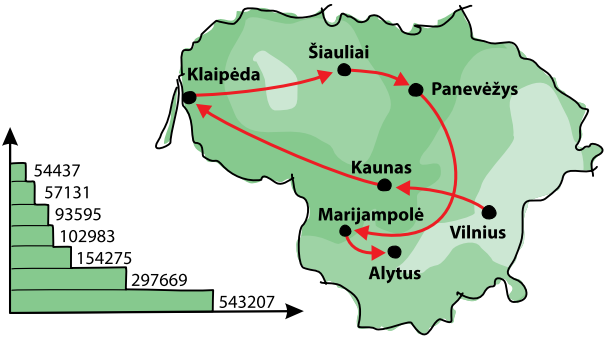
MIESTAI

Rodyklėmis rodoma, kaip aplankyti pagrindinius Lietuvos miestus. Pradedama nuo Vilniaus, turinčio daugiausia gyventojų (543 207), toliau lankomi miestai mažėjimo tvarka.

Pateikta diagrama vaizduoja gyventojų skaičių šiuose miestuose, deja, nenurodyti miestų pavadinimai.



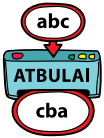
Kiek gyventojų Panevėžyje?

Duomenų ir informacijos apdorojimas – svarbi informatikos dalis. Informacija gali būti pateikiama įvairiais būdais, reikia pastebėti dėsningumus ir atrasti ryšius tarp skirtingai vaizduojamų duomenų. Šiuo atveju tuos pačius duomenis naudojame ir žemėlapyje, ir juostinėje diagramoje. Svarbu pastebėti ryšį tarp miestų rikiavimo eilę nurodančių rodyklių žemėlapyje ir surikiuotų skaičių diagramoje.

TEKSTO MAŠINA

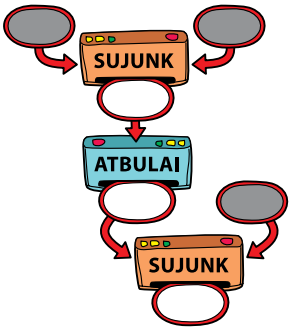
Tekstui apdoroti sukonstruotos dvi mašinos:

Tekstų sujungimo mašina paima du tekstus ir juos sujungia.



Teksto apsukimo mašina paima pateiktą tekstą ir jį perrašo atbulai.

Iš dviejų tekstų sujungimo mašinų ir vienos apsukimo mašinos gauname sudėtingesnę teksto mašiną. Tokiai mašinai reikia trijų tekstų – tai pažymėta pilkais ovalais. **Kokius tekstus turėtume pateikti šiai teksto mašinai, kad gautume žodį INFORMACIJA?**



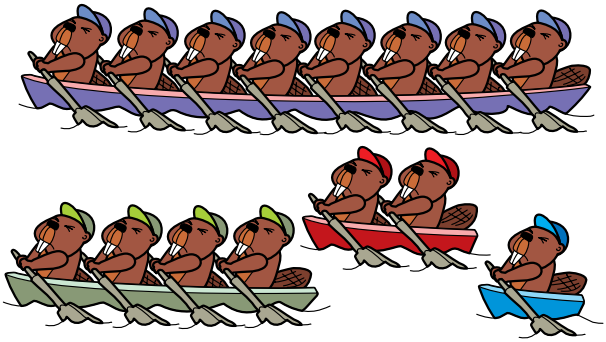
1. FNI AMRO AJIC
2. AMR OFNI AJIC
3. AMR OFNI CIJA
4. INF ORMA CIJA

Formalioji kalba – tai žodžių rinkinys, kurį sudaro baigtinė raidžių, simbolių ir ženklų aibė. Šią kalbą apibrėžia formalioji gramatika – taisyklės, pagal kurias gaunami nauji žodžiai. Formaliąsias kalbas nagrinėja informatikos ir lingvistikos mokslai. Šias kalbas generuoja ir analizuoja baigtiniai automatai. Automatas – tai skaitmeninio aparato modelis, kurio išėjimo simbolis priklauso nuo įėjimo simbolio, o tolesnė būsena – nuo esamos būsenos ir įėjimo simbolio. Baigtinio automato būsenų skaičius baigtinis.

IRKLAVIMO TURNYRAS

B

Bebrai rengiasi dalyvauti irklavimo turnyre. Jie turi keturias valtis: aštuonvietę, keturvietę, dvivietę ir vienvietę.

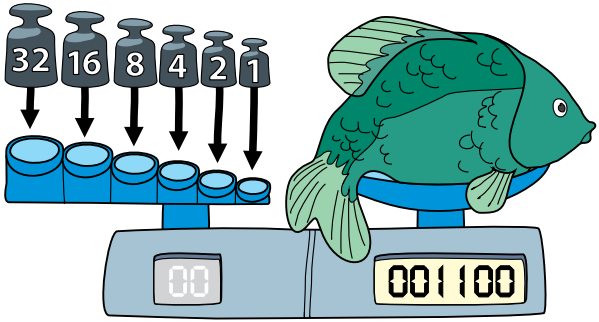


Kiekvienas bebras gali dalyvauti tik vienoje rungtyje. Bebrų treneris registruoja, kurios valtys pasirengusios plaukti turnyre. Jis pradeda nuo didžiausios aštuonvietės valties ir, jei ji sukomplektuota, rašo 1. Analogiškai daro su kitomis valtimis. Pavyzdžiui, jei dalyvauja 10 bebrų, tai treneris užrašo 1010.

Turnyre rengiasi dalyvauti 13 bebrų.
Ką užrašys treneris?

Kompiuteriai naudoja dvejetainę skaičiavimo sistemą – 0 ir 1. Šia sistema efektyviai sprendžiama daugelis uždavinių. Kiekvieno skaitmens svoris dvejetainiame užraše yra 2, pakeltas *n*-uoju laipsniu (čia *n* nusako skaitmens vietą nuo skaičiaus galo). Didžiausias keturių pozicijų dvejetainis skaičius yra 1111, t. y. 15 mums įprasta dešimtaine sistema.

Bebro svarstyklės rodo svorį dešimtainiais (kairėje) ir dvejetainiais (dešinėje) skaičiais.
Žuvis sveria 1100 kg (dvejetainiu pavidalu).



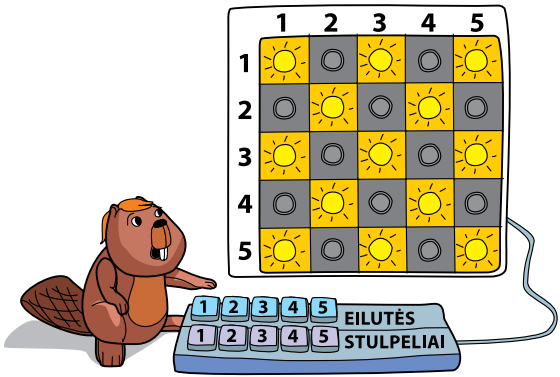
Kuriuos svarsčius reikia uždėti, kad matytume, kiek sveria žuvis mums įprastu dešimtainiu pavidalu?

Kompiuteryje duomenys koduojami dvejetainiais skaičiais – nuliu ir vienetu. Kaip dvejetainį skaičių užrašyti mums įprastu dešimtainiu pavidalu? Vienas paprastesnių būdų – įsivaizduoti svarstykles su dvejeto kartotinių svorių svarsčiais (1, 2, 4, 8, 16, 32 ir t. t.). Pradedame nuo dvejetainio skaičiaus galo ir dedame tik tuos svarsčius, kurie atitinka dvejetainio skaičiaus vienetus (nulius atitinkančias lėkšteles paliekame tuščias).

LEMPUTĖS

Bebras valdo tinkleliu išdėstytų lempučių sistemą. Bebro pultelis – klaviatūra, kurios vienu klavišu valdomos visos vienos eilutės arba vieno stulpelio lemputės: jei lemputė nedega – įjungiama, jei dega – išjungiama.

Bebras nori, kad lemputės šviestų štai taip:



Kuriuos pultelio klavišus reikia spausti?

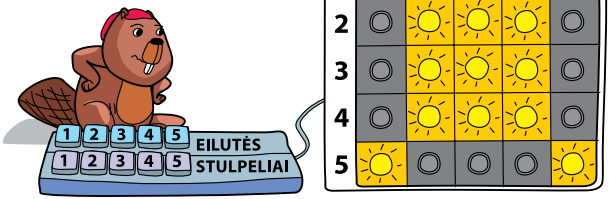
Dvejetainė (binarinė) logika operuoja bitais – mažiausiais informacijos vienetais, įgyjančiais dvi reikšmes. Elektroninių grandinių veikimas iš esmės grindžiamas dvejetaine logika, arba Bulio algebra (pavadinta anglų matematiko Džordžo Bulio (George Boole) vardu). Lemputės veikimas – akivaizdus dvejetainės logikos pavyzdys, nes lengvai pastebimos dvi būsenos: lemputė įjungta (šviečia) – lemputė išjungta (nešviečia). Čia taikoma loginio neigimo (inversijos) operacija, kai esama būsena keičiama priešinga. Dar daugiau – loginio neigimo operacija taikoma visoms vienos eilutės arba stulpelio lemputėms.

PROGRAMUOJAMOS LEMPUTĖS

Bebras valdo tinkleliu išdėstytų lempučių sistemą. Viena komanda valdomos visos vienos eilutės arba vieno stulpelio lemputės: jei lemputė nedega – įjungiama, jei dega – išjungiama. Bebras parašė keturias programas:

A. 1S 5S 2E 3E 4E
B. 1E 5E 2S 3E 4E
C. 1E 2E 3E 4E 5E 1S 5S
D. 1S 2S 3S 4S 5S 1S 5S 1E 5E

S žymi stulpelį, **E** – eilutę, pavyzdžiui, **1S** komanda įjungiamos visos pirmojo stulpelio lemputės, tą pačią komandą parašius antrą kartą – visos pirmojo stulpelio lemputės išjungiamos. Trijų programų rezultatas yra lemputės, šviečiančios taip, kaip parodyta paveiksle.



Viena programa pateikia kitokį rezultatą. Kuri?

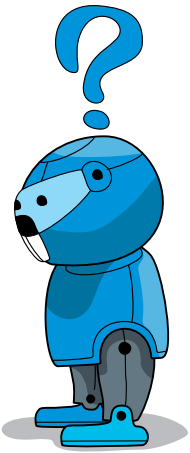
Lemputės veikimas grindžiamas dvejetaine logika, t. y. lemputė yra kurios nors iš dviejų būsenų: įjungta (šviečia) – išjungta (nešviečia). Lemputės būsena programuojant nusakoma komanda – viena komanda valdoma visa lempučių eilutė arba visas stulpelis. Kiekviena lemputė priklauso kuriai nors eilutei, taip pat ir kuriam nors stulpeliui, todėl atliekant komandas reikia atminti, kad lemputės būseną gali keisti ir eilutės, ir stulpelio komandos.

TIK DEŠINĖN

Bebriukas sukonstravo robotą, kuris moka vykdyti tik dvi komandas:

paeik 10 cm į priekį;

pasisuk į dešinę ketvirtadalį apskritimo – stačiu kampu.

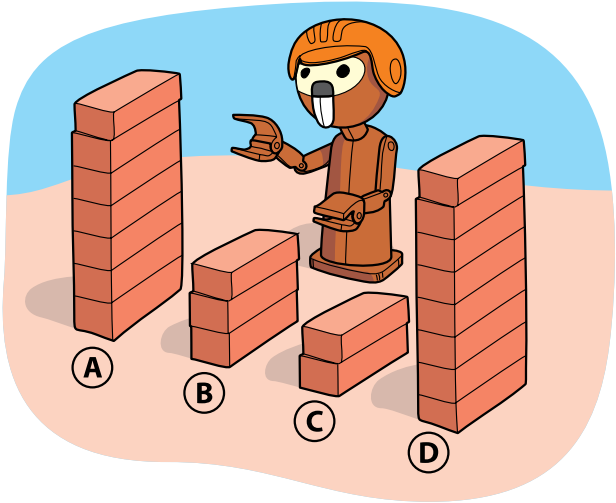


Ar galima naudojantis tik šiomis komandomis pasiekti, kad robotas būtų pasisukęs į kairę stačiu kampu? Jei taip, nurodykite komandų seką.

Pradėdamas programuoti turi galvoti veiksmus: komandas ir būsenas. Programuojamos kompiuterinės sistemos veiksmus gali riboti techninės galimybės. Pavyzdžiui, bebriuko robotas nemoka sukti į kairę, t. y. negali atlikti posūkio į kairę komandos. Daug būsenų kompiuterinėse sistemose turi ribotų veiksmų galimybes, kaip ir bebriuko robotas, kuris negali pasisukti į kairę. Informatikus domina, kaip turimai būsenų aibei parinkti tinkamų veiksmų aibę.

PLYTŲ BOKŠTAI

Robotas gali perkelti plytas iš vieno bokšto į kitą. Viena komanda robotui užrašoma taip: (bokštas, skaičius, bokštas). Pavyzdžiui, (A, 3, D) reiškia: iš A bokšto paimti 3 plytas ir dėti jas ant D bokšto.



Užrašykite komandas robotui perkelti plytas taip, kad visi bokštai būtų vienodo aukščio.

Mokantis sudaryti algoritmus ir programas svarbi komandos sąvoka. Komanda – tai tikslus ir aiškus vienareikšmis nurodymas, kurį gali atlikti kompiuteris, robotas ar automatas. Komanda užrašoma sutartiniais ženklais, žodžių santrumpomis ar žodžiais. Svarbu susitarti, kas ką reiškia, ir tiksliai laikytis nurodymų. Komandos vartojamos programavimo kalbose, kompiuterių ir kitokių įrenginių sąsajose.

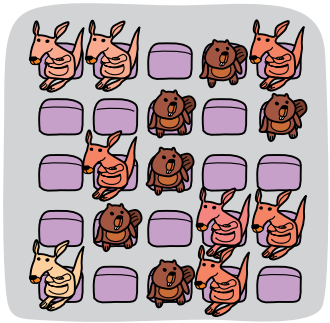
Bebras paleido skraidyti droną virš salės, kurioje sėdi kengūros ir bebrai.

Dronas valdomas komanda **SKRISK *kryptis žingsnis.***

– Parametras ***kryptis*** atitinka vieną iš raidžių: **Š** (šiaurė), **R** (rytai), **P** (pietūs), **V** (vakarai).
– Parametras ***žingsnis*** atitinka atstumą: tarp gretimų kėdžių ar eilių yra vienas žingsnis.

Dronas pradeda skristi nuo salės centro ir vykdo šias komandas:

SKRISK V 1
SKRISK Š 2
SKRISK R 3
SKRISK P 3
SKRISK V 1
SKRISK P 1
SKRISK V 3



Įvykdęs kiekvieną komandą, dronas fotografuoja.
Viena kengūra liko nenufotografuota – trūksta paskutinės komandos. Užrašykite.

Komanda – viena svarbiausių programavimo sąvokų. Programuotojui svarbu išsiaiškinti, kas tiksliai atliekama pagal vieną ar kitą komandą. Kad komandą galima būtų taikyti įvairioms situacijoms, universaliai, ji paprastai turi vadinamuosius parametrus – dydžius, kurių reikšmės keičiamos. Šiuo atveju komanda turi du parametrus. Svarbi šių parametrų pateikimo tvarka: pirmas parametras nusako kryptį (jo reikšmė – viena iš keturių raidžių), antras parametras nusako atstumą (išreiškiamą natūraliuoju skaičiumi).

DRONAS VIRŠ SALĖS

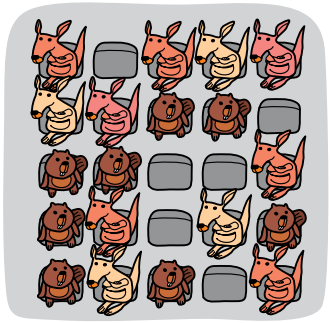
B

Bebras paleido skraidyti droną virš salės, kurioje sėdi kengūros ir bebrai.
Dronas valdomas komanda **SKRISK *kryptis žingsnis*.**

– Parametras ***kryptis*** nusakomas raidėmis **Š** (šiaurė), **R** (rytai), **P** (pietūs), **V** (vakarai).
– Parametras ***žingsnis*** nusakomas atstumu: tarp gretimų kėdžių ar eilių yra vienas žingsnis.

Dronas pradeda skristi nuo salės centro ir vykdo šias komandas:

SKRISK V 2
SKRISK Š 2
SKRISK R 3
SKRISK P 3
SKRISK R 1
SKRISK P 1
SKRISK V 4
SKRISK Š 3



Įvykdęs komandą, dronas fotografuoja.
Kiek bebrų ir kiek kengūrų nufotografavo dronas?

Kaip matome, dronas valdomas viena komanda, nurodančia skristi. Kuria kryptimi ir kaip toli skristi, nusakoma komandos parametrais. Parametrai daro komandą universalią, tinkančią daugeliu atvejų. Keisdami parametrų reikšmes, atrodo, gauname vis kitas komandas. Svarbūs ne vien parametrai, bet ir jų pateikimo tvarka. Komandos vykdomos iš eilės taip, kaip yra užrašytos.

MIESTO STATULA

Kengūriukas apsistojo viename iš Bebriškių miesto viešbučių. Norėdamas pamatyti garsiąją Bebro statulą, jis vadovavosi viešbučio instrukcijomis:

1. Išeik pro viešbučio duris ir suk į kairę.
2. Priėjęs pirmąsias dvi sankryžas, eik tiesiai.
3. Priėjęs trečiąją sankryžą, suk į dešinę.
4. Eik tiesiai. Priėjęs pirmąją sankryžą, suk į kairę.
5. Eik tiesiai. Priėjęs pirmąją sankryžą, suk į dešinę.

Kengūriukas sėkmingai rado statulą ir dabar ją fotografuoja.

Kuriame viešbutyje apsistojo kengūriukas?

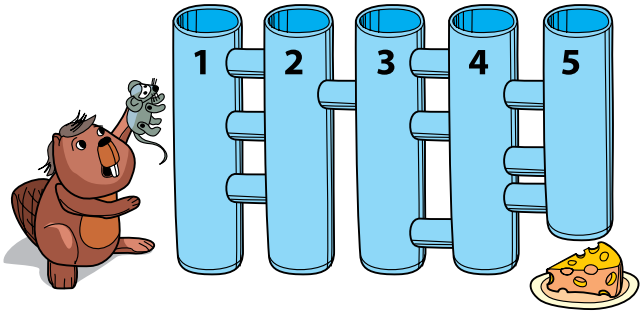
Instrukcijų – komandų rinkinio – pateikimas ir tikslus vykdymas yra esminė algoritmų, sudarančių informatikos pagrindą, idėja. Algoritmas yra uždavinio (problemos) sprendimas žingsnis po žingsnio. Siekiant išspręsti uždavinį, žingsniai turi būti kuo mažesni ir tikslesni. Algoritmai užrašomi programavimo kalbomis. Juos vykdo transliatoriai (kompiliatoriai, interpretatoriai).

PASIEK SŪRĮ

Bebras sukonstravo robotą – pelę. Pelė juda pagal šias komandas:

1. Bėga stačiu vamzdžiu žemyn, kol pamato skersinį vamzdelį.
2. Pasuka į skersinį vamzdelį ir juo perbėga į kitą vamzdį.

Šias komandas kartoja tol, kol išbėga laukan.



Kuriuo vamzdžiu turi pradėti bėgti pelė, kad atsidurtų prie sūrio?

Daugelis automatų suprogramuoti taip, kad tiksliai vykdytų komandas. Taip ir pelė robotė šiame uždavinyje juda tiksliai pagal nurodytas komandas: keliauja vamzdžiu žemyn, kai pamato skersinį vamzdelį, juo perbėga į kitą vamzdį. Šios komandos yra deterministinės, t. y. priklauso nuo pasirinkto vamzdžio ir kelio sujungtuose vamzdžiuose. Daugelis kompiuterių programų yra deterministinės. Tai reiškia, kad kiekvieną kartą programa, naudodama tuos pačius duomenis, atlieka tuos pačius veiksmus ir gauna tą patį rezultatą.

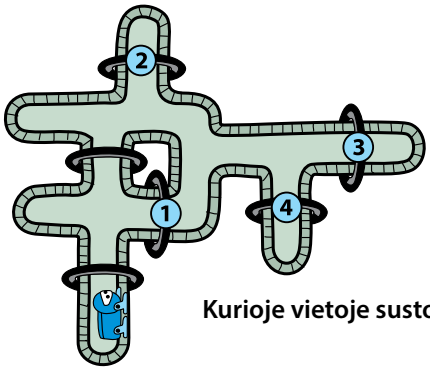
SLANKUSIS ROBOTAS

Slankusis robotas juda prisišliejęs prie kelkraščio. Robotas geba atlikti keturias komandas:

PRADĖK – pradeda judėti į priekį.
EIK – juda toliau tuo pačiu kelkraščiu.
PEREIK – pereina į kitą kelio pusę ir toliau juda į priekį.
SUSTOK – sustoja.

Komanda **PRADĖK** vykdoma bet kurioje vietoje, t. y. ten, kur stovi robotas. Kiekviena kita komanda vykdoma tik tada, kai robotas praslenka pro tamsų magnetinį kontrolės punktą. Slankusis robotas vykdo štai tokią programą:

PRADĖK, PEREIK, EIK, EIK, EIK, SUSTOK.

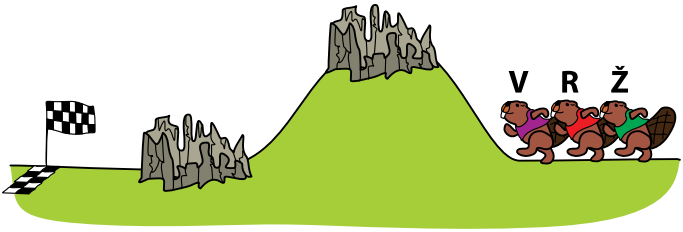


Kurioje vietoje sustos robotas?

Automatiškai judančios transporto priemonės reikalingos daugelyje darbo vietų, pavyzdžiui, tuneliuose, oro uostuose, gamyklose. Šias automatines priemones, arba kitaip – robotus, valdo kompiuterių programos. Iš esmės kompiuterių programa yra komandų seka. Komandos yra susietos su transporto priemonių fiziniais jutikliais – tokiu būdu valdomos šios transporto priemonės.

KROSAS

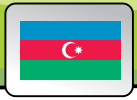
Trys bebrai bėga krosą: iš pradžių bėgama į kalną, tada tarp uolų, nusileidžiama pakalnėn ir vėl tarp uolų. Vilė pradeda pirma, paskui bėga Rudis ir tada Žalė.



Bėgdamas į kalną, Rudis pralenkia vieną bebrą.	R	
Bėgdama pakalnėn, Vilė pralenkia vieną bebrą.	V	
Bėgdama tarp uolų, Žalė pralenkia vieną bebrą.	Ž	

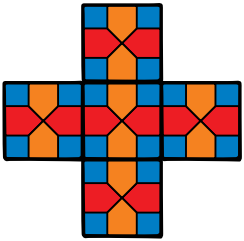
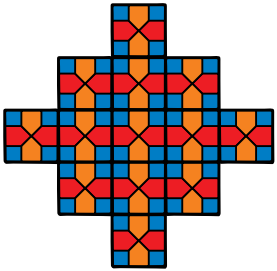
Kokia eilės tvarka bebrai pasieks finišą?

Rašant programų komandas, svarbu numatyti tų komandų vykdymo sąlygas. Čia bėgimo komanda priklauso nuo sąlygų: ar bėgama į kalną, ar pakalnėn, ar tarp uolų. Skirtingomis sąlygomis keičiasi ir bebrų išsidėstymo eilė. Programuojant reikia atidžiai sekti sąlygas, o parašytą programą būtina testuoti – įsitikinti, ar programa tinkamai veikia, ar tiksliai atsižvelgiama į sąlygas.



VITRAŽAS

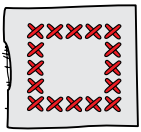
Robotas puošia langus vitražais iš trijų spalvų stiklo gabaliukų: mėlynų, raudonų ir oranžinių. Kiekviename vitraže kartojamas vienas toks pat fragmentas 

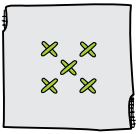
3 stulpelių vitražą sudaro 5 fragmentai	5 stulpelių vitražas atrodo šitaip:
	

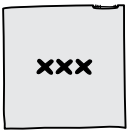
Kiek reikės mėlynų kvadratinių stiklo gabaliukų 7 stulpelių vitražui sudaryti?

Robotas veikia pagal programą. Čia programos funkcionalumas aprašomas pateikiant porą nedidelių pavyzdžių. Reikia suvokti funkcinį uždavinio aprašą ir paversti jį programa, t. y. sugalvoti algoritmą, kaip kurti vitražą. Tam reikia atpažinti pasikartojančias dalis – informatikoje tai vadinama paprogramėmis (arba procedūromis ar funkcijomis). Šitaip išanalizavus uždavinį nesunku apskaičiuoti prašomą rezultatą.

Bebras turi mašiną kryželiams siuvinėti. Vienu žingsniu išsiuvinėjamas vienas kryželis.
Kengūros mašina modernesnė: vienu žingsniu gali išsiuvinėti vieną iš šių trijų figūrų:

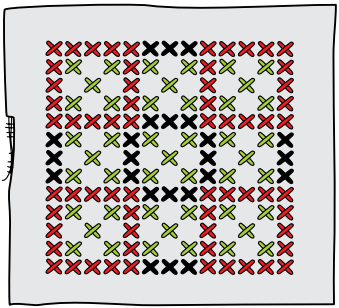






Vieno kryželio ant kito siuvinėti negalima.

Bebras išsiuvinėjo ornamentą:



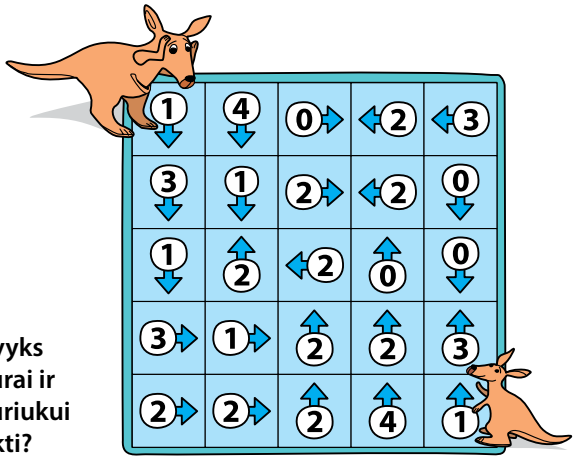
Siuvinėdama bebro mašina padarė 133 žingsnius. Kengūra juokiasi: jos mašinai prireiktų kur kas mažiau žingsnių. **Kiek žingsnių padarys kengūros mašina?**

Atpažinę pasikartojančias dalis ir joms suteikę tam tikrą formą, galime kur kas efektyviau atlikti tolesnius veiksmus. Programuojant tokios pasikartojančios dalys vadinamos paprogramėmis (arba procedūromis ar funkcijomis). Iš dalies komandų sudarytą paprogramę toliau galime naudoti kaip vientisą ir taip sutaupyti laiko. Šitaip daro kengūros mašina – siuvinėja ne pavienius kryželius, o tam tikras figūras.

KENGŪRA IR KENGŪRIUKAS

Kengūra nori susitikti su sūnumi.

– Pirmą šuolį daro kengūra, tada kengūriukas – ir taip paeiliui.
– Šuoliuojama pagal rodyklių kryptis per tiek langelių, kiek nurodyta prie rodyklės.
– Langelių, per kuriuos šokama, rodyklės ignoruojamos.
– Kengūra susitinka su kengūriuku, kai abudu atsiduria viename langelyje.



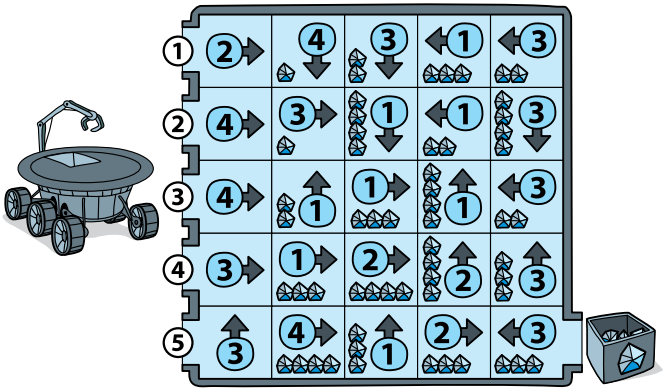
Ar pavyks kengūrai ir kengūriukui susitikti?

Aikštelę, kurioje šuoliuoja kengūros, galima laikyti paprasta programa: yra komandos, nusakančios, ką daryti, nurodyta veiksmų pradžia ir pabaiga. Rodyklės ir skaičiai langeliuose aiškiai ir nedviprasmiškai aprašo, kur ir kaip toli eiti. Tai – komandos, nesvarbu, ar jos užrašomos žodžiais, ar vaizduojamos rodyklėmis. Programą galima automatiškai vykdyti – sąlygos apibrėžtos, veiksmai nusakomi vienareikšmiškai.

MĖNULEIGIS

Bebras pagamino mėnuleigį robotą, kuris važinėja Mėnulio paviršiumi ir renka mineralus.

1. Mėnuleigis įvažiuoja per vieną iš įėjimų kairėje.
2. Važiuoja pagal rodyklių kryptis per tiek langelių, kiek nurodyta prie rodyklės.
3. Langelių, per kuriuos važiuoja, rodyklės ignoruojamos.



Kurį įėjimą ar įėjimus turi pasirinkti mėnuleigis, kad surinktų daugiausia mineralų?

Mėnuleigio judėjimą aprašantį lauką galima laikyti paprasta programa: yra komandos, nusakančios veiksmus, nurodyta pradžia ir pabaiga. Rodyklės langeliuose – tai tikrų tikriausios komandos, jos aiškios ir nedviprasmiškos, nesvarbu, kad užrašytos ne raidėmis, o pavaizduotos rodyklėmis. Prie rodyklių esančius skaičius galima laikyti komandos parametrais: jie nurodo, per kiek langelių einama. Kita komanda galėtų būti paimti esančius mineralus. Mėnuleigį galima valdyti automatiškai: komandos nusakomos vienareikšmiškai.

ROBOTĖ VOVERĖ

Robotė voverė juda plytelių taku pagal šias komandas:

– šokti ant kitos plytelės;

– padėti riešutą ant plytelės;

– kartoti nurodytą komandą 4 kartus, šiuo atveju komandą „šokti ant kitos plytelės“.

Ant vienos plytelės gali būti ir voverė, ir kiek norima riešutų.

Pagal kurią iš šių komandų sekų voverė sudėlios 4 riešutus į eilę – ant keturių viena po kitos einančių plytelių?

Ⓐ 

Ⓑ 

Ⓒ 

Ⓓ 

Šiuo uždaviniu siekiama supažindinti su iteracija, arba kartojimo komanda, ir parodyti, kaip ji veikia. Kartojimas, arba, kalbant informatikos terminais, ciklas, – vienas iš pagrindinių programavimo konceptų. Norint suvokti ciklą, reikia padirbėti praktiškai, pačiam atlikti komandas tiksliai pagal nurodymus.

ROBOTAS IR DEIMANTAI

Robotas burtininkas eina plytelių taku pagal šias komandas:


– žengti ant kitos plytelės;


– padėti deimantą ant plytelės;

– kartoti nurodytą komandą 4 kartus, šiuo atveju komandą „žengti ant kitos plytelės“.

Kas norima kartoti keletą kartų, suskliaudžiama:

– robotas 4 kartus kartoja dvi komandas: paeina 8 žingsnius į priekį.

Ant vienos plytelės gali būti ir burtininkas, ir kiek norima deimantų.
Pagal kurį iš šių komandų sekų burtininkas sudėlios 4 deimantus į eilę – ant keturių viena po kitos einančių plytelių?

Ⓐ 

Ⓑ 

Ⓒ 

Ⓓ 

Norint išsiaiškinti, kaip veikia programa, pirmiausia reikia suprasti, kaip teisingai derinti įvairias komandas. Ypač tai svarbu, kai komandos kartojamos, tai yra sudaromas ciklas: reikia išmokti atpažinti, kurios komandos vykdomos ciklo viduje, kurios išorėje, kada ciklas pradedamas ir baigiamas. Pateiktame uždavinyje komandos, kurios priklauso ciklui, t. y. atliekamos ciklo viduje, suskliaudžiamos.

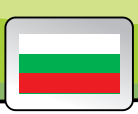
JAPONŲ KALENDORIUS

Senovės japonų kalendorius susideda iš 60 metų ciklo. Metai numeruojami nuo 1 iki 60 ir grupuojami poromis, kaip parodyta lentelėje. Metų poros nuspalvintos iš eilės: žaliai, raudonai, geltonai, baltai ir juodai (toliau vėl kartojama):

1	2	11	12	21		51	52
3	4	13	14			53	54
5	6	15	16			55	56
7	8	17	18			57	58
9	10	19	20			59	60

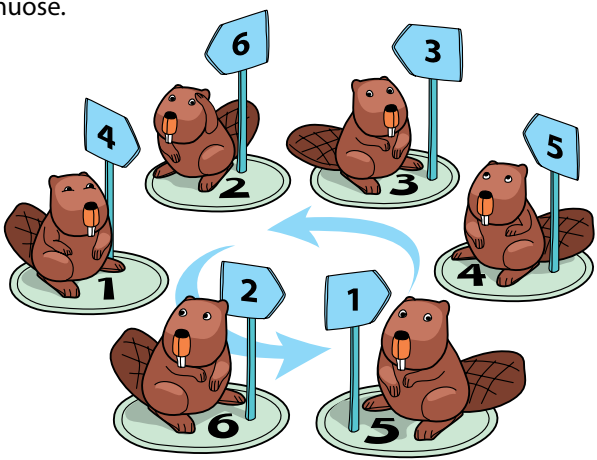
Žinoma, kad 1984 metai yra pirmieji 60 metų ciklo metai.
Kokios spalvos yra 2017 metų langelis?

Metai spalvinami pagal tam tikras taisykles, galima sakyti, algoritmą. Algoritmai – viena pagrindinių informatikos sričių. Metų porų spalvinimo algoritmą galima perprasti, nagrinėjant paveikslėlį. Tai paprasta – svarbu įžvelgti dėsningumus ir pasikartojimus. Kartojamiems veiksmams atlikti skirtas ciklas – viena svarbiausių algoritmavimo konstrukcijų.



ŠOKIS RATU

Bebriukai žaidžia žaidimą. Yra šeši sunumeruoti apskritimai, kuriuose stovi bebriukai. Kiekviename apskritime yra rodyklė, rodanti, į kurį apskritimą bebriukas turės pereiti. Po signalo kiekvienas bebriukas eina ratu pagal rodykles, kol visi bebriukai vėl atsiduria savo pradiniuose apskritimuose.



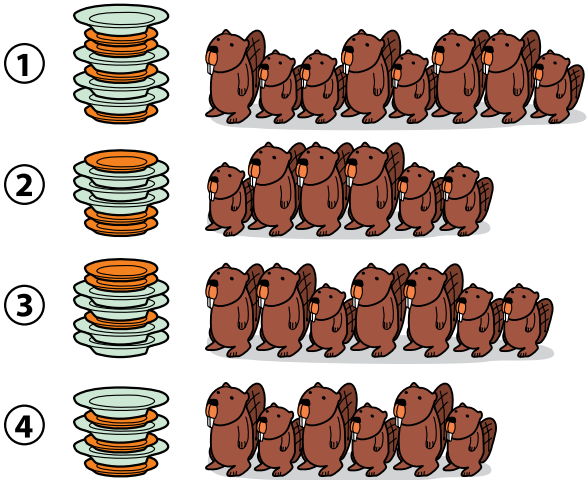
Po kelių ratų baigsis žaidimas?

Pasikartojančios programų veiksmų dalys užrašomos kartojimo komandomis – vadinamaisiais ciklais. Visos programavimo kalbos turi ciklų komandas ir netgi kelių rūšių. Kiek kartų atliekamas ciklas, nurodo sąlyga. Ciklas gali būti atliekamas nulį kartų, t. y. visai neatliekamas, daug kartų arba gali tęstis be galo – tai vadinamasis amžinasis ciklas.

BEBRIUKŲ LĖKŠTĖS

Dideli bebrai valgo iš didelių lėkščių, maži – iš mažesnių. Bebriukai laukia eilėje pietų. Virėjai turi sudėti lėkštes vieną ant kitos taip, kad lėkščių eilė atitiktų laukiančių bebriukų eilę.

Viena iš lėkščių krūvų sukrauta blogai. Kuri?

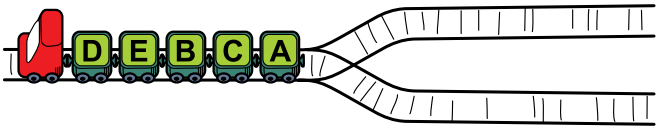


Kompiuterių programos dirba su duomenimis. Duomenys pateikiami ir apdorojami tam tikra tvarka. Čia kalbama apie dvi klasikines duomenų struktūras: eilę ir dėklą. Eilei būdinga tai, kad duomenys dedami gavimo tvarka, t. y. į eilės pabaigą, o pirmiausia išimamas pirmasis į eilę įdėtas duomuo. Dėklas – duomenų struktūra, kurios duomenys dedami iš eilės tokia tvarka, kokia pateikiami, o juos išimti galima tik pradedant nuo paskiausio įdėto duomens. Eilė apibūdinama santrumpa *FIFO* (angl. *first in, first out* – pirmas į maišą, pirmas iš maišo), o dėklas – *LIFO* (angl. *last in, first out* – paskutinis į maišą, pirmas iš maišo). Laukiantys bebriukai gali būti vaizduojami eile, o lėkščių krūvelės – dėklu.

KROVININIS TRAUKINYS

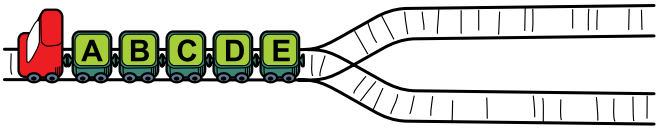
B

Bebrų traukinių stotyje stovintis traukinys atrodo štai taip:



Garvežys gali judėti pirmyn ir atgal. Jis taip pat gali pastumti ar patraukti ne daugiau kaip penkis prijungtus vagonus. Kiekvienas kartas, kai vagonai yra prijungiami arba atskiriami, skaičiuojamas kaip viena operacija.

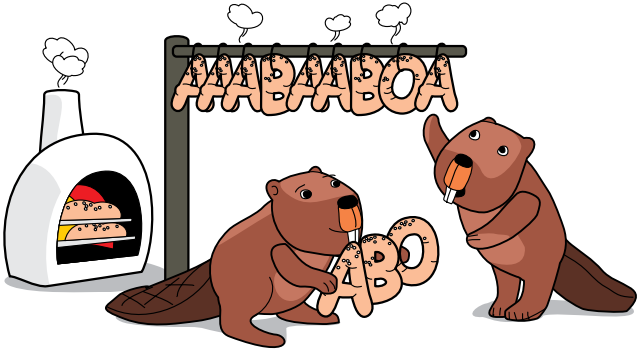
Kiek mažiausiai reikia operacijų, kad traukinys atrodytų taip:



Šiuo uždaviniu vaizduojama vadinamoji dėklo duomenų struktūra. Dėklo elementai gali būti dedami tik ant viršaus ir imami tik nuo viršaus. Angliškai dėklas dar vadinamas *LIFO* duomenų struktūra (angl. *last in, first out*) – tai reiškia, kad paskutinis į struktūrą padėtas elementas paimamas pirmas. Dėklo pavyzdžiai gali būti kaladėlių bokšto statyba ir griovimas, knygų krovimas į siaurą dėžę (vienu bokštu) ir išėmimas iš jos.

RIESTAINIAI

Du draugai dirba kepykloje. Lina kepa riestainius, traukia iš krosnies po tris skirtingų formų ir kabina juos ant kartelės iš dešinės: pirmiausia pakabina A riestainį, tada B, paskiau O. Linas pardavinėja riestainius. Nuo kartelės jis visada ima dešiniausią riestainį. Lina kepa riestainius greičiau, negu Linas juos parduoda.

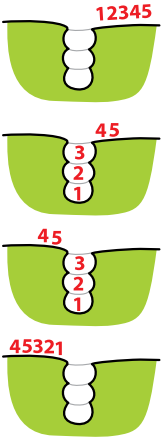


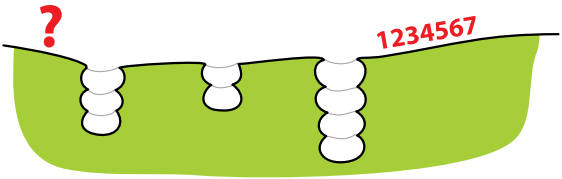
Kiek mažiausiai riestainių pardavė Linas, jei kartelė atrodo taip, kaip parodyta paveikslėlyje?

Šiuo uždaviniu iliustruojama duomenų struktūra – vadinamasis dėklas. Dėklo elementai gali būti dedami tik ant viršaus ir imami tik nuo viršaus – tai reiškia, kad paskutinis į šią struktūrą padėtas elementas paimamas pirmas. Taip pat atkreipiame dėmesį į lygiagrečių procesų ypatumus: darbai atliekami greičiau, tačiau riestainių eilė sudarkoma (kai kada tai gali būti svarbu).

KIŠKIŲ DUOBĖ

Bebrai mėgsta vaikštinėti po mišką. Takeliai miške siauri, todėl bebrai turi eiti vorele vienas paskui kitą. Nedorėliai kiškiai išrausė duobių takeliuose. Bebrai duobę įveikia, sušokdami į ją vienas ant kito, kad likusieji galėtų pereiti. Kai likusieji bebrai pereina duobę, paskutinis ištraukia visus bebrus nuo viršutinio iki apatinio. Bebrai vėl keliauja vorele. Dešinėje parodyta, kaip penki bebrai įveikia duobę.





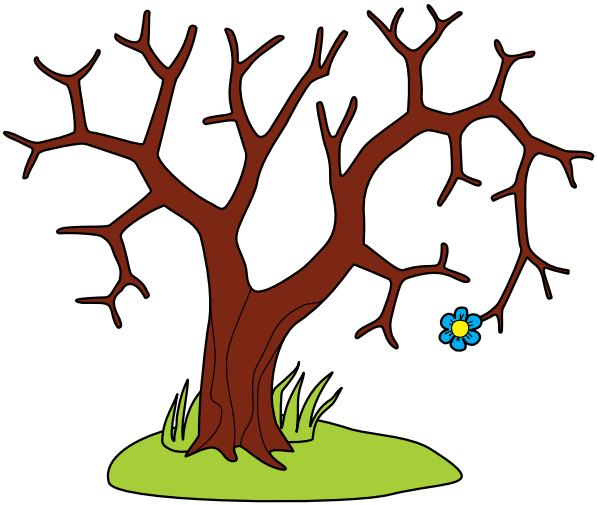
Kaip bus išsirikiavę septyni bebrai, kai pereis visas tris duobes?

Apdorojant duomenis, pavyzdžiui, atliekant jų rikiavimą, svarbu pasirinkti tinkamas duomenų struktūras. Šiam uždaviniui tinkama vadinamoji dėklo duomenų struktūra. Su dėklu gali būti atliekamos įdėjimo (elementas dedamas tik ant viršaus) ir išėmimo (imamas tik viršutinis elementas) operacijos. Sudedant ir išimant dėklo elementus, keičiama jų išdėstymo tvarka.

DVEJETAINIS MEDIS

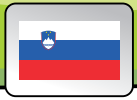
Informatikoje naudojami įvairūs kodavimo būdai. Vienas iš jų – dvejetainis medis.
Vaikščiojimas medžiu žymimas komandomis:

– pradžia (kamienas) – T
– posūkis į kairę – K
– posūkis į dešinę – D



Parašykite komandų seką, vedančią nuo kamieno prie žiedo.

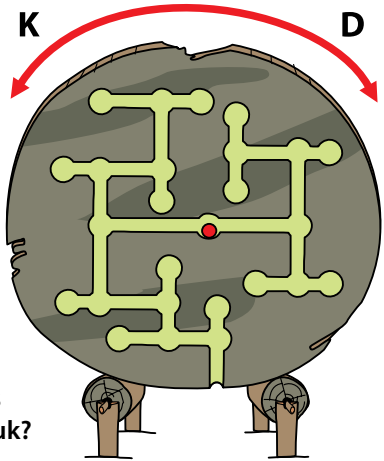
Medis – tai viena efektyviausių struktūrų duomenims sistemiškai sutvarkyti. Dvejetainis medis – tai visuma atkarpų, iš kurių prie kiekvienos galo galima jungti ne daugiau kaip dvi kitas atkarpas (vadinamas šakomis). Tokiu būdu galima taip sutvarkyti duomenis, kad jie būtų greitai randami. Dideliam medžiui aprašyti pakanka vos kelių ženklų.



LABIRINTAS BERŽO TOŠYJE

Bebriukai rado beržo tošį, kurioje kirminai išgraužė tunelių ir duobučių. Nagingas tėtis iš tošies padarė žaidimą: apdrožė tošį apskritai ir į vidurį įdėjo stiklo rutuliuką, kaip parodyta paveikslėlyje.

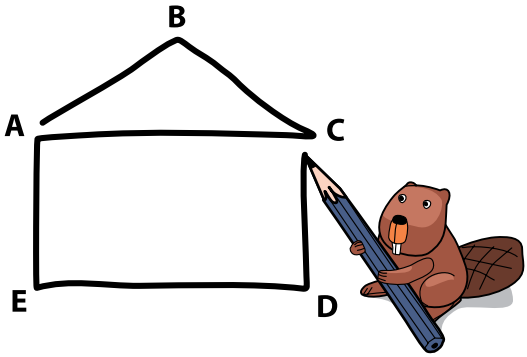
Žaidimo tikslas – išridenti rutuliuką iš tošies, sukiojant ją į kairę (K) arba į dešinę (D). Kiekvieną kartą pasukus, rutuliukas įkrenta vis į kitą duobutę.



Kokia krypčių seka sukiojant tošį pavyks rutuliuką išridenti lauk?

Nuo tošies žaislo, kuriame yra tunelių ir duobučių sistema, reikia pereiti prie abstraktaus modelio. Pastebime, kad iš kiekvienos duobutės išeina du keliai: į kairę ir į dešinę. Vadinasi, galima taikyti dvejetainio medžio struktūrą. Medžio viršūnės – duobutės, o šakos – tuneliai, po du tunelius iš kiekvienos duobutės. Dvejetainis medis – tai duomenų (informacijos) modelis, kai duomenys aprašomi viršūnėmis ir šakomis ir iš kiekvienos viršūnės išeina ne daugiau kaip po dvi šakas.

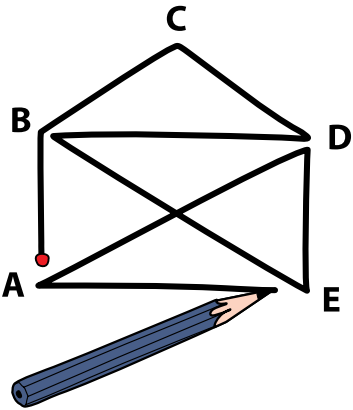
Nepakėlęs pieštuko ir tą pačią liniją brėždamas tik vieną kartą, gali nubraižyti tokį namelį:



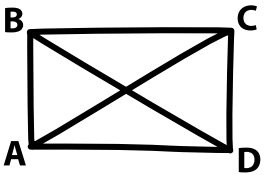
Šiuo atveju pradėta nuo taško A.
Ar gali pradėti nuo kitų taškų? Nuo kurių?

Grafu vadinama struktūra, kurią sudaro viršūnės (taškai) ir briaunos (linijos). Grafą, turintį lygiai 2 nelygines viršūnes (kai yra nelyginis briaunų, išeinančių iš viršūnės, skaičius), galima nubraižyti, neatitraukiant pieštuko nuo popieriaus ir nekartojant linijų, jei pradedama vienoje nelyginėje viršūnėje, o baigiama antroje. Tai, pavyzdžiui, optimalus maršrutas, kuriuo aplankomos visos kurio nors miesto ar krašto įžymios vietos (jei transportas visomis briaunomis vienodas), valomos gatvės ir pan.

Nepakėlęs pieštuko ir tą pačią liniją brėždamas tik vieną kartą, gali nubraižyti voką:

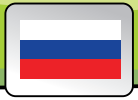


O dabar reikia nubraižyti uždarą voką.



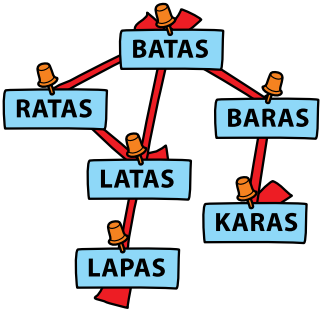
Kodėl pirmu atveju pavyksta, o antru – ne? Kaip paaiškintum?

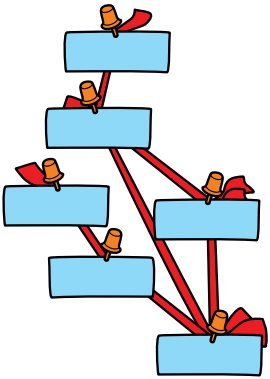
Informatikoje grafas yra viršūnių (taškų) ir briaunų (linijų) aibė. Grafą, kuris turi lygiai 2 nelygines viršūnes (kai yra nelyginis briaunų, išeinančių iš viršūnės, skaičius), galima nubraižyti neatitraukiant pieštuko nuo popieriaus ir nekartojant linijų, jei pradedama vienoje nelyginėje viršūnėje, o baigiama antroje. Tačiau grafo, kurio visos viršūnės nelyginės, negalima nubraižyti, neatitraukiant pieštuko nuo popieriaus ir nekartojant linijų. Reikia skaičiuoti, kiek briaunų išeina iš kiekvienos viršūnės.



ŽODŽIŲ GRANDINĖLĖS

Bebriukė surašė žodžius ant lentelių. Tada gumine juosta sujungė visas žodžių poras, kurios skiriasi tik viena raide (parodyta dešinėje).





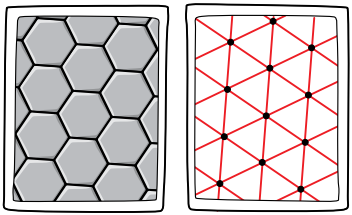
Išdykęs bebriukės broliukas ištrynė žodžius ir net korteles visiškai sumaišė (parodyta kairėje). Supratęs savo išdaigą, broliukas nusiminė, bet bebriukė jį ramino: „Nesijaudink, aš žinau, kaip teisingai sudėlioti žodžius. Tai nėra sunku.“

Kaip turi būti teisingai surašyti žodžiai lentelėse?

Bebriukė sukonstravo sistemą iš lentelių ir guminių juostų, kurią informatikai vadina grafu. Grafas – tai matematinė abstrakcija arba modelis, kurį sudaro taškai ir atkarpos. Atkarpos vadinamos briaunomis, taškai – viršūnėmis. Uždavinyje viršūnes atitinka lentelės, briaunos yra jas jungiančios juostos. Grafą galime „tampyti“, kaip tik norime.

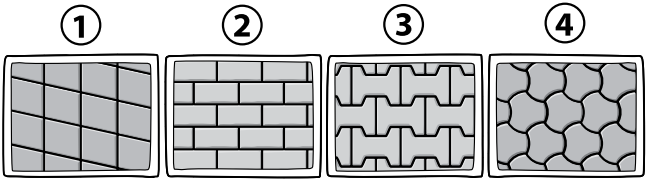
ŠALIGATVIO PLYTELĖS

Rimas nufotografavo šaligatvį prie savo namų ir nubraižė jo rašto schemą (parodyta dešinėje).



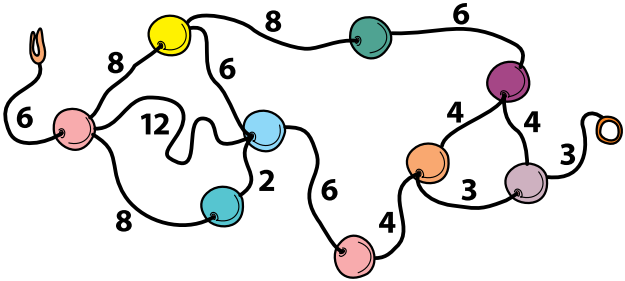
Taškas schemoje reiškia plytelę. Du taškus jungianti linija reiškia dvi kraštine besiliečiančias plyteles.
Vėliau Rimas vaikštinėjo po miestą ir fotografavo kitus šaligatvius. Grįžęs namo ir peržiūrėjęs nuotraukas, nustebo, kad visi raštai, išskyrus vieną, atitinka jo nubraižytą schemą.

Kuris iš pateiktų šaligatvių raštų neatitinka Rimo schemos?



Uždavinyje pateikta schema yra grafas, nors taip ir neįvardijama. Grafas – tai struktūra, kurią sudaro viršūnės ir briaunos. Grafas yra svarbi informatikos priemonė duomenims aprašyti ir objektų ryšiams nusakyti, pavyzdžiui, telekomunikacijų tinklams arba organizacijų struktūroms pavaizduoti. Modelio (grafo) sudarymas ar atpažinimas, turint realų objektą (šaligatvio plyteles), yra dažnas informatikos uždavinys.

Bebriukė Elena veriasi karolius. Baigusi suabejojo, ar karoliai ne per trumpi. Skaičiai žymi siūlų ilgį tarp perlų. Sagtelės žiedeliai yra kairėje ir dešinėje.



Koks bus susegtų karolių ilgis?

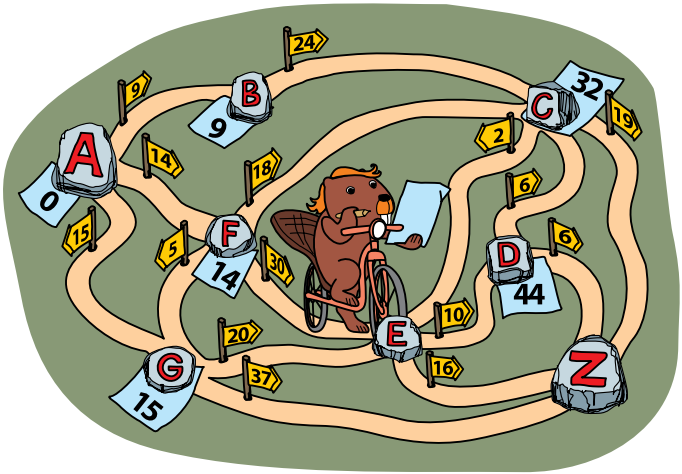
Trumpiausio kelio uždaviniai sprendžiami, ieškant trumpiausio kelio tarp dviejų grafo viršūnių (arba tinklo mazgų). Kelio ilgis priklauso nuo jį sudarančių atkarpų reikšmių sumos. Mažiausia kelio ilgio reikšmė ir yra trumpiausias kelias. Trumpiausiam keliui rasti sukurta daug algoritmų, vienas populiariausių – vadinamasis Deikstros (Dijkstros) algoritmas.

DVIRATININKĖ

B

Dviratininkė renkasi trumpiausią kelią nuo A iki Z. Dviračių takai yra tik vienos krypties, tačiau ji žino algoritmą, kaip rasti trumpiausią kelią. Kiekvienoje sankryžoje dviratininkė parašo užuominą (skaičių) ant lapo.

Kokį skaičių šiuo metu rašo dviratininkė?



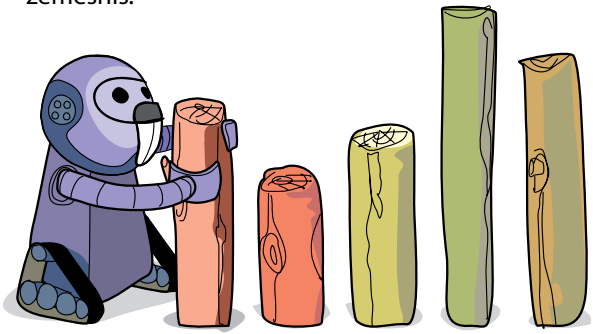
Esama efektyvių algoritmų, padedančių rasti reikiamą kelią tarp grafo viršūnių. Akivaizdu, kad trumpiausio kelio algoritmai yra naudingi, planuojant maršrutus. Pavyzdžiui, vežėjų įmonių svetainėse siūlomi trumpiausio kelio algoritmu apskaičiuoti maršrutai, kaip kuo greičiau nuvykti nuo vienos stoties iki kitos. Algoritmas, kuriuo šiame uždavinyje buvo ieškoma trumpiausio kelio, vadinamas Deikstros (Dijkstros) algoritmu. Kai kurių grafų briaunos gali turėti kryptį arba svorį.

PALYGINK IR SUKEISK

Bebras moka atlikti šias komandas:

1 komanda: palygink du gretimus kuolus ir nustatyk, kuris iš jų žemesnis;

2 komanda: sukeisk kuolus vietomis, jei antrasis žemesnis.

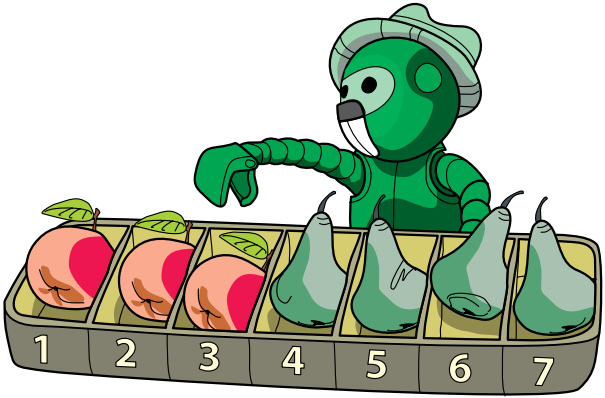


Robotas lygina ir keičia vietomis kuolus iš eilės: pirmą su antru, tada antrą su trečiu ir t. t. Baigęs eilę, robotas pradeda vėl iš kairės.

Kiek komandų atliks robotas, kol surikiuos visą kuolų eilę?

Duomenų rikiavimas yra viena iš svarbiausių informatikos temų. Surikiuotų duomenų paieška atliekama daug sparčiau. Pavyzdžiui, miesto gyventojų pavardės duomenų banke surikiuotos abėcėlės tvarka, todėl informacija apie bet kurį miesto gyventoją pateikiama labai greitai. Šiame uždavinyje rikiuojama didėjančiai, tai yra nuo mažiausio iki didžiausio elemento. Lygių duomenų išdėstymo tvarka nesvarbi. Kai reikia matematinio tikslumo, tada vietoj „rikiuoti didėjančiai“ sakoma „rikiuoti nemažėjančiai“.

Robotas bebras stovi prie ilgo lovelio su obuoliais ir kriaušėmis:



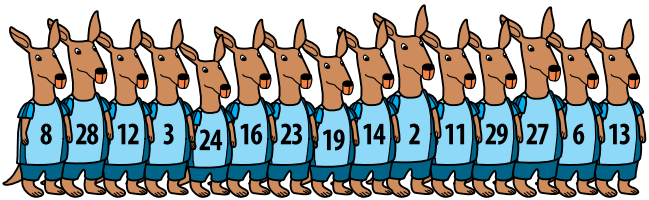
Robotas gali apkeisti vietomis du gretimus vaisius.
Kiek mažiausiai kartų turės apkeisti vaisius robotas, kad visos kriaušės atsidurtų pradžioje, o obuoliai – gale?

Sprendžiant šį uždavinį, tenka rikiuoti duomenis. Čia taikomas vienas paprasčiausių būdų, vadinamasis burbulinio rikiavimo algoritmas. Algoritmo principas – nuosekliai iš eilės peržiūrėti gretimų elementų poras, prireikus elementus sukeisti, perkeliant reikiamą elementą arčiau pradžios. Algoritmas dažniausiai taikomas skaičiams išdėstyti didėjančiai arba mažėjančiai. Algoritmo veikimo principas primena virimo procesą, kai oro burbuliukai kyla į paviršių, dėl to algoritmas taip ir vadinamas – burbulinio rikiavimo.

ŽAIDĖJŲ RIKIAVIMAS

Dvi komandos, kuriose yra po 15 žaidėjų, vilki marškinėlius su numeriais.
Pirmosios komandos žaidėjai išsirikiavę pagal marškinėlių numerius:

Antrosios komandos žaidėjai sustoję bet kaip:



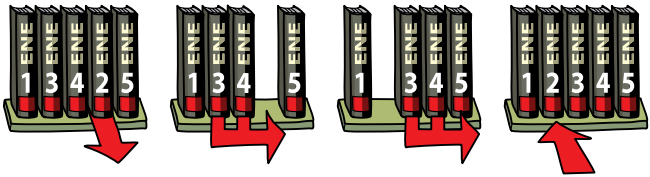
Kiek pirmosios komandos žaidėjų turi tokius pat numerius kaip antrosios?

Paieška sąrašuose yra vienas pagrindinių informatikos uždavinių. Sąrašų rikiavimas yra būtinas informatiko įgūdis. Prieš atliekant paiešką, patartina nerikiuotą sąrašą surikiuoti, ypač jei teks ieškoti daug kartų. Surikiuoto sąrašo elemento paieška daug spartesnė nei nesurikiuoto. Šiuo uždaviniu parodoma, kad kur kas sparčiau randami antrosios komandos žaidėjų numeriai pirmojoje komandoje, o ne atvirkščiai. Kuo sąrašai ilgesni, tuo labiau skiriasi sprendimo laikas. Jei turėtume tūkstančio numerių sąrašą, paieška surikiuotame sąraše būtų maždaug 50 kartų spartesnė nei nesurikiuotame.

Bibliotekininkas kas vakarą turi surikiuoti skaitytojų sumaišytus enciklopedijų tomus didėjimo tvarka. Jis tai norėtų atlikti per kiek galima mažiau žingsnių.
Vieną žingsnį sudaro trys veiksmai:

a – paimti iš lentynos vieną knygą,
b – pastumti kita ranka keletą knygų į kairę arba į dešinę,
c – padėti laikomą knygą į atsiradusią tuščią vietą.

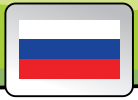
Štai kaip vienu žingsniu galima surikiuoti penkis tomus:



Vieną dieną bibliotekininkas rado 9 sumaišytus enciklopedijų tomus.

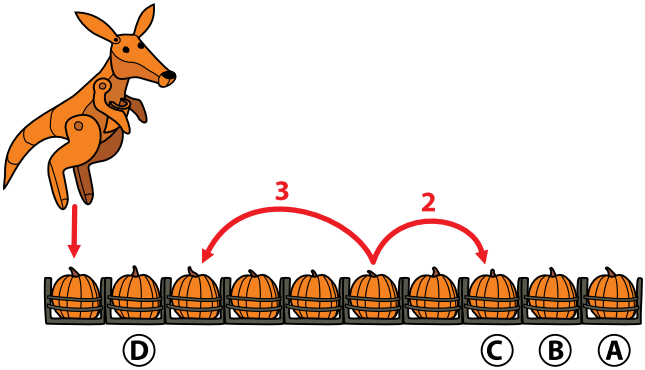
Kiek mažiausiai žingsnių reikės, norint išrikiuoti visus šiuos tomus?

Sprendžiant šį uždavinį, susipažįstama su duomenų rikiavimu ir imama mąstyti apie duomenų sekų dalis – posekius. Reikia surasti didžiausią didėjantį tomų numerių posekį ir šio posekio tomus palikti lentynoje. Uždavinio formuluotė pateikia tam tikrų sąlygų, informatikoje vadinamų ribojimais. Programuojant būtina atsižvelgti į visus nurodytus ribojimus ir jų griežtai laikytis.



ŠOKLIOJI KENGŪRA

Eilėje 10 dėžių, kiekvienoje kurių – po moliūgą.
Bebras pagamino šoklųjį robotą – kengūrą.
Kiekvienu šuoliu kengūra gali šokti arba per dvi dėžes į priekį, arba per tris dėžes atgal. Prišokusi pasiima moliūgą. Kengūra šoka tik prie dėžių su moliūgais. Pradeda nuo kairiausiojo moliūgo.



Jei kengūra surinks visus moliūgus, kurį paims paskutinį?

Vienas iš šio uždavinio sprendimo būdų – kengūros šuolių modeliavimas ir visų galimų dėžių sekų analizė. Ieškoma sekos, atitinkančios galimus šuolius. Tai vadinamasis visiškas perrinkimas, jam atlikti reikia labai daug laiko. Perrinkimo uždaviniuose svarbus sisteminis tyrimas, sisteminė galimybių analizė.

ALKANAS BEBRAS

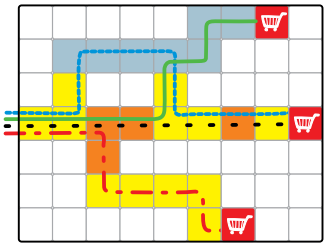
Bebras išalko, todėl nori vykti į parduotuvę riešutų. Kad būtų greičiau, nusprendė važiuoti dviračiu. Bebras pasirengė schemą ir pabandė išsirinkti tinkamiausią maršrutą. Netoliese yra trys parduotuvės ir keletas kelių į jas nuvykti. Bebras žino, kad yra lengvesnių ir sunkesnių kelių atkarpų, todėl tas atkarpas pasižymėjo skirtingos spalvos kvadratėliais. Pateikta schema ir paaiškinimai:


parduotuvė


keliavimo laikas: 1 minutė


keliavimo laikas: 2 minutės


keliavimo laikas: 5 minutės



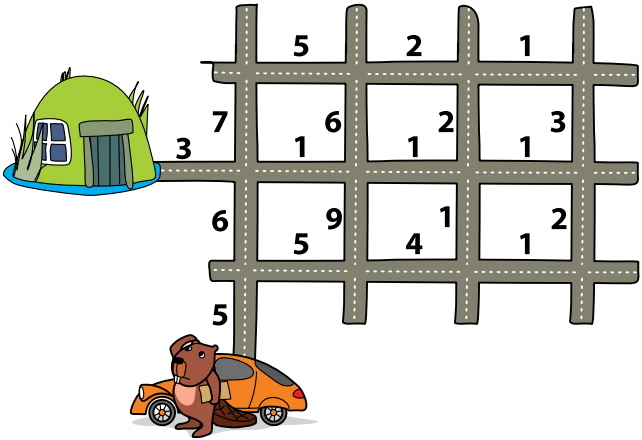
Kuriuo keliu Bebras greičiausiai pasieks parduotuvę?

1. Mėlynuoju keliu (punktyrinė linija)
2. Žaliuoju keliu (ištisinė linija)
3. Juoduoju keliu (taškinė linija)
4. Raudonuoju keliu (brūkšninė-taškinė linija)

Sistemingas galimybių perrinkimas reikalingas, sprendžiant informatikos uždavinius. Visiško perrinkimo metodu tikrinami visi galimi atvejai. Deja, tai tinka, tik kai galimybių skaičius baigtinis ir kai tų galimybių nėra labai daug. Kitaip skaičiavimai net ir galingais kompiuteriais gali trukti šimtmečius. Perrinkimą stengiamasi optimizuoti, įvertinant situaciją ir atmetant akivaizdžiai netinkamus sprendimus – tai vadinama ribotu perrinkimu.

SUKTI KAIRĖN NEGALIMA!

Mieste – transporto spūstys. Mašinos rieda taip arti viena kitos, kad net negalima pasukti kairėn…
Tėtis, baigęs darbą, skuba automobiliu namo. Pateiktas gatvių planas. Skaičius šalia gatvės rodo, kiek minučių trunka važiuoti šia gatve. Greičiausio kelio namo tėtis ieško GPS įrenginiu.

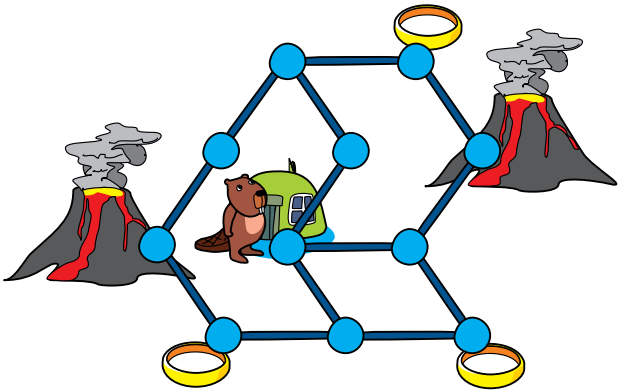


Kiek mažiausiai laiko tėtis užtruks, važiuodamas namo?

Sprendimą, kuriuo keliu važiuoti, reikia priimti kiekvienoje sankryžoje, remiantis kriterijais: laiku (reikalingu atkarpai įveikti) ir kryptimi (važiuoti tiesiai ar dešinėn). Sistemingas galimybių perrinkimas reikalingas, sprendžiant informatikos uždavinius. Visišką perrinkimą, kai tikrinami visi galimi atvejai, stengiamasi optimizuoti, įvertinant situaciją ir atmetant akivaizdžiai netinkamus sprendimus – tai vadinamasis ribotas perrinkimas.

BEBRAS PRAMOGAUJA

Bebras užsimanė pramogų. Išėjęs iš namų, jis turi surinkti tris žiedus ir įmesti juos į ugnikalnį. Tada grįžta namo. Bebras turi žemėlapį. Kiekvienas kelias (atkarpa tarp dviejų taškų) įveikiamas per vieną dieną. Tuo pačiu keliu bebras gali eiti kad ir kelis kartus, o kai kuriais keliais iš viso nevaikšto.



Kiek mažiausiai dienų prireiks bebrui, kad surinktų tris žiedus, tada įmestų juos į vieną iš ugnikalnių ir grįžtų namo?

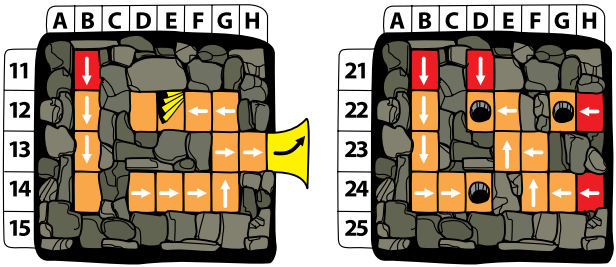
Universalus metodas – visų galimų variantų paieška ir apdorojimas, vadinamasis visiškas perrinkimas. Kadangi realiuose uždaviniuose duomenų būna labai daug, visiško perrinkimo metodas netinka, nes reikėtų šimtmečių įvertinti visus galimus sprendimų būdus. Sprendimus siekiama optimizuoti, atmetant akivaizdžiai neoptimalius sprendinius (vadinamasis ribotas perrinkimas), svarbu tik nepamesti teisingų variantų. Tam būtina sisteminė galimybių analizė.

LABIRINTAS

Bebras pastatė dviejų aukštų labirinto modelį. Robotė skruzdė ropinėja akmeniniame labirinte pagal tam tikras taisykles. Kai atsistoja:

– ant rodyklės langelio – paropoja vienu langeliu rodyklės kryptimi;
– ant duobės (apskritimo) langelio – nusileidžia į po ta duobe esantį langelį;
– ant laiptų langelio – kyla į antro aukšto langelį;
– ant tuščio langelio – sustoja (baigia darbą).

Pirmame aukšte yra vienas įėjimo langelis – B11 ir vienas išėjimo langelis – H13. Antrame aukšte yra keturi įėjimo langeliai: B21, D21, H22, H24.



Pro kurį įėjimo langelį įropojusi skruzdė sėkmingai įveiks kelią ir išropos iš labirinto?

Čia pateikiamas veiksmų algoritmas. Jį reikia įgyvendinti, atsižvelgiant į nurodytas sąlygas – reikalavimą, kad skruzdė išropotų iš labirinto. Universalus metodas – visų galimų variantų paieška, vadinamasis visiškas perrinkimas. Šį metodą siekiama optimizuoti, atmetant akivaizdžiai neoptimalias šakas, svarbu tik nepamesti teisingų variantų. Tobulinami išsamios baigtinės paieškos įgūdžiai.

DVI KUPRINĖS

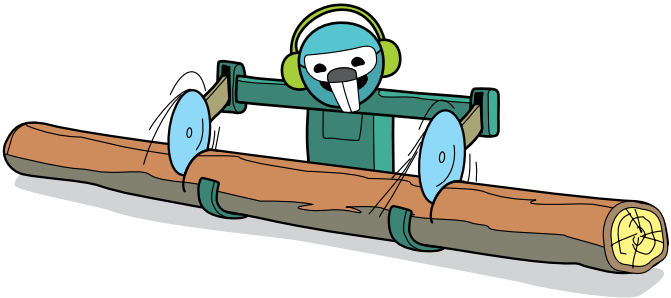
B

Du bebrai ruošiasi į kelionę. Jie krauna daiktus į kuprines.

Kuprinės svoris negali viršyti 8 kg.
Kaip paskirstyti daiktus į kuprines, kad kuo daugiau tilptų?

Daugelyje gyvenimo sričių reikia optimaliai pasirinkti veiklas ar atlikti sprendimus. Iki šiol sukurta daug optimizavimo algoritmų. Vienas iš jų – godusis algoritmas (angl. *greedy*), kai kiekvienu žingsniu imamas geriausias (didžiausias, sunkiausias, pelningiausias) komponentas. Deja, daugeliu atvejų godumo principas nepateikia optimalaus sprendinio ir tenka ieškoti tobulesnės sprendimo strategijos.

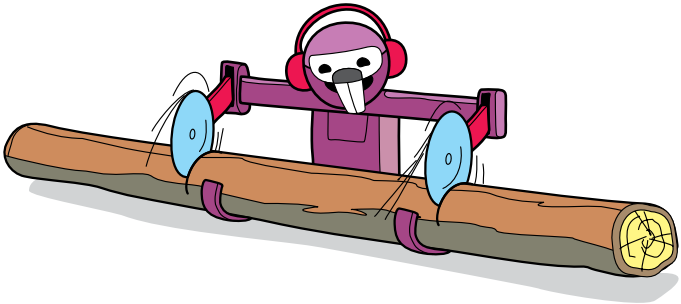
Robotas bebras per 1 sekundę supjausto rąstą į 3 dalis:



Kiek reikia robotų, norint rąstą supjaustyti į 9 dalis per 1 sekundę?

Kelis darbus galima atlikti vienu ir tuo pačiu metu, kitaip tariant, lygiagrečiai. Šitaip dirba kompiuterio procesorius – lygiagrečiai atlieka daugelį veiksmų. Iš tiesų lygiagretumas – savybė, kurią turintys keli procesai darbus atlieka vienu metu, naudodami skirtingus išteklius (kiekvienas robotas bebras turi savo pjūklą, nors visi pjauna vieną rąstą, be to, jie gali atsistoti taip, kad nekliudytų vienas kitam).

Robotas bebras per 1 sekundę supjausto rąstą į 3 dalis:



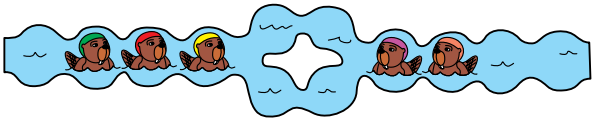
Kiek reikia robotų, norint rąstą supjaustyti į 9 dalis per 2 sekundes?

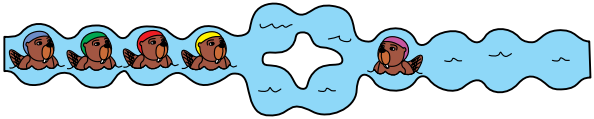
Kompiuterio procesorius vienus darbus atlieka nuosekliai – vieną po kito, o daugelį kitų darbų atlieka lygiagrečiai – visus tuo pačiu metu. Darbas atliekamas sparčiau, kai procesai vyksta lygiagrečiai. Tačiau tokiu atveju reikia daugiau išteklių (pavyzdžiui, pjūklų). Šiame uždavinyje derinami nuoseklieji ir lygiagretieji procesai.

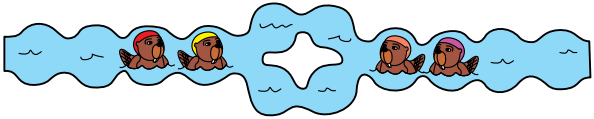
BEBRUPĖ

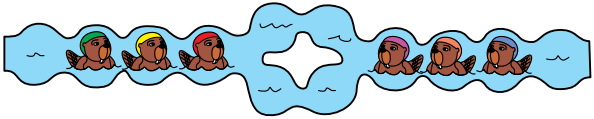
Bebrijoje teka siaura upė Bebrupė. Bebras nenori plaukti ja atbulas, todėl Bebrupėje iškasė atšakų, kad galėtų prasilenkti su kitu bebru. Vienoje atšakoje gali tilpti tik du bebrai.

Kada bebrai negalės prasilenkti?

A 

B 

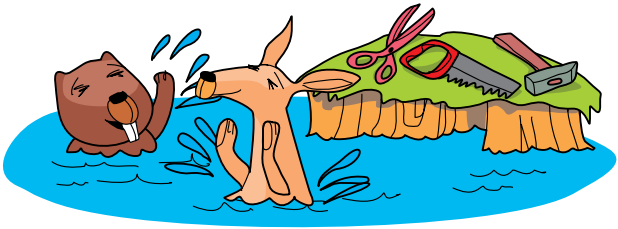
C 

D 

Situacija, kai kiekvienas sąveikaujančios procesų aibės procesas laukia įvykio, kurį gali pateikti tik tos pačios aibės kitas procesas, vadinama aklaviete. Kai programoms reikia bendrų išteklių, procesų blokavimas yra svarbus veiksnys, siekiant išvengti aklavietės. Bendras išteklių naudojimas ir aklavietės vengimas yra svarbios informatikos sąvokos.

BENDRI ĮRANKIAI

Bebras ir kengūra gamina žaislus, naudodamiesi vienu iš šių įrankių: plaktuku, žirklėmis arba pjūklu. Įrankiai laikomi saloje. Kuriam prireikia įrankio, tas jį ir pasiima. Jei reikiamo įrankio nėra saloje, laukiama, kol įrankis bus grąžintas, o turimas įrankis nepadedamas.
Kartais vienu metu abiem reikia įrankio, kurio nėra saloje. Jei taip nutinka, draugai nutraukia darbą ir eina maudytis.



Kurioje iš šių situacijų bebras ir kengūra tikrai eis maudytis?

A. Bebras turi ir jam reikia . Kengūra turi ir jai reikia .

B. Bebras turi ir jam reikia . Kengūra turi ir jai reikia .

C. Bebras turi ir jam reikia . Kengūra turi ir jai reikia .

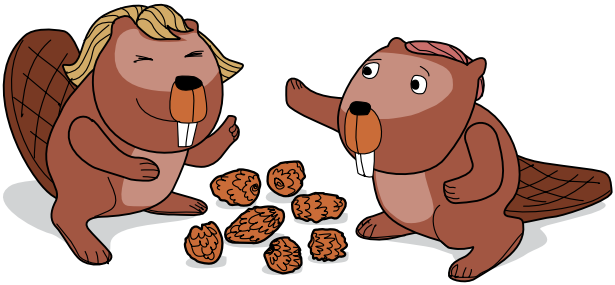
D. Bebras turi ir jam reikia . Kengūra turi ir jai reikia .

Įrankio paėmimas – tai proceso blokavimas. Situacija, kai kiekvienas sąveikaujančios procesų aibės procesas laukia įvykio, kurį gali pateikti tik tos pačios aibės kitas procesas, vadinama aklaviete. Kai programoms reikia bendrų išteklių, procesų blokavimas yra svarbus veiksnys, siekiant išvengti aklavietės. Bendras išteklių naudojimas ir aklavietės vengimas yra svarbios informatikos sąvokos.

PASKUTINIS KANKORĖŽIS

Bebrai Alė ir Benas yra kankorėžių žaidimo čempionai, jie abu puikiai žaidžia. Taisyklės tokios: paeiliui ima vieną arba du kankorėžius. Pralaimi tas, kuris paima paskutinį kankorėžį.

Alė pradeda, bet ar ji gali laimėti?

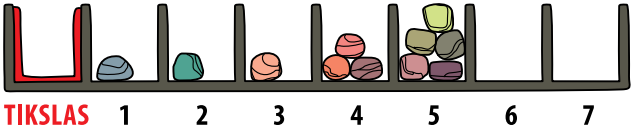


Žaidimai skiriasi taisyklėmis, sudėtingumu, tačiau yra daug ir panašumo. Žaidėjai daro ėjimus paeiliui. Kiekvienas žaidėjas iš visų galimų ėjimų stengiasi pasirinkti geriausią, t. y. tokį, kuris užtikrintų laimėjimą. Žaidėjai turi susidaryti žaidimo strategiją. Šį žaidimą reikėtų pradėti nagrinėti nuo pabaigos: vienas kankorėžis – pralaimima, du kankorėžiai – laimima, trys kankorėžiai – taip pat laimima ir t. t.

AKMENUKŲ PERDĖLIOJIMAS

B

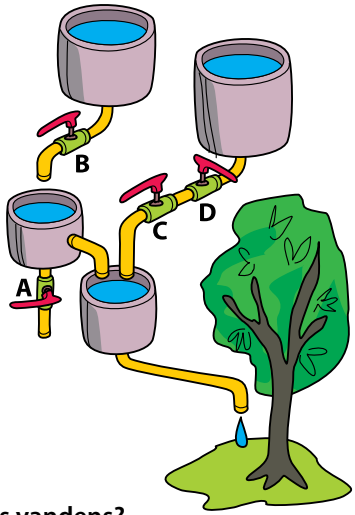
Žaidimo tikslas – perkelti visus akmenukus į kairįjį lovelį (užrašyta **Tikslas**). Reikia laikytis šios taisyklės: lovelį su numeriu nuo 1 iki 7 galima ištuštinti, jei jo akmenukų skaičius yra lygus lovelio numeriui. Tada šio lovelio akmenukai paskirstomi po vieną į visus kairiau esančius lovelius. Žaidimas laimimas, kai visi akmenukai sukeliami į lovelį **Tikslas**. Pateikiama situacija, kai žaidimas gali būti laimimas.



Kuria tvarka reikia tuštinti lovelius?

Sprendžiant šį uždavinį, lemiamą vaidmenį vaidina veiksmų seka – tai būdinga daugeliui informatikos problemų, ypač susijusioms su žaidimais ar procesų modeliavimu. Pateikiama laimėjimo strategija yra ne kas kita, kaip sąrašas algoritminių žingsnių, kuriuos atliekant žaidimas gali būti laimėtas, jei iš viso įmanoma laimėti. Reikia ieškoti optimalios strategijos, dažniausiai pradedama nagrinėti nuo žaidimo pabaigos.

Bebras sukonstravo vamzdžių sistemą savo obelaitei laistyti. Vamzdžių vožtuvai sužymėti raidėmis – kintamaisiais **A, B, C, D,** kurie gali įgyti tik vieną iš dviejų reikšmių: „tiesa“, angl. ***true*** arba „netiesa“, angl. ***false***. Jei vožtuvas atidarytas, tai jį žyminčio kintamojo reikšmė – ***true***. Jei vožtuvas uždarytas, tai jį žyminčio kintamojo reikšmė – ***false***.

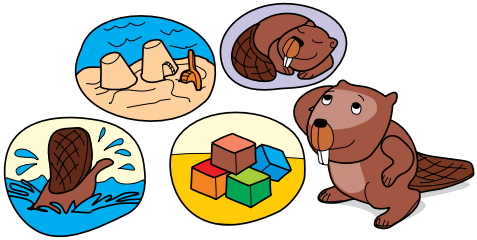


Kuriuo atveju obelaitė gaus vandens?

1. A = ***true***, B = ***true***, C = ***false***, D = ***false***
2. A = ***true***, B = ***false***, C = ***false***, D = ***true***
3. A = ***false***, B = ***true***, C = ***false***, D = ***false***
4. A = ***false***, B = ***false***, C = ***false***, D = ***true***

Kompiuterių programos atlieka veiksmus su realių objektų modeliais. Modelis yra tam tikra abstrakcija, supaprastintas realybės atspindys. Vamzdyno vožtuvai vaizduojami kintamaisiais, kurie įgyja reikšmes (*false* arba *true*) – tai abstrakcija, nes visos kitos galimos vožtuvų padėtys ignoruojamos (pavyzdžiui, pusiau uždarytas vožtuvas). Vandens srovės tekėjimas arba netekėjimas vamzdžiu gali būti skaičiuojamas logikos operacijomis. Taigi nuoseklus dviejų vožtuvų išdėstymas atitinka loginę IR operaciją (teisinga tik tuomet, kai abiejų vožtuvų reikšmės yra *true*), o lygiagretus – loginę ARBA operaciją (teisinga, kai bent vieno vožtuvo reikšmė yra *true*).

KĄ VEIKĖ BEBRAS?



Pagal oro sąlygas bebras sprendžia, ką veikti. Jis elgiasi pagal šias taisykles:

– Jei šiandien saulėta, bet vakar lijo, maudosi upėje.
– Jei šiandien saulėta, vakar taip pat buvo saulėta, žaidžia su smėliu upės pakrantėje.
– Jei šiandien lyja, bet vakar buvo saulėta, žaidžia su kaladėlėmis namie.
– Jei šiandien lyja ir vakar lijo, bebras miega.

Orai lapkričio 1–8 d. buvo tokie:

Data	1 d.	2 d.	3 d.	4 d.	5 d.	6 d.	7 d.	8 d.
Oro sąlygos								

Ką bebras veikė lapkričio 7-ąją?

Šį uždavinį galime spręsti dvejopai: atlikdami logines operacijas (iš tikrųjų pakanka loginės daugybos) arba taikydami baigtinio automato modelį. Baigtinis automatas – tai matematinė abstrakcija, kuria modeliuojama sistemos būsenų kaita, atsižvelgiant į ankstesnę būseną ir įėjimo signalus, kai būsenų ir galimų įėjimo signalų skaičius yra baigtinis. Įėjimo duomenys yra dienos (šiandien, vakar), o būsenos – koks oras (saulėta, lyja).

Du bebrai susirašinėja žinutėmis. Kad būtų patikimiau, žinutes koduoja savo sugalvotu būdu: paima vieną raidę, lentelėje suranda ją atitinkantį skaičių ir padaugina iš dviejų.

A	1	H	8	O	25	V	37
B	2	I	9	P	26	W	38
C	3	J	15	U	27	X	39
D	4	K	16	R	28	Y	45
E	5	L	17	S	29	Z	46
F	6	M	18	T	35	!	47
G	7	N	19	U	36	?	48

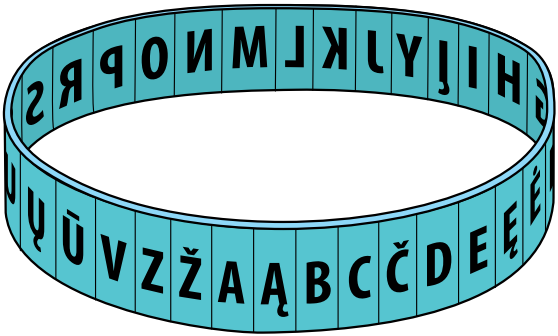
Taip žinutės tekstas pakeičiamas skaičių seka, pavyzdžiui, žodis BEBRAS būtų užkoduotas šitaip: 410456258.

Kaip būtų užkoduotas žodis INFORMATIKA?

Kodavimas – veiksmas, kuriuo ženklui suteikiamas kodas (sutartinis ženklas). Koduojama dėl įvairių priežasčių: patogesnio informacijos vartojimo, techninių galimybių, informacijos apsaugos (kad neperskaitytų pašaliniai). Kai kalbama apie duomenų apsaugą, tikslingiau vartoti šifravimo sąvokas (šifras, šifravimas).

ŠIFRAS

Bebrai siunčia žinutes, užkodavę šifru: kiekviena abėcėlės raidė perstumiama per dvi raides.
Pavyzdžiui, A->B, B->Č, Ž->Ą.



Gauta žinutė: ĖYPBO SBT MĖPIZŠBT
Ką parašė bebras?

Duomenys slepiami užšifruojant – perkoduojant tuos pačius duomenis kitokiais ženklais pagal tam tikras taisykles. Kai norima perskaityti pradinius duomenis, reikia juos iššifruoti. Mokslas, tiriantis informacijos užšifravimo ir iššifravimo metodus, vadinamas kriptografija. Šiais laikais kriptografija plačiai naudojama slaptai informacijai saugoti ar siųsti atviraisiais tinklais. Perstūmimo per porą raidžių metodas žinomas labai seniai, jis vadinamas Cezario šifru.

INFORMATIKOS KONCEPTAI, SLYPINTYS UŽDAVINIUOSE

1	Informacijos analizė, diagrama
2	Informacijos analizė, automatas
3	Dvejetainiai skaičiai
4	Dvejetainiai skaičiai
5	Dvejetainė logika, būsena
6	Dvejetainė logika, komanda
7	Komanda, būsena, ribojimas
8	Komanda, komandų seka
9	Komanda, parametras
10	Komanda, parametras
11	Algoritmas, tikslus vykdymas
12	Algoritmas, tikslus vykdymas
13	Algoritmas, programa
14	Programa, testavimas
15	Paprogramė, procedūra
16	Paprogramė, procedūra
17	Programa, tikrinimas
18	Programa, automatizavimas
19	Komanda, kartojimas
20	Komandų seka, kartojimas
21	Algoritmas, kartojimas
22	Kartojimas, sąlyga
23	Duomenų struktūra, dėklas, eilė
24	Dėklas, operacija
25	Dėklas, lygiagretūs procesai
26	Dėklas, įdėjimas ir išėmimas
27	Dvejetainis medis, komanda
28	Modelis, dvejetainis medis
29	Grafas, viršūnių skaičius
30	Grafas, viršūnių skaičius
31	Grafas, abstrahavimas
32	Grafas, abstrahavimas
33	Grafas, trumpiausias kelias
34	Grafas, trumpiausias kelias
35	Rikiavimas, komanda
36	Burbulinis rikiavimas
37	Rikiavimas, paieška
38	Rikiavimas, posekis
39	Visiškas perrinkimas
40	Visiškas ir ribotas perrinkimas
41	Perrinkimas optimizuojant
42	Perrinkimas optimizuojant
43	Optimizavimas, baigtinė paieška
44	Godusis algoritmas
45	Lygiagretūs veiksmai
46	Lygiagretūs veiksmai
47	Procesas, aklavietė
48	Proceso blokavimas, aklavietė
49	Žaidimas, laimėjimo strategija
50	Žaidimas, laimėjimo strategija
51	Loginės operacijos
52	Loginės operacijos, automatas
53	Kodavimas
54	Šifravimas
55	Informatikos konceptai
56	Žaidimo taisyklių pavyzdys

Kortelės apačioje pilkame fone pateikiamas paaiškinimas mokytojui ar tėveliams, tai neįeina į uždavinio sąlygą.

ŽAIDIMO PAVYZDYS

„Bebro“ kortelės skiriamos informatikai mokyti(s). Galimi įvairūs mokymo(si) būdai. Vienas jų pateikiamas čia, daugiau rasite svetainėje **www.bebras.lt**

1. Mokiniai susiskirsto į grupes po 2–3. Sudaroma 12 porų (ar trejetų). Mokytojas paaiškina žaidimo taisykles. (5 min.)

2. Kiekviena pora gauna vieną kortelę. (2 min.)

3. Darbas dviese (ar trise): mokiniai drauge sprendžia abu kortelės uždavinius (10 min.), jų sprendimus ir atsakymus užrašo į lapą.

4. Darbas grupėje: dvi poros ar trejetai sudaro grupę. Aiškina išspręstus uždavinius vieni kitiems. Grupė išsirenka vieną įdomiausią uždavinį ir pasiruošia jo sprendimą pristatyti visai klasei. (10 min.)

5. Kiekviena grupė pristato po vieną išsirinktą uždavinį ir jo sprendimą. (6 x 2 min. = 12 min.)

6. Klausimai ir papildomi paaiškinimai. (6 min.)
Laikas: 45 min.

Reikmenys: 12 „Bebro“ kortelių, popieriaus lapas, rašymo priemonė.