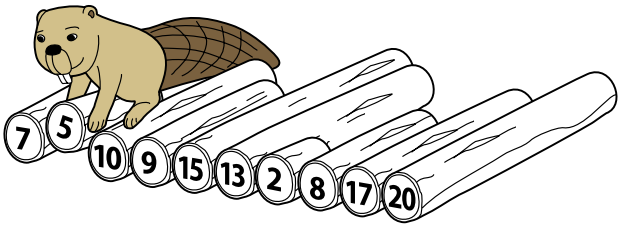
1 RĄSTŲ RIKIAVIMAS
KEITIMŲ ALGORITMAS

Bebras nori surikiuoti rąstus nuo trumpiausio iki ilgiausio. Rikiavimas vyksta lyginant du gretimus rąstus. Pradedama nuo antrojo rąsto. Jei jis trumpesnis už pirmąjį, bebras sukeičia rąstus vietomis. Po to lygina paskesnį su prieš jį einančiu ir jei reikia, pakeičia vietomis. Rąstas keičiamas tol, kol atrandama jam tinkama vieta. Tai kartojama su kiekvienu rąstu tol, kol galiausiai visi rąstai yra surikiuojami pagal ilgį.

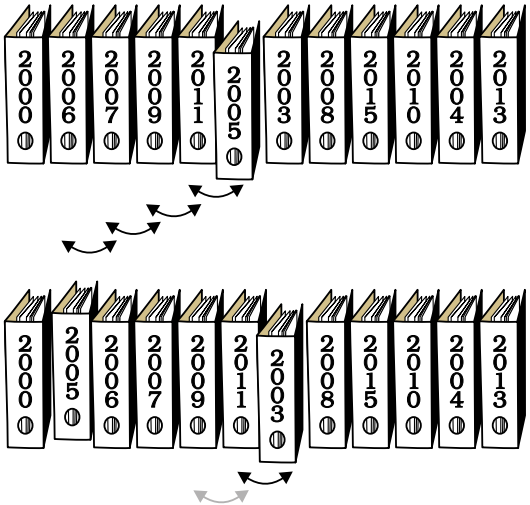


Kiek iš viso kartų rąstai bus keičiami vietomis?

Keitimų rikiavimo algoritmas yra vienas paprastesnių, tačiau jis efektyvus tik tada, kai rikiuojamų objektų nėra daug arba kai objektai beveik surikiuoti, t. y. kai žinoma, jog tik keletas objektų yra ne savo vietoje. Kitais atvejais rikiavimas užims labai daug laiko.

2 BYLŲ RIKIAVIMAS
KEITIMŲ ALGORITMAS

Biure ant bylų nugarėlių nurodyti metai, tačiau lentynoje jos sudėliotos padrikai. Biuro darbuotojo Bebro buvo paprašyta surikiuoti bylas didėjančia tvarka, nes taip patogiau ieškoti reikiamos bylos.



Bebras lygina dvi gretimas bylas: jei jų metai eina ne didėjančia tvarka, tuomet sukeičia vietomis. Byla keičiama tol, kol randama jai tinkama vieta. Tai kartojama su kiekviena byla, kol galiausiai visos bylos surikiuojamos pagal metus didėjančia tvarka.

Jeigu dviejų bylų sukeitimas užtrunka 5 sekundes, tai kiek laiko užtruks pateiktojo pavyzdžio bylų surikiavimas?

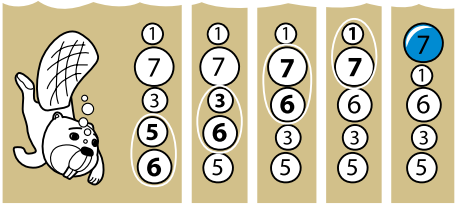
Algoritmas paprastas, tačiau, esant dideliam objektų skaičiui, - neefektyvus, užimantis labai daug laiko. Šiam uždaviniui jis gerai tinka, nes bylų pagal metus nebūna daug.

3 DIDELI BURBULAI KYLA

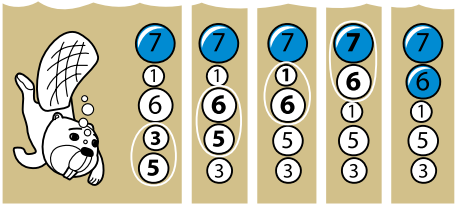
BURBULO RIKIAVIMO ALGORITMAS

Bebras žaidžia su vandens burbulais: rikiuoja juos nuo mažiausio (apačioje) iki didžiausio taikydamas burbulo rikiavimo algoritmą. Pradeda nuo apačios. Ima du burbulus ir lygina: jei didesnis yra apačioje, sukeičia vietomis ir vėl lygina su gretimu burbulu. Lyginama ir keičiama tol, kol burbulas atsiduria savo vietoje. Ir vėl kartojama iš pradžių (apačios).

1 žingsnis



2 žingsnis



Bebrui reikia 4 žingsnių, kad surikiuotų šio paveikslėlio burbulus (kiekvienu žingsniu atliekami 4 lyginimai).

Kaip atrodys ketvirtasis žingsnis?

Algoritmo veikimo principas primena virimo procesą, kai oro burbulai kyla į paviršių, dėl to jis ir yra taip vadinamas.

4 TRIUŠIAI IR VĖŽLIAI

BURBULO RIKIAVIMO ALGORITMAS

Burbulo rikiavimo metodas - tai algoritmas, kuris remiasi principu: iš eilės peržiūrėti gretimų elementų poras, o prireikus - elementus sukeisti, perkeliant mažesnįjį arčiau pradžios. Rikiuojant burbulo metodu, elementai yra perkeliami (juda) skirtingu greičiu:

- Elementas, kuris turi būti perkeltas į paskutiniąją poziciją, juda itin greitai, nes rikiuojama stumiant didesnįjį elementą į galą.
- Elementas, kuris turi būti perkeltas į pradžią, negali judėti greičiau nei vieną poziciją per ėjimą, nes toliau lyginamas ir keičiamas tik didesnysis elementas. Todėl elementai link pradžios juda lėtai.

Žinodami, kad vėžliai lėti, o triušiai greiti, **kuris piešinys tiksliausiai nusako burbulo rikiavimo metodo esmę?**

A.

B.

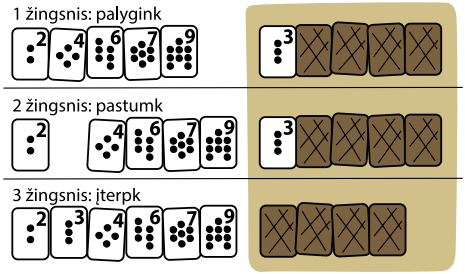
C.

D.

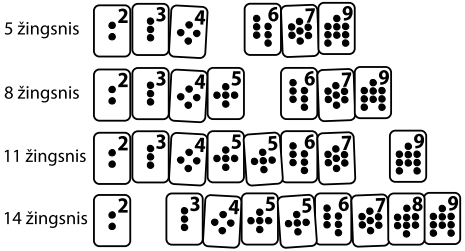
Burbulo rikiavimo metodas yra vienas iš paprastų, bet nelabai efektyvių rikiavimo algoritmų. Algoritmo principas – nuosekliai iš eilės peržiūrėti gretimų elementų poras, prireikus elementus sukeisti, perkeliant didesnįjį į viršų.

5 KORTELIŲ RIKIAVIMAS PERSTUMIANT
ĮTERPIMO ALGORITMAS

Bebras laiko žaidimo korteles rankoje, surikiavęs jas didėjančia tvarka pagal akučių skaičių. Po to ima iš kaladės naują kortelę ir iš eilės ją lygina su kortelėmis laikomomis rankoje tol, kol randa jai tinkamą vietą. Suradęs įterpia naują kortelę. Tokiu būdu turi visas korteles surikiuotas didėjančia tvarka pagal akučių skaičių ir vėl traukia kortelę. Procesas yra kartojamas iš pradžių. Tokia šio rikiavimo algoritmo esmė.



Keli tolesni žingsniai atrodytų šitaip:

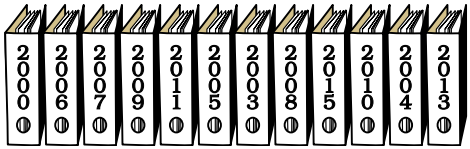


Nustatykite, kurios kortelės yra įterptos (žingsniuose 6, 9, 12 ir 15)?

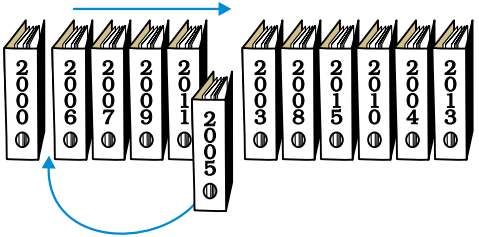
Kai žmonės rankiniu būdu surikiuoja korteles rankoje, dauguma jų naudoja panašų metodą į įterpimo rikiavimo algoritmą perstumiant.

6 BYLŲ RIKIAVIMAS PERSTUMIANT
ĮTERPIMO ALGORITMAS

Biure ant bylų nurodyti metai, tačiau lentynoje bylos sudėliotos padrikai. Kontoros darbuotojo Bebro buvo paprašyta surikiuoti bylas didėjančia tvarka, kad prireikus būtų galima greitai surasti norimą bylą.



Bebras nori surikiuoti bylas taikydamas įterpimo perstumiant algoritmą. Jis peržiūri bylas iš eilės, randa ne vietoje stovinčią ir ieško jai vietos. Radęs perstumia visus segtuvus į dešinę ir įdeda bylą.

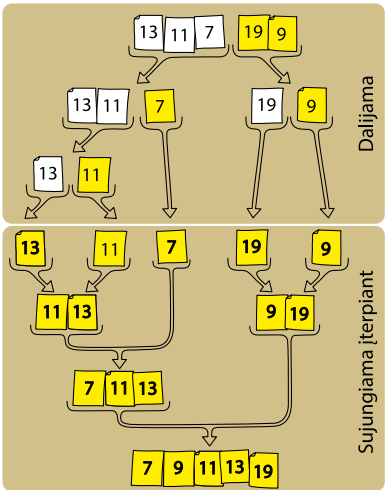


Jei perkelti bylą į tinkamą vietą užtrunka 5 sekundes (ištraukti, rasti vietą, pastumti bylas, įdėti), tai kiek laiko užtruks surikiuoti visus bylas lentynoje?

Perkėlimas perstumiant reiškia, kad bylą reikia perkelti suradus jai tinkamą vietą ir pastūmus kitas bylas.

7 KORTELIŲ DALIJIMAS IR JUNGIMAS
SĄLAJOS RIKIAVIMO ALGORITMAS

Bebrų šeima rikiuoja Bebro žaidimo korteles taikydami sąlajos rikiavimo algoritmą. Pirmiausia jie dalina kortelių rinkinį į dvi lygias ar beveik lygias dalis ir tai kartoja tol, kol lieka 1 kortelė. Tada lygina kiekvieną kortelę su gretimu kortelių rinkiniu, kad galėtų rikiuoti ir sujungti (sulieti) abu kortelių rinkinius į vieną. Algoritmas pateikiamas pavyzdžiu:



Pateiktas kortelių rinkinys:

5 17 11 1 3 7 8 20

Kuris atsakymas vaizduoja pirmus du kortelių rinkinius, kurie bus sujungti?

A. 5 17 ir 11 1

B. 1 5 11 17 ir 3 7 8 20

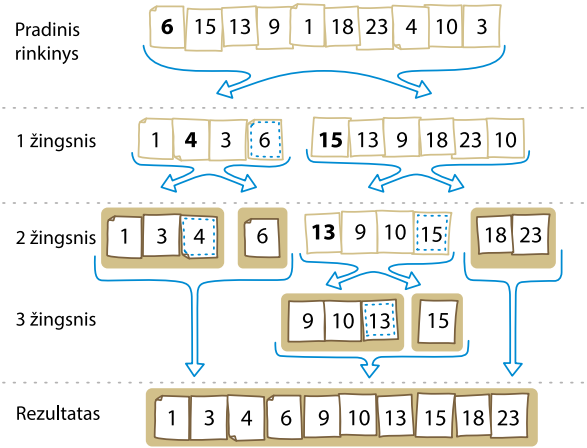
C. 5 ir 17

D. 3 ir 7

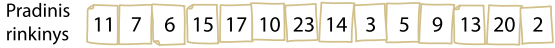
Sąlajos rikiavimo algoritmo principas yra skaidyti duomenis į dvi dalis, kiekvieną dalį atskirai surikiuotą ir po to sujungti.

8 DALIJAME IR JUNGIAME
GREITOJO RIKIAVIMO ALGORITMAS

Greitojo rikiavimo algoritmas yra itin efektyvus ir dažnai naudojamas metodas. Algoritmo esmė - elementų seka dalijama į dvi dalis pagal kurį nors požymį. Imkime pirmąją ne vietoje esančią kortelę (6) ir pagal ją padalinkime rinkinį į dvi dalis. Kiekviena dalis analogiškai dalijama į dvi dalis ir taip vyksta iki kol nebegalime dalyti. Pateikiame pavyzdį:



Duota kortelių seka:

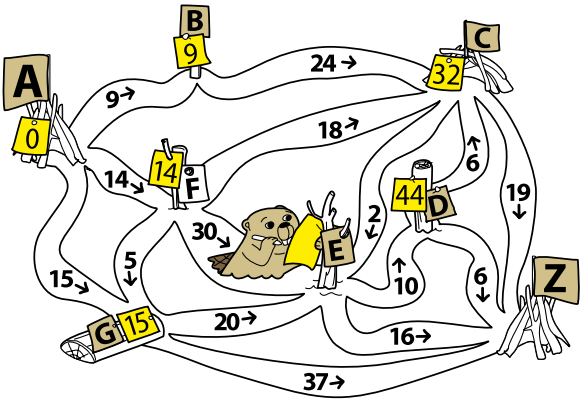


Kaip atrodys kortelių rinkinys trečiame žingsnyje?

Quicksort algorithm was developed by the famous British computer scientist C.A.R. Hoare in 1961 and it is still a commonly used algorithm for sorting.

9 BEBRAS TYRINĖJA
DIJKSTROS ALGORITMAS

Bebras tyrinėja maršrutus upėmis tarp urvų A (pradžia) ir Z (pabaiga). Nurodoma kiekvienos upės tėkmės kryptis ir ilgis.



Bebras plaukdamas iš A į Z ties kiekviena upių sankirta palieka lapelius su skaičiais.

Ką reiškia skaičius lapelyje?

A. Kelio ilgis nuo A iki šios upių sankirtos, plaukiant per kuo mažiau sankirtų.

B. Trumpiausio kelio ilgis nuo A iki šios upių sankirtos.

C. Kelio ilgis nuo A iki šios upių sankirtos, plaukiant tik kairėn visose sankirtose (jei tai yra įmanoma).

D. Ilgiausio kelio ilgis nuo A iki šios upių sankirtos.

Olandų informatiko Edgaro Dijkstros sukurtas algoritmas trumpiausiam keliui nuo vienos viršūnės iki kitos grafe su neneigiamais svoriais rasti. Paskelbtas 1958 metais.

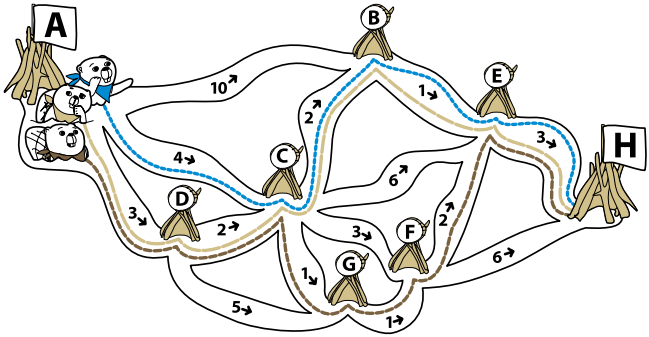
10 TRUMPIAUSIO KELIO IEŠKANT
DIJKSTROS ALGORITMAS

Trys bebrai keliavo iš urvo A į urvą H plaukdami tik pasroviui. Kiekvieno bebro kelias pažymėtas kita linija.

Vienas iš bebrų žino Dijkstros algoritmą trumpiausiam keliui rasti:

1. Pirmajam urvui (A) priskiriamas 0. Laikoma, jog A yra "dabartinis" urvas.
2. Skaičiuojami atstumai nuo "dabartinio" urvo iki kiekvieno jam gretimo urvo.
3. Šie atstumai lyginami tarpusavyje ir nustatomas mažiausias skaičius.
4. Mažiausią skaičių pelnęs urvas taps "dabartiniu", o prieš tai buvęs urvas pažymimas aplankytu - aplankytas urvas daugiau nebetikrinamas.
5. Grįžtama prie 2-ojo žingsnio.
6. Baigiama, kai pasiekiamas H urvas.

Kuris iš bebrų naudojo Dijkstros algoritmą?

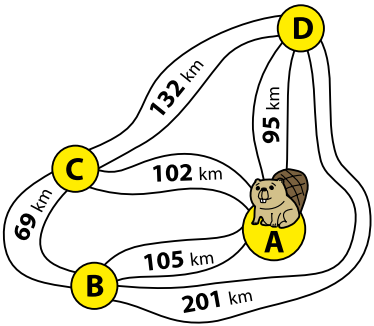


Dijkstros algoritmas yra vienas iš populiariausių metodų ieškant trumpiausio kelio. Jis taikomas sprendžiant dirbtinio intelekto problemas, atliekant paieškas MapQuest ar Google Maps svetainėse.

11 DAUG MARŠRUTŲ

KELIAUJANČIO PIRKLIO UŽDAVINYS

Bebriukė išplaukia iš savo buveinės A, turi aplankyti 3 kaimus B, C, D ir grįžti atgal į A.



Bebriukė taiko perrinkimo algoritmą: išrašo visus galimus maršrutus ir apskaičiuoja jų ilgius.

1. A C D B A $102 + 132 + 201 + 105 = 538$
2. A B D C A $105 + 201 + 132 + 102 = 538$
3. A D B C A $95 + 201 + 69 + 102 = 467$
4. A C B D A $102 + 69 + 201 + 95 = 467$

Perrinkimą galima apibrėžti kaip uždavinių sprendimo metodą, kai išbandomi visi galimi sprendiniai. Šiam uždaviniui (kuris vadinamas "Keliaujančio pirklio problema") perrinkimo algoritmas atrodo šitaip:

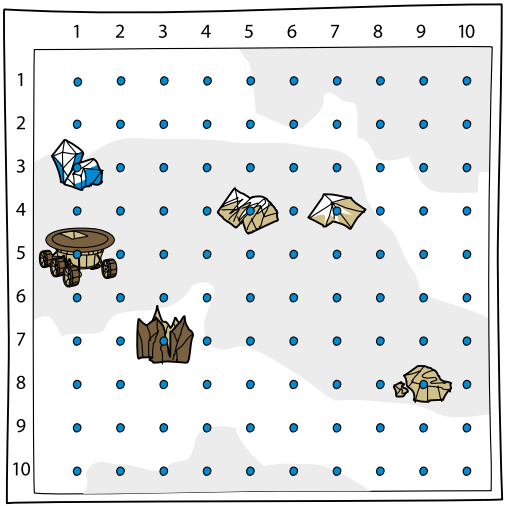
- Surašomi visi galimi Hamiltono ciklai: pradedama kurioje nors grafo viršūnėje, pereinama kiekvieną viršūnę lygiai vieną kartą ir vėl grįžtama į pradinę viršūnę.
- Sudedami kiekvieno Hamiltono ciklo visų briaunų svoriai.
- Iš visų ciklų išrenkamas mažiausio svorio ciklas - jis ir yra uždavinio sprendinys.

Tačiau pastebi, kad trūksta kelių maršrutų? Kurių?
Kokio ilgio trumpiausias maršrutas?

Keliaujančio pirklio problema yra vienas iš populiariausių ir labiausiai išnagrinėtų uždavinių kompiuterių moksle. Turint tam tikrą skaičių miestų ir kelionės iš vieno miesto į kitą kainas, reikia rasti pigiausią maršrutą, kad aplankius kiekvieną miestą, maršrutas baigtųsi pradiniame mieste.

12 MĖNULEIGIS
KELIAUJANČIO PIRKLIO UŽDAVINYS

Mėnuleigis buvo išsiųstas į mėnulį rinkti mineralų. Jį galime matyti žemėlapyje. Mėnuleigis, surinkęs visus mineralus, turi grįžti į tašką, kuriame stovi.



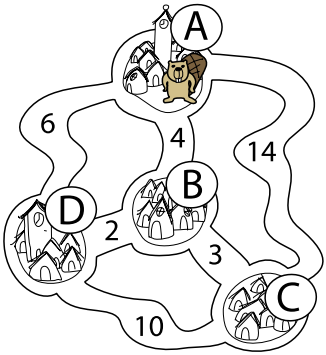
Atstumas tarp kiekvieno taško yra lygiai 1 km.

Nubraižykite grafiką, kuris parodytų šio uždavinio sprendimą. Nuveskite mėnuleigį taip, kad jis rastų trumpiausią (optimaliausią) maršrutą visiems mineralams surinkti.

Keliaujančio pirklio uždavinys yra grafų teorijos uždavinys, kai pilnajame svoriniame grafe ieškoma mažiausio svorio Hamiltono ciklo.

13 KAIMYNŲ LANKYMAS
ARTIMIAUSIOJO KAIMYNO ALGORITMAS

Bebras gyvena bebrinėje A. Jis nori aplankyti visus draugus, gyvenančius B, C ir D. Keliavimas upe mokamas - skaičiai nurodo kainą bebkoinais. Bebras išvyksta iš A, užsuka pas kiekvieną draugą (tik po vieną kartą) ir grįžta namo.



Bebras nori taikyti artimiausio kaimyno algoritmą:

1. Kelionės pradžia - A.
2. Pasirenkame pigiausią kelią į miestą, kuriame dar nesame buvę.
3. Kartojame 2-ąjį žingsnį tol, kol aplankysime visus miestus.
4. Iš paskutinio miesto grįžtame namo.

Štai 3 galimi maršrutai:
1. A B D C A = 4 + 2 + 10 + 14 = 30
2. A B C D A = 4 + 3 + 10 + 6 = 23
3. A C B D A = 14 + 3 + 2 + 6 = 25

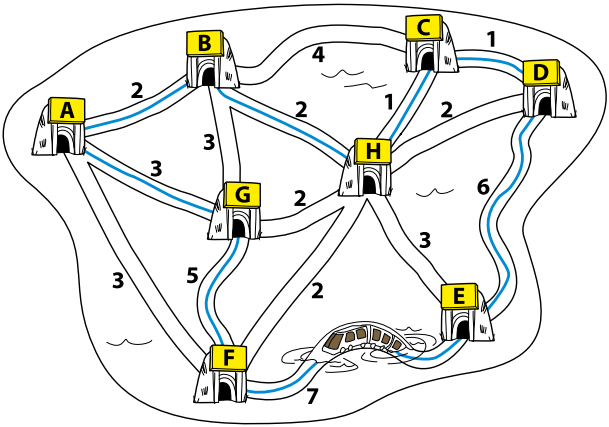
Kuris maršrutas gaunamas taikant artimiausio kaimyno metodą?

Artimiausiojo kaimyno algoritmą beveik visiškai apibūdina jo pavadinimas.

14 RENKANTIS PIGIAUSIĄ NE VISADA PIGU

ARTIMIAUSIOJO KAIMYNO ALGORITMAS

Bebras nori išbandyti naują povandeninį metro. Išvykęs iš namų stotelės A, jis nori apsilankyti kiekvienoje stotelėje, bet tik po vieną kartą! Bebras taiko artimiausio kaimyno algoritmą: kiekvienoje stotelėje pasirinks artimiausią stotelę iš likusių dar nekeliautų (trumpiausią atstumą kilometrais). Iš paskutinės stotelės grįš namo.



Bebro maršrutas pagal artimiausio kaimyno algoritmą būtų: ABHCDEFGA ir tai sudarytų 27 km.

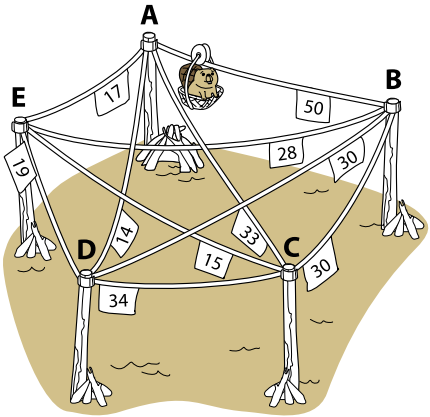
Įrodykite, kad šis bebro pasirinktas maršrutas nebus trumpiausias.

Artimiausiojo kaimyno metodo principas - pradedame nuo kažkurio taško, nuolat renkamės iš neaplankytų taškų patį „artimiausią" (su kuo mažesne verte jam pasiekti). Kai nebelieka neaplankytų taškų – grįžtame į pradinį.

15 PRAMOGŲ TINKLAS

KARTOTINIS ARTIMIAUSIOJO KAIMYNO ALGORITMAS

Pramogų centras siūlo smagią kelionę: iš taško A aplankyti visus taškus B, C, D, E - tik po vieną kartą - ir grįžti atgal į A. Nurodyta kiekvieno kelio kaina beurais. Bebriukė nori pasivažinėti, tačiau jai rūpi rasti pigų maršrutą. Jei taikytų artimiausio kaimyno algoritmą, tai keliautų šitaip: Iš A pigiausia judėti į D (14 b€), tada - į E (19 b€), toliau - į C (15 b€) ir iš čia - į B (taške E jau buvo). Visi taškai aplankyti ir galima grįžti į A. Ar tai pigiausias maršrutas? Deja, ne - įsitikinkite!



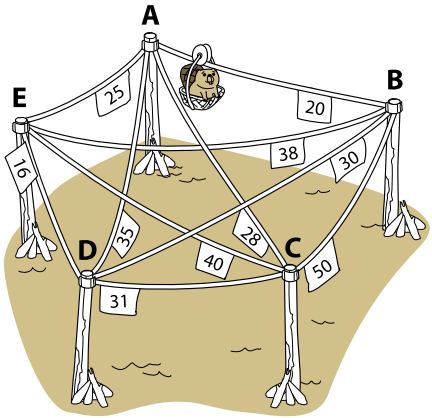
Pabandykime pritaikyti patobulinimą, vadinamąjį kartotinį artimiausiojo kaimyno algoritmą. Šio algoritmo esmė - apskaičiuoti maršrutų ilgius, kai pradžia imamas vis kitas taškas. Šiuo atveju reikia apskaičiuoti iš viso 5 maršrutus, palyginti ir išrinkti pigiausią.

Taigi artimiausiojo kaimyno algoritmą galima pradėti nuo bet kurio taško (nebūtinai iš pradžios), o gautą maršrutą po to galima perrašyti taip, kad būtų pradedama iš pradinio taško.

16 DAR KARTĄ PRAMOGŲ CENTRE
PIGIAUSIOS JUNGTIES ALGORITMAS

Pramogų centras siūlo kelionę: aplankyti visus taškus A, B, C, D, E - tik po vieną kartą - ir grįžti atgal į pradinį tašką A. Nurodyta kiekvienos jungties kaina beurais. Bebriukė nori pasivažinėti šiais keliais, tačiau jai rūpi rasti pigų maršrutą. Ji žino ir nori pasinaudoti pigiausios jungties algoritmu. Jo principas toks:

1. Imame pigiausią jungtį (jei tokių kelios – rinktis bet kurią). Pažymime.
2. Imame kitą pigiausią tinkamą jungtį ir ją pažymime. Tinkama jungtis yra kai: a) ji nepažymėta; b) ji neuždaro mažesnio ciklo; c) ji nėra trečia pažymėta jungtis, išeinanti iš to paties taško.
3. Kartojame 2-ąjį žingsnį, kol aplankysime visus taškus.



Padėkime bebriukei apskaičiuoti maršruto kainą pagal pigiausios jungties algoritmą.

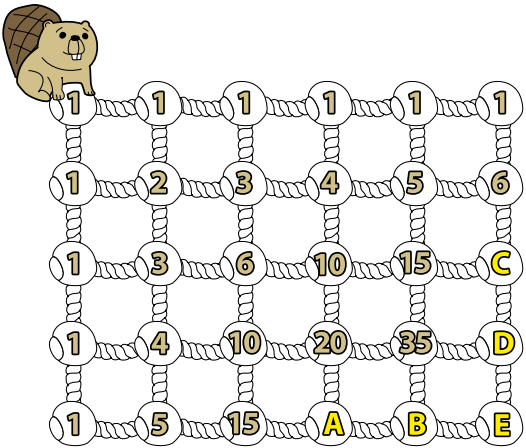
Artimiausiojo kaimyno algoritmas ir jo modifikacija - kartotinis artimiausiojo kaimyno metodas, abu sugeneruoja sprendimus, kurie nėra itin optimalūs. Tačiau šie metodai mums pateikia algoritmus, kurie lengvai suprantami, paprastai atliekami ir daugumai tipiškų uždavinių duoda apytikslį sprendinį su priimtinomis paklaidomis.

17 KIEK KELIŲ
DINAMINIS PROGRAMAVIMAS

Bebriukas nori išspręsti, atrodo, paprastą uždavinį: suskaičiuoti, kiek yra trumpiausių kelių, vedančių iš pradžios iki kiekvieno tinklo mazgo. Jei bebras kelius skaičiuos sistemingai, judėdamas iš kairės į dešinę ir leisdamasis po eilę žemyn ir kiekviename mazge užrašys suskaičiuotą kelių skaičių, vargo nebus.

Iš tiesų, šitokiu skaičiavimu paremtas dinaminio programavimo metodas - principu, jog uždavinys dalijamas į smulkesnes dalis, apskaičiuojami tų dalių rezultatai, užsirašomi (paprastai į lentelę) ir jais remiamasi toliau skaičiuojant.

Raskite kelių nuo pradžios iki mazgų A, B, C, D ir E ilgius.



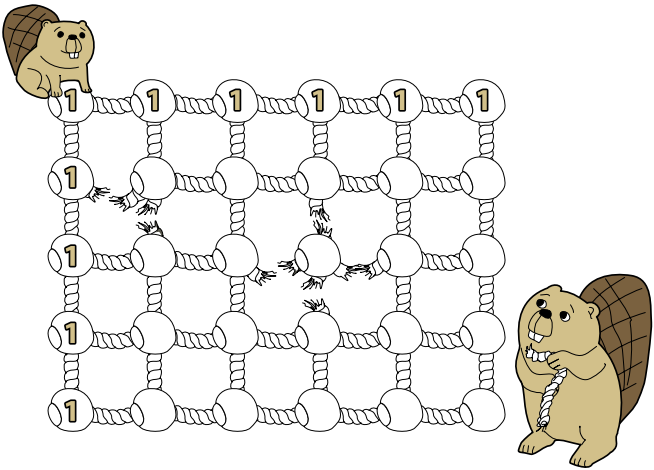
Dinaminis programavimas - tai programavimo metodas, taikomas spręsti sudėtiniams uždaviniams, kai juos galima skaidyti į dalis, apskaičiuoti tų dalių rezultatus ir juos taikyti tolesniems skaičiavimams.

18 SUARDYTI TAKAI

DINAMINIS PROGRAMAVIMAS

Bebrų gyvenvietė - urvai, sujungti taisyklingų takų tinkleliu. Bebras nori suskaičiuoti, kiek yra trumpiausių takų nuo pradžios iki kiekvieno urvo. Skaičiuojama sistemingai judant iš kairės į dešinę eilėmis žemyn. Deja, kai kurie takai suardyti. Bebras nori taikyti dinaminio programavimo metodą, kurio esmė - skaidyti uždavinį į smulkesnes dalis, išspręsti jas ir kaupti tarpinius rezultatus.

Padėkite Bebrui užpildyti tuščius skrituliukus.



Dinaminis programavimas yra efektyvus sprendinių radimo būdas, kurį galima pritaikyti kai kuriems, ypač optimizavimo, uždaviniams spręsti.

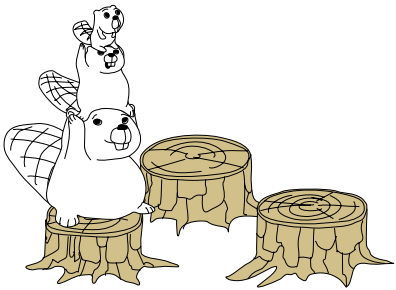
19 BEBRŲ AKROBATIKA

HANOJAUS BOKŠTAI

Bebrų trijulė - geri akrobatai. Jie rengia specialų pasirodymą: didžiausias bebras apačioje, virš jo - vidurinis, o viršuje - mažiausias - stovi ant vieno iš trijų kelmų ir visa trijulė turi peršokti ant kito kelmo pagal šias taisykles:

- šoka po vieną;
- turi stovėti tik ant kelmo arba vienas ant kito;
- didesnis bebras negali stovėti ant mažesniojo.

Vieno bebro peršokimas ant kito kelmo užtrunka 1 minutę. Štai kaip atrodytų trijų bebrų peršokimas:



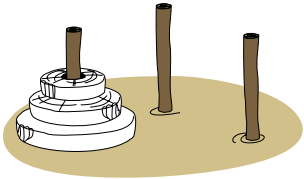
Kaip matome, bebrų trijulė šiam peršokimui sugaiš 7 minutes.

Kiek minučių užims keturių bebrų akrobatų peršokimas nuo vieno iš trijų kelmų ant kito pagal tas pačias taisykles?

Tai klasikinio Hanojaus bokštų uždavinio variantas. Šį uždavinį 1883 metais suformulavo prancūzų matematikas Edvardas Lukas.

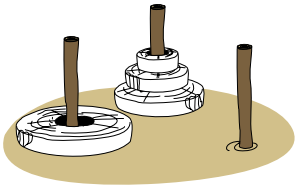
20 VĖL IR VĖL - REKURSIJA
HANOJAUS BOKŠTAI

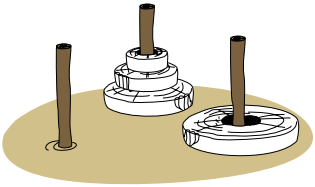
Hanojaus bokštai - senas uždavinys: yra trys strypai, o ant vieno iš jų užmauti diskai mažėjančia tvarka.



Reikia perkelti visus diskus ant vieno iš tuščių strypų pagal šias taisykles;
- perkeliama po vieną diską;
- diską galima užmauti ant tuščio strypo arba uždėti ant kito, didesnio, disko;
- didesnis diskas negali būti padėtas ant mažesnio.

Tarkime, vieno disko perkėlimas trunka 1 minutę. Jei turime tris diskus, uždavinys nėra sunkus, lengvai įsitikiname, kad jų perkėlimas užimtų 7 minutes. Mokėdami perkelti tris diskus, keturis galime perkelti šitaip: kai 3 diskai perkelti, tada paskutinį (didžiausią) diską perkeliame ant likusio tuščio strypo. O tada lieka perkelti 3 diskus, tik dabar ant strypo, kuriame uždėtas didžiausias diskas.





Taigi 4 diskų Hanojaus bokštą perkeltume per 15 min.
Kiek minučių užtruktų 6 diskų perkėlimas?

Hanojaus bokštai susiję su legenda: sakoma, kad kadaise Hanojuje prieš šventyklą buvo trys bokštai ir ant vieno iš jų buvo užmauti 64 diskai. Legenda byloja, kai vienuoliai perkels visus diskus (vieną diską perkelia per sekundę!), įvyks pasaulio pabaiga…

21 LENGVAS BŪDAS PIRMINIAMS SKAIČIAMS RASTI

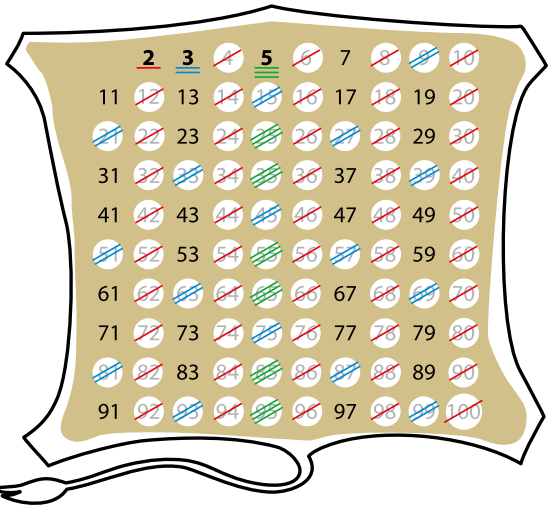
ERATOSTENO RĖTIS

Vienas seniausių žinomų algoritmų pirminiams skaičiams rasti vadinasi Eratosteno rėčiu.

1. Surašome visus natūrinius skaičius nuo 2 iki n (pavyzdžiui, n = 100).
2. Skaičius 2 pirminis, taigi išbraukiame visus didesnius skaičius, kurie dalijasi iš 2, pvz. 4, 6, 8 ...
3. Imame tolesnį neužbrauktą skaičių (pvz. 3) ir išbraukiame visus jo kartotinius.
4. Kartojame 3 žingsnį tol, kol galiausiai nelieka ko imti.

Tie skaičiai, kurie liko neužbraukti - yra visi pirminiai skaičiai mažesni už n.

Kiek pirminių skaičių yra tarp 2 ir 100?



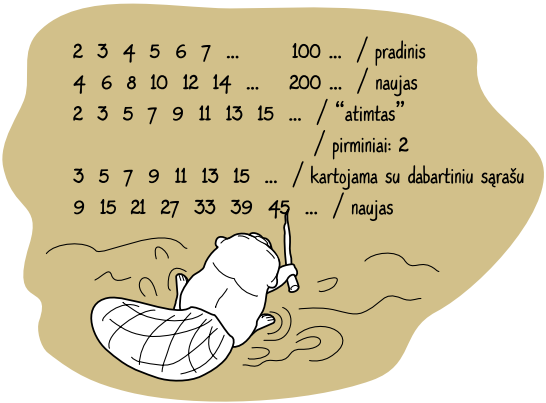
Graikų matematikas Eratostenas (~275-~195 m. pr. m. e.) pasiūlė paprastą metodą kaip rasti visus pirminius skaičius nuo 2 iki n. Jis ant papiruso surašė visus natūrinius skaičius nuo 2 iki 1000 ir pradurdavo sudėtinius skaičius. Tokiu būdu liko tarsi rėtis su „išsijotais" sudėtiniais skaičiais, o pirminiai skaičiai liko.

22 GREITESNIS BŪDAS PIRMINIAMS SKAIČIAMS RASTI

OILERIO RĖTIS

Patobulintas algoritmas pirminių skaičių suradimui veikia taip:

1. Sudarome pradinį skaičių sąrašą nuo 2 iki kiek norime; didžiausią skaičių vadinkime n.
2. Įtraukiame pirmąjį skaičių iš sąrašo į pirminių skaičių sąrašą, tuomet padarome naują sąrašą, kuriame kiekvienas skaičius, įskaitant ir pirmąjį, yra padauginamas iš pirmojo įtraukto skaičiaus.
3. Pašaliname naujo sąrašo skaičius iš pradinio skaičių sąrašo, tokiu būdu perkeldami į rezultatų sąrašą tik tuos skaičius, kurių nėra naujame sąraše. Padarome rezultatų sąrašą dabartiniu sąrašu. Pašaliname iš šio sąrašo pirmąjį skaičių.
4. Kartojame 2, 3 ir 4 žingsnius su dabartiniu sąrašu iki kol jis tampa tuščias.



Kiek kartų naujo sąrašo skaičius teks atimti iš dabartinio sąrašo skaičių, kad gautume pirmus 10 pirminių skaičių?

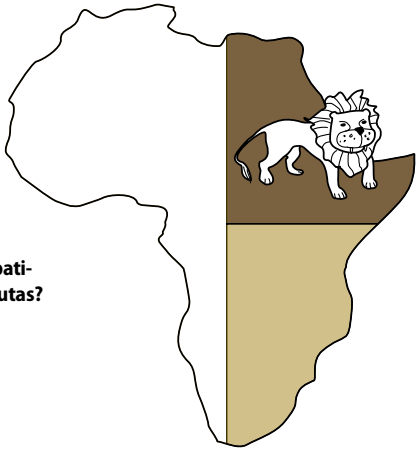
Senovinis Eratosteno rėtis nėra efektyvus, kai skaičiai gali būti dalinami daugiau nei vieną kartą; pavyzdžiui, 21 dalus iš 3 ir 7. Didysis šveicarų matematikas L. Oileris sugalvojo rėtį, kuris dalina kiekvieną skaičių tik tai vieną kartą.

23 PAGAUK LIŪTĄ
DVEJETAINĖ PAIEŠKA

Informatikai žino kaip pagauti liūtą Afrikoje! Štai, legendinis algoritmas yra paremtas dvejetaine paieškos technika.

1 žingsnis. Padaliname plotą per pusę.
2 žingsnis. Tikriname pirmąją pusę - jeigu liūtas yra čia, tyrinėjame toliau. Jeigu ne, peržiūrime kitą pusę.
3 žingsnis. Tikriname ar tyrinėjamas plotas yra mažesnis nei 0,5 km2, jeigu taip - da rome prielaidą, kad liūtas yra pagautas ir algoritmas sustoja. Kitu atveju grįžtame prie 1-ojo žingsnio.

Afrikos plotas yra apie 30 370 000 km2.



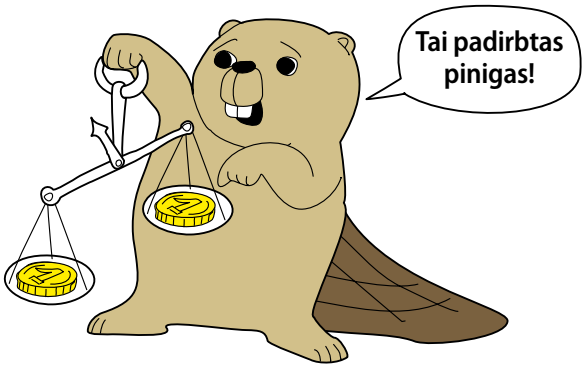
Kiek skirtingų plotų reikia patikrinti, kad liūtas būtų pagautas?

Dvejetainė paieška yra dažniausiai naudojama technika problemoms spręsti. Dvejetainės paieškos algoritmas buvo paskelbtas Donaldo Knuto knygoje "Programavimo menas" 1971 metais.

24 NETIKRA MONETA
SKALDYMO ALGORITMAS

Dvejetainės paieškos principas paprastas: ieškomasis elementas palyginamas su surikiuotos sekos viduriniu nariu. Jei jie yra lygūs, vadinasi, radome ieškomą elementą sekoje. Jei ieškomasis elementas yra mažesnis už vidurinį, tai reiškia jis mažesnis ir už visus „dešiniuosius" sekos narius, todėl paieška tęsime kairiojoje sekos dalyje. Analogiškai, jei ieškomas elementas didesnis už vidurinį, paieška tęsime dešiniojoje masyvo dalyje. Toliau ieškoma tuo pačiu principu, kol randamas ieškomas elementas arba paieškos sritis tampa tuščia.

Tarkime, bebras turi devynias monetas, kurios yra tokio pačio svorio, išskyrus vieną - netikrą, ji yra lengvesnė. Bebras turi svarstykles, kuriomis galima lyginti monetų svorius.
Ar įmanoma nustatyti netikrą monetą sveriant tik du kartus?

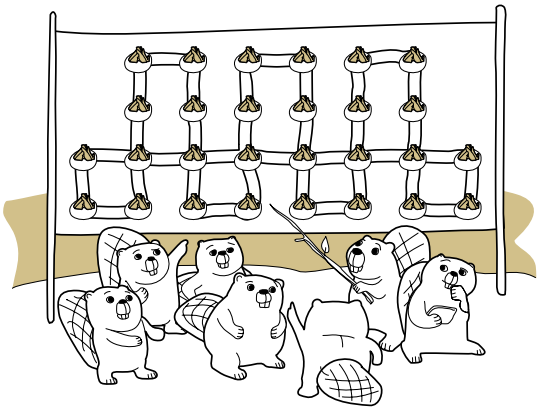


Skaldymo metodas (angl. divide and conquer) yra efektyvi uždavinių sprendimo strategija, kai uždavinys skaidomas į mažesnius ir jiems ieškomas sprendimas. Dvejetainės paieškos algoritmas yra skaldymo metodo atskiras atvejas, kai dalijama perpus.

25 AR TAI ĮMANOMA?
OILERIO TEOREMA

Paveiksle kanalais sujungti miesteliai. Toks abstrahuotas vaizdas vadinamas grafu ir labai dažnai naudojamas sprendžiant uždavinius. Grafą sudaro viršūnės (taškai) ir jas jungiančios briaunos (linijos). Grafe yra daug kelių. Kai kelias eina per kiekvieną briauną tik po vieną kartą ir apima visas briaunas, tai jis vadinamas Oilerio keliu. Bebrai nori plaukti kiekvienu kanalu ir tik po vieną kartą. Vienas iš bebrų yra susipažinęs su Oilerio teorema: jeigu grafas turi ne daugiau kaip dvi nelyginio laipsnio viršūnes, tuomet šis grafas turi bent vieną Oilerio kelią (dažniausiai daugiau).

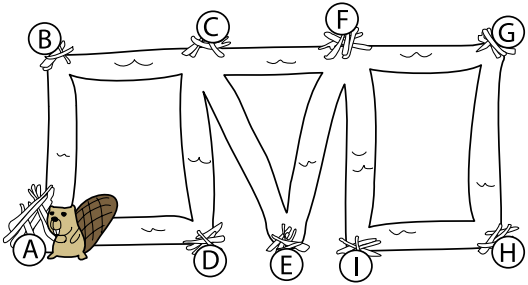
Kai kurie bebrai nežino, ką reiškia viršūnės laipsnis. Sugalvok, kas tai galėtų būti.
Nustatyk, ar žemiau pateiktas grafas (kanalų sistema) turi Oilerio kelią.



Oilerio maršrutas – maršrutas grafe, kuriuo galima pereiti visas jo briaunas po vieną kartą. Jei tokiu maršrutu grįžtama į pradinę viršūnę, jis vadinamas Oilerio ciklu, jei ne – Oilerio keliu.

26 OILERIO CIKLO RADIMAS
FLERIO ALGORITMAS

Bebras ketina plaukti kiekvienu kanalu ir tik po vieną kartą, pradėdamas ir užbaigdamas savo buveinėje A. (Kai Oilerio kelias prasideda ir pasibaigia toje pačioje viršūnėje, jis vadinamas Oilerio ciklu). Maršrutui rasti Bebras naudoja Flerio algoritmą, kurio esmė tokia:
Kiekviename žingsnyje, jeigu tik galima, bebras pasirenka dar neplauktą kanalą, kuris nėra "tiltas" į nekeliautą dalį (tiltu vadinama briauna, kurią perėjus grafas suskiltų į dvi dalis - sudegink tiltą už savęs ir niekada negalėsi grįžti atgal). Tiltą bebras renkasi tik tada, jei nėra jokio kito "ne tiltiško" kanalo.



1 žingsnis: iš A į B
2 žingsnis: iš B į C
3 žingsnis: iš C į E (C į D yra tiltas!)
4 žingsnis: iš E į F
5 žingsnis: Kuris iš kanalų: iš F į C, iš F į G ar iš F į I turėtų būtų pasirinktas?

Kiek žingsnių liko pabaigti Oilerio ciklą?

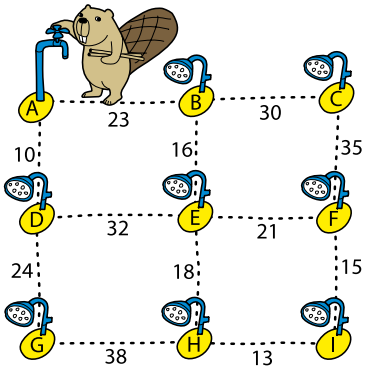
Flerio algoritmas atrastas 1883 metais. Jis paprastas ir elegantiškas, vienintelis dalykas, su kuriuo reikia elgtis atsargiai - tai tilto interpretacija. Reikia nepamiršt, kad turima omeny nekeliautos dalies tiltus, tad jie nuolat keičiasi ir verčia būti budrius.

27 VAMZDYNAS
KRUSKALO ALGORITMAS

Bebras nori sudaryti vamzdžių tinklą taip, kad vanduo nuo pagrindinio šaltinio A galėtų nusigauti iki kiekvienos purkštuvo galvutės (B, C, D, E, F, G, H, I). Skaičius ant vamzdžio nurodo šio vamzdžio tiesimo išlaidas. Siekiama darbą atlikti kaip galima pigiau.

Bebras žino Kruskalo algoritmą:

1. Rasti mažiausių išlaidų vamzdį (jeigu yra daugiau nei vienas, pasirinkti atsitiktinai). Jį pažymėti.
2. Rasti kitą pigiausią nepažymėtą vamzdį. Jeigu jis sudaro ciklą su jau pažymėtais vamzdžiais, reikia pašalinti šį vamzdį iš tolimesnio pasirinkimo. Jeigu nesudaro ciklo, tuomet jį pažymėti.
3. Kartoti 2-ąjį žingsnį tol, kol visos maišytuvo galvutės bus sujungtos.



Bebras pritaiko algoritmą vamzdyno projektavimui:

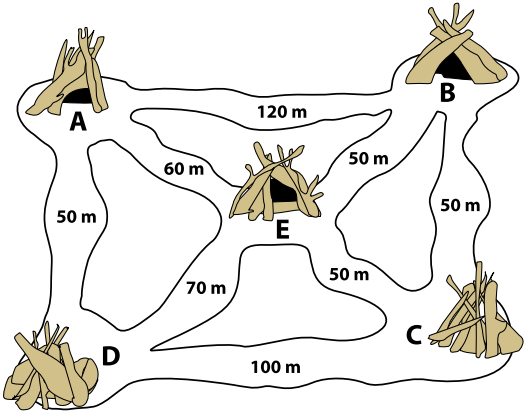
1. Pažymi AD.
2. Pažymi HI.
3. Pažymi FI.
4. Pažymi BE.
5. Pažymi EH.
6. Pažymi EF. Ciklas. Pašalina EF.
7. . . .

Tęsia paiešką ir suprojektuoja pigiausią vamzdžių tinklą. Kaip atrodys vamzdynas? Kiek kainuos jo nutiesimas?

Kruskal's algorithm always finds a minimum spanning tree (with the least total weight) for a connected weighted graph.

28 KANALAI
KRUSKALO ALGORITMAS

Penkios bebrų buveinės (A, B, C, D, E) įsikūrė, kaip parodyta žemėlapyje.



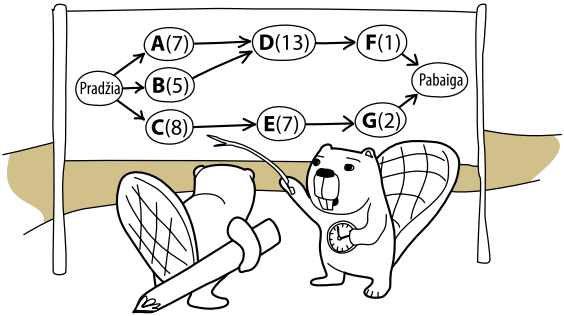
Bebrai norėtų sujungti šias penkias buveines vandens kanalais ir taip, kad tiesiamų kanalų bendras ilgis būtų kuo mažiausias. Pasiūlyk bebrams visus šiuos galimus kanalų tiesimo variantus. Pasinaudok Kruskalo algoritmu:

1. Rasti trumpiausią kanalą (jeigu yra daugiau nei vienas, pasirinkti atsitiktinai). Jį pažymėti.
2. Rasti kitą trumpiausią nepažymėta kanalą. Jeigu ji sudaro ciklą su jau pažymėtais kanalais, reikia pašalinti šį kanalą iš tolimesnio pasirinkimo. Jeigu nesudaro ciklo, tuomet jį pažymėti.
3. Kartoti 2-ąjį žingsnį tol, kol visos bebrų buveinės bus sujungtos.

Minimalaus jungiančiojo medžio radimo uždavinys - tai vienas iš retų atvejų, kai viskas vyksta gerai: turime lengvai suprantama ir lengvai atliekamą Kruskalo algoritmą, be to, jis yra optimalus ir tuo pat metu efektyvus.

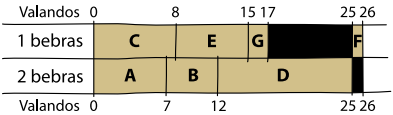
29 DU BEBRAI DIRBA
MAŽĖJANČIŲ TRUKMIŲ ALGORITMAS

Du bebrai stato užtvanką ir turi atlikti 7 užduotis: A(7), B(5), C(8), D(13), E(7), F(1), G(2). Skaičiai skliaustuose nurodo valandas, reikalingas užduočiai atlikti. Kai kurios užduotys turi būti padarytos pirmiau kitų - atlikimo tvarka nurodoma rodyklėmis. Pavyzdžiui užduotis D(13) gali būti atliekama tik tada, kai A ir B yra pabaigtos.



Bebrai naudoja mažėjančių trukmių algoritmą, kurio esmė: iš visų užduočių, kurias tuo metu reikia atlikti, pasirenkama ilgiausiai trunkanti užduotis (Lygiųjų atveju rinktis atsitiktinai).

Mažėjančių trukmių sąrašas atrodytų taip: D(13), C(8), A(7), E(7), B(5), G(2), F(1). Tačiau užduoties D(13) negalima atlikti tol, kol neatliktos užduotys A ir B. Sudarome tvarkaraštį:



Taikydami šį algoritmą, bebrai gali baigti statyti užtvanką per 26 valandas.

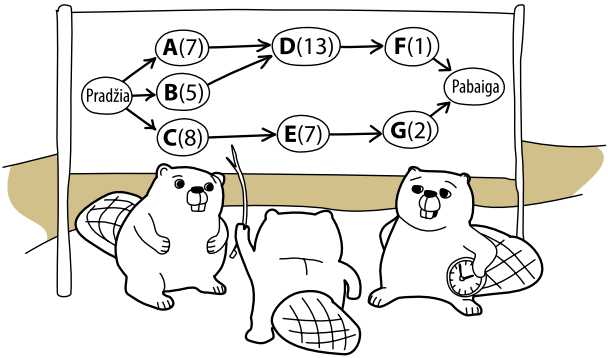
Ar yra įmanoma pastatyti užtvanką per trumpesnį laiką?
Pagrįskite atsakymą.

Mažėjančių trukmių algoritmas yra paremtas paprasta strategija: daryti ilgiausius darbus pirma ir pataupyti trumpesniuosius darbus pabaigai.

30 TRYS BEBRAI DIRBA

MAŽĖJANČIŲ TRUKMIŲ ALGORITMAS

Trys bebrai nori pastatyti užtvanką ir jiems reikia atlikti 7 užduotis: A(7), B(5), C(8), D(13), E(7), F(1), G(2). Skaičiai skliaustuose nurodo valandas, reikalingas užduočiai atlikti. Kai kurios užduotys turi būti padarytos pirmiau kitų - atlikimo tvarka nurodoma rodyklėmis.



Bebrai naudoja mažėjančių trukmių algoritmą, kurio esmė: iš visų užduočių, kurias tuo metu reikia atlikti, pasirenkama ilgiausiai trunkanti užduotis. Lygiųjų atveju galima rinktis bet kurią.

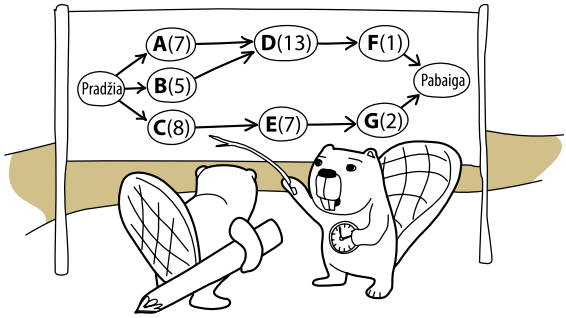
Sudarykite tvarkaraštį, kaip bebrai turi atlikti darbus. Per kiek laiko bus pastatyta užtvanka?

Nors ir strategija pirmiausia padaryti ilgiausias užduotis skamba gerai ir yra prasminga, tačiau praktiškai duoda ne visai efektyvius tvarkaraščius.

31 UŽDUOČIŲ PLANAVIMAS

KRITINIS KELIAS: PRIEŠSROVIO ALGORITMAS

Bebrai stato užtvanką ir turi atlikti 7 užduotis: A(7), B(5), C(8), D(13), E(7), F(1), G(2). Skaičiai skliaustuose nurodo valandas, reikalingas užduočiai atlikti. Kai kurios užduotys turi būti padarytos pirmiau kitų - atlikimo tvarka nurodoma rodyklėmis.



Bebrai žino apie kritinį kelią ir nori jį rasti: kritiniu keliu laikomas kelias, kurio užduotims atlikti sugaištama daugiausiai laiko. Jam rasti taikoma paprasta procedūra, vadinama priešsrovio algoritmu. Pagrindinė idėja yra pradėti nuo pabaigos ir keliauti link pradžios kiekvienoje viršūnėje (užduotyje) sumuojant vykdymo laikus.

1 žingsnis. Užrašome kritinių kelių ilgius (laužtiniuose skliaustuose): F(1)[1] ir G(2)[2].
2 žingsnis. Užrašome kritinio kelio D(13)[13+1=14] ilgį.
3 žingsnis. Užrašome kritinio kelio E(7)[7+2=9] ilgį.
4 žingsnis. ...

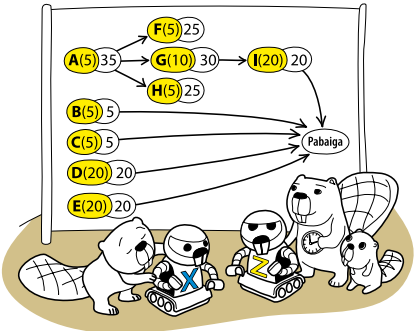
Pratęskime priešsrovio algoritmą ir nustatykime užtvankos statymo kritinį kelią.

The critical time of a project is the shortest possible time required to complete the project. The backflow algorithm finds the critical path. NĖRA VERTMO

32 DARBŲ PLANAVIMAS DVIEMS ROBOTAMS

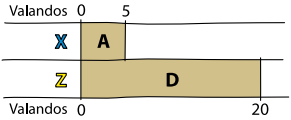
KRITINIO KELIO ALGORITMAS

Bebro šeima naudoja du robotus X ir Z, kuriems pavesta atlikti 9 užduotis: A(5), B(5), C(5), D(20), E(20), F(5), G(10), H(5), I(20). Skaičiai skliaustuose nurodo valandas, reikalingas užduočiai atlikti. Kai kurios užduotys turi būti padarytos pirmiau kitų - atlikimo tvarka nurodoma rodyklėmis.



Bebrai žino kritinio kelio algoritmą ir norėtų jį naudoti. Algoritmas nurodo, jog kai tik robotas yra laisvas, jis turėtų peržvelgti visas tuo momentu laisvas užduotis ir pasirinkti vieną, kurios vykdymo laikas ilgiausias - didžiausias kritinis kelias. Lygiųjų atveju, užduotis pasirenkama atsitiktinai. Kiekvienai užduočiai nurodytas kritinio kelio ilgis (baltame skrituliuke).

Tęskite naudodami kritinio kelio algoritmą suplanuoti užduotis dviems robotams.



Bendras darbo kritinis kelias rodo minimalų laiką, per kurį darbas gali būti atliktas. Deja, ne visada galima sudaryti tokį tvarkaraštį, kad užduotys eitų viena paskui kitą žąsele, be pertrūkių.

33 KAS PARAŠYTA?
CEZARIO ŠIFRAS

Kriptografijoje Cezario šifras yra vienas iš paprasčiausių šifravimo būdų. Tai kodavimo šifras, kuriuo kiekviena raidė tekste pakeičiama kita raide pastūmus abėcėlę per tam tikrą skaičių (vadinama raktu). Pavyzdžiui, kai raktas lygus 3, tai A raidė bus keičiama į C, Ą keičiama į Č ir taip toliau, t. y. raide, kuri stovi abėcėlėje trimis pozicijomis dešiniau. Paskutiniąsias tris raides tenka keisti atitinkamomis pirmosiomis abėcėlės raidėmis.

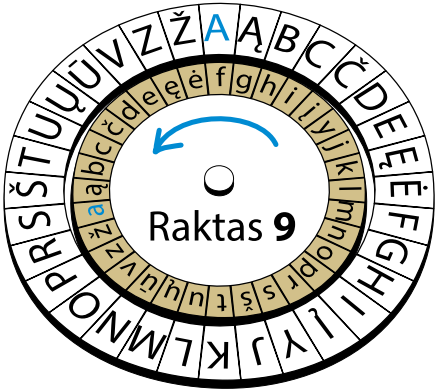
Ką parašė bebras?

Cezario šriftas yra vadinamas Romos imperatoriaus Juliaus Cezario vardu, nes jis šį šriftą naudojo savo privatiems susirašinėjimams.

34 KAS UŽSAKYTA PIETUMS?

ALBERTI ŠIFRAVIMO DISKAS

Vienoje iš bebravietės kavinių stovi šifravimo ratas ir visi patiekalai užkoduoti. Sukama tik vidinė rato dalis (su mažosiomis raidėmis) ir tik prieš laikrodžio rodyklę. Išorinė rato dalis reikalinga žinutei koduoti.

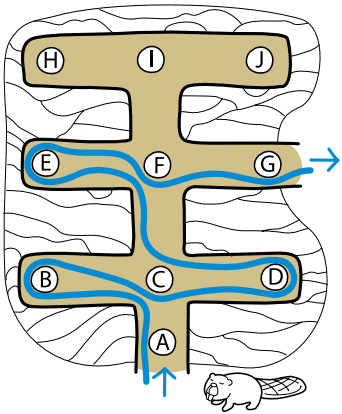


Pavyzdžiui, kai raktas lygus 9, tai vidinis ratas pasukamas per 9 pozicijas ir raidė A užkoduojama raide f. Kiekvienam naujam žodžiui ratas nustatomas į pradinę padėtį: A sutampa su a. Šiandien pirmo žodžio raktas lygus 7. Antro žodžio raktas yra 4.

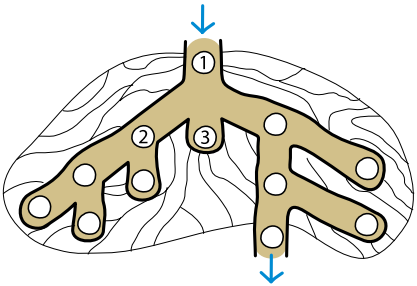
Ką užsisakė bebras, jei kodas yra: fįveų ųzpč?

Šį šifravimo ratą sukūrė italų architektas Leon Battista Alberti 1470 metais. Ratą sudarė du koncentriniai diskai, užmauti vienas ant kito. Didysis diskas buvo stacionarus, o mažasis – slankus.

Bebras įlindo į požeminį labirintą ir ieško išėjimo. Jis žino paieškos į plotį algoritmą ir juo vadovaujasi: labirinte juda pirmiausia patikrindamas visus nusukimus į šonus ir tik neradęs išėjimo tęsia paiešką į priekį. Paieškos į plotį algoritmą paaiškinsime pavyzdžiu. Pradedama iš taško A ir einama, kaip parodyta, link išėjimo G.



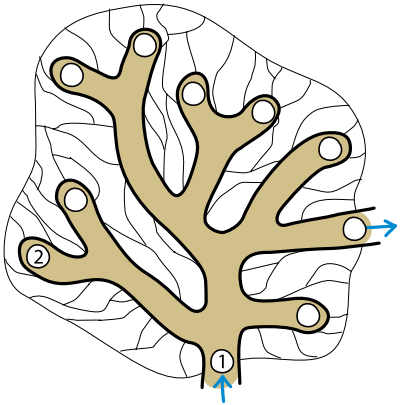
Naudodamas paieškos į plotį algoritmą, padėk Bebrui išeiti iš antrojo labirinto.
Sunumeruok apskritimus, kokia eilės tvarka jie aplankomi.



Paieškos į plotį (Breatch-First Search) algoritmą paskelbė pirmojo dvejetainio kompiuterio Z1 kūrėjas Konradas Cūsė 1945 metais savo disertacijoje apie Plankalkül programavimo kalbą (disertacija nebuvo apginta). Pirmą kartą algoritmas išspausdintas 1972 metais.

36 URVUOSE
PAIEŠKA Į GYLĮ

Bebras įlindo į požeminius urvus ir ieško išėjimo. Jis žino paieškos į gylį algoritmą ir juo vadovaujasi: urvuose eina pirmiausia kiek įmanoma giliau, renkantis vis naują viršūnę; kai paskutinė aplankyta viršūnė naujos (dar neaplankytos) kaimynės nebeturi, tada grįžtama iki artimiausios neaplankytos briaunos ir vėl ieškoma kuo giliau tol, kol bus rastas išėjimas.



Naudodamas paieškos į plotį algoritmą, padėk Bebrui išeiti iš urvų.
Sunumeruok apskritimus, kokia eilės tvarka jie aplankomi.

Paieškos į gylį (angl. Depth-First Search) algoritmas priešpastatomas paieškai į plotį, kur esant kurioje nors viršūnėje pirmiausia aplankomos visos šiai viršūnei gretimos viršūnės, po to viršūnės, gretimos pastarosioms ir taip toliau.

SPRENDIMAI

1. Rąstų rikiavimas

Rąstų {7, 5, 10, 9, 15, 13, 2, 8, 17, 20} keitimų procesą parodysime žingsniais. Pabrauktas skaičius reiškia, jog šis rąstas yra lyginamas su gretimu rąstu (prieš jį einančiu). Kai rąstas būna sukeistas (ar paliktas savo vietoje, jei atitinka eiliškumą), jo skaičius paryškinamas.

7, 5, 10, 9, 15, 13, 2, 8, 17, 20 - bus atliktas 1 keitimas
5, 7, 10, 9, 15, 13, 2, 8, 17, 20
5, 7, 10, 9, 15, 13, 2, 8, 17, 20 - bus atliktas 1 keitimas
5, 7, 9, 10, 15, 13, 2, 8, 17, 20
5, 7, 9, 10, 15, 13, 2, 8, 17, 20 - bus atliktas 1 keitimas
5, 7, 9, 10, 13, 15, 2, 8, 17, 20 - bus atlikti 6 keitimai
2, 5, 7, 9, 10, 13, 15, 8, 17, 20 - bus atlikti 4 keitimai
2, 5, 7, 8, 9, 10, 13, 15, 17, 20
2, 5, 7, 8, 9, 10, 13, 15, 17, 20
2, 5, 7, 8, 9, 10, 13, 15, 17, 20

Iš viso atlikta 13 keitimų.

Blogiausiu atveju, kai visi duomenys pateikti atvirkščia norimo rikiavimo eile, keitimų skaičius bus lygus $1 + 2 + 3 + \ldots + (n-1) = n \times (n-1) / 2$

2. Bylų rikiavimas

Bylų {2000, 2006, 2007, 2009, 2011, 2005, 2003, 2008, 2015, 2010, 2004, 2013} keitimo procesą parodysime žingsniais. Pabrauktas skaičius reiškia, jog ši byla yra lyginama su gretima byla (prieš ją einančia). Kai byla būna sukeista (ar palikta savo vietoje, jei atitinka eiliškumą), jos skaičius paryškinamas.

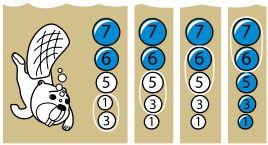
2000, 2006, 2007, 2009, 2011, 2005, 2003, 2008, 2015, 2010, 2004, 2013
2000, **2006**, 2007, 2009, 2011, 2005, 2003, 2008, 2015, 2010, 2004, 2013
2000, 2006, **2007**, 2009, 2011, 2005, 2003, 2008, 2015, 2010, 2004, 2013
2000, 2006, 2007, **2009**, 2011, 2005, 2003, 2008, 2015, 2010, 2004, 2013
2000, 2006, 2007, 2009, **2011**, 2005, 2003, 2008, 2015, 2010, 2004, 2013
2000, **2005**, 2006, 2007, 2009, 2011, 2003, 2008, 2015, 2010, 2004, 2013 - bus atlikti 4 keitimai
2000, **2003**, 2005, 2006, 2007, 2009, 2011, 2008, 2015, 2010, 2004, 2013 - bus atlikti 5 keitimai
2000, 2003, 2005, 2006, 2007, **2008**, 2009, 2011, 2015, 2010, 2004, 2013 - bus atlikti 2 keitimai
2000, 2003, 2005, 2006, 2007, 2008, 2009, 2011, **2015**, 2010, 2004, 2013
2000, 2003, 2005, 2006, 2007, 2008, 2009, **2010**, 2011, 2015, 2004, 2013 - bus atlikti 2 keitimai
2000, 2003, **2004**, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2015, 2013 - bus atlikti 8 keitimai
2000, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, **2013**, 2015 - bus atliktas 1 keitimas
2000, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2013, **2015**

Iš viso bus atlikti 22 keitimai, o tai sudaro $22 \times 5 = 110$ sekundžių darbo.

Blogiausiu atveju, kai visi duomenys pateikti atvirkščia norimo rikiavimo eile, keitimų skaičius bus lygus $1 + 2 + 3 + \ldots + (n-1) = n \times (n-1) / 2$

3. Dideli burbulai kyla

Ketvirtas sukeitimas yra pavaizduotas dešinėje:
Ketvirtuoju žingsniu surikiuojami keturi viršutiniai burbulai, taigi ir paskutinysis burbulas turi būti savo vietoje.
Norint išanalizuoti burbulo metodo rikiavimu, turėtume atkreipti dėmesį kaip burbulai išdėstyti pradžioje. Jeigu yra n burbulų, tai blogiausiu atveju reikės n-1 ėjimo.

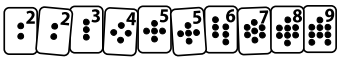


4. Triušiai ir vėžliai

Teisingas atsakymas D.
Turėtume atkreipti dėmesį į:
- Elementas, judantis į paskutiniąją poziciją, gali judėti greitai - turi būti triušiai (8, 7, 6).
- Elementas, judantis į pirmąją poziciją, negali judėti greičiau nei vieną pasikeitimą per ėjimą, taigi į pirmąją poziciją judama lėtai - turi būti vėžliai (1, 3, 5).

5. Kortelių rikiavimas perstumiant

Kartojami trys žingsniai: palyginama, pastumiama ir įterpiama. Parodyti žingsniai po pastūmimo (5, 8, 11 ir 14), tad mums reikia nustatyti įterpiamas korteles. 5-tame žingsnyje pastūmimas yra tarp kortelių 4 ir 6, vadinasi, kitu žingsniu gali būti įterpiama kortelė 5 arba 6. Atsižvelgę į tolesnius žingsnius, matome, kad buvo įterpta kortelė 5. Analogiški samprotaudami nustatome įterptas korteles: 5 5 8 2. Paskutiniame žingsnyje gauname tokia kortelių eilė:



6. Bylų rikiavimas perstumiant

Jei užtrunka 5 sekundes ištraukti bylą ir perkelti ją į kitą vietą, tai kiek laiko užtruks surūšiuoti visas bylas?
Pabrauktas skaičius reiškia, jog ši byla buvo ištraukta.
2000, 2006, 2007, 2009, 2011, 2005, 2003, 2008, 2015, 2010, 2004, 2013
2000, 2005, 2006, 2007, 2009, 2011, 2003, 2008, 2015, 2010, 2004, 2013 - 5 sekundės
2000, 2003, 2005, 2006, 2007, 2009, 2011, 2008, 2015, 2010, 2004, 2013 - 5 sekundės
2000, 2003, 2005, 2006, 2007, 2008, 2009, 2011, 2015, 2010, 2004, 2013 - 5 sekundės
2000, 2003, 2005, 2006, 2007, 2008, 2009, 2011, 2015, 2010, 2004, 2013
2000, 2003, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2015, 2004, 2013 - 5 sekundės
2000, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2015, 2013 - 5 sekundės
2000, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2015, 2013
2000, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2013, 2015 - 5 sekundės
Rikiavimas naudojant šį algoritmą užtruks 30 sekundžių; 6 x 5 = 30 sekundžių.

7. Kortelių dalijimas ir jungimas

Teisingas variantas: C.
A. Neteisinga. Sujungę pirmuosius du kortelių rinkinius gautume pagrindą tolesniems jungimams, tačiau mes dar iki to nepriėjome.
B. Neteisinga. Tai būtų paskutinieji du sujungti rinkiniai.
C. Teisinga! Kortelės 5 ir 7 yra pirmosios pagrindinio jungimo kortelės ir todėl jos bus pirmi du kortelių rinkiniai, kurie bus sujungti.
D. Neteisinga. Nors kortelės 3 ir 7 yra greta viena kitos, tačiau pradedant pirmą dalijimą jos atsidurs skirtingose rinkinių pusėse.

8. Kaip įmanoma greičiau

Kadangi algoritmas nėra lengvas, pademonstruosime visus tris žingsnius:

1 žingsnis. Gautos dvi dalys:

2 žingsnis. Gautos 4 dalys:

3 žingsnis. Gautos 7 dalys:

9. Bebras tyrinėja

Atsakymas B. Panagrinėkime kiekvieną variantą.

A. Norėdamas plaukti per kuo mažiau sankirtų iš A į D bebras turi rinktis kelią AFED arba AGED. Pirmuoju atveju skaičius lapelyje turėtų būti D = 54, antruoju atveju D = 45. Abu atvejai prieštarauja užrašytam skaičiui D = 44.

B. Prieštaravimų nerandame.

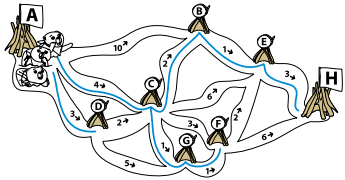
C. Sukdamas kairėn iš A į C bebras turi plaukti ABC, tada C = 33 ir gaunamas prieštaravimas.

D. Iš A į G bebras turėtų plaukti AFG ir tada G = 19 - prieštaravimas užrašytam skaičiui.

Vadinasi, skaičius ant geltono lapuko rodo trumpiausio kelio ilgį nuo A iki šios upių sankirtos.

10. Trumpiausio kelio ieškant

Atlikę Dijkstros algoritmo žingsnius, apskaičiuosime trumpiausius kelius tarp pradinio urvo A ir bet kurio kito urvo.



Kadangi bebrai plaukia iš A į H, tai jų trumpiausias kelias yra A C B E H ir lygus 10.

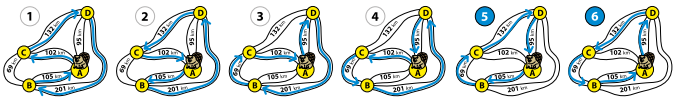
11. Daug maršrutų

Maršrutus reikia perrinkti sistemingai, pavyzdžiui, imame vieną kelio atkarpą AC ir sudarome visus galimus maršrutus, jų bus du: ACDBA ir ACBDA. Tada imame kitą atkarpą, tarkime, AD ir sudarome maršrutus: ADCBA ir ADBCA. Analogiškai elgiamės su likusia AB atkarpa, gauname dar du maršrutus: ABCDA ir ABDCA. Matome, kad trūksta dviejų maršrutų, apskaičiuojame jų ilgius:

ADCBA 95 + 132 + 69 + 105 = 401

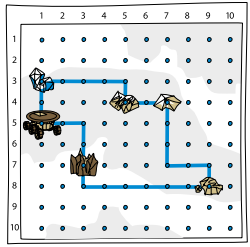
ABCDA 105 + 69 + 132 + 95 = 401

Tai ir bus trumpiausias maršrutas, jo ilgis 401.



12. Mėnuleigis

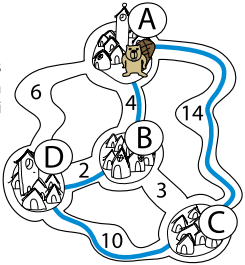
Tai keliaujančio pirklio uždavinio atskiras atvejis, tik dar sudėtingesnis, nes nenurodyti keliai. Kaip žinote, šiam uždaviniui nėra patikimo algoritmo. Mėnuleigis turi aplankyti visus deimantus ir grįžti į pradinę padėtį. Kadangi deimantų (grafo viršūnių) nedaug, tai žmogus gali intuityviai rasti sprendimą. Logika sako, kad optimaliausia keliauti ratu nuo vieno prie kito deimanto einant kuo trumpiausiu keliu. Tokių trumpiausių kelių daug, vienas iš jų parodytas paveiksle.



Reikėtų ištirti visus posūkius ir įsitikinti, kad neįmanoma labiau sutrumpinti šio kelio.

13. Kaimynų lankymas

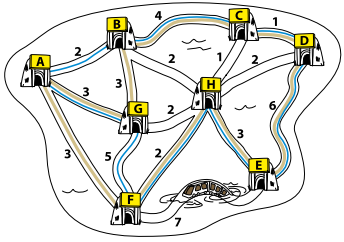
Taikydamas artimiausio kaimyno algoritmą bebras iš A turi rinktis pigiausią kelią (4), vadinasi, keliauja į B. Iš B veda trys keliai: A (čia būta - bebro namas), D (2) ir C (3) - pigiausias kelias yra D. Taigi bebro maršrutas Nr 1: A B D C A.



14. Renkantis pigiausią ne visada pigu

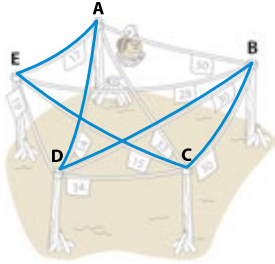
Norint įrodyti, kad bebro maršrutas nėra trumpiausias, pakanka rasti bent vieną trumpesnį maršrutą. Galima išrašyti visus įmanomus maršrutus ir apskaičiuoti atstumus. Tačiau tai daug darbo. Ar įmanoma efektyviau spręsti? Patyrinėkime atkarpas. Pagerinimą pasiektume, jei nenaudotume ilgiausių atkarpų, pavyzdžiui, 7. Randame tokį maršrutą:
ABCDEHFGA - jo ilgis 26 km.

Įdomumo dėlei patyrinėkime, ar įmanomas trumpesnis maršrutas. Deja, 6 km. kelio nepavyks atsisakyti (kodėl?), tačiau 5 galime nenaudoti, tuomet maršrutas būtų: AFHEDCBGA - jo ilgis 25 km.



15. Pramogų tinklas

Uždavinyje siūlomas maršrutas taikant artimiausio kaimyno algoritmą kainuotų 14 + 19 + 15 + 30 + 50 = 128 b€. Kad parodytume, kad tai nėra pigiausias maršrutas, pakanka rasti bent vieną pigesnį maršrutą. Intuicija kužda, kad reikėtų rinktis taip, kad gale išvengtume labai brangaus kelio už 50 b€. Taigi iš karto galima rinktis ne patį pigiausią kelią, o kiek brangesnį, pavyzdžiui, E. Toliau sektų atitinkamai B, C, D ir A. Viso maršruto kaina būtų: 17 + 28 + 30 + 34 + 14 = 123 b€.



Artimiausio kaimyno algoritmą pritaikykime paėmę vis kitą pradžios tašką: B, C, D ir E.

Palyginkime gautus skaičiavimus:

1. BECADB = 128 b€
2. CEADBC = 106 b€
3. DAECBD = 106 b€
4. ECBDAE = 106 b€

Matome, kad yra net trys pigesni maršrutai, nors iš tiesų tai yra vienas ir tas pats maršrutas, tik skirtinga pradžia, norėdami pradėti nuo A, maršrutas bus: ADBCEA.

16. Dar kartą pramogų centre

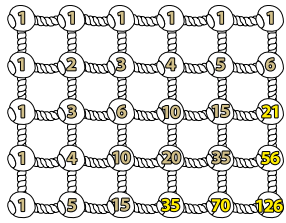
Taikome pigiausios jungties algoritmą išrašydami žingsnius:

1. Pigiausia jungtis – ED (16). Pažymime ją.
2. Kita pigiausia tinkama jungtis – AB (20). Pažymime ją.
3. Kita pigiausia tinkama jungtis – AE (25). Pažymime ją.
4. Toliau pigiausia jungtis yra AC (28), bet iš taško A jau išeina dvi jungtys, vadinasi, AC netinka. BD taip pat netinka, nes pasirinkus susidarytų ciklas EABDE. Todėl kita pigiausia tinkama jungtis – CD (31). Pažymime ją.
5. Ir toliau tinkama jungtis – BC (50).

Gavome maršrutą ABCDEA, kurio kainą 142 beurai: nors ir ne pigiausias, bet gerokai pigesnis už brangiausią (179).

17. Kiek kelių?

Pirmiausia pabandykime pagrįsti, kad šis skaičiavimas teisingas. Tarkime, P[i, j] yra trumpiausių kelių skaičius nuo pradžios taško iki susikirtimo taško, kurio koordinatės i (1<=i<=5) ir j (1<=j<=6). Bet kurį trumpiausią kelią sudaro atitinkami horizontalių ir vertikalių atkarpų segmentai. Taigi trumpiausių kelių skaičius nuo pradžios iki susikirtimo taško i ir j gali būti rastas sumuojant:



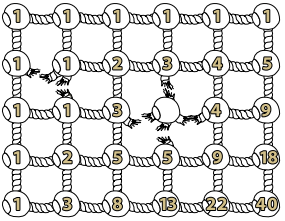
P[i, j] = P[i-1, j] + P[i, j-1] kiekvienam 1<i<=5, 1<j<=6, čia P[1, j] = 1 kiekvienam 1<=j<=6 ir P[i, 1]=1 kiekvienam 1<=i<=5.

Taikydami šią formulę galime apskaičiuoti kiekvieną P[i, j] reikšmę judėdami iš kairės į dešinę ir leisdamiesi eilutėmis žemyn.

18. Suardyti takai

Kelių skaičiavimas tinkle yra klasikinis dinaminio programavimo taikymo būdas. Remiantis šiuo metodu randamas trumpiausių takų skaičių nuo pradžios iki kiekvieno susikirtimo taško.

Kadangi uždavinyje kai kur tinklas pažeistas, tai suardytos dalys nėra skaičiuojamos. Jeigu susikirtimo taškas turi kairįjį ir viršutinį kaimynus, tai šio taško trumpiausių kelių skaičių sudaro šių kaimynų skaičių suma; jeigu susikirtimo taškas turi tik vieną kaimyną, tai trumpiausių kelių skaičius tiesiog sutaps.



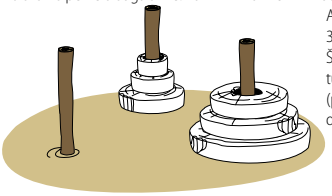
19. Bebrų akrobatika

Pažymėkime didžiausią bebrą D, vidurinį - V ir mažiausią - M. Tuomet galėtume išrašyti visus bebrų šokinėjimus. Tačiau galime rasti ir trumpiau užrašomą sprendimą. Iš tiesų, 4 bebrų peršokimas yra tas pats, kas trijų, tik vienetu mažesnis uždavinys. Tarkime, kad žinome, kaip 3 bebrai peršoka. Tuomet lieka tik paskutinis, didžiausias bebras. Jis peršokdamas sugaiš 1 minutę.

Kai tik didžiausias bebras peršoks ant tuščio kelmo, tuomet visi 3 bebrai galės peršokti ant jo - tai mokame ir žinome, kad sugaištamos 7 minutės. Taigi iš viso bebrai sugaiš: 7 + 1 + 7 = 15 minučių.

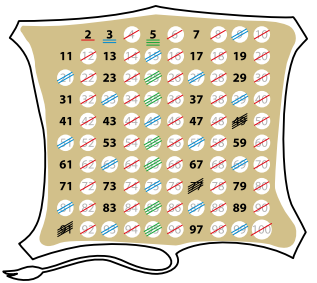
20. Vėl ir vėl - rekursija

Norėdami perkelti 6 diskus, turime mokėti perkelti 5 diskus. Norėdami perkelti 5 diskus, turime mokėti perkelti 4 diskus - tai mokame ir žinome, kad sugaištama 15 minučių. Mokėdami perkelti 4 diskus, penkis perkeltume paprastai: kai 4 diskai perkelti, tada paskutinįjį (didžiausią) diską perkeliame ant likusio tuščio strypo ir belieka perkelti 4 diskus, tik dabar ant strypo, kuriame uždėtas didžiausias diskas. Taigi 5 diskams perkelti sugaišime: 15 + 1 + 15 = 31 minutę.



Analogiškai samprotaudami, 6 diskams sugaišime: 31 + 1 + 31 = 63 minutes.

Šiam uždaviniui spręsti pritaikome rekursiją - tai tų pačių veiksmų kartojimas, tik kai ką sumažinus (pavyzdžiui, Hanojaus bokštas kaskart mažėja vienu disku).



21. Lengvas būdas surasti pirminius skaičius

Atlikę algoritmo žingsnius su visais neišbrauktais skaičiais, pabaigoje liks pirminiai skaičiai. Jų pirmajame šimte yra 25.

22. Greitesnis būdas pirminiams skaičiams rasti

Reikia pastebėti dėsningumą: pirmą kartą atimdami gauname pirmą pirminį skaičių - 2. Tada sąrašo pradžioje atsiduria antras pirminis skaičius - 3, vėl atimame. Trečią kartą atimdami gauname trečiąjį pirminį skaičių - 5, ketvirtą - ketvirtąjį pirminį skaičių 7 ir t. t. Vadinasi, kad gautume pirmus 10 pirminių skaičių, reikės sąrašus atiminėti 10 kartų.

23. Pagauk liūtą

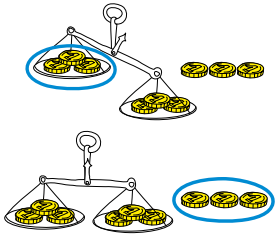
Norint sužinoti, kiek reikės patikrinti plotų, reikia Afrikos plotą 30 370 000 dalinti iš 2 tol, kol gausime mažiau nei 0,5 km2. Pirmas žingsnis: tikriname plotą 30 370 000:2 = 15 185 000. Antras žingsnis: tikriname plotą 7 592 500. Toliau tikriname plotus:

3) 3 796 250;	8) 118 632.82;	13) 3 707.28;	18) 115.86;	23) 3.63;
4) 1 898 125;	9) 59 316.41;	14) 1853.64;	19) 57.93;	24) 1.82;
5) 949 062.5;	10) 29 658.21;	15) 926.82;	20) 28.97;	25) 0.91;
6) 474 531.25;	11) 14 829.11;	16) 463.41;	21) 14.49;	26) 0.46.
7) 237 265.63;	12) 7 414.56;	17) 231.71;	22) 7.25;	

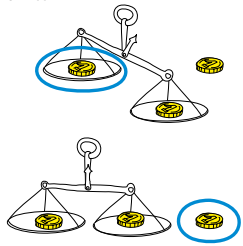
24. Netikra moneta

Kadangi pasakyta, kad netikra moneta lengvesnė, tai įmanoma nustatyti netikrą monetą sveriant du kartus. Kaip tai padaryti, pavaizduota piešiniais.

1 svėrimas



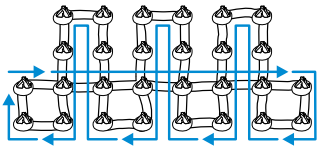
2 svėrimas



25. Ar tai įmanoma?

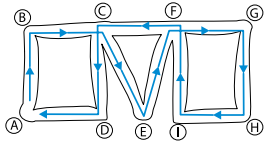
Viršūnės laipsnis - tai visų jos briaunų (linijų) skaičius.

Visos šio grafos viršūnės turi lyginius laipsnius, 2 arba 4. Vadinasi, jis tikrai turi Oilerio kelią.



26. Oilerio ciklo radimas

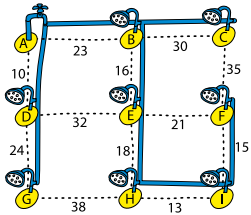
FC 5 žingsnyje tampa tiltu: perėję juo, nebegalėsime sugrįžti į nekeliautą dešiniąją grafo dalį. Kanalai FG ir FI yra visiškai vienodi, galime rinktis bet kurį iš jų.
Šio grafo Oilerio ciklas randas per 11 žingsnių.



27. Vamzdynas

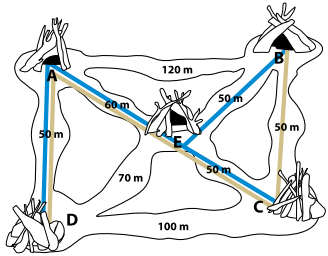
Tolesniais žingsniais nutiestume vamzdžius AB, DG ir BC, visi kiti vamzdžiai sudarys ciklus ir jų netiesime.

Tinklo nutiesimas kainuos:
10+24+23+30+16+18+13+15 = 149.



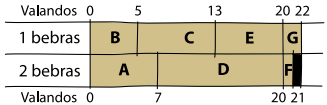
28. Kanalai

Pritaikome Kruskalo algoritmą: 1. Pažymime CE (40); 2. Pažymime AD (50); 3. Pažymime BC (50) arba BE (50): abejų negalima, nes susidarytų ciklas; 4. Pažymime AE (60). Visos kitos briaunos sudarytų ciklus. Bendras mažiausias kanalų ilgis bus 200 m. Kadangi yra dvi galimybės rinktis vieną iš briaunų BC arba BE, tai galima sudaryti du skirtingus kanalus:



29. Du bebrai dirba

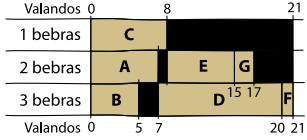
Iš parengto tvarkaraščio matome, kad kažkas negerai: vienas bebras darbuosis 26 valandas, o kitas - tik 17. Mažėjančių trukmių algoritmas patarė mums rinktis dvi ilgiausias užduotis - C(8) ir A(7). Tai buvo trumparegiška strategija. Jei būtume pažvelgę, kas laukia toliau, būtume pamatę, kad B(5) yra kur kas geresnė užduotis, nes atlikę A ir B užduotis bebrai galės imtis ilgiausios užduoties D(13).
Pakeitę užduočių atlikimo tvarką - pradžioje vietoj C paėmę B - pamatytumėme, kad užtvanką galima pastatyti per 22 valandas! Štai tvarkaraštis:



Dar kartą įsitikiname, kad siekdami greitos naudos ir neatsižvelgdami į ilgalaikes veiksmų pasekmes, galime priimti blogą sprendimą.

30. Trys bebrai dirba

Mažėjančių trukmių sąrašas atrodytų taip: D(13), C(8), A(7), E(7), B(5), G(2), F(1). Tačiau užduoties D(13) negalima atlikti tol, kol neatliktos užduotys A ir B. Sudarome tvarkaraštį pagal mažėjančių trukmių algoritmą:



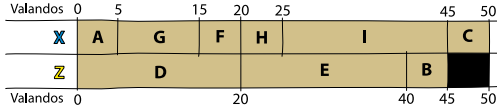
Taikydami mažėjančių trukmių algoritmą trys bebrai pastatys užtvanką per 21 valandą.

31. Užduočių planavimas

1 žingsnis. Užrašome kritinių kelių ilgius (laužtiniuose skliaustuose): F(1)[1] ir G(2)[2].
2 žingsnis. Užrašome kritinio kelio D(13)[13+1=14] ilgį.
3 žingsnis. Užrašome kritinio kelio E(7)[7+2=9] ilgį.
4 žingsnis. Užrašome kritinio kelio A(7)[7+14=21] ilgį.
5 žingsnis. Užrašome kritinio kelio B(5)[5+14=19] ilgį.
6 žingsnis. Užrašome kritinio kelio C(8)[8+9=17] ilgį.
7 žingsnis. Viso darbo kritinis kelias (ir užduočių atlikimo laikas) bus didžiausias iš A, B ir C kritinių kelių, t. y. 21.

32. Darbų planavimas dviems robotams

Sudarome kritinių kelių sąrašą: A(5)[35], G(10)[30], F(5)[25], H(5)[25], D(20)[20], E(20)[20], I(20)[20], B(5)[5], C(50[5]. Naudodamiesi šiuo prioritetiniu darbų sąrašu, sudarysime tvarkaraštį. Pastebėkime, kad nors G, F ir H užduotys yra prioritetinės, robotas Z negali jų rinktis, kol nebaigta A užduotis.

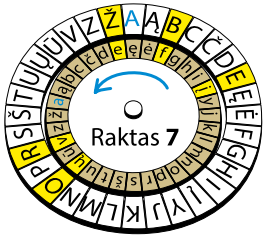


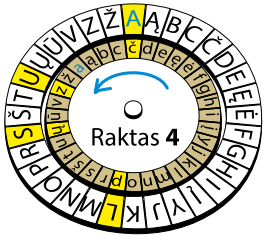
Darbo trukmė visoms užduotims atlikti yra 50 valandų.

33. Kas parašyta?

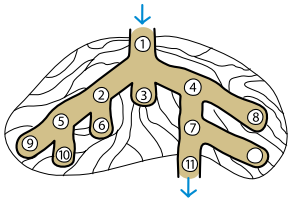
Tekstas užkoduotas stumiant raides per 3 pozicijas į dešinę, vadinasi, atkoduojant, reikės stumti į kairę. Bebras parašė: AR ĮMANOMAS DARNUS BEBRO IR ŽMOGAUS SAMBŪVIS

34. Kas užsakyta pietums?
Bebro užsisakė: BERŽO SULA

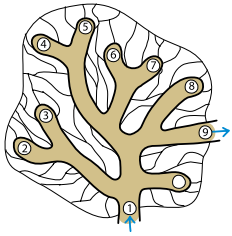


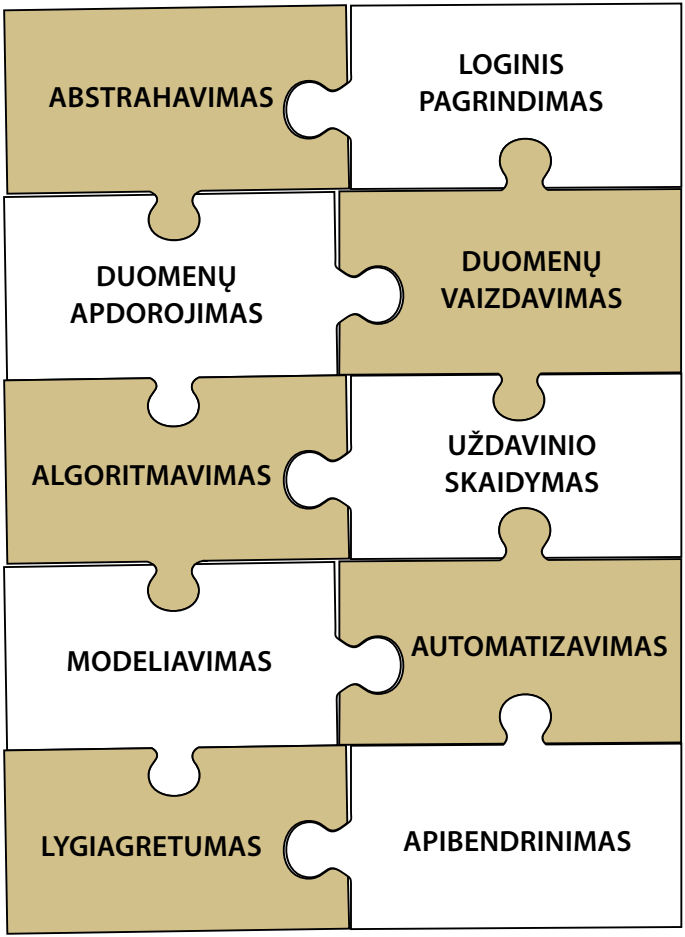


35. POŽEMYJE
Paieškos į plotį algoritmas veikia pagal griežtą taisyklę: pradėjus nuo pradžios, aplankomos visos viršūnės, kurios pasiekiamos vienu ėjimu, tuomet - kurios pasiekiamos dviem ėjimais ir taip toliau.



36. URVUOSE
Skirtingai negu grįžimo metodas, paieškos į gylį algoritmas yra efektyvus algoritmas, kadangi kiekviena grafo viršūnė aplankoma tik vieną kartą.




ABSTRAHAVIMAS
LOGINIS PAGRINDIMAS
DUOMENŲ APDOROJIMAS
DUOMENŲ VAIZDAVIMAS
ALGORITMAVIMAS
UŽDAVINIO SKAIDYMAS
MODELIAVIMAS
AUTOMATIZAVIMAS
LYGIAGRETUMAS
APIBENDRINIMAS

Informatinis mąstymas	Aprašymas
Abstrahavimas	Mokytis įžvelgti svarbias detales reiškiniuose ar procesuose, nepaisyti perteklinių, nebūtinų komponentų. Atskleisti uždavinio esmę. Pavaizduoti procesus schemomis ar kitokiais vaizdavimo būdais.
Loginis pagrindimas	Sprendžiant uždavinius remtis loginiu samprotavimu, indukcija ir dedukcija, taikyti logines operacijas, dėsnius. Nuosekliai argumentuoti, pagrįsti teiginius, daryti išvadas.
Duomenų apdorojimas	Įvaldyti veiksmus su duomenimis: rinkti, atrinkti, rūšiuoti, klasifikuoti, sisteminti duomenis. Atpažinti pagrindinius duomenų tipus ir struktūras. Tvarkyti duomenų rinkinius.
Duomenų vaizdavimas	Pavaizduoti ir susisteminti duomenis jiems tinkamose diagramose, lentelėse, žodžiuose ar nuotraukose.
Algoritmavimas	Spręsti uždavinius, kuriems būdingas taisyklių ir komandų taikymas. Skaityti algoritmų tekstus. Suprasti algoritmo vykdymą. Atpažinti paprastus algoritmus ir taikyti juos įvairioms problemoms spręsti.
Uždavinio skaidymas	Skaidyti duomenis, procesus ar problemas į mažesnes, lengviau tvarkomas dalis.
Modeliavimas	Kurti modelį, kuris imituoja realaus pasaulio procesus. Modeliavimas taip pat įtraukia eksperimentų bandymus.
Automatizavimas	Užprogramuoti kompiuterius ar mechanizmus, kad atliktų pasikartojančias ar nuobodžias užduotis.
Lygiagretumas	Tuo pačiu metu atlikti mažesnes užduotis iš didelės užduoties, kad efektyviau būtų pasiektas bendras tikslas.
Apibendrinimas	Identifikuoti šablonus, panašumus ir dėsningumus. Naujas problemas spręsti remiantis analogiškomis išspręstomis problemomis. Gautus sprendimus apibendrinti (indukcija).