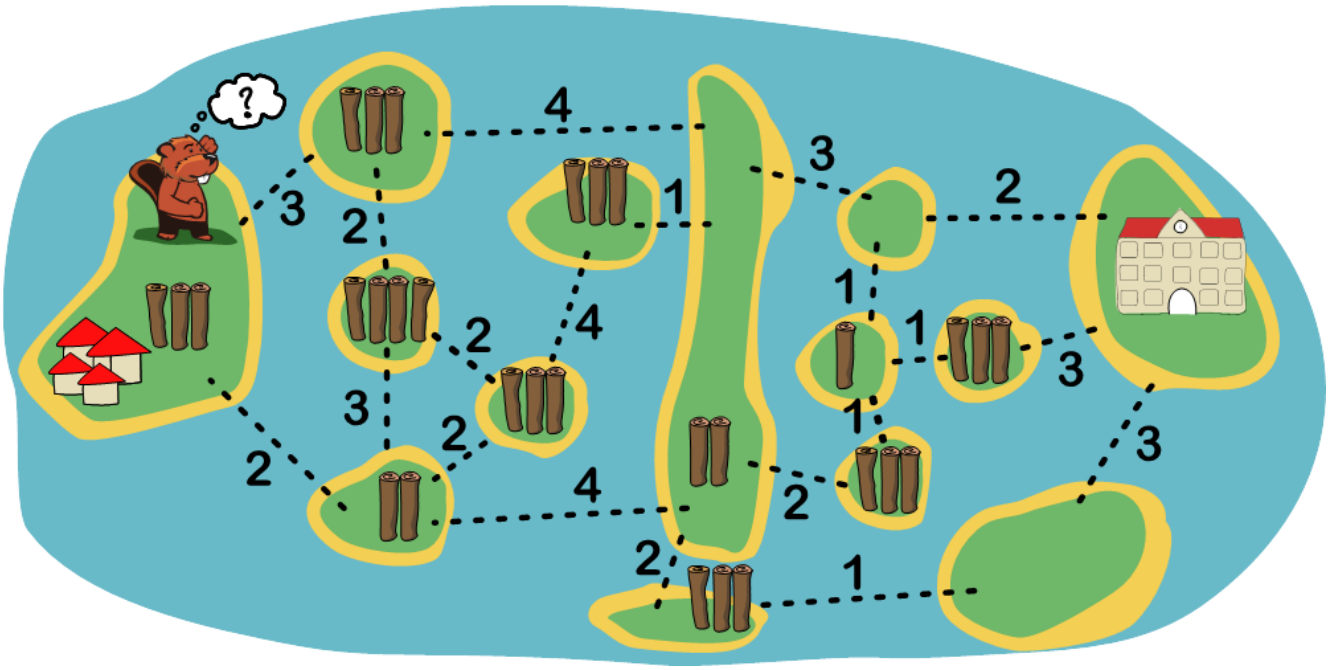
Planuojant autonominių mobiliųjų robotų (kaip ir autonominių automobilių) veiklą labai svarbu atsižvelgti į jų baterijos įkrovą. Viena vertus, robotas turi laiku įkrauti bateriją, kad ji visiškai neišsikrautų ir netaptų nefunkcionali. Kita vertus, jei robotas įkraunamas per dažnai, sumažėja jo našumas. Norint atsikratyti šios dilemos (kuri skiriasi nuo šioje užduotyje pateiktos dilemos), reikia sukurti strategiją, kaip naudoti įkroviklius arba pakaitines baterijas taip, kad būtų užtikrintas saugus mobiliųjų robotų veikimas ir kartu maksimaliai padidintas jų našumas.

Mokslininkai nustatė, jog mobiliųjų robotų įkrovimo stotys turi būti išdėstytos taip, kad įkrovos stokojantis robotas galėtų sėkmingai pasiekti vieną iš įkrovimo stočių iš bet kurios vietos be kliūčių, kai baterijos įkrovos likutis pasiekia nustatytą ribą.

38. Tilto statyba

Bronius stato tiltus. Naujas jo projektas – pastatyti tiltus, kad Broniaus kaimo  mokiniai galėtų nukeliauti į naująją mokyklą  .

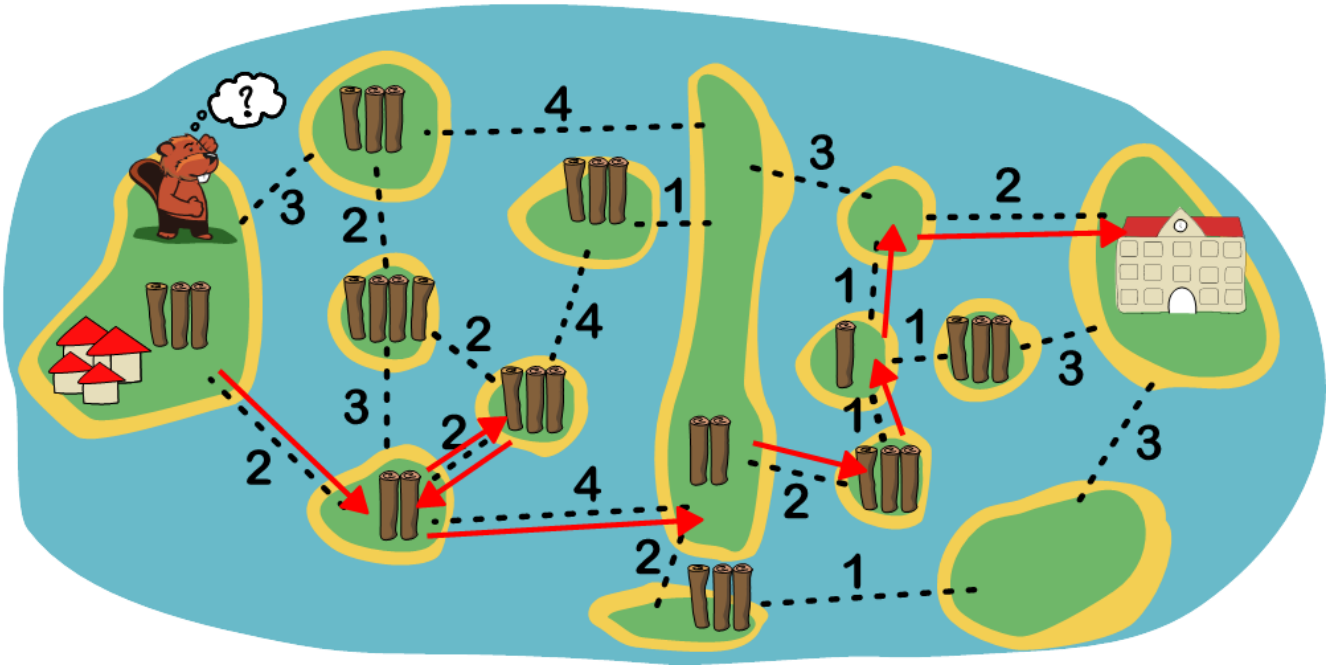
- Bronius nemoka plaukti, todėl pirmiausia jis turi pastatyti tiltą, kuriuo būtų galima pereiti į kitą salą tiek kartų, kiek jam reikės.
- Kiekvieno tilto statybai reikia tam tikro skaičiaus rąstų. Savo kaime Bronius turi tik 3 rąstus. Bronius gali keliauti jo pastatytais tiltais iš vienos salos į kitą ir pasiimti daugiau rąstų. Paveiksle pavaizduota, kiek rąstų reikia tiltui pastatyti tarp salų ir kiek rąstų yra kiekvienoje saloje.
- Kiekvienas rąstas gali būti naudojamas tik vieną kartą, tačiau nebūtina panaudoti visų turimų rąstų.

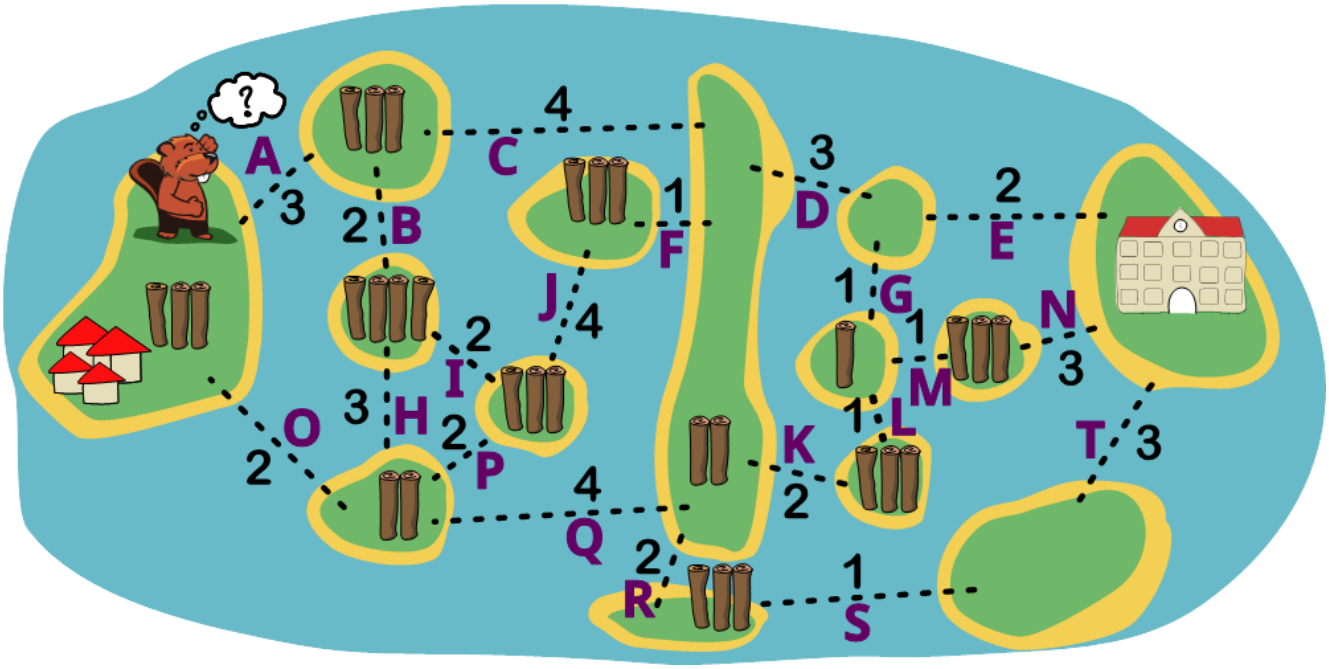


Kuriuos tiltus turėtų pastatyti Bronius, kad sunaudotų mažiausią rąstų skaičių ir mokiniai galėtų nukeliauti iš kaimo į mokyklą? Kiek rąstų jam užteks?

Paaiškinimas

Teisingas atsakymas: 14 rąstų.





Teisingas kelias užrašyti bendrą surinktų ir panaudotų rąstų skaičių kiekviename žingsnyje būtų toks (kiekviename žingsnyje surinkti rąstai >= panaudoti rąstai):

(3, 2) (5, 4) (8, 4) (8, 8) (10, 10) (13, 11) (14, 12) (14, 14).

Įrodysime, kad tai yra mažiausiais įmanomas rąstų kiekis. Surasime visus kelius, jungiančius kaimą su mokykla, kuriuose yra <= 14 panaudotų rąstų. Svarbu tai, kad visi keliai turi eiti per didžiąją salą tvenkinio viduryje, todėl uždavinį galima padalyti į dvi dalis. Keliui iš kaimo į

didžiąją salą reikia ne mažiau kaip 6 rąstų, o keliui iš didžiosios salos į mokyklą reikia ne mažiau kaip 5 rąstų (šiuos kelius vadiname apatiniais rėžiais).

Pirmiausia išsprendžiame kairiąją dalį naudodamiesi dešiniuoju apatiniu rėžiu. Tai reiškia, kad ieškome visų įmanomų kelių, kurie sunaudoja <= 9 rąstus (nes bet koks kairės dalies sprendimas >= 10 iš viso sunaudotų bent 15 rąstų).

Kairėje dalyje turime šiuos kelius, kurie jungiasi su didžiąja sala ir naudoja <= 9 rąstus:

{A, C} = (6, 7), {A, B, C} = (10, 9), {O, Q} = (5, 6), {O, H, Q} = (9, 9), {O, P, Q} = (8, 8), {O, P, J, F} = (11, 9)

Pašalinus netinkamus sprendinius, kuriuose surinktų rąstų mažiau negu panaudotų rąstų, lieka:

{A, B, C} = (10, 9), {O, H, Q} = (9, 9), {O, P, Q} = (8, 8), {O, P, J, F} = (11, 9)

Tuomet dešinę dalį sprendžiame naudodami apatinės kairiosios dalies 8 panaudotų rąstų rėžį, ir ieškome visų kelių, kuriems reikia <= 6 rąstų:

{D,E}=(2,5),{D,G,E}=(3,6),{K,L,G,E}=(6,6),{R,S,T}=(5,6)

Sujungiame kairiosios ir dešiniosios dalių sprendinius, kad rastume visas įmanomas poras, kuriose būtų <= 14 panaudotų rąstų, ir randame tik

{O,P,Q} = (8,8) + {K,L,G,E} = (6,6) → (14, 14),

nes visos kitos poros turi daugiau paimtų rąstų nei panaudotų.

Tai informatika!

Žemės plotus (salas) ir galimus statyti tiltus galima pavaizduoti grafo viršūnėmis ir briaunomis. Kiekvienoje saloje surinktų rąstų ir panaudotų kiekvienam tiltui statyti rąstų skaičių galima priskirti atitinkamoms viršūnėms ir briaunoms. Broniaus kelią galima pavaizduoti kaip briaunų sąrašą tiltų pastatymo tvarka.

Sprendžiant šį uždavinį galima naudoti kelio paieškos algoritmus su išteklių apribojimais. Išsami visų tinkamų kelių paieška būtų rezultatyvi, tačiau užtruktų eksponentinį laiką grafo dydžio atžvilgiu.

Taip pat šią užduotį galima suformuluoti kaip „skaldyk ir valdyk“ metodo paiešką, kaip tai daroma sprendimo paaiškinime. Taikant šį metodą naudojami apribojimų optimizavimo metodai – kelio nuoseklumo ir šakų bei ribų (angl. *branch-and-bound*) metodas. Šiais metodais grindžiami sudėtingiausių realaus pasaulio išteklių paskirstymo uždavinių, galvosūkių (pvz., sudoku) ir dirbtiniu intelektu paremtų žaidimų sprendimai.

Informatinis mąstymas

Abstrahavimas (salos dydis ar forma neturi reikšmės) – todėl tinka vaizduoti grafu.

Dekomponavimas – uždavinį galima padalyti į du panašaus pobūdžio uždavinius (kairioji ir dešinioji grafo dalys).

Algoritminis mąstymas – kelio paieška – yra polinominio laiko nuoseklusis algoritmas ir reikalauja kaupiamųjų išteklių apribojimų. Taikant išsamios paieškos metodą sudėtingumas būtų eksponentinis (atsižvelgiant į tai, kad mūsų kelias yra medis, o grafe gali egzistuoti iki n^n – 2 medžių) ir neefektyvus tokio dydžio uždaviniui spręsti.

39. Postfiksinis užrašas

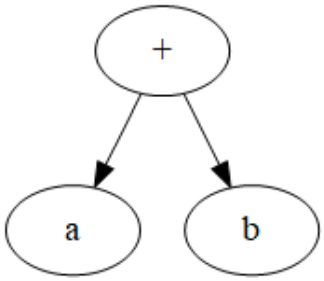
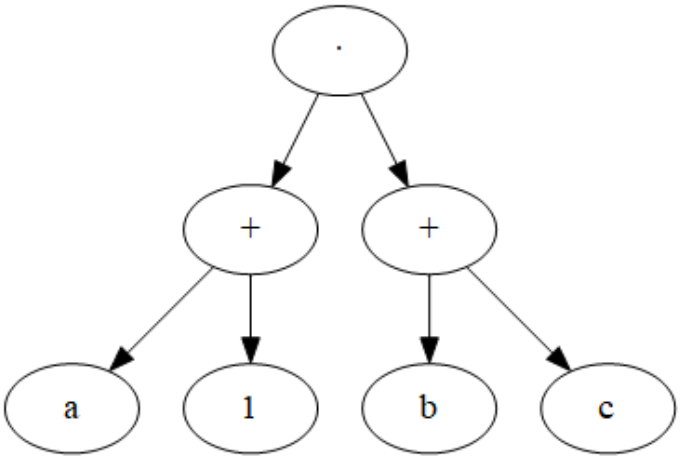
Matematinį reiškinį sudaro:

- operatoriai, pavyzdžiui, „+“, „–“, „·“ arba „:“
- operandai, kurie gali būti skaičiai, pavyzdžiui, 1, 2, ..., arba reiškiniai, pavyzdžiui, (1 + 2).

Matematinių reiškinių struktūra vaizduojama operatorių ir operandų diagrama. Tokia diagrama dar vadinama sintaksės medžiu.

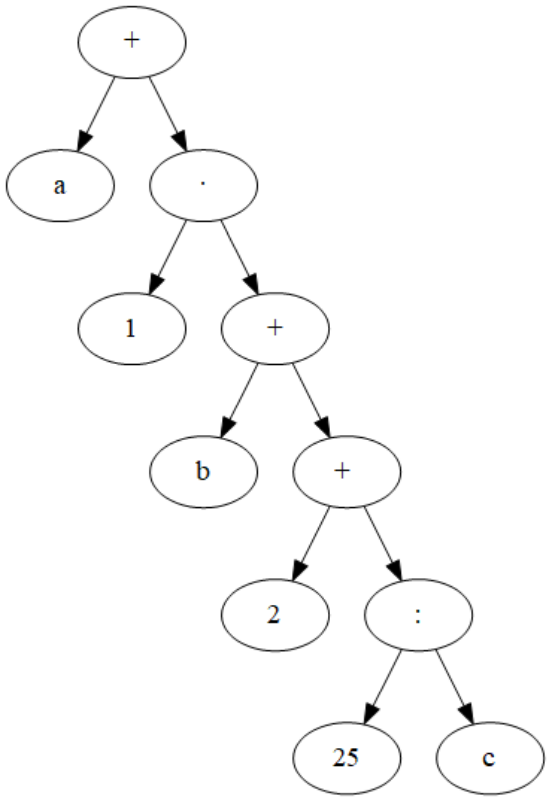
Iš sintaksės medžio galima sudaryti matematinio reiškinio postfiksinį (angl. *postfix*) užrašą. Šiuo atveju pirmiausia užrašomi operandai, po jų – operatoriai.

Dviejų matematinių reiškinių perrašymo eigą ir rezultatus galite matyti atitinkamuose sintaksės medžiuose ir postfiksiniuose užrašuose:

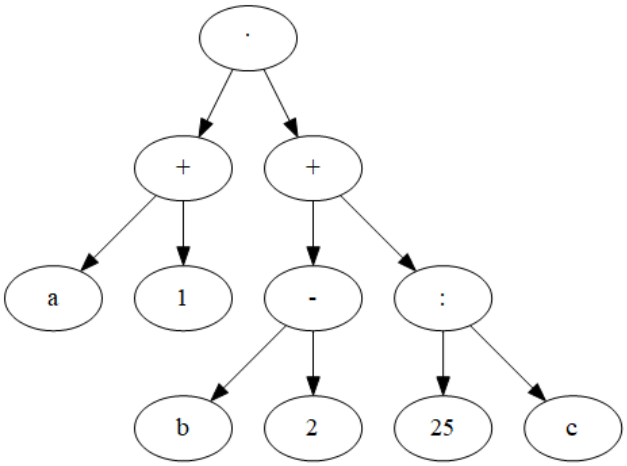
Matematinis reiškinys	a + b	(a + 1) · (b + c)
Sintaksės medis		
Postfiksinis užrašas	a b +	a 1 + b c + ·

Kuris sintaksės medis vaizduoja tokį postfiksinį užrašą: a 1 + b 2 + · 25 c : +

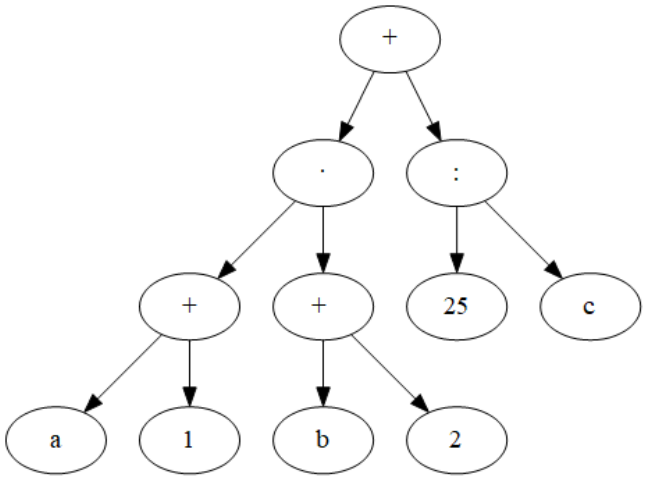
A



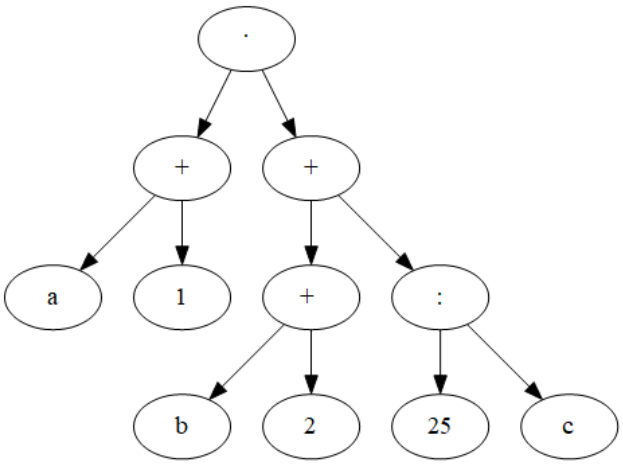
B



C



D



Paaiškinimas

Teisingas atsakymas: C.

Jei postfiksinį užrašą vėl paversite sintaksės medžiu, svarbiausias yra pats paskutinis simbolis – operatorius „+“. Jis turi būti medžio viršūnėje, nes yra paskutinis užrašo simbolis. Todėl galimi tik atsakymai A arba C.

Operatorius „+“ turi du operandus: vieną iš kairės, kitą iš dešinės. Dešiniojo operando vienintelis galimas simbolis yra „:“, nes tai yra priešpaskutinis postfiksinio užrašo simbolis. Taigi teisingas gali būti tik C atsakymas.

Tačiau ar jis tikrai teisingas? Galima postfiksinį užrašą žingsnis po žingsnio konvertuoti toliau. Tačiau paprasčiau sintaksės medį žingsnis po žingsnio konvertuoti į postfiksinį užrašą.
Kiekvieną iš trijų žemiausių medžio šakų sudaro 3 elementai, atitinkamai: a 1 +, b 2 + ir 25 c :
Kylant iš kairės aukščiau operacija „–“ sujungia du operandus: a 1 + b 2 + -. Iš dešinės lieka 25 c :
Atsakymo postfiksinis užrašas yra a 1 + b 2 + 25 c : +
Taigi C atsakymas yra teisingas.

Tai informatika!

Norint išspręsti šią užduotį, sudėtingą matematinį reiškinį reikia išskaidyti į mažesnius matematinius reiškinius. Užduoties sąlygoje paaiškinamas išskaidymo būdas. Be to, procesas turi būti analizuojamas taip, kad jo veikimą būtų galima suprasti ir atvirkštine tvarka. Iš esmės ta pati informacija konvertuojama iš vieno vaizdavimo būdo į kitą.

Postfiksinis užrašas (angl. *postfix notation*, *reverse Polish notation*, RPN) – tai formatas, naudojamas kompiliatoriuose, kurie yra tam tikros rūšies transliatoriai, konvertuojantys žmonėms suprantamas programavimo kalbas į kompiuteriams suprantamą mašininį kodą. Šis užrašymo būdas naudojamas bet kurioje programoje esantiems reiškiniams įvertinti.

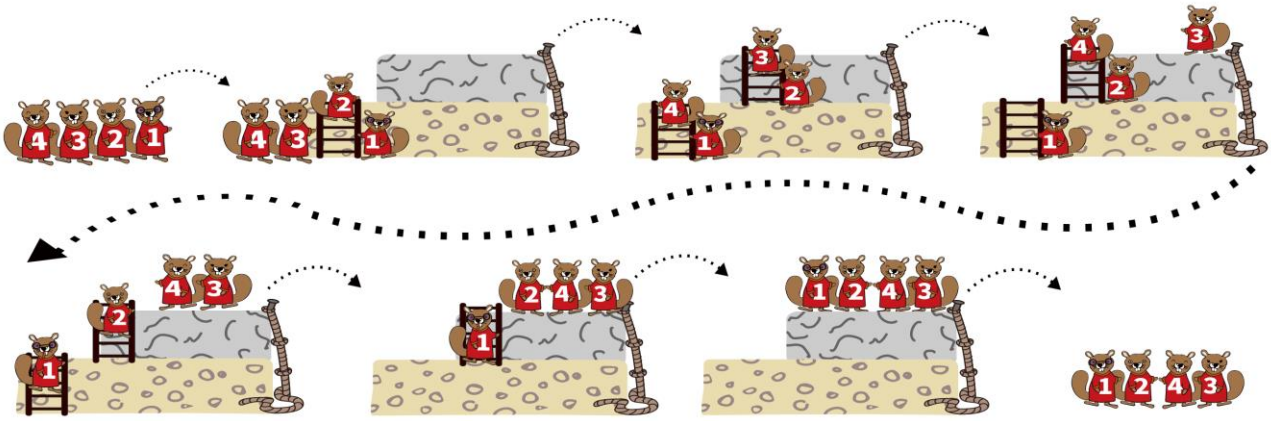
Informatinis mąstymas

Šią užduotį galima spręsti skaidant reiškinį į mažesnes dalis ir naudojant pavyzdžius, kurie veikia kaip modeliai kuriant su reiškiniu susijusį medį.

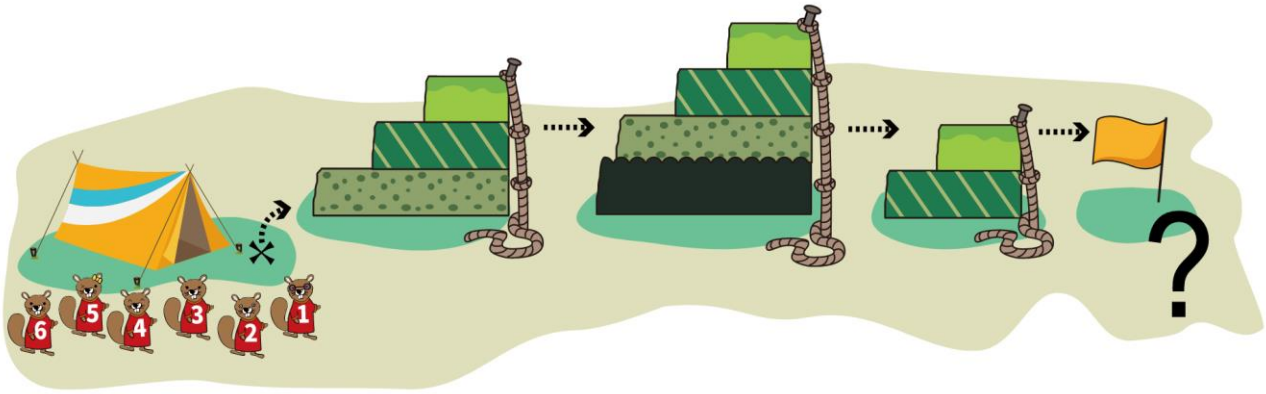
40. Į kalnus

Ruošdamiesi žygiui, bebrai treniruojasi lipti stačiomis uolomis. Bebrai rikiuojasi ir eina eilute vienas paskui kitą. Kiekvienas bebras nešasi kopėčias. Kai prieina stačią uolą, pirmas eilėje esantis bebras pastato kopėčias, kad galėtų lipti kiti. Kiekvienas pastatęs kopėčias bebras laukia vietoje kol visi, kas nelaiko pakeltų kopėčių, užlipa į uolos viršūnę. Kiekvienas kopėčias pastatęs bebras savo kopėčiomis lipa į kitą lygį. Taip jie visi lipa tol, kol pasiekia uolos viršūnę. Kiekvienas bebras jungiasi į eilės pabaigą. Baigę kopti į uolą bebrai nusileidžia žemyn naudodamiesi virve ir laikydamiesi savo eilės tvarkos.

Paveiksle parodyta, kaip jie kyla.



Šeši bebrai ruošiasi į žygį vietovėje, pavaizduotoje paveiksle. Jų tikslas – vėliava ir klaustuku pažymėta vieta. Nutempkite ten bebrus jų atvykimo į tikslą eilės tvarka.

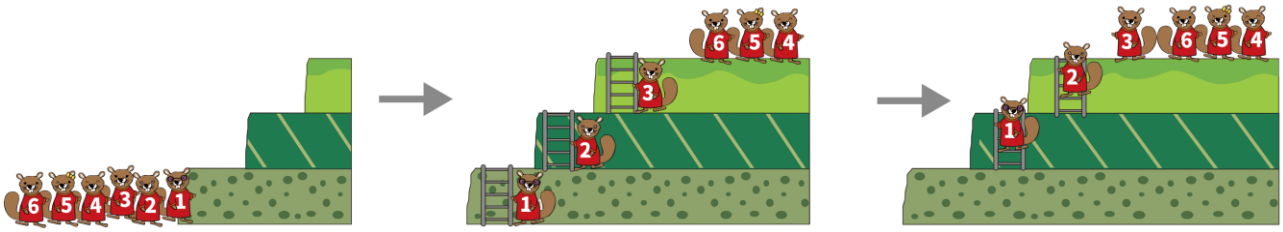


Paaiškinimas

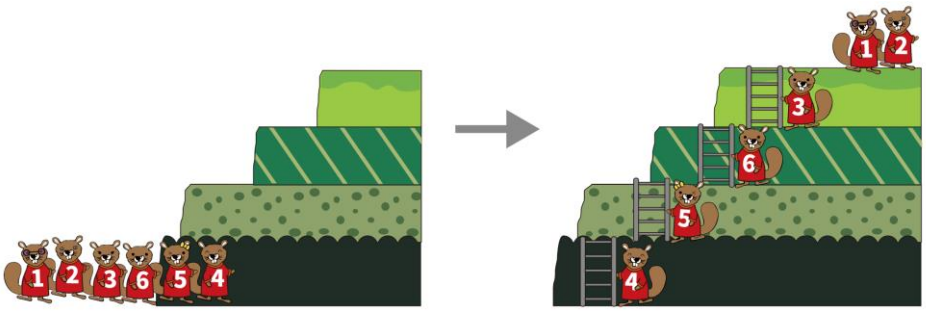
Teisinga bebrų eilės tvarka: 365412.

Bebrai pradeda žygį eilės tvarka: 123456. Taip jie atvyksta prie pirmosios 3-jų lygių uolos. Bebras, kurio numeris 1, laiko savo kopėčias 1-ame lygyje, 2-as bebras laiko savo kopėčias 2-ame lygyje, 3-as bebras savo kopėčias laiko 3-ame lygyje, o 4, 5 ir 6 bebrai šia tvarka kyla į viršūnę.

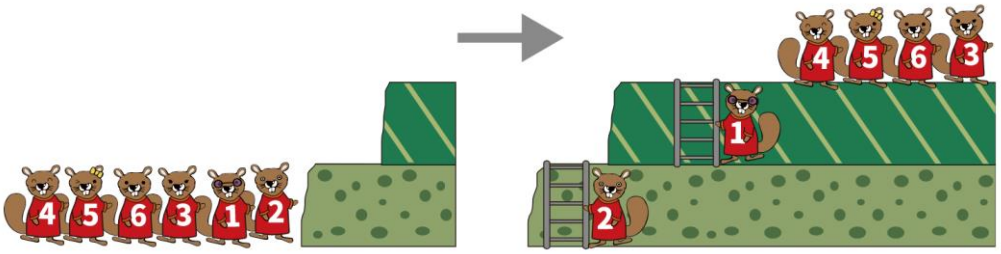
Pastariesiems bebrams pasiekus viršūnę, 3-as, 2-as ir 1-as bebrai lipa į viršų ta pačia tvarka, kol visi jie atsiduria pirmosios uolos viršūnėje. Po šio kopimo bebrų eilės tvarka yra 456321.



Tokia tvarka bebrai atvyksta prie antros 4-ių lygių uolos. 4-as bebras, būdamas pirmas eilėje, laiko kopėčias 1-ame lygyje, 5-as bebras – 2-ame lygyje, 6-as bebras – 3-ame lygyje, 3-as bebras – 4-ame lygyje, todėl 2-as ir 1-as bebrai gali šitokia tvarka užlipti į antrosios uolos viršūnę. 2-am ir 1-am bebrams pasiekus viršūnę, 3-as, 6-as, 5-as ir 4-as bebrai ta pačia tvarka lipa aukštyn, kol visi jie atsiduria ant antrosios uolos viršūnės. Dabar bebrų eilės tvarka yra tokia: 213654.



Bebrai prie paskutiniosios dviejų lygių uolos atvyksta tokia tvarka: 213654. Tad 2-as bebras yra pirmas eilėje, jis laiko kopėčias 1-ame lygyje, 1-as bebras – 2-ame lygyje, o kiti bebrai lipa į viršų eilės tvarka: 3, 6, 5 ir 4. Šiems bebrams pasiekus viršūnę, ta pačia tvarka į viršų lipa 1-as ir 2-as bebrai.



Įkopus į paskutinę uolą bebrų eilės tvarka yra 365412. Tai reiškia, kad bebrai atvyksta į pažymėtą tikslą tokiu eiliškumu: 365412.

Tai informatika!

Tai, kaip šiame uždavinyje bebrai lipa į uolas, atspindi dėklo ir eilės duomenų struktūras informatikoje.

Dėklas ir eilė yra dviejų tipų duomenų struktūros. Duomenys dėkle laikomi panašiai, kaip ir objektai dėkle realiame gyvenime. Dėklo duomenų struktūra veikia pagal principą: „paskutinis į maišą, pirmas iš maišo“, t. y., duomenys į ją dedami iš eilės tokia tvarka, kokia pateikiami, tad skaityti arba pašalinti galima tik paskiausiai įdėtą duomenį.

Į eilės duomenų struktūrą nauji duomenys dedami jų gavimo tvarka (į eilės pabaigą), o išimamas pirmasis į eilę įdėtas duomuo (esantis eilės pradžioje). Kitaip tariant, jei duomenis (objektus) iš eilės išimame vieną po kito, pirmas į eilę padėtas duomenų objektas bus išimtas pirmas.

Informatinis mąstymas

Mokiniai gali taikyti šablonų (struktūrų) atpažinimą ir išsiaiškinti, kokia tvarka bebrai pakelia kopėčias ir kokia tvarka lipa į uolas. Mokiniai taiko algoritminį mąstymą, kad galėtų vadovautis komandomis ir susekti bebrų eilės tvarką.

41. Elektroninė spyna

Bebras Benas savo name įrengė elektroninę kodinę spyną. Spyna atrakinama teisinga tvarka paspaudus keletą skirtingų mygtukų, ant kurių užrašyti skaitmenys.

Pradžioje kodas buvo toks:

0	2	4	3	1

Benas aprašė kodą, bet šiek tiek jį pakeitė: *n* >> *c* reiškia, kad į kairę nuo skaitmens *c* yra lygiai *n* skaitmenų, didesnių už *c*. Pavyzdžiui, užrašas 1 >> 3 reiškia, kad į kairę nuo skaitmens 3 yra vienas skaitmuo, kuris yra didesnis už 3 (tai 4). Taigi dabartinį kodą Benas aprašė taip:

0 >> 0; 3 >> 1; 0 >> 2; 1 >> 3; 0 >> 4

Penkių skaitmenų kodas Benui atrodo labai nesaugus. Todėl galvoja naują kodą, kuriame būtų skaitmenys nuo 0 iki 7. Naująjį kodą Benas aprašo taip:

0>>3, 1>>2, 2>>4, 3>>4, 4>>1, 5>>1, 6>>1, 7>>0.

Koks yra naujasis kodas?

A) 7 0 1 4 5 6 2 3

B) 7 4 1 0 5 6 2 3

C) 7 4 1 6 5 3 2 0

D) 7 4 1 0 5 3 2 6

Paaiškinimas

Teisingas atsakymas: 7 4 1 0 5 6 2 3.

Norėdami rasti kodą, atidžiau panagrinėkime Beno užrašus, po vieną skaitmenis nuo 0 iki 7.

3 >> 0: į kairę nuo 0 yra lygiai 3 skaitmenys, didesni už 0. Taigi skaitmuo 0 turi būti ketvirtoje kodo pozicijoje.

2 >> 1: į kairę nuo 1 yra lygiai 2 skaitmenys, didesni už 1. Todėl skaitmuo 1 turi būti trečioje kodo pozicijoje.

4 >> 2: į kairę nuo 2 yra lygiai 4 skaitmenys, didesni už 2. Kadangi mažesni skaitmenys 1 ir 0 jau yra trečioje ir ketvirtoje pozicijose, 4 didesni skaitmenys turi būti pirmoje, antroje, penktoje ir šeštoje pozicijose. Todėl skaitmuo 2 turi būti septintoje kodo pozicijoje.

4 >> 3: į kairę nuo 3 yra lygiai 4 skaitmenys, didesni už 3, todėl skaitmuo 3 turi būti aštuntoje, paskutinėje kodo pozicijoje.

1 >> 4: į kairę nuo 4 yra lygiai 1 skaitmuo, didesnis už 4, todėl skaitmuo 4 turi būti antras iš likusių skaitmenų, tai yra antrasis kodo skaitmuo.

1 >> 5: į kairę nuo 5 yra lygiai 1 skaitmuo, didesnis už 5, todėl skaitmuo 5 turi būti antras iš likusių skaitmenų, tai yra penktasis kodo skaitmuo.

1 >> 6: į kairę nuo 6 yra lygiai 1 skaitmuo, kuris yra didesnis už 6, todėl skaitmuo 6 turi būti antras iš likusių skaitmenų, tai yra šeštasis kodo skaitmuo.

0 >> 7: nėra jokio skaitmens, didesnio už 7, todėl skaitmuo 7 turi būti paskutinis laisvas skaitmuo, tai yra pirmasis kodo skaitmuo.

Tai informatika!

Visų įmanomų derinių generavimas ir vienintelio teisingo derinio radimas priklauso kombinatorikos, informacijos kodavimo ir kriptografijos klasikinių algoritmų sritims. Ši užduotis yra puikus įvadas į kombinatorikos algoritmus pradedantiesiems.

Kombinatorikoje į kairę nuo tam tikro skaitmens esančių didesnių skaitmenų kiekis vadinamas inversija. Manome, kad daug naudingiau pasiūlyti mokiniams pirmiausia išspręsti tokio tipo uždavinį ir tik tada pateikti inversijos apibrėžtį.

Nuoseklus reikšmių priskyrimo taikymas sprendiniui rasti yra lanko nuoseklumo algoritmo tenkinant apribojimus pavyzdys. Šis algoritmas yra daugelio realaus pasaulio išteklių paskirstymo uždavinių ir populiarių galvosūkių, pavyzdžiui, sudoku, sprendimų pagrindas.

Be šių metodų išspręsti uždavinį gali būti daug sudėtingiau.

Informatinis mąstymas

Sudėtingas kombinatorikos uždavinys sprendžiamas taikant paprastą tiesinį algoritmą. Tai skatina mokinių informatinį mąstymą.

42. Ornamentai

Bebrų įsteigta programinės įrangos įmonė gamina ornamentuotus atvirukus ir dovanų popierių.

Ornamentai kuriami pagal toliau pateiktas instrukcijas (objekto kūrimas reiškia jo sukūrimą kompiuterio atmintyje, o ne nupiešimą ekrane):

Sukurti atsitiktinės spalvos skritulį ir pavadinti jį C.

Tolesnė instrukcijų seka kartojama atsitiktinį kartų skaičių:

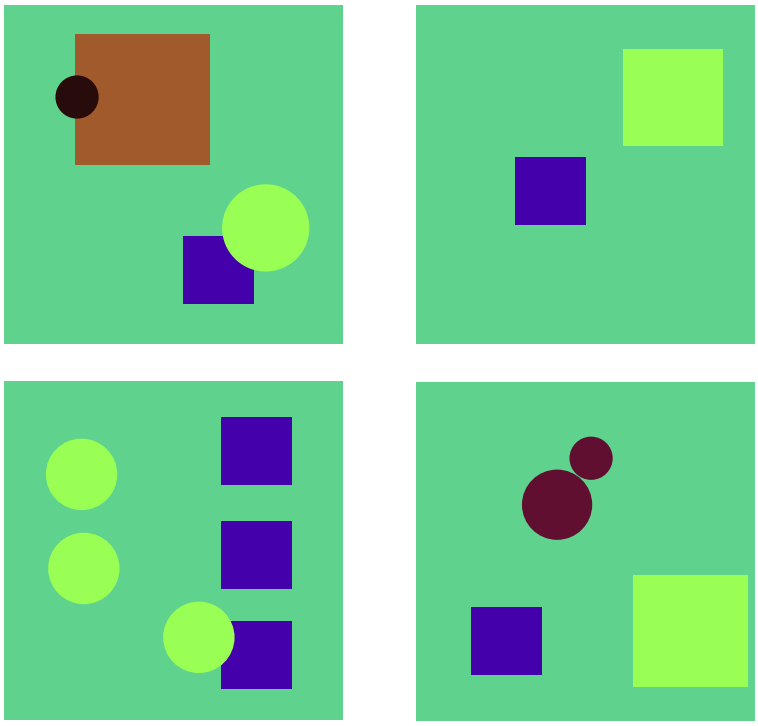
Sukurti atsitiktinio dydžio ir atsitiktinės spalvos kvadratą ir pavadinti jį S.

Pasirinkti C dydį, vieną iš dviejų variantų – didelis arba mažas.

Spausdinti C kurioje nors ornamento vietoje.

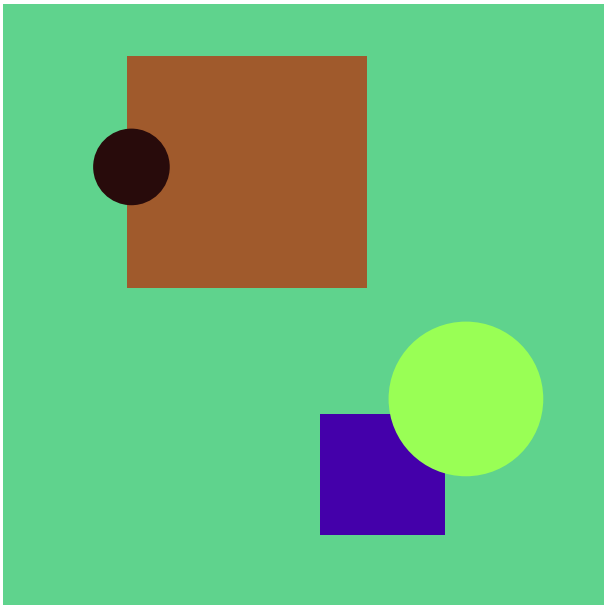
Spausdinti S kurioje nors ornamento vietoje.

Kurio iš pateiktų ornamentų bebrų įmonė negalėtų sukurti?



Paaiškinimas

Teisingas atsakymas:



Tai informatika!

Šiame ornamente yra du skirtingo dydžio ir skirtingų spalvų skrituliai. Pateiktame instrukcijų apraše skritulys pasirenkamas vieną kartą. Tik jo dydis vėliau keičiamas.

Kiti vaizdai yra galimi algoritmo rezultatai, nors iš pirmo žvilgsnio jie nepanašūs.

Kadangi programa spausdina kvadratą kiekvienam skrituliui, skritulių turi būti tiek pat, kaip ir kvadratų. Tačiau kvadratas gali uždengti skritulį ir jį paslėpti. Taip pat turite atsižvelgti į tai, kad kvadratas arba skritulys gali būti tokios pat spalvos kaip fonas.

Skritulių ir kvadratų padėtis parenkama atsitiktinai. Tačiau įmanoma (nors ir labai mažai tikėtina), kad gali atsirasti taisyklingų ir simetriškų struktūrų.

Informatinis mąstymas

Tokiuose filmuose kaip „Įsikūnijimas“ (angl. „Avatar“, rež. Jamesas Cameronas, JAV, 2009 m.) yra daug vaizdinio turinio, kurį automatiškai sukuria kompiuteriai. Iš esmės dailininkas tiesiog pateikia taisyklių rinkinį. Visas smulkias sudėtingos dirbtinės struktūros detales kompiuteris sukuria nedeterministiniu būdu, bet vis tiek laikydamasis pateiktų taisyklių.

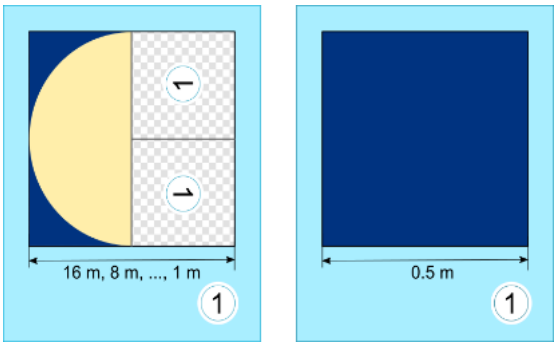
43. Rekursinis dažymas

Ina ir Benas padeda rengti parodą Informatikos mokslo muziejuje. Ant parodų salės grindų jie turi nutapyti 16 x 16 metrų dydžio paveikslą.

Jiems vadovaujantis dailininkas pateikia paveikslo tapymo instrukcijos lapų rinkinį, kuriame nurodyti paveikslo elementai, mastelis ir pasukimai. Kai kuriuose instrukcijos lapuose yra laukai su numeriais, kurie nurodo į kitus lapus.

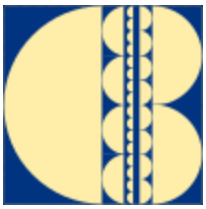
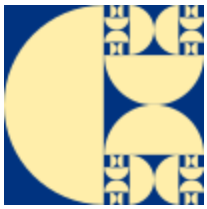
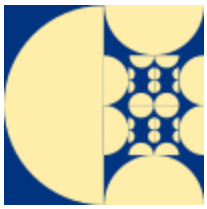
Štai pavyzdys iš ankstesnio tapybos projekto. Teisingai sudėliojus visus tris instrukcijos lapus, gaunamas bebro paveikslas:

Naujam projektui Ina ir Benas gauna štai šiuos du tapybos instrukcijos lapus:



Benas susiraukia: „Kaip tai suprasti? Kairysis lapas rodo pats į save, be to, abiejuose lapuose yra tas pats numeris!“ Ina nusijuokia: „Man jau aišku! Pirmiausia pasinaudokime tik kairiuoju lapu. Dešinysis lapas rodo, kada nustoti piešti“.

Kaip atrodys ištapytos parodų salės grindys?

A 
B 
C 
D 

Paaiškinimas

Teisingas atsakymas – B.

Panagrinėkime kairįjį instrukcijos lapą, kuriame pavaizduotas grindų dekoravimas pusapskritimiu. Grindų dešinioji pusė aprašoma naudojant du kartus tą patį lapą, bet ši pusė tapoma tik tada, jei yra mažiausiai 1 metro ilgio. Pastebėkite, kad plane nurodytų dviejų vienetų orientacija yra priešinga, todėl svarbu teisingai pasukti lapą.

Atkreipkite dėmesį, kad visi elementai turi liestis.

Tai informatika!

Save nurodančios instrukcijos vadinamos rekursinėmis. Rekursija – labai svarbus informatikos konceptas. Rekursiniai modeliai dažnai aptinkami ir gamtoje. Rekursiniai sprendimai paprastai yra trumpesni ir kompaktiškesni, tačiau kartais juos sunkiau suprasti. Siekiant išvengti begalinių ciklų rekursijoje svarbi baigties sąlyga. Ji pateikta ir šiame pavyzdyje.

44. Šifro „nulaužimas“

Prefikso šifre raidės yra koduojamos kintamo ilgio skaitmenų kodu ir joks kodas neprasideda kitu kodu. Pavyzdžiui, jei raidė A yra koduojama 12, tai raidė B gali būti koduojama 2, nes 12 neprasideda 2, bet negali būti koduojama 121. Raidė C gali būti koduojama 11, nes abu kodai – 12 ir 2 – neprasideda 11, bet C negali būti koduojama nei 21 (nes 2 jau yra raidės B kodas), nei 121 (nes kodą 12 turi A).

Tarp skaitmenų įterpkite tarpus taip, kad kiekviena seka, atskirta tarpu, būtų raidės kodas ir sudarytų žodį BEBRAS.

12112233321

Paaiškinimas

Teisingas atsakymas yra 1 21 1 22 33 321.

Pradėkime nuo pradžių. Raidės B kodas negali būti didesnis negu 2 (121, 1211, 12112,...), nes raidė B turi pasikartoti, o tokių pasikartojimų nėra koduotame pranešime. Jei raidė B būtų koduojama 12, tada raidė E turėtų būti koduojama 1, o tai neatitinka reikalavimo, nes raidės B kodas prasidėtų E kodu. Taigi raidės B kodas turi būti 1. Raidė E yra po B, todėl ji gali būti koduojama 2, 21 arba 211223332. E negali būti koduojama 2, nes tada gautume žodį BEBB. Ji negali būti 211223332, nes tada gautume žodį BEB. Taigi E kodas yra 21. Kol kas žinome, kad 1_21_1 koduoja BEB, taigi belieka padėti tarpus tarp 2233321. Kadangi raidė S yra žodžio pabaigoje, tai ji negali būti 1 arba 21, nes šiuos kodus naudoja B ir E. Taigi raidė S gali būti 321, 3321, 33321,233321 arba 2233321. S negali būti koduojama 2233321, nes tada gautume žodį BEBS. Ji taip pat negali būti 233321, nes tada liktų tik vienas skaitmuo raidėms R ir A koduoti. Kodas 33321 taip pat negalimas, nes liktų 22, o vienodi skaitmenys 2 ir 2 turėtų koduoti skirtingas raides R ir A. Jei S būtų 3321, tada raidės RA turėtų būti koduojamos 223. Tačiau R negali būti koduojama 22, o A negali būti 3, nes S prasideda 3. Taigi S kodas turi būti 321, o RA koduojama 2233. Panašiai nagrinėdami gauname, kad R kodas turi būti 22, o A kodas yra 33.

Tai informatika!

Prefikso šifras šiame uždavinyje yra kintamo ilgio kodo pavyzdys, kuriame raidės yra paverčiamos kodais, turinčiais savybę, kad nė vienas iš jų neprasideda kitos raidės kodu. Ši prefikso kodų savybė užtikrina vienareikšmį pranešimo dekodavimą, nes nėra būtinybės naudoti tarpų tarp kodų. Su kintamo ilgio kodais yra susijusi informacijos suspaudimo problema, sprendžiama siekiant kuo trumpesnio perduodamo pranešimo. Šiuo tikslu gali būti naudojamas Hufmano (David A. Huffman) kodavimo algoritmas, pasižymintis duomenų nepraradimo ir suspaudimo savybėmis.

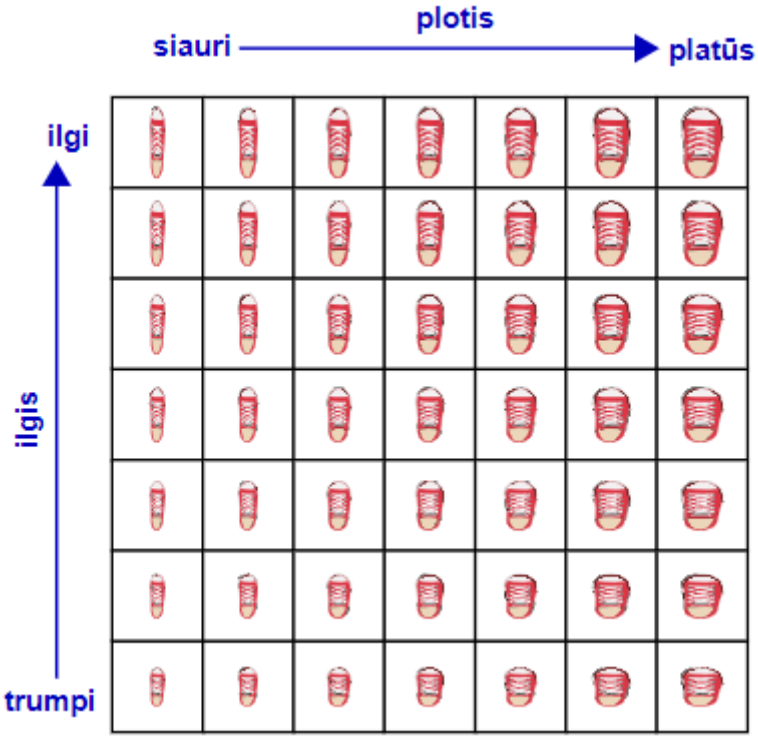
Daugiau skaitykite čia: https://en.wikipedia.org/wiki/Prefix_code

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

45. Batų pirkimas

Bebras nuėjo į parduotuvę nusipirktų batų porą. Vitrinoje jis pamatė keletą batų, išdėliotų taip, kaip parodyta paveikslėlyje.

Batai buvo išdėlioti didėjimo tvarka pagal dydį ir plotį. Siauriausi ir trumpiausi buvo padėti kairiajame apatiniame kampe, o didžiausi ir plačiausi – dešiniajame viršutiniame kampe. Visi batai buvo skirtingo dydžio ir pločio.

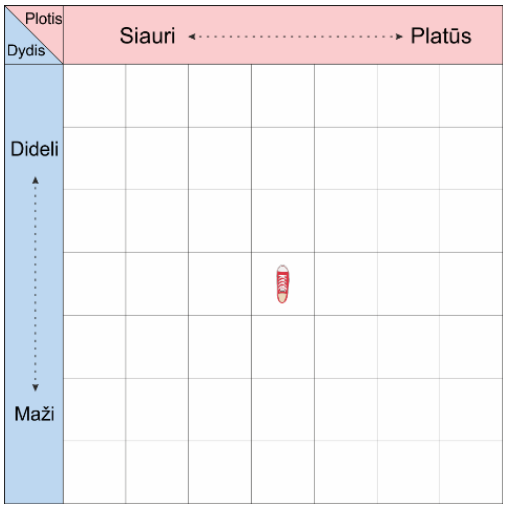


Bebras buvo užmaršus ir neprisiminė savo batų dydžio, todėl turėjo matuotis tol, kol rado tinkamus. Tinkami batai – tai tinkamo dydžio ir pločio batai. Bebras naudojosi metodu, užtikrinančiu, kad tinkamus batus blogiausiu atveju jis randa per mažiausią matavimų skaičių. Kuriuos batus jis turėtų matuotis? Spustelėkite batą, kol ilgis ir plotis taps tinkamais.

Paaiškinimas

Atsakymas: 2.

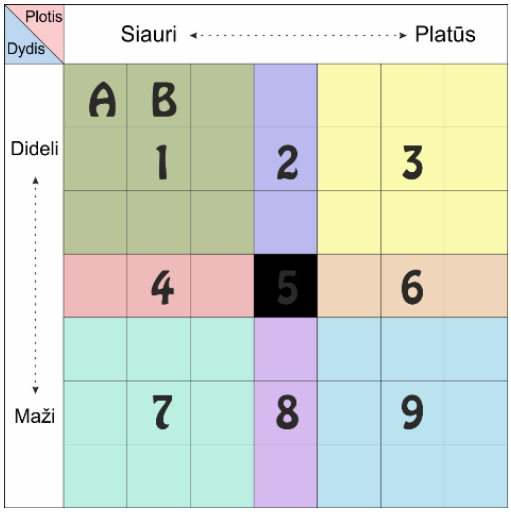
Bebrui gali pasisekti ir tinkamus batus jis rastų jau pirmojo bandymo metu. Kitu atveju, tinkančius batus jis gali atrasti antruoju bandymu.



Bebras gali pradėti matuotis nuo batų, esančių centre, kaip parodyta paveikslėlyje.

- Batas bus tinkamo dydžio, mažesnis, didesnis ir tinkamo pločio, per siauras arba per platus. Priklausomai nuo rezultato, bebras žinos, kad tinkamas batas bus vienoje iš devynių skirtingomis spalvomis pažymėtų zonų.
- Jei batas tiko, jis rado reikiamą apavą.
- Jei batas per mažas ir per platus, jis pasimatuos 1-oje zonoje esančius batus.
- Jei batas yra per mažas, tačiau tinkamo pločio, jis pasimatuos 2-oje zonoje esančius batus.
- Jei batas per didelis ir per siauras, jis matuosis 9-oje zonoje esančius batus ir taip toliau.

Sakykime, kad bebras pasimatavo per mažus ir per plačius batus. Dabar jis turėtų matuotis didesnius ir siauresnius, kurie padėti 1-oje zonoje.



Jis pasimatuoja batą, kuris padėtas 1-osios zonos centre (lentynėlė pažymėta skaičiumi 1).

- Jei batas jam tinka, jis rado tinkamą apavą.
- Jei batas per mažas ir per platus, jam tiks batai A lentynėlėje.
- Jei batas jam per mažas, bet gero pločio, jam tiks batai B lentynėlėje.

Kaip matome, bebras turi pasimatuoti daugiausiai du skirtingus batus, kad rastų tinkamus.

Jei jis pradėtų matuotis iš bet kurios kitos pozicijos, tektų pasimatuoti daugiau batų.

Tai informatika!

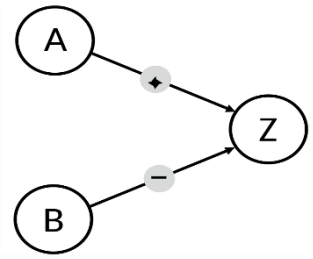
Batai vitrinoje išdėlioti didėjančio dydžio ir pločio tvarka pagal dvi ašis. Toks išdėliojimas vadinamas rikiavimu. Surikiuotiems duomenims naudojami dvejetainės paieškos algoritmai reikiamą reikšmę randa labai greitai. Dvejetainė paieška paieškos erdvę kaskart mažina per pusę tam, kad teisingą reikšmę rastų naudojant kuo mažiau bandymų. Dvejetainę paiešką galima naudoti ir žaidžiant, pavyzdžiui, bandant atspėti skaičių nuo 1 iki 100 ir panaudojant kuo mažiau bandymų.

Ši užduotis iliustruoja dvejetainės paieškos algoritmą, kai reikiamos reikšmės ieškoma dviejų matavimų erdvėje.

46. Konfliktų detektorius

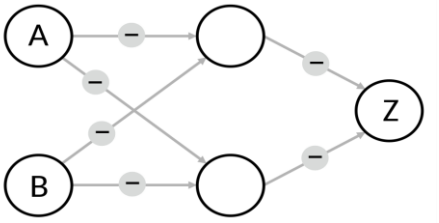
Ana ir Benas nori sukurti konfliktų detektorių, kuris parodytų, ar jų nuomonės skiriasi. Jie naudoja įtaisus, turinčius dvi būsenas: TAIP ir NE. Du tokius įtaisus galima sujungti jungtimi. Kai pradinio įtaiso būsena yra TAIP, tada šia jungtimi siunčiamas teigiamas (+) arba neigiamas (–) signalas į visus dešiniau prijungtus įtaisus. Kai pradinio įtaiso būsena yra NE, joks signalas prijungtiems įtaisams nesiunčiamas. Įtaisas, kuris gauna signalą, pereina į būseną TAIP, jei jis gauna daugiau teigiamų nei neigiamų signalų, o jei gauna daugiau neigiamų arba visai negauna signalų, jo būsena bus NE. Kiekvieną detektorių, sukurtą iš šių komponentų, sudaro du įvesties įtaisai A ir B ir išvesties įtaisas Z. Ana nustato A įtaiso būseną, o Benas – B įtaiso būseną.

Pradžioje Ana ir Benas pagamina tokį detektorių:



Jie pastebi, kad įtaiso Z būsena yra TAIP tik tuo atveju, jei A yra TAIP ir siunčia teigiamą signalą prijungtam įtaisui, o B yra NE (ir jokio signalo nesiunčia). Tai nėra tai, ko jie nori.

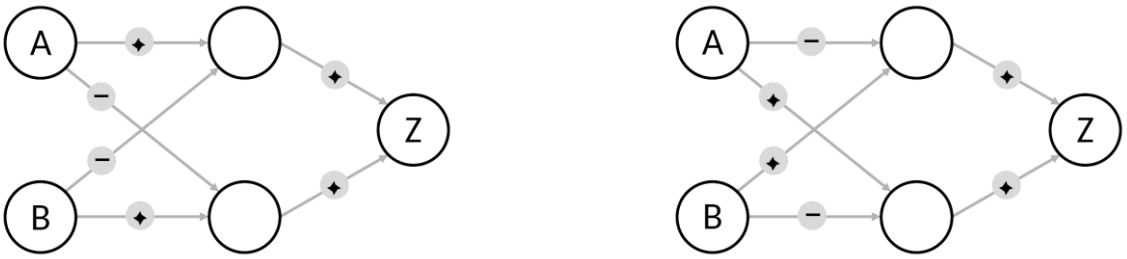
Tada Ana ir Benas sukonstruoja sudėtingesnį konfliktų detektorių (žr. paveikslą žemiau). Z būsena turi būti TAIP tik tada, kai A ir B yra skirtingų būsenų (TAIP ir NE arba NE ir TAIP). Priešingu atveju (kai A ir B yra vienodų būsenų) Z būsena turi būti NE. Deja, jungtys siunčiamiems signalams dar nėra teisingai sutvarkytos.



Kiekvienai jungčiai nustatykite siunčiamą teigiamą (+) ar neigiamą (–) signalą, kad konfliktų detektorius veiktų tinkamai (atsižvelgiant į visas A ir B būsenas).

Paaiškinimas

Šiuose paveiksluose parodyti du galimi sprendimai:

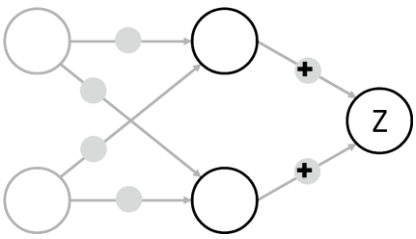


Remdamiesi pirmojo sprendimo pavyzdžiu, paaiškinsime, kaip veikia įtaisas. Toliau pateiktos schemos aprašo sistemos būsenas visiems įvesties deriniams. Jos nusako pagrindinį principą: tik tuo atveju, jei įtaisai A ir B yra skirtingų būsenų, vieno iš viduryje esančių įtaisų būsena yra TAIP. Šiuo atveju įtaisas, kurio būsena yra TAIP, siunčia teigiamą signalą įtaisui Z ir jo būsena tampa TAIP.

Antrojo sprendimo schema veikia taip pat. Jei, neatjungdami jungčių, sukeisite vietomis du vidurinius įtaisus (viršutinį ir apatinį), turėsite tą pačią schemą kaip ir pirmojo sprendimo atveju.

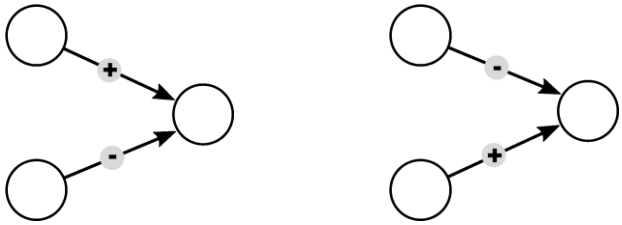
Kitų sprendimų nėra. Tai galima įrodyti taip: išmėginkite visų įmanomų jungčių derinius, t. y., signalų + ir – derinius, ir patikrinkite, kaip kiekvienu atveju veikia schema. Pamatysite, kad jokia kita schema, išskyrus dvi pateiktas, neveikia taip, kaip norima.

Konflikto detektorius turi siųsti teigiamą signalą esant lygiai dviem skirtingiems įėjimams (TAIP ir NE arba NE ir TAIP). Įtaisai sukonstruoti taip, kad siunčia signalą tik tada, kai gauna daugiau teigiamų nei neigiamų signalų. Taigi logiška, kad įtaisas Z turi gauti teigiamą signalą kiekvienai iš skirtingų jungčių (jei abiejų jungčių būsenos būtų neigiamos, įtaiso būsena niekada netaptų TAIP, jei vienos iš dviejų jungčių būsena būtų neigiama, galima nagrinėti tik vieną atvejį). Taigi abi dešiniųjų jungčių būsenos turi būti +.

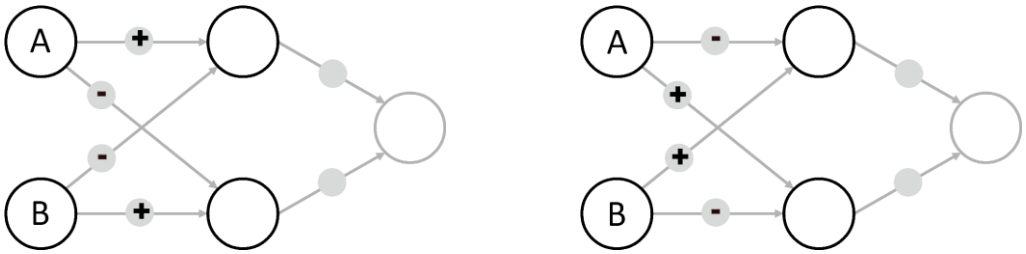


Schema simetriška. Taigi akivaizdu, kad vienam atvejui reikia pasirinkti vieną iš dviejų įtaisų, o kitam atvejui – kitą įtaisą. Kiekvienas iš dviejų įtaisų turi siųsti teigiamą signalą lygiai vienu konflikto atveju ir neturi siųsti signalo (ar siųsti neigiamą signalą) kitu atveju. Būtent, toks atvejis yra pagal užduoties pavyzdyje pateiktą schemą (jei abu signalai būtų teigiami, signalas būtų

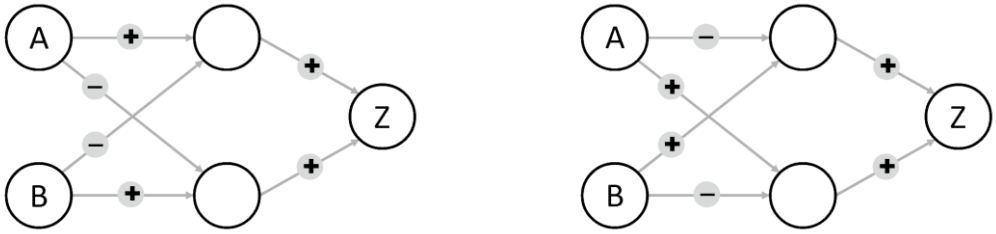
TAIP ir TAIP arba simetriškai kitu atveju, o jei abu būtų neigiami, signalas nebūtų siunčiamas): taigi vienas iš signalų turi būti +, o kitas signalas turi būti -.



Kadangi abu atvejai yra simetriški, pasirenkame vieną jungtį + ir vieną - vienam iš vidurinių įtaisų ir sukeičiam juos su kitu viduriniu įtaisu.



Galimi tik du sprendimai:



Tai informatika!

Konflikto detektorius apdoroja dvi įvesties reikšmes (TAIP arba NE) ir pateikia išvesties rezultatą TAIP tik tada, kai abi įvesties reikšmės skiriasi. Ši loginė funkcija vadinama išskirtine (angl. *exclusive*) arba (XOR). Pirmoji Anos ir Beno mašinos versija (du jungikliai ir vienas išvesties blokas) yra supaprastinta perceptrono versija, kurią 1957 m. aprašė Frankas Rosenblattas (Frank Rosenblatt). Išėjimo blokas atitinka nervinę ląstelę (neuroną), kuri gali apdoroti įėjimo signalus ir sukurti išėjimo signalą. Naudojant šį perceptroną galima atlikti logines operacijas AND ir OR, bet ne išskirtinę operaciją XOR. Tam reikia dar vieno perjungimo elemento, kaip ir šio uždavinio atveju. Tai buvo pripažinta tik devintajame dešimtmetyje (Rumelhart, Hinton ir Williams, 1986) ir vėliau buvo galima programuoti dirbtinius neuroninius tinklus, kurie veikia panašiai kaip žmogaus smegenys ir gali, pavyzdžiui, vertinti nuotraukų vaizdus ir atpažinti objektus.

47. Domino

Domino kauliuką sudaro dvi pusės. Kiekvienoje pusėje yra nuo 1 iki 6 taškų.

Šioje užduotyje kauliukus reikia sudėti iš eilės taip, kad:

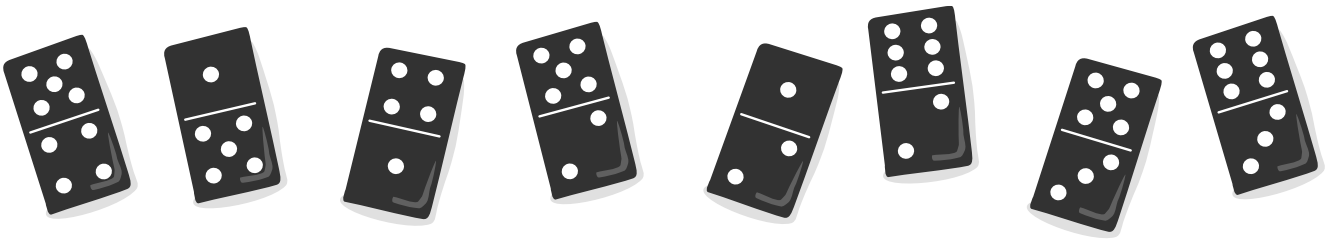
- visi aštuoni kauliukai būtų panaudoti ir
- dviejų gretimų kauliukų šalia esančiose pusėse būtų tas pats akių skaičius.

Tinkamas sprendimas yra kauliukų grandinė, kurioje visi aštuoni kauliukai išdėstyti tam tikra tvarka. Teisingi sprendiniai gali būti keliais skirtingais variantais išdėlioti kauliukai.

Sprendimas turėtų atrodyti taip:

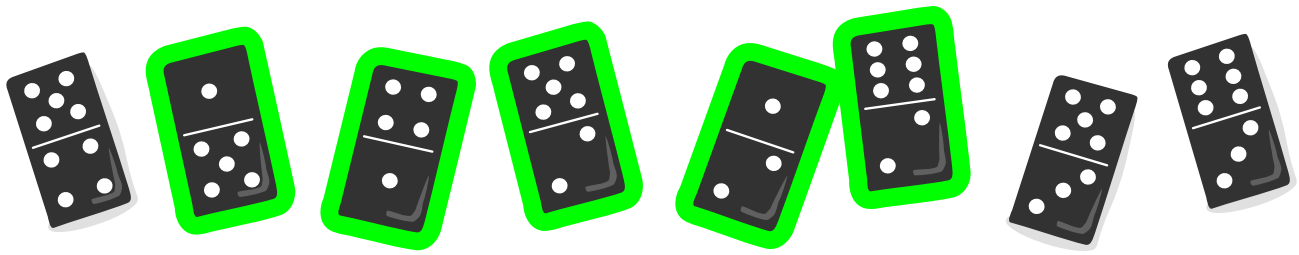
Pastebėsime, jog kai kurių kauliukų negalima dėti pažymėtuose grandinės galuose.

Nurodykite visus kauliukus, kurie gali būti panaudoti grandinės galuose.



Paaiškinimas

Iš aštuonių domino kauliukų penki gali būti grandinės galuose:



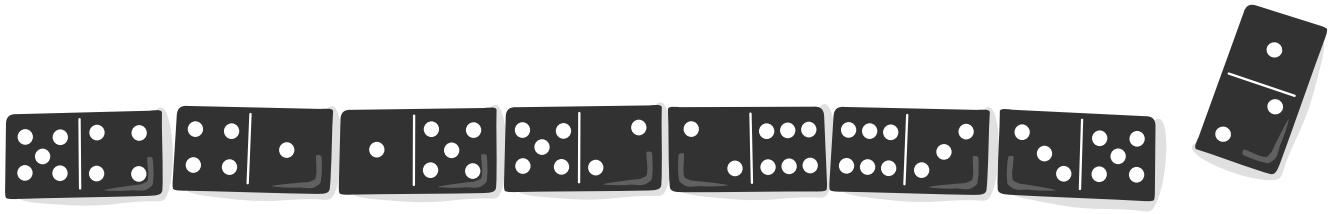
Pirmiausia suskaičiuokime taškų skaičių ant kiekvieno domino kauliuko ir panagrinėkime, kiek dažnai atsiranda kiekvienas iš pateiktų taškų skaičius. Mūsų atveju turime aštuonis domino kauliukus, o tai reiškia 16 kauliukų pusių. Lentelėje pateikta apžvalga kartu su papildomu stulpeliu, kuriame nurodyta, ar taškų atsiradimo dažnis yra lyginis, ar nelyginis:

Taškai	**Dažnis**	**Lyginis ar nelyginis**
	3	nelyginis
	3	nelyginis
	2	lyginis
	2	lyginis
	4	lyginis
	2	lyginis

Atkreipkite dėmesį, kad paviršiai su 3, 4, 5 ir 6 taškais pasitaikė lyginiu dažnumu, o paviršiai su 1 ir 2 taškais – nelyginiu dažnumu. Tai svarbu, nes gretimos dviejų kauliukų pusės visada turi turėti vienodą taškų skaičių, todėl tam tikroje domino grandinėje jos būna poromis.

Kauliukai su 3, 4, 5 ir 6 taškų paviršiais gali būti sugrupuoti į poras be liekanos, o su 1 ir 2 paviršiais taip nėra.

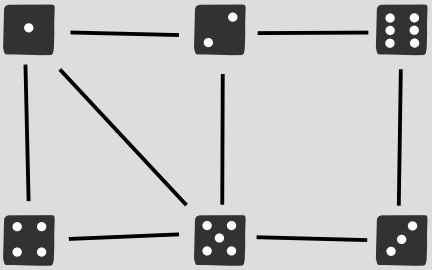
Norint išrikiuoti kauliukus svarbu atskirti lyginio ir nelyginio dažnio kauliukus. Būtent 1 ir 2 (nelyginį dažnį turintys paviršiai) turi būti kruopščiai išdėstyti, nes dėl jų gali susidaryti situacija, kai seka negali būti tęsiama, nors yra likusių kauliukų. Pavyzdžiui, jei kauliukus išdėstome taip, kad visi kauliukai su vienu tašku būtų panaudoti, o grandinė baigtųsi kauliuku su vienu tašku, sekos negalima pratęsti, net jei dar yra laisvų kauliukų.



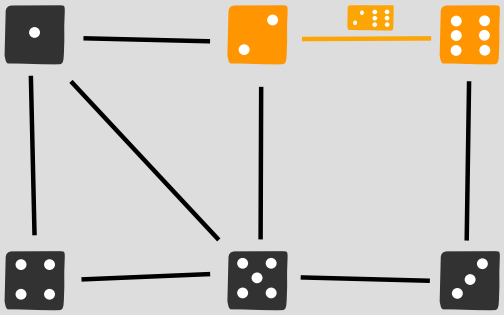
Tačiau šią problemą galima išspręsti sąmoningai naudojant kauliuką su 1 tašku kaip galinį. Tas pats pasakytina ir apie paviršius su dviem taškais. Sekos pradžia ir pabaiga yra unikalios, nes joms nereikia nei pirmesnio (pirmojo kauliuko atveju), nei tolesnio (paskutinio kauliuko atveju) kauliukų. Iš aštuonių domino kauliukų penkiuose iš jų vienoje arba abiejose pusėse yra 1 arba 2 taškai. Todėl tinkamais galiniais kauliukais laikomi šie penki domino kauliukai: (1,2), (1,4), (1,5), (2,5), (2,6).

Tai informatika!

Yra keletas tinkamų sprendimų, kuriuose naudojami visi aštuoni pateikti kauliukai ir kurie atitinka nurodytas taisykles. Norėdami geriau apžvelgti visus galimus sprendimus, galime šią užduotį pavaizduoti kitaip – informatikai tam naudoja vadinamuosius grafus:

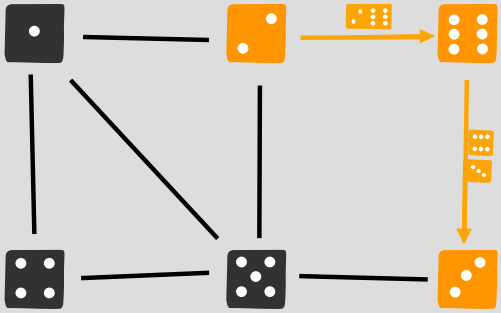


Pateiktame grafe kvadratukai (vadinami viršūnėmis) vaizduoja visus šešis galimus domino kauliukų paviršius. Linijos (vadinamos briaunomis) vaizduoja domino kauliukus, o kiekviena linija jungia dvi domino kauliuko puses. Šioje užduotyje buvo pateikta aštuoni konkretūs domino kauliukai, todėl grafe matome lygiai aštuonias briaunas. Pavyzdžiui, 2-6 kauliuką vaizduoja ši briauna:



(Pastebėjimas: tarp kai kurių paviršių (pvz., 5-6) nėra linijos, nes šio derinio nėra tarp mūsų užduotyje pateiktų domino kauliukų.)

Kad rastume, kaip tinkamai išdėstyti visus aštuonis domino kauliukus, turime rasti tokią seką, kurioje sutaptų gretimų segmentų (kauliuko pusių) taškai. Taigi, padėjus pirmąjį kauliuką jau aišku, kokiu taškų skaičiumi turi prasidėti antras kauliukas, nes gretimos dviejų kauliukų pusės visada turi turėti vienodą taškų skaičių. Grafe tai akivaizdu iš to, kad kauliukai gali būti dedami viena šalia kito tada ir tik tada, kai jų briaunos susitinka toje pačioje viršūnėje. Pavyzdžiui, kauliukai (2,6) ir (6,3) gali būti dedami vienas šalia kito, nes abiejuose yra pusės su 6 taškais:



Kauliukų išdėstymą galima suprasti kaip kelią (briaunų seką) grafe. Šiuo keliu kiekviena briauna turėtų būti aplankyta lygiai vieną kartą, kad būtų užtikrinta, jog visi aštuoni kauliukai bus panaudoti ir nė vienas nebus panaudotas kelis kartus. Kelias, kuris per kiekvieną briauną eina lygiai vieną kartą, vadinamas Oilerio keliu.

Šis pavadinimas – iš šveicarų matematiko Leonardo Oilerio, kuris sukūrė grafų teorijos sritį, pavardės. Oileris sugebėjo įrodyti, kad sujungtame grafe toks kelias egzistuoja tada ir tik tada, kai yra ne daugiau kaip dvi viršūnės su nelyginiu briaunų skaičiumi.