Skaičiavimo metodų ir matematinio modeliavimo

seminaras Nr. 1 (108)

Kaip sprendžiame neišsprendžiamus uždavinius

Raimondas ČIEGIS

Vilniaus Gedimino Technikos Universitetas,

Saulėtekio al. 11, Vilnius, Lietuva

e-mail: rc@fm.vgtu.lt

Matematinių uždavinių sudėtingumas

Matematinių uždavinių sudėtingumo vertinimo būdai yra labai įvairūs.

Egzistuoja daug pavyzdžių apie klasikinių uždavinių (jų formuluotės labai paprastos, suprantamos ir penktokui) sprendinio egzistavimo arba nebuvimo įrodymo sudėtingumą.

Prisiminkime didžiąją Ferma teoremą

Lygtis

$$x^n + y^n = z^n, \quad n > 2$$

neturi sprendinių teigiamų sveikųjų skaičių aibėje.

Lengviausias priešingo teiginio įrodymo būdas yra surasti (atspėti) kontrapavyzdį.

Pavyzdžiui, jei $n = 2$, tai:

$$3^2 + 4^2 = 5^2.$$

Todėl ir didžiąją Fermą teoremą bandyta įrodyti arba paneigti remiantis paprastais pertvarkymais, loginiais samprotavimais arba variantų perrinkimu.

Teoremos įrodymo istorija

Atvejį, kai n=3, įrodė Leonardas Oileris, kai n=4 – pats Ferma, n=5 – Ležandras, n=7 – Lamė, n=14 – Leženas-Dirichlė.

Teisingas pilnas teoremos įrodymas pateiktas tik po 357 metų matematiko Andrew Wiles, išspausdinusio įrodymą 1995 metais (Fieldso medalis 1998 metais Berlyne)

Ryšys tarp Ferma teoremos ir Shimura–Taniyama hipotezės

If p is an odd prime and a, b, and c are positive integers such that $ap + bp = cp$, then a corresponding equation

$$y^2 = x(x - ap)(x + bp)$$

defines a hypothetical elliptic curve, called the Frey curve, which must exist if there is a counterexample to Fermat's Last Theorem.

Algoritmų sudėtingumo analizė

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.

Nagrinėkime dviejų matricų sandaugos

$$C = AB$$

skaičiavimo algoritmą

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

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

Ar galima apskaičiuoti dviejų matricų sandaugą greičiau?

Taip!

Štraseno algoritmu, atliekame $\mathcal{O}(n^{\log 7})$ veiksmų.

Uždaviniai, kurių sprendiniui rasti žinome **polinominio** sudėtingumo $\mathcal{O}(n^p)$ algoritmus, yra "išsprendžiami" net kai jų dimensija $n \gg 1$ (aišku, jei p nėra didelis skaičius).

- tiesinė algebra,
- tikrinių reikšmių radimo uždaviniai,
- matematinės fizikos uždaviniai
- ...

Nepolinominio sudėtingumo uždaviniai

Turime n skirtingų objektų $(a_1, a_2, \ldots, a_n)$.

Reikia sugeneruoti visus skirtingus šių objektų dėstinius (*kėlinius*)

$$P_n = \left\{(a'_1, a'_2, \ldots, a'_n)\right\}.$$

Skirtingų kėlinių yra $n!$, todėl net ir efektyviausių generavimo algoritmų sudėtingumas $\mathcal{O}(n!)$.

Naudodami Stirlingo formulę, įvertiname

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

Kėlinių P_n generavimo laikai

$$T(11) = 0.28, \quad T(12) = 3.4(s)$$
$$T(13) = 44.7 \quad T(14) = 626(s)$$
$$T(15) \approx 2.61, \quad T(16) \approx 41.7(h)$$
$$T(17) \approx 29.5, \quad T(18) \approx 532(d)$$

$T(20) \approx 553$ metai.

Galime kurti lygiagrečiuosius algoritmus. Naudodami 2056 procesorius išspręsime šį uždavinį per 3 mėnesius.

Tačiau P_{21} aibę generuosime ilgiau nei 5 metus.

RSA kodavimo algoritmas

RSA viešojo rakto kodavimo algoritmas yra naudojamas daugelyje svarbių duomenų kodavimo algoritmų, pvz. SSH (*secure shell*), SCP programose.

Viešojo rakto kodavimo sistemoje visi vartotojai gali užkoduoti pranešimą, bet jį perskaityti gali tik tas, kas žino kodo raktą.

Turime 2^{56} skirtingus kodo rakto variantus, todėl kodo patikimumas priklauso nuo to, ar greitai galime patikrinti visus raktus.

Ar uždavinys "išsprendžiamas" per kritinį laiko intervalą?

Išeitis: padidinti kodo ilgį iki 128 bitų.

Sudoku žaidimas

Turime 9×9 dydžio matricą. Visa lentelė suskirstyta į devynis mažesnius kvadratėlius, kurie susideda iš 3×3 langelių. Kai kurie langeliai yra užpildyti skaičiais nuo 1 iki 9.

Žaidimo tikslas – užpildyti visus stulpelius, eilutes ir 3×3 kvadratėlius skaičiais nuo 1 iki 9 (nei vienas skaičius juose negali kartotis).

8	1					7		3
			6		7			8
9		2	3	1		6		
	4			7		5	6	
		7	9		1	2		
	6	3		4			9	
		4		9	2	1		6
6			5		4			
7		8					5	9

Užpildyta lentelė

8	**1**	6	4	5	9	**7**	2	**3**
4	3	5	**6**	2	**7**	9	1	**8**
9	7	**2**	**3**	**1**	8	**6**	4	5
2	**4**	9	8	**7**	3	**5**	**6**	1
5	8	**7**	**9**	6	**1**	**2**	3	4
1	**6**	**3**	2	**4**	5	8	**9**	7
3	5	**4**	7	**9**	**2**	**1**	8	**6**
6	9	1	**5**	8	**4**	3	7	2
7	2	**8**	1	3	6	4	**5**	**9**

Variantų perrinkimo algoritmas

```
int   Tikrink (int n, Inf pozicija)
begin
  (1)   if   ( End (pozicija) == Pabaiga ) then
  (2)        Spausdink (pozicija);
  (3)        return(1);
  (4)   else
  (5)        S = BandymųAibė(pozicija)
  (6)        while  ( S ≠ ∅ ) do
  (7)                  NaujaPozicija(S, pozicija);
  (8)                  if (Tikrink (n+1, pozicija) == 1) return (1);
  (8)                  else
  (10)                    SenaPozicija(S, pozicija);
                       end if
             end do
  (11)       return (0);
        end if
end   Tikrink
```

S leistinų ėjimų aibė

- Siekiame minimizuoti S.
- Kuo greičiau išsiaiškinti, kad pozicija neleistina.
- Nei vienas skaičius 3 aibėse negali kartotis (iš viso 21 laukelyje).

Nauja pozicija

- Leksikografinė tvarka.

4×4 Sudoku variantas

Testas 1: variantų skaičius $7.75835 \cdot 10^{101}$
("lengvas" uždavinys)

Uždavinio sprendimo laikas – 589 sekundės.

S aibės parinkimo modifi kacija

- Nei vienas skaičius 3 aibėse negali kartotis (iš viso 21 laukelyje).

- Patikriname ar lentelėje egzistuoja bent 2 priklausomi laukai, kuriuose leistinas tik vienas ir tas pats skaičius.

Tikrinimas sudėtingesnis, bet $\Rightarrow$

Uždavinio sprendimo laikas – 190 sekundžių.

Testas 2: variantų skaičius $2.42879 \cdot 10^{105}$
("sunkus" uždavinys)

Uždavinio sprendimo laikas – 34.5 valandos.

Naujos pozicijos pasirinkimo algoritmas

- Laukelis, kuriame leistinų ėjimų skaičius yra mažiausias (heap duomenų struktūra).

Uždavinio sprendimo laikas – 3.2 sekundės (pagreitėjimas 3600 kartų).

Testas 3: variantų skaičius $3.64359 \cdot 10^{112}$
("labai sunkus" uždavinys)

Uždavinio sprendimo laikas – 350 sekundžių.

Lygiagretieji algoritmai

Labai netrivialus uždavinys, nes

- užduočių aibė sudaroma dinamiškai;
- algoritmo sudėtingumas priklauso nuo pasirinktos krypties, kai egzistuoja daugiau nei viena pasirinkimo galimybė.

NP sudėtingumo uždaviniai

Palyginkime n^2 ir 2^n sudėtingumo algoritmus.

Padidinus duomenų skaičių dvigubai, polinominio sudėtingumo algoritmo skaičiavimo trukmė padidėja keturis kartus.

Eksponentinio sudėtingumo algoritmo skaičiavimo trukmė padidėja du kartus, pridėjus tik vieną papildomą duomenį, ir keturis kartus, pridėjus du duomenis.

Išskiriame NP (angl. *nondeterministic polynomial time*) uždavinių klasę.

NP uždaviniams nežinome polinominio sudėtingumo sprendimo algoritmų, tačiau, atlikę $\mathcal{O}(n^k)$ veiksmų, galime patikrinti, ar duotasis objektas yra uždavinio sprendinys.

$$P \subset NP, \quad \text{bet ar} \quad P \neq NP \ ?$$

1. *Hamiltono ciklas*. Reikia patikrinti ar duotajame grafe $G = (V, E)$ egzistuoja ciklas, jungiantis visas jo viršūnes.

2. *Keliaujančio pirklio uždavinys*. Duotas svertinis grafas $G = (V, E)$, reikia rasti trumpiausią pirklio maršrutą, kai jis po vieną kartą aplanko visas grafo viršūnes ir grįžta į pradinę viršūnę.

3. *Diskretusis kuprinės užpildymo uždavinys*. Turime n daiktų, kurių tūriai yra $v_1, v_2, \ldots, v_n$, o kaina $p_1, p_2, \ldots, p_n$. Reikia rasti tokį daiktų rinkinį, kuris tilptų į V tūrio kuprinę ir krovinio vertė būtų didžiausia.

4. *Dėžių užpildymo uždavinys*. Turime keletą vienetinio tūrio dėžių ir n daiktų, kurių dydžiai v_1, v_2, …, v_n, čia $0 < v_j < 1$. Šiuos daiktus reikia sudėti į kuo mažesnį skaičių dėžių.

5. *Grafo viršūnių dažymo uždavinys*. Turime neorientuotąjį grafą $G = (V, E)$. Kiekvieną jo viršūnę nudažome kokia nors spalva. Kiek mažiausiai reikės parinkti spalvų, kad visos grafo briaunos jungtų skirtingų spalvų viršūnes?

6. *Darbų tvarkaraščio sudarymas*. Reikia atlikti $\{t_1, t_2, \ldots, t_n\}$ trukmės darbus, jų baigimo terminai – $\{d_1, d_2, \ldots, d_n\}$. Jei darbai nebus atlikti laiku, tai teks mokėti baudas $\{b_1, b_2, \ldots, b_n\}$. Kokia tvarka reikia vykdyti darbus, kad bauda būtų mažiausia?

Euristikos

Keliaujančio pirklio uždavinys.

Maršruto paiešką pradedame grafo viršūnėje v_0. Iš kiekvienos viršūnės einame į artimiausią dar neaplankytą grafo viršūnę. Paskutiniu žingsniu vėl grįžtame į pradinę maršruto viršūnę v_0 (godžioji strategija GS).

Teorema. Tegul C yra $n \times n$ dydžio matrica, apibrėžianti atstumus tarp miestų. Tarsime, kad matrica yra simetrinė, t. y. $c_{ij} = c_{ji}$ ir atstumai tenkina trikampio nelygybę

$$c_{ij} \leq c_{ik} + c_{kj}, \quad 1 \leq i, j, k \leq n.$$

Tada teisingas toks maršruto, apskaičiuoto GS algoritmu, ilgio įvertis:

$$L_n(GS) \leq \frac{1}{2}\big(\lceil \log n \rceil + 1\big) L_n.$$

Diskretusis kuprinės užpildymo uždavinys.

Pirmiausia apibrėžiame santykinę kiekvieno daikto vertę. Visus daiktus rūšiuojame šios vertės mažėjimo tvarka. Kuprinę stengiamės užpildyti didžiausios santykinės vertės daiktais. Juos imame vieną po kito ir tikriname, ar daiktas dar telpa į kuprinę, jei ne – imame kitą daiktą.

Turime aštuonis daiktus, kuriuos žymėsime (v_j, p_j):

$$(25, 50),\ \ (20, 80),\ \ (20, 50),\ \ (15, 45),$$
$$(30, 105),\ \ (35, 35),\ \ (20, 10),\ \ (10, 45).$$

Surūšiuojame daiktus večių didėjimo tvarka

$$(10, 45),\ \ (20, 80),\ \ (30, 105),\ \ (15, 45),$$
$$(20, 50),\ \ (25, 50),\ \ (35, 35),\ \ (20, 10).$$

Imkime kuprinę, kurios tūris $V = 80$. Naudodami godųjį algoritmą į ją įdedame pirmuosius keturis daiktus, jų bendras tūris – 75, o vertė – 275 litai.

Imdami pirmuosius tris ir penktąjį daiktus, užpildome visą kuprinę, o tokio krovinio vertė – 280 litų.

OPTCABLE projektas

$D = \{d_k, \ \ k = 1, \ldots, K\}$ yra laidų leistinų dydžių aibė.

$S_j = \{(d_{k_m}, I_j^m), \ 1 \le k_m \le K, \ m = 1, \ldots, M\}, \ j = 1, \ldots, J$

yra laidų apkrovimo kritiniai režimai. Tikslo funkcija W yra visų pluošto laidų plotų suma (svoris, metalo kiekis)

$$W(d_{k_1}, \ldots, d_{k_M}) = \frac{\pi}{4} \sum_{m=1}^{M} d_{k_m}^2.$$

Sprendžiame diskrečiojo minimizavimo uždavinį

$$\min_{d_{k_m} \in D, \ s.t. \ T \le T_{Max}} W(d_{k_1}, \ldots, d_{k_M}) = W(d_{k_1}^0, \ldots, d_{k_M}^0). \qquad (1)$$

T_{Max} yra duota kritinė temperatūra, T apskaičiuota maksimali laidų temperatūra (atsižvelgiant į visus darbo režimus)

$$T = \max_{p(d) \in P} \Big[\max_{1 \le j \le J} \max_{1 \le m \le M} U_m(S_j) \Big], \qquad (2)$$

$U_m(S_j)$ yra apskaičiuota m-jo laido temperatūra, $p(d)$ apibrėžia laidų išdėstymą pluošte.

Du diskrečiojo optimizavimo (min-max) uždaviniai.