Įvadas: algoritmų pavyzdžiai ir jų sudėtingumo teorijos pradmenys

Raimondas Čiegis

Matematinio modeliavimo katedra, e-paštas: rc@vgtu.lt

Rugsėjo 1 d., 2022

Informatikos, matematikos ir kitų dalykų paskaitose jau nagrinėjote daug algoritmų.

Informatikos, matematikos ir kitų dalykų paskaitose jau nagrinėjote daug algoritmų.

Laisvai interpretuodami šią sąvoką, sakysime, kad algoritmas yra veiksmų ir taisyklių rinkinys, juos atlikę pasiekiame savo tikslą (išsprendžiame uždavinį).

Informatikos, matematikos ir kitų dalykų paskaitose jau nagrinėjote daug algoritmų.

Laisvai interpretuodami šią sąvoką, sakysime, kad algoritmas yra veiksmų ir taisyklių rinkinys, juos atlikę pasiekiame savo tikslą (išsprendžiame uždavinį).

Nesunku surasti įdomių pavyzdžių ir aplink mus.

1. Norime pietums pagminti cepelinus.

Surandame kulinarinių receptų knygą, joje yra aprašyti visi veiksmai, kuriuos reikia atlikti, bei jų eiliškumas. Taigi turime cepelinų gaminimo algoritmą.

2. Reikia išlaikyti "Algoritmų teorijos" egzaminą.

Vėl naudosime atitinkamą algoritmą:

a) atlikite visus laboratorinius darbus,

b) išspręskite namų darbų užduotis,

c) išlaikykite tarpinį egzaminą,

d) gaukite teigiamą sesijos egzamino vertinimą.

Viskas paprasta, tikslu ir aišku, lieka tik tvarkingai įvykdyti šį algoritmą.

3. Norime surinkti Rubiko kubą.

Internete rasite keletą (daug) algoritmų, kaip spręsti šią užduotį.

3. Norime surinkti Rubiko kubą.

Internete rasite keletą (daug) algoritmų, kaip spręsti šią užduotį.

Natūralus klausimas, kuris iš šių algoritmų yra geresnis?

Kaip atlikti tokį palyginimą?

Šiame kurse susisteminsime Jūsų jau turimas žinias, taip pat išmoksite daug naujų naudingų dalykų:

a) kaip sudaryti naują algoritmą;

b) kaip efektyviai realizauoti algoritmus;

c) susipažinsite su svarbių uždavinių sprendimo algoritmais, skirtais informacijos paieškos, duomenų rūšiavimo, logistikos uždaviniams.

Šiame kurse susisteminsime Jūsų jau turimas žinias, taip pat išmoksite daug naujų naudingų dalykų:

a) kaip sudaryti naują algoritmą;

b) kaip efektyviai realizauoti algoritmus;

c) susipažinsite su svarbių uždavinių sprendimo algoritmais, skirtais informacijos paieškos, duomenų rūšiavimo, logistikos uždaviniams.

Kur naudojami šie algoritmai?

Paminėsime tik kelias sritis: didieji duomenys ir jų analizė, virtualioji realybė, dirbtinis intelektas, daiktų internetas, skaitmeninis garso, vaizdo, informacijos saugojimas ir perdavimas, medicina, robotika, dirbtiniai neuroniniai tinklai, mašininis mokymasis, kriptovaliutos, kompiuteriniai žaidimai...

Algoritmas yra tiksliai apibrėžta skaičiavimo procedūra, kuria, imdami pradinius duomenis ir atlikę baigtinį skaičių operacijų, gauname rezultatą.

Skaičiavimo procedūrą galime suprasti kaip kompiuterio programą, užrašytą viena iš programavimo kalbų.

Algoritmas yra tiksliai apibrėžta skaičiavimo procedūra, kuria, imdami pradinius duomenis ir atlikę baigtinį skaičių operacijų, gauname rezultatą.

Skaičiavimo procedūrą galime suprasti kaip kompiuterio programą, užrašytą viena iš programavimo kalbų.

Tikslesnis apibrėžimas gaunamas, kai skaičiavimo įrenginys yra garsioji Tiuringo mašina. Pastaroji teorinė mašina ir buvo sukurta, kai Tiuringas pateikė konstruktyvų algoritmo apibrėžimą.

Dabar aptarsime, nuo ko priklauso ir kaip apibrėžiamas algoritmo sudėtingumas.

Dabar aptarsime, nuo ko priklauso ir kaip apibrėžiamas algoritmo sudėtingumas.

Tai labai svarbu, kai norime prognozuoti, kiek laiko reikės kompiuteriui, kad duotuoju algoritmu išspręstų uždavinį.

Dabar aptarsime, nuo ko priklauso ir kaip apibrėžiamas algoritmo sudėtingumas.

Tai labai svarbu, kai norime prognozuoti, kiek laiko reikės kompiuteriui, kad duotuoju algoritmu išspręstų uždavinį.

Informacija apie rytojaus orus bus visai neaktuali, jeigu modeliavimo uždavinį teks spręsti dvi dienas.

Dabar aptarsime, nuo ko priklauso ir kaip apibrėžiamas algoritmo sudėtingumas.

Tai labai svarbu, kai norime prognozuoti, kiek laiko reikės kompiuteriui, kad duotuoju algoritmu išspręstų uždavinį.

Informacija apie rytojaus orus bus visai neaktuali, jeigu modeliavimo uždavinį teks spręsti dvi dienas.

Automatinė automobilio vairavimo sistema turi reaguoti į pasikeitusią situaciją greičiau, nei įvyks avarija.

Dabar aptarsime, nuo ko priklauso ir kaip apibrėžiamas algoritmo sudėtingumas.

Tai labai svarbu, kai norime prognozuoti, kiek laiko reikės kompiuteriui, kad duotuoju algoritmu išspręstų uždavinį.

Informacija apie rytojaus orus bus visai neaktuali, jeigu modeliavimo uždavinį teks spręsti dvi dienas.

Automatinė automobilio vairavimo sistema turi reaguoti į pasikeitusią situaciją greičiau, nei įvyks avarija.

Kiek laiko reikės kenkėjiškai programai, kad ji "nulaužtų" Jūsų banko sąskaitos slaptažodį?

Ar blokų grandinėje patikimai apsaugota informacija apie kriptovaliutas?

Pradinių duomenų skaičius yra labai svarbi uždavinio charakteristika, nes kuo daugiau duomenų, tuo didesnės kompiuterio atminties reikia duomenims saugoti, ir ilgiau vykdoma skaičiavimo programa.

Pradinių duomenų skaičius yra labai svarbi uždavinio charakteristika, nes kuo daugiau duomenų, tuo didesnės kompiuterio atminties reikia duomenims saugoti, ir ilgiau vykdoma skaičiavimo programa.

Pavyzdžiui:

- vektoriaus X duomenų dydis yra lygus jo elementų skaičiui n,

Pradinių duomenų skaičius yra labai svarbi uždavinio charakteristika, nes kuo daugiau duomenų, tuo didesnės kompiuterio atminties reikia duomenims saugoti, ir ilgiau vykdoma skaičiavimo programa.

Pavyzdžiui:

- vektoriaus X duomenų dydis yra lygus jo elementų skaičiui n,
- matricos A, turinčios m eilučių ir n stulpelių, dydis yra mn,

Pradinių duomenų skaičius yra labai svarbi uždavinio charakteristika, nes kuo daugiau duomenų, tuo didesnės kompiuterio atminties reikia duomenims saugoti, ir ilgiau vykdoma skaičiavimo programa.

Pavyzdžiui:

- vektoriaus X duomenų dydis yra lygus jo elementų skaičiui n,
- matricos A, turinčios m eilučių ir n stulpelių, dydis yra mn,
- grafo $G = (V, E)$, kurio viršūnių V skaičius n, o briaunų aibės E dydis m, pradinių duomenų skaičius yra $(m + n)$.

Nors duomenų skaičius ir charakterizuoja uždavinio dydį, tačiau vien tik jis dar neapibūdina algoritmo sudėtingumo. Nagrinėkime du svarbius matricų veiksmus: dviejų matricų sumos $A + B$ ir sandaugos AB skaičiavimą.

Tegul matricų dydis yra n eilučių ir stulpelių, t. y. turime $n \times n$ dydžio kvadratines matricas.

Nors duomenų skaičius ir charakterizuoja uždavinio dydį, tačiau vien tik jis dar neapibūdina algoritmo sudėtingumo. Nagrinėkime du svarbius matricų veiksmus: dviejų matricų sumos $A + B$ ir sandaugos AB skaičiavimą.

Tegul matricų dydis yra n eilučių ir stulpelių, t. y. turime $n \times n$ dydžio kvadratines matricas.

Nagrinėkime matricų sumos $A + B$ skaičiavimo algoritmą

$$C = A + B, \quad C = (c_{ij}), \ \ 1 \leqslant i, j \leqslant n\,,$$

$$c_{ij} = a_{ij} + b_{ij}\,.$$

Taigi iš viso atliekame n^2 sumavimo veiksmų. Šį kartą veiksmų skaičius yra tos pačios eilės dydis, kaip ir pradinių duomenų skaičius, nes abiejų matricų koeficientų yra $2n^2$.

Dabar nagrinėkime dviejų matricų sandaugos $C = AB$ skaičiavimo algoritmą

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \quad 1 \leqslant i, j \leqslant n\,.$$

Iš viso atliekame n^3 daugybos ir $n^2(n-1)$ sumavimo veiksmų, arba $(2n^3 - n^2)$ aritmetinių veiksmų.

Dabar nagrinėkime dviejų matricų sandaugos $C = AB$ skaičiavimo algoritmą

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \quad 1 \leqslant i, j \leqslant n\,.$$

Iš viso atliekame n^3 daugybos ir $n^2(n-1)$ sumavimo veiksmų, arba $(2n^3 - n^2)$ aritmetinių veiksmų.

Taigi matome, kad dviejų matricų daugybos veiksmas yra sudėtingesnis už jų sumos skaičiavimą. Abiejų operacijų sudėtingumą įvertinome naudodami kiekybinį matą – aritmetinių veiksmų skaičių.

Tačiau ne visiems algoritmams tinka šis matas, pavyzdžiui, duomenų rūšiavimo ir paieškos algoritmuose svarbiausi yra duomenų lyginimo ir sekos elementų keitimo vietomis veiksmai.

Tačiau ne visiems algoritmams tinka šis matas, pavyzdžiui, duomenų rūšiavimo ir paieškos algoritmuose svarbiausi yra duomenų lyginimo ir sekos elementų keitimo vietomis veiksmai.

Todėl vartosime bendresnį apibrėžimą:

Algoritmo sudėtingumas yra lygus atliekamų bazinių veiksmų skaičiui.

Tačiau ne visiems algoritmams tinka šis matas, pavyzdžiui, duomenų rūšiavimo ir paieškos algoritmuose svarbiausi yra duomenų lyginimo ir sekos elementų keitimo vietomis veiksmai.

Todėl vartosime bendresnį apibrėžimą:

Algoritmo sudėtingumas yra lygus atliekamų bazinių veiksmų skaičiui.

Tada uždavinio dydžiu (arba apimtimi) vadiname geriausio žinomo jo sprendimo algoritmo sudėtingumą.

Čia verta trumpam sustoti ir aptarti svarbius taikymuose atvejus.

Prisiminsime ir kai kuriuos nesudėtingus matematinius rezultatus, kurie smarkiai palengvina algoritmų analizę.

Čia verta trumpam sustoti ir aptarti svarbius taikymuose atvejus.

Prisiminsime ir kai kuriuos nesudėtingus matematinius rezultatus, kurie smarkiai palengvina algoritmų analizę.

Dažniausiai tą patį algoritmą naudojame spręsdami mus dominantį uždavinį su įvairiais pradiniais duomenimis. Pavyzdžiui, kompanija kiekvieną dieną iš naujo rūšiuoja informaciją apie savo klientus ar duomenų saugyklos įrašus. Kaip matysime duomenų paieška ypač efektyvi, kai duomenys surūšiuoti.

1. Geriausiojo atvejo sudėtingumas.

Duotasis algoritmas gali būti ypatingai efektyvus kai kurių tipų duomenims. Mums svarbu žinoti, kaip greitai tada galime išspręsti uždavinį ir mokėti atpažinti tokius duomenis.

Dažnai toks įvertis yra vadinamas optimistiniu.

1. Geriausiojo atvejo sudėtingumas.

Duotasis algoritmas gali būti ypatingai efektyvus kai kurių tipų duomenims. Mums svarbu žinoti, kaip greitai tada galime išspręsti uždavinį ir mokėti atpažinti tokius duomenis.

Dažnai toks įvertis yra vadinamas optimistiniu.

Pažymėkime D_n visus pradinių duomenų variantus apibrėžiančią aibę, kai kiekviename variante yra n duomenų:

$$D_n = \{d_m : \ d_m = (a_1, a_2, \ldots, a_n), \quad a_j \in A, \ \ j = 1, 2, \ldots, n\}.$$

Pažymėkime $T(d_m)$ algoritmo sudėtingumą, kai turime konkretų pradinių duomenų pasiskirstymą d_m.

1. Geriausiojo atvejo sudėtingumas.

Duotasis algoritmas gali būti ypatingai efektyvus kai kurių tipų duomenims. Mums svarbu žinoti, kaip greitai tada galime išspręsti uždavinį ir mokėti atpažinti tokius duomenis.

Dažnai toks įvertis yra vadinamas optimistiniu.

Pažymėkime D_n visus pradinių duomenų variantus apibrėžiančią aibę, kai kiekviename variante yra n duomenų:

$$D_n = \{d_m : \ d_m = (a_1, a_2, \ldots, a_n), \quad a_j \in A, \ \ j = 1, 2, \ldots, n\}.$$

Pažymėkime $T(d_m)$ algoritmo sudėtingumą, kai turime konkretų pradinių duomenų pasiskirstymą d_m.

Geriausiojo atvejo sudėtingumas

$$T_G(n) = \min_{d_m \in D_n} T(d_m).$$

2. Blogiausiojo atvejo sudėtingumas.

Duotasis algoritmas gali būti neefektyvus kai kurių tipų duomenims. Mums svarbu žinoti, kiek laiko tada gali tekti laukti rezultato ir mokėti atpažinti tokius duomenis.

Dažnai toks įvertis yra vadinamas pesimistiniu.

2. Blogiausiojo atvejo sudėtingumas.

Duotasis algoritmas gali būti neefektyvus kai kurių tipų duomenims. Mums svarbu žinoti, kiek laiko tada gali tekti laukti rezultato ir mokėti atpažinti tokius duomenis.

Dažnai toks įvertis yra vadinamas pesimistiniu.

Blogiausiojo atvejo sudėtingumas

$$T_B(n) = \max_{d_m \in D_n} T(d_m)\,.$$

3. Vidutiniojo atvejo sudėtingumas.

Duotasis algoritmas gali būti neefektyvus kai kurių tipų duomenims, bet dažnai tokių rinkinių yra nedaug. Todėl daugumai vartotojų svarbiausia žinoti, kiek laiko vidutinškai teks laukti rezultato. Tokia informacija leidžia racionaliai išnaudoti žmogiškuosius ir gamybinius resursus.

3. Vidutiniojo atvejo sudėtingumas.

Duotasis algoritmas gali būti neefektyvus kai kurių tipų duomenims, bet dažnai tokių rinkinių yra nedaug. Todėl daugumai vartotojų svarbiausia žinoti, kiek laiko vidutinškai teks laukti rezultato. Tokia informacija leidžia racionaliai išnaudoti žmogiškuosius ir gamybinius resursus.

Vidutinis (arba tikėtinas) algoritmo sudėtingumas

$$T_V(n) = \frac{1}{|D_n|} \sum_{d_m \in D_n} T(d_m),$$

3. Vidutiniojo atvejo sudėtingumas.

Duotasis algoritmas gali būti neefektyvus kai kurių tipų duomenims, bet dažnai tokių rinkinių yra nedaug. Todėl daugumai vartotojų svarbiausia žinoti, kiek laiko vidutinškai teks laukti rezultato. Tokia informacija leidžia racionaliai išnaudoti žmogiškuosius ir gamybinius resursus.

Vidutinis (arba tikėtinas) algoritmo sudėtingumas

$$T_V(n) = \frac{1}{|D_n|} \sum_{d_m \in D_n} T(d_m)\,,$$

Matematinius metodus, kurie leidžia įvertinti svarbiausių algoritmų sudėtingumą, prisiminsime, kai nagrinėsime konkrečius taikomuosius algoritmus.

Algoritmų sudarymo metodai

Naujų uždavinių, kuriuos turime išspręsti, atsiranda kasdien. Dažnai jie yra netrivialūs, jų sprendimui neužtenka naudoti tik jau egzistuojančius algoritmus.

Algoritmų sudarymo metodai

Naujų uždavinių, kuriuos turime išspręsti, atsiranda kasdien. Dažnai jie yra netrivialūs, jų sprendimui neužtenka naudoti tik jau egzistuojančius algoritmus.

Kaip sugalvoti (sukonstruoti, sudaryti) uždaviniui tinkamą algoritmą? Nuo ko pradėti? Kiek laiko teks mokytis, kad įgytume reikalingus įgūdžius?

Algoritmų sudarymo metodai

Naujų uždavinių, kuriuos turime išspręsti, atsiranda kasdien. Dažnai jie yra netrivialūs, jų sprendimui neužtenka naudoti tik jau egzistuojančius algoritmus.

Kaip sugalvoti (sukonstruoti, sudaryti) uždaviniui tinkamą algoritmą? Nuo ko pradėti? Kiek laiko teks mokytis, kad įgytume reikalingus įgūdžius?

Geroji žinia yra ta, kad egzistuoja tik keletas pagrindinių metodų, kurie ir naudojami naujų algoritmų sudarymui. Taigi nereikės klaidžioti tarp gausybės hipotezių, kad rastume tinkamą uždavinio sprendimo būdą.

Deja, tai nereiškia, kad susipažinus su pagrindiniais algoritmavimo metodais be didesnių pastangų galėsite sukonstruoti naujo uždavinio sprendiklį.

Deja, tai nereiškia, kad susipažinus su pagrindiniais algoritmavimo metodais be didesnių pastangų galėsite sukonstruoti naujo uždavinio sprendiklį.

Nepažadu, kad nereikės skaityti vadovėlių – nebus taip, kad po mėnesio galėsite patys atrasti, sukurti bet kurį, net ir garsiausią algoritmą.

Bet pamatysime, kad net ir Top 10 sąrašo algoritmai yra sukurti naudojant tuos bendrus metodus, kuriuos dabar ir nagrinėsime. O tai jau stipriai motyvuojantis argumentas.

Uždavinio skaidymas

Dauguma algoritmavimo metodų remiasi tokiu paprastu bendruoju principu – uždavinį dalijame į mažesnes užduotis, kurias išsprendę sukonstruojame viso uždavinio sprendinį.

Uždavinio skaidymas

Dauguma algoritmavimo metodų remiasi tokiu paprastu bendruoju principu – uždavinį dalijame į mažesnes užduotis, kurias išsprendę sukonstruojame viso uždavinio sprendinį.

Tenka atsakyti į tokius pagrindinius du klausimus:

Uždavinio skaidymas

Dauguma algoritmavimo metodų remiasi tokiu paprastu bendruoju principu – uždavinį dalijame į mažesnes užduotis, kurias išsprendę sukonstruojame viso uždavinio sprendinį.

Tenka atsakyti į tokius pagrindinius du klausimus:

1. Kaip padalinti uždavinį į baigtinį skaičių mažesnių užduočių (variantų).

Uždavinio skaidymas

Dauguma algoritmavimo metodų remiasi tokiu paprastu bendruoju principu – uždavinį dalijame į mažesnes užduotis, kurias išsprendę sukonstruojame viso uždavinio sprendinį.

Tenka atsakyti į tokius pagrindinius du klausimus:

1. Kaip padalinti uždavinį į baigtinį skaičių mažesnių užduočių (variantų).
2. Kaip sumažinti nagrinėjamų variantų skaičių, nes tiesioginis visų variantų patikrinimas gali būti neįvykdomas net su greičiausiais superkompiuteriais.

Variantų perrinkimas

Nagrinėsime vieną tokį metodą – variantų perrinkimą. Šis metodas ypač išpopuliarėjo, kai atsirado kompiuteriai.

Variantų perrinkimas

Nagrinėsime vieną tokį metodą – variantų perrinkimą. Šis metodas ypač išpopuliarėjo, kai atsirado kompiuteriai.

Jo svarbus pranašumas yra tai, kad, išnagrinėjus visus variantus, nebereikia atlikti jokių papildomų skaičiavimų.

Variantų perrinkimas

Nagrinėsime vieną tokį metodą – variantų perrinkimą. Šis metodas ypač išpopuliarėjo, kai atsirado kompiuteriai.

Jo svarbus pranašumas yra tai, kad, išnagrinėjus visus variantus, nebereikia atlikti jokių papildomų skaičiavimų.

Kaip sužinoti Petro vaikų amžių?

Susitiko du senokai nesimatę draugai – Jonas ir Petras. Pokalbio metu paaiškėjo, kad ši diena ypatinga Petrui, nes visi trys jo vaikai – Inga, Julija ir Justas – šiandien švenčia gimtadienius. Petras pasiūlė Jonui, geram matematikui, pabandyti atspėti, koks yra kiekvieno vaiko amžius.

Norėdamas palengvinti užduotį, jis nurodė, kad seserys yra ne vyresnės už Justą, o Inga neturi jaunesnės sesers.

Norėdamas palengvinti užduotį, jis nurodė, kad seserys yra ne vyresnės už Justą, o Inga neturi jaunesnės sesers.

Taip pat Petras informavo, kad sudauginę visų trijų vaikų metus, gauname skaičių 36.

Norėdamas palengvinti užduotį, jis nurodė, kad seserys yra ne vyresnės už Justą, o Inga neturi jaunesnės sesers.

Taip pat Petras informavo, kad sudauginę visų trijų vaikų metus, gauname skaičių 36.

Šiek tiek pagalvojęs Jonas pareiškė, kad jam dar neužtenka informacijos. Tada Petras pasakė, kad vaikų metų suma sutampa su namo, prie kurio jie stovi, langų skaičiumi.

Norėdamas palengvinti užduotį, jis nurodė, kad seserys yra ne vyresnės už Justą, o Inga neturi jaunesnės sesers.

Taip pat Petras informavo, kad sudauginę visų trijų vaikų metus, gauname skaičių 36.

Šiek tiek pagalvojęs Jonas pareiškė, kad jam dar neužtenka informacijos. Tada Petras pasakė, kad vaikų metų suma sutampa su namo, prie kurio jie stovi, langų skaičiumi.

Jonas vėl pagalvojo ir pareiškė, kad naujoji informacija tikrai labai svarbi, bet jos dar nepakanka, kad galėtų pasakyti atsakymą. Todėl reikėtų mažos pagalbos.

Norėdamas palengvinti užduotį, jis nurodė, kad seserys yra ne vyresnės už Justą, o Inga neturi jaunesnės sesers.

Taip pat Petras informavo, kad sudauginę visų trijų vaikų metus, gauname skaičių 36.

Šiek tiek pagalvojęs Jonas pareiškė, kad jam dar neužtenka informacijos. Tada Petras pasakė, kad vaikų metų suma sutampa su namo, prie kurio jie stovi, langų skaičiumi.

Jonas vėl pagalvojo ir pareiškė, kad naujoji informacija tikrai labai svarbi, bet jos dar nepakanka, kad galėtų pasakyti atsakymą. Todėl reikėtų mažos pagalbos.

Petro pastaba buvo trumpa – vyriausiojo vaiko akys yra mėlynos. Sužinojęs tai, Jonas iš karto pasakė kiekvieno vaiko amžių!

Iš pirmosios sąlygos sužinome, kad trijų vaikų metų sandauga lygi 36. Nesunku patikrinti, kad yra tik aštuoni skirtingi variantai, kai įvykdoma ši sąlyga. Jie pateikti lentelėje.

TABLE : Aštuoni variantai, kai vaikų metų sandauga lygi 36

Vardas	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
Inga	1	1	1	1	1	2	2	3
Julija	1	2	3	4	6	2	3	3
Justas	36	18	12	9	6	9	6	4

Darome prielaidą, kad Jonas žinojo, kiek langų turi namas, prie kurio susitiko draugai, taigi iš antrosios sąlygos jis sužinojo, kam lygi vaikų metų suma. Tačiau tokios informacijos jam vis dar neužteko, kad galėtų pasakyti atsakymą.

Apskaičiuokime kiekvieno varianto vaikų metų sumas:

$$\begin{aligned} 1+1+36&=38, & 1+2+18&=21,\\ 1+3+12&=16, & 1+4+\ \ 9&=14,\\ 1+6+\ \ 6&=13, & 2+2+\ \ 9&=13,\\ 2+3+\ \ 6&=11, & 3+3+\ \ 4&=10. \end{aligned}$$

Darome prielaidą, kad Jonas žinojo, kiek langų turi namas, prie kurio susitiko draugai, taigi iš antrosios sąlygos jis sužinojo, kam lygi vaikų metų suma. Tačiau tokios informacijos jam vis dar neužteko, kad galėtų pasakyti atsakymą.

Apskaičiuokime kiekvieno varianto vaikų metų sumas:

$$\begin{aligned} 1+1+36&=38, & 1+2+18&=21,\\ 1+3+12&=16, & 1+4+\ \ 9&=14,\\ 1+6+\ \ 6&=13, & 2+2+\ \ 9&=13,\\ 2+3+\ \ 6&=11, & 3+3+\ \ 4&=10. \end{aligned}$$

Dabar tampa aišku, kad ši metų suma yra lygi 13, nes visais kitais atvejais Jonas jau žinotų kiekvieno vaiko amžių.

Liko du variantai – (1, 6, 6) ir (2, 2, 9).

Liko du variantai – (1, 6, 6) ir (2, 2, 9).

Kadangi tik antruoju atveju Justas yra vyriausias vaikas (tai, kad jo akys mėlynos, aišku, neturi jokios reikšmės), darome išvadą, kad Petras augina dvynukes Ingą ir Juliją, kurioms sukako dveji metukai, ir devynerių metų sūnų Justą.

Informacijos šifravimas, slaptažodžiai

Dažniausiai ieškome naujų algoritmų, kurių sudėtingumas būtų minimalus. Tačiau dabar pateiksime pavyzdį, kai visų variantų perrinkimo uždavinio sudėtingumas yra siektina ir naudinga savybė.

Informacijos šifravimas, slaptažodžiai

Dažniausiai ieškome naujų algoritmų, kurių sudėtingumas būtų minimalus. Tačiau dabar pateiksime pavyzdį, kai visų variantų perrinkimo uždavinio sudėtingumas yra siektina ir naudinga savybė.

Komunikacijos slaptumas ir pranešimo perdavimo saugumas šiuo metu yra labai svarbi užduotis ir jos svarba tik didės ateityje. Smarkiai auga nesankcionuoto informacijos perskaitymo ar pakeitimo, hibridinių atakų grėsmė.

Informacijos šifravimas, slaptažodžiai

Dažniausiai ieškome naujų algoritmų, kurių sudėtingumas būtų minimalus. Tačiau dabar pateiksime pavyzdį, kai visų variantų perrinkimo uždavinio sudėtingumas yra siektina ir naudinga savybė.

Komunikacijos slaptumas ir pranešimo perdavimo saugumas šiuo metu yra labai svarbi užduotis ir jos svarba tik didės ateityje. Smarkiai auga nesankcionuoto informacijos perskaitymo ar pakeitimo, hibridinių atakų grėsmė.

Šiuolaikinių viešojo rakto kriptografinių algoritmų saugumas yra grindžiamas matematiniais rezultatais, kad kai kurių skaičių teorijos uždavinių sprendimas yra galimas tik atlikus pilnąjį variantų perrinkimą, o variantų skaičius yra toks didelis, kad slaptažodžio "nulaužimas" nėra galimas per laiką, kol informacija dar yra aktuali.

Rekursijos metodas

Sakysime, kad objektas (uždavinys, matematinė struktūra, skaitmeninių duomenų rinkinys) yra apibrėžtas rekursijos būdu, jei apibrėžime vėl naudojami vienas arba keli tokie patys objektai, tik mažesnio dydžio.

Rekursijos metodas

Sakysime, kad objektas (uždavinys, matematinė struktūra, skaitmeninių duomenų rinkinys) yra apibrėžtas rekursijos būdu, jei apibrėžime vėl naudojami vienas arba keli tokie patys objektai, tik mažesnio dydžio.

Faktorialas. Natūraliojo skaičiaus $n \in \mathbb{N}$ faktorialas yra apibrėžiamas taip:

$$n! = \begin{cases} n \cdot (n-1)!, & \text{jei } n > 0, \\ 1, & \text{jei } n = 0. \end{cases}$$

Rekursijos metodas

Sakysime, kad objektas (uždavinys, matematinė struktūra, skaitmeninių duomenų rinkinys) yra apibrėžtas rekursijos būdu, jei apibrėžime vėl naudojami vienas arba keli tokie patys objektai, tik mažesnio dydžio.

Faktorialas. Natūraliojo skaičiaus $n \in \mathbb{N}$ faktorialas yra apibrėžiamas taip:

$$n! = \begin{cases} n \cdot (n-1)!, & \text{jei } n > 0, \\ 1, & \text{jei } n = 0. \end{cases}$$

Iš šio apibrėžimo gauname tokią lygybę

$$n! = n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1 \,.$$

Fibonačio skaičiai. Fibonačio skaičiai yra apibrėžiami tokiu būdu:

$$f_n = \begin{cases} f_{n-1} + f_{n-2}, & \text{jei } n > 1, \\ f_0 = 1, \quad f_1 = 1\,. \end{cases}$$

Fibonačio skaičiai. Fibonačio skaičiai yra apibrėžiami tokiu būdu:

$$f_n = \begin{cases} f_{n-1} + f_{n-2}, & \text{jei } n > 1, \\ f_0 = 1, \quad f_1 = 1\,. \end{cases}$$

Naudodami šį apibrėžimą apskaičiuokite f_4 reikšmę.

Fibonačio skaičiai. Fibonačio skaičiai yra apibrėžiami tokiu būdu:

$$f_n = \begin{cases} f_{n-1} + f_{n-2}, & \text{jei } n > 1, \\ f_0 = 1, \quad f_1 = 1\,. \end{cases}$$

Naudodami šį apibrėžimą apskaičiuokite f_4 reikšmę.

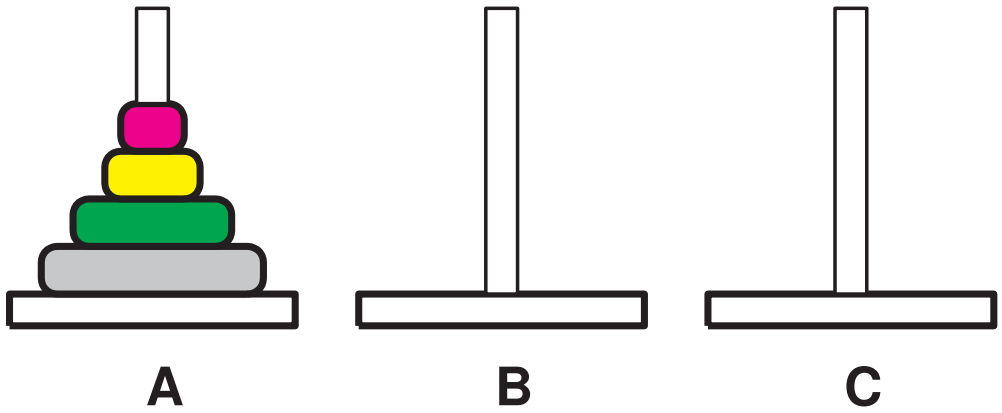
O dabar apskaičiuokite f_{30} ir f_{31} reikšmes. Kokią išvadą galite padaryti?

Parodysime, kaip rekursija padeda išspręsti loginių žaidimų ir galvosūkių užduotis.

Senovės Kinijoje buvo žaidžiamas toks žaidimas: turime tris virbus A, B ir C. Ant A virbo suverti n skirtingo skersmens žiedai: apačioje yra didžiausias žiedas, ant jo uždėtas mažesnis ir taip toliau iki mažiausio, kuris dedamas viršuje. Šiuos žiedus reikia perkelti ant kito virbo C, kai ant viršaus galima dėti tik mažesnio skersmens žiedą bei galime naudoti pagalbinį B virbą.

Parodysime, kaip rekursija padeda išspręsti loginių žaidimų ir galvosūkių užduotis.

Senovės Kinijoje buvo žaidžiamas toks žaidimas: turime tris virbus A, B ir C. Ant A virbo suverti n skirtingo skersmens žiedai: apačioje yra didžiausias žiedas, ant jo uždėtas mažesnis ir taip toliau iki mažiausio, kuris dedamas viršuje. Šiuos žiedus reikia perkelti ant kito virbo C, kai ant viršaus galima dėti tik mažesnio skersmens žiedą bei galime naudoti pagalbinį B virbą.



Žaidimo strategiją patogu apibrėžti naudojant rekursiją.

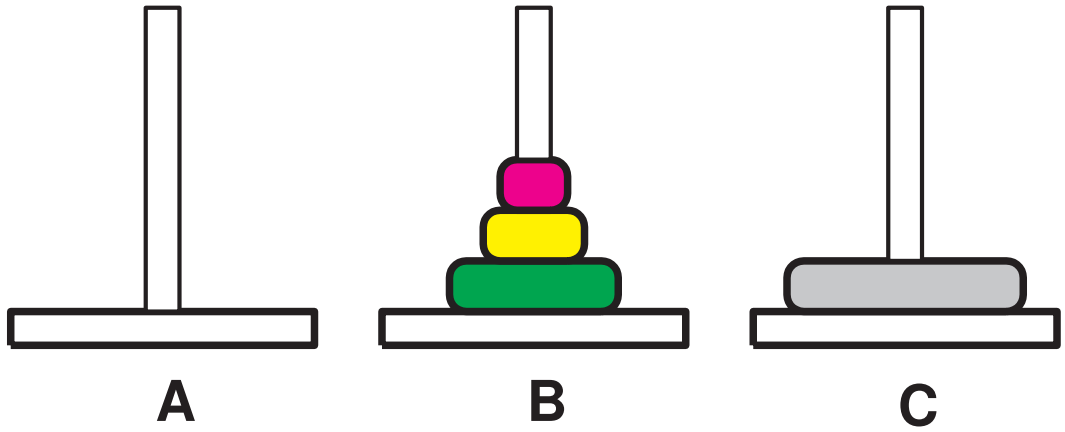
Žaidimo strategiją patogu apibrėžti naudojant rekursiją.

Pirmiausia, laikydamiesi žaidimo taisyklių, perkeliame $(n-1)$ viršutinius A žiedus ant B virbo, o C virbu naudojamės kaip pagalbiniu.

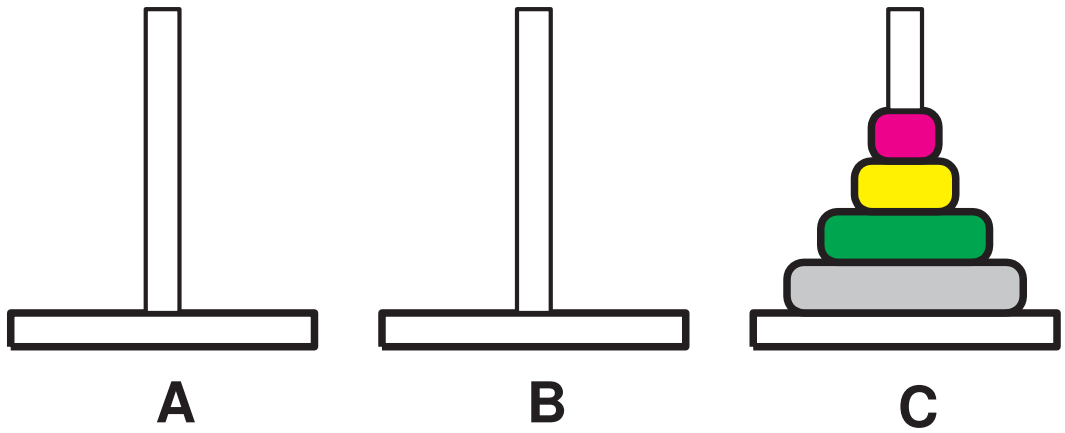
Žaidimo strategiją patogu apibrėžti naudojant rekursiją.

Pirmiausia, laikydamiesi žaidimo taisyklių, perkeliame $(n-1)$ viršutinius A žiedus ant B virbo, o C virbu naudojamės kaip pagalbiniu.

Tada perkeliame nuo A ant C didžiausią žiedą



Paskutiniu žingsniu tuo pačiu būdu perkeliame žiedus nuo B ant C virbo, o A naudojame kaip pagalbinį.



Hanojaus bokštų uždavinio sprendimo algoritmas

```
HanojausBokštai (n, A, B, C)
begin
  (1)  if  (  n > 0  )
  (2)     HanojausBokštai (n-1, A, C, B);
  (3)     move (A, C);
  (4)     HanojausBokštai (n-1, B, A, C);
  (5)  end if
end  HanojausBokštai
```

Skaldyk ir valdyk metodas

Skaldyk ir valdyk metodas

1. Uždavinį skaidome į kelis mažesnius uždavinius.

Skaldyk ir valdyk metodas

1. Uždavinį skaidome į kelis mažesnius uždavinius.
2. Randame šių uždavinių sprendinius.

Skaldyk ir valdyk metodas

1. Uždavinį skaidome į kelis mažesnius uždavinius.
2. Randame šių uždavinių sprendinius.
3. Iš jų sudarome viso uždavinio sprendinį.

Skaldyk ir valdyk metodas

1. Uždavinį skaidome į kelis mažesnius uždavinius.
2. Randame šių uždavinių sprendinius.
3. Iš jų sudarome viso uždavinio sprendinį.

Skaldyk ir valdyk metodas

1. Uždavinį skaidome į kelis mažesnius uždavinius.
2. Randame šių uždavinių sprendinius.
3. Iš jų sudarome viso uždavinio sprendinį.

Mažesnius uždavinius vėl galime spręsti tokiu pačiu metodu, taip skaidome tol, kol gautieji uždaviniai yra lengvai išsprendžiami.

Skaldyk ir valdyk metodas

1. Uždavinį skaidome į kelis mažesnius uždavinius.
2. Randame šių uždavinių sprendinius.
3. Iš jų sudarome viso uždavinio sprendinį.

Mažesnius uždavinius vėl galime spręsti tokiu pačiu metodu, taip skaidome tol, kol gautieji uždaviniai yra lengvai išsprendžiami.

Toks rekursyvus algoritmas ir vadinamas skaldyk ir valdyk metodu (angl. *divide and conquer*).

Šis metodas yra darbinis arkliukas konstruojant daug elitinių algoritmų, todėl šioje paskaitoje nengrinėsime rimtesnių jo taikymų.

Šis metodas yra darbinis arkliukas konstruojant daug elitinių algoritmų, todėl šioje paskaitoje nengrinėsime rimtesnių jo taikymų.

Apsiribosime tik populiariu pavyzdžiu, kuris net vaikams paaiškina šio metodo esmę.

Šis metodas yra darbinis arkliukas konstruojant daug elitinių algoritmų, todėl šioje paskaitoje nengrinėsime rimtesnių jo taikymų.

Apsiribosime tik populiariu pavyzdžiu, kuris net vaikams paaiškina šio metodo esmę.

Saloje medžioja liūtas, kurį vietiniai gyventojai nori sugauti ir perkelti į kito draustinio teritoriją.

Šis metodas yra darbinis arkliukas konstruojant daug elitinių algoritmų, todėl šioje paskaitoje nengrinėsime rimtesnių jo taikymų.

Apsiribosime tik populiariu pavyzdžiu, kuris net vaikams paaiškina šio metodo esmę.

Saloje medžioja liūtas, kurį vietiniai gyventojai nori sugauti ir perkelti į kito draustinio teritoriją.

Tada jie panaudoja tokį skaldyk ir valdyk algoritmo variantą. Per salą užtveria tvorą ir visą teritoriją padalina į dvi dalis. Liūtas medžioja vienoje iš jų. Po pirmojo algoritmo žingsnio pavyko sumažinti teritoriją dvigubai.

Šis metodas yra darbinis arkliukas konstruojant daug elitinių algoritmų, todėl šioje paskaitoje nengrinėsime rimtesnių jo taikymų.

Apsiribosime tik populiariu pavyzdžiu, kuris net vaikams paaiškina šio metodo esmę.

Saloje medžioja liūtas, kurį vietiniai gyventojai nori sugauti ir perkelti į kito draustinio teritoriją.

Tada jie panaudoja tokį skaldyk ir valdyk algoritmo variantą. Per salą užtveria tvorą ir visą teritoriją padalina į dvi dalis. Liūtas medžioja vienoje iš jų. Po pirmojo algoritmo žingsnio pavyko sumažinti teritoriją dvigubai.

Toliau algoritmas kartojamas jau toje dalyje, kurioje gyvena liūtas. Ir taip po baigtinio skaičiaus dalijimų pavyksta liūtą uždaryti mažoje salelėje.