04.04.2023

AA

curs 7

- Primul curs din partea a doua
- NOTARE: PT PTI
  lab 30 - 60
  examen 30 - 60

- Th: orice fct. care e algoritmic calculabilă e calculabilă de o mașină Turing
- Def. algoritm + cplx. in timp. | spațiul îl poți refolosi, timpul nu!

input de lungime $n \to f(n)$ pași

Clasa O: limită superioară pt. nr. de operații

$O(n)$ inclus în $O(n^2)$

$$\boxed{\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty}$$

$O(g(n)) \Leftrightarrow \exists g, n_0$ a.î. $c \times g(n) \geq f(n) \geq 0 \quad | \forall n > n_0$

funcție input

$O(n^2+2n) = O(n^2)$

Clasa Ω: limită inferioară

$f(n) \geq c \cdot g(n) \geq 0 \quad | \forall n > n_0$

$\Omega(n)$ inclus în $\Omega(n^2)$

$$\boxed{\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0}$$

Clasa θ: mărg. atât sup. cât și inf $\Rightarrow$

$\exists g, n_0$ a.î. $c_1 \cdot g(n) \geq f(n) \geq c_2 \cdot g(n) \quad \forall n > n_0$

$$\boxed{\lim_{n\to\infty} \frac{f(n)}{g(n)} \in \mathbb{R}_+}$$

- Complexitate worst case v.s. complexit. medie
  - atunci când pe aceeași lung. de input sunt pași mai mulți/puțini

"Turing Machine"

$M = (Q, \Gamma, b, \Sigma, P, q_0, F)$

- Q → mult. stărilor
- Γ → alfabet
- b → simbol blank
- Σ → alfab. de intrare
- P → funcție de tranziție
- q_0 → st. init.
- F → st. finală

$\boxed{Q}$ state register

read-write head

a	b	a	c	a	b	a	a	b	a

determinist: $\delta : \Gamma \times Q \to \Gamma \times Q \times \{left, right\}$

nedeterminist: $P : \Gamma \times Q \to 2^{\Gamma \times Q \times \{left, right\}}$

Clasele de complexit.:

- P = polynomial — $O(n^c)$ — pb ce pot fi rez. în timp.
  = putem afla sol. în timp polinomial
- NP = nondeterministic Polynomial — = — " — de o mașină Turing nedeterministă
  = putem verifica în timp polinomial dacă rez. este corect.

$\boxed{P \subset NP}$

$P \subseteq NP$ totuși asta nu-i demonstrat încă

(inclus strict)

Problemă de optim: Prob. rucsacului

greut(o) ≤ capacit. rucsac, 1/0. (ori luăm ob, ori nu) ≠ 0

- Fie L – lista ob. sortate descrescător după raportul valoare / greut
  cu P.D. → nu are cplx. polinomială
- Fie O_{max} – ob. cu prof. cel mai mare din listă
  S = 0, G = capacit. rucsacului
  Pt. fiecare O din L: dacă greut(o) ≤ G, atunci
  S += val(o)
  G −= greutate(o)
  altfel ~~break~~

  ALG = max(S, O max)

  | SORTARE + ↑

cplx: O(n log n) + O(n)

- Sol. e fezabilă? DA, Σ greut. obiectelor nu dep. capacit. rucsacului
- Trebuie justif. factorul de aproximare.

Fie $OPT_{1/0}$ soluția optimă pt. probl. rucsacului 1/0.

Fie OPT_G sol. pt. probl. rucsacului cu var. fracționară

$$\boxed{OPT_{1/0} \le OPT_G}$$

Fie O_k ultimul obiect inclus pt a obț. OPT_G. Eventual O_k este acel obiect fracționat

$$OPT_G \le \sum_{1 \le i \le k} val(O_i) = \sum_{1 \le i < k} val(O_i) + val(O_k) \le \sum_{1 \le i < k} val(O_i) + val(O_{max})$$

$$\le S + val(O_{max}) \le ALG + ALG = 2\,ALG$$

$$\Rightarrow OPT_G \le 2\,ALG$$

$$OPT_{1/0} \le OPT_G \le 2\,ALG$$

11.04.2023

AA
cursul 8

P - problemă ; i - intrare

OPT(i) = val. sol. optime   OPT, ALG > 0

ALG(i) = "valoarea" unei sol

ALG - pb de minimiz. : ρ - aprox dacă ALG $\leq \rho \cdot$ OPT

pb de maximiz. : ρ - aprox dacă ALG $\geq \rho \cdot$ OPT

$\rho = 1 \Rightarrow$ ALG. OPTIM

$> 1 \Rightarrow$ maximizare ?

$< 1 \Rightarrow$ mi

ρ = "tight bounded" = supremum; $\frac{ALG(i)}{OPT(i)}$

Load balancing problem - minimizare

m calc.

n activit. → fiecare are nevoie de t_j unit. de timp -

~~atr~~ potriv. activit. unui calc. pt. a minimiza timpul de lucru pe întregul proiect.

$J(i)$ - submulț. joburilor de pe i

L_i - load-ul mașinii i ; $L_i = \sum_{j \in J(i)} t(j)$

Vrem să minimizăm L_k, unde $k = \max_i (L_i)$.

PSEUDOCOD

for i = 1, m :

$L_i = 0$, $J(i) = \emptyset$

for j = 1, n :

i = arg (min (L_k | k ∈ {1, ..., m})   încărcătura minimă + arg - sau

$J(i) = J(i) \cup \{j\}$

L_i += j

→ NU E OPTIM :c

Care e factorul de aproximare?

① $OPT \geq \max \{ \underbrace{\frac{1}{m} \sum_{1 \leq j \leq n} t_j}_{\text{timpul mediu de pe fiecare mașină}}, \underbrace{\max \{ t_j \mid j = \overline{1,n} \}}_{\text{cea mai lungă activitate}} \}$

② alg. este 2- aproximativ

~~ALG = max~~

$ALG \leq 2^k \; OPT$

k = indicele mașinii cu load-ul maxim în urma rulării

l = nr. ultimului de pe maș. k

load'(x1) - load-ul maș. M = cât aveau maș. înainte de asign. jobului l

Deci:

$ALG = \underbrace{load'(k)}_{\leq OPT} + \underbrace{t_l}_{\leq OPT}$

$\hookrightarrow load'(l) \leq \overbrace{\frac{1}{m} \sum_{1 \leq i \leq m} load'(i)}^{\text{media timpului înainte să vină jobul } l}$

$= \underbrace{\frac{1}{m} \sum_{1 \leq j < l} t_j}_{\text{timpul de lucru necesar până la } l} < \frac{1}{m} \sum_{1 \leq j \leq n} t_j \leq OPT$

$t_l \leq \max \{ t_j \mid 1 \leq j \leq n \} \leq OPT$

$\Rightarrow ALG \leq 2 \times OPT$

Îl putem îmbunătăți?

a) Același alg., mai bună ef. (sch. ceva la o comparație mai „barbară")

• load'(k) $\le$ ___ $\frac{1}{m}\sum_{1\le j<q} t_j \le \frac{1}{m}\sum_{1\le j\le n}(t_j - t_q)$

scădem jobul t_q.

$\rightarrow$ $ALG \le \underbrace{\left(2-\frac{1}{m}\right)}_{}$ OPT

noul factor de aproximare

b) Același alg, alt LB. $\rightarrow$ NU

c) Alt alg.

Este $2-\frac{1}{m}$ TB? DA

$ALG = 2m-1$

$OPT = m$.

Să sortăm joburile după timp, desc.

$t_1 \ge t_2 \ge \dots t_n$

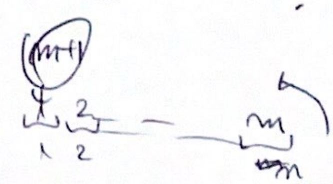
$\underline{n > m} \Rightarrow OPT \ge t_m + t_{m+1}$

le lucr. pe cele "mai mici"

Intuitiv:

$\Rightarrow$ Am $n > m$

$\Rightarrow \exists$ min. o mașină cu 2 activități!



TEME SEMINAR: deadl. săpt. 12-13

Load balance $\rightarrow$ pb. 3 (ARE 2P)

$\frac{3}{2}$ APPROX $\rightarrow$ $\frac{3}{2}-\frac{1}{2m}$

$\rightarrow$ pb. 1 (1P) a $\rightarrow$ sigur nu

b $\rightarrow$ 10 unit. ~~R~~ nu

$\rightarrow$ pb 2 (1P)

25.04.2023

SAPT. 9

AA

curs 9

CURS 3 DIN PARTEA II → { DEADL. TEME : sem/lab. 12/13 PART. 2 }

- Alg^α-aprox

α se întâmplă în anumite cazuri → maxif.

TASK-uri

sortăm descr. ~> preprocesare → 1,5 aprox.

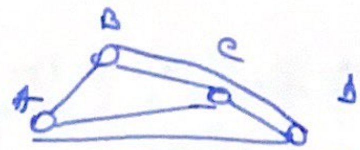
atribuim mașinii cu cel mai mic timp $\frac{3}{2} - \frac{1}{2n}$

PROBLEMĂ: ciclu hamiltonian: cont. ciclu cu f. nod. o sg. dată (HCP)

- pb. de decizie NP-completă
- ∃ sau ∄

PROBLEMĂ: traveling salesman problem (TSP):

- G hamilt.
- lanț închis cu suma costurilor muchiilor e minimă



- pb. NP-Hard : dacă ∃ să îl și spunem

COND. NECESARE PT. A FI GR. HAMILTONIAN
- fiecare nod are gr ≥ n/2 → gr. hamilt.
- ∀ 2 noduri vecine au suma gr > n → gr. hamilt.

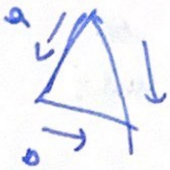
- nu ∃ un alg. aprox. care să rezolve problema
- P=NP ~~decât~~ așa vom avea alg. aprox. ce să rez. pb.

demo: REDUCERE LA ABSURD

∄ ..

TSP particular

regula Δ



$a+b \geq c$

(în spațiul euclidian)

→ pt. un graf ~~complet~~, ponderat, cu ~~...~~ resp. regulii Δ
→ ∃ alg.

ALG: MST + pre-ordine
minimum spanning tree

2APROX.

– Se poate mai bine? DA

Alg. lui Christofides

~~cuplaj = mult. de muchii cu prop. că ∄ 2 muchii vf~~

~~cuplaj = mult. de muchii~~ muchii

cuplaj = vf nesc. A 2 ~~...~~ muchii
fiecare vf are o rel.

DE ASTA vrem nr. par de vf.

cost diag $\sqrt{2}$.

Pb (2a)

→ pătrat și mai adăugam 1pct în mijloc

(2b) mulț. de pct P + 1pct → MST

mai mic și mai mic ⇒ – –

cel mai bun cost e de cel mult 2^x mai bun

decât cel de la care am plecat

02.05.2023

AA

curs 10

Vertex Cover Problem, Programare liniară

1. Vertex Cover

prog. testează doar calc. legate direct

vrem să instalăm prog. pe cât mai puține calc.

formal: vrem să găsim o ~~[illegible]~~ acoperire de card. minim.

prob. NP-hard

$G = V, E$

$E = E'$, $S = \emptyset$ → polinomial

cât timp $E' \neq \emptyset$:

aleg $(x, y) \in E'$

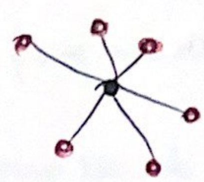
$S = S \cup \{x, y\}$

ștergem din E' toate muchiile incid. lui x dar și lui y

return S

Întrebări:

1. Mulț. de noduri S e o acop. pt. graful G? AA
2. Cât de apr. e de optim? → poate fi și de 100 de ori mai slab decât un alg. optim



3. Cum îmbunătățim? → alg. 2-aproximativ acum

Dem că S = acop. validă, alg este 2-aprox.

Lower bound = card E^* → mulț. de muchii nod-disj

$OPT \geq |E^*|$ $= 2|S|$

• WVCP → avem și ponderi (pe vârfuri!!)

2. Programare liniară

constrângeri → ce vrei minimizat.

$$\left.\begin{array}{l} a_{11} x_1 + \ldots a_{1d} x_d \le b_1 \\ a_{21} x_1 - \ldots + a_{2,d} x_d \le b_2 \\ \\ a_{m1} x_1 + \ldots a_{md} x_d \le b_3 \end{array}\right\} \begin{array}{l} \text{minimizat} \\ c_1x_1 + \ldots c_dx_d. \end{array}$$

ALGORITMI SIMPLEX → rezolvă în timp polinomial

și dacă ∄, ne spune asta.

Programarea eronată nu poate rezolva WVCP.

Programarea

... un f. de aprox < 1.36 nu poate

• orice alg aprox.

rula în timp polinomial.

$X = \{x_1, x_2 \ldots x_n\} \to$ 0 nu e luat, 1 e luat

$$\underbrace{\min \sum_{i=1,n} f(v_i) * x_i}$$

• Selectăm un prag → 0,5 1 1 0 prag

$$ALG = \left\{ \cancel{x} \ \tfrac{1}{2} \right. \quad \le \sum f(v_i) * 2x_i \quad 2\sum f(v_i) * x_i \le OPT.$$

⇒(doar când e zero)

$\Rightarrow ALG \le 2^k OPT$

Analiz. fact. de aprox → sol. e de cel mult k ori ... decât opt.

↳ Înainte de asta dem. că sol. este o sol!

măcar una să fie true

14.06.2023 ORA 10:00
examen | laptop, nu, materiale

09.05.2023

AA

Algoritmi Genetici

curs săpt. 11

Definiții:

cromozom = mulț. ordonată de gene | $\boxed{1}$0011010110 | → culoarea ochilor, etc.

genele det. caracteristicile unui individ

populație = mulț. de indivizi care trăiesc într-un mediu

fitness = fcț. de ~~cost~~ adecvare → gradul de adaptare la mediu pt. fiecare individ

generație = etapă în evoluția populației

selecție = alegerea indivizilor cu grad de adaptare (fitness) ridicat

operatori genetici { încrucișare : indiv. noi pe baza car. părinților; mutație : —— " —— cu alte car. f.d. ale părinților (apar trăsături noi)

• discretizarea interv.

[3, 10)

$[a; b) \rightarrow b-a$ elem întregi

$(b-a) \cdot 10$ elem cu 1 cifră după , 3,1 3,2 3,3 ... 9.8 to 9,9

$(b-a) \cdot 100$ — —

$(b-a) \cdot 10^p$ pt. o precizie de p zecimale

• lung. cromoz. $2^{l-1} < (b-a) \cdot 10^p \leq 2^l \Rightarrow l = \lceil \log_2((b-a) \cdot 10^p) \rceil$

(parte întreagă superioară)

• val. codificată $x_{(2)} \rightarrow x_{(10)} \rightarrow \frac{b-a}{2^l - 1} x_{10} + a$

string baza 10

- funcția de fitness, f
- selecție → proporțională
  → elitistă
  → turneu
  ...

Selecția proporțională:
- $\boxed{p_i = \text{prob. de selecție a indiv. } x_i = \frac{f(x_i)}{\sum f(x_i)}}$
- selectăm n indivizi j cu prob. $(p_1, p_2 \dots p_n)$
- luăm o var. unif. între 0,1
- det j a.î. var $\in [q_{j-1}, q_j]$
- indiv. j trece în generația următoare

Selecția elitistă → cel mai bun individ trece automat în gen. următoare

- încrucișare → prob. de încrucișare < ca să participe

$$
\begin{array}{l|l}
x_1 \; — \; x_i & x_{i+1} \; — \; x_l \\
\hline
y_1 \; — \; y_i & y_{i+1} \; — \; y_l
\end{array}
$$

i = pct. de rupere generat aleator

finalul se interschimbă

⇒ 2 copii din generația următoare

- mutație - se schimbă val. unei gene din cromozom

$\frac{10100100}{10110100}$ sau $\frac{10100100}{10000100}$ din 0 în 1 sau din 1 în 0.

TSP: n=6
X = 2 3 5 1 4 6 → fitness-ul ar fi $\frac{1}{\text{~~costul total al muchiilor~~}}$ 2-3, 3-5, 5-1, 4-6, 6-2

$\frac{1}{\Sigma \dots}$ (cost↗ fitness↘ cost↘ fitness↗)

- crossing-over și mutație → NU AU SENS
- putem face flip sau swap

→ De ce trecem elitistul mai dep. mereu? Ca să nu ne poată scădea maximul, WORST-CASE rămâne cel de data trecută.

→ grafic cu cum evoluează maximul / fitness-ul mediu

→ la prez cod → modif. pe loc

AA

Curs 12

Algoritmi Randomizați

= probabiliști

Def: alg. în care se gen. aleator un element și ef. deciziî în funcție de valoarea acestuia.

→ un zar cu 12 fețe v.s. două zaruri cu 6 fețe

1→6 distr. unif
1→6 distr. unif } 2 val. cu distrib. gaussiană

Alg. poate avea nr. dif. de pași / outputul.

2 clase de alg. probabiliști (sau 3):

MONTE CARLO → rapid
→ probabil

LAS VEGAS → corect
→ timp probabil rapid

(+) probabil rapid
probabil corect | ↔ Atlantic City

1. MONTE CARLO: înmulțirea matricilor

$A, B \in \mathcal{M}(n, n)$; $A \times B = ?$

- naiv $\Rightarrow O(n^3)$
- Strassen $\Rightarrow O(n^{\log 7}) = O(n^{2,81})$
- Coppersmith $O(n^{2,376})$

NU ⇒ Matrix Multipl Check → verif. dc. $A \times B = C$

Se poate mai bine decât direct? DA

$A \times B = C \Rightarrow$ întotdeauna "DA"

$A \times B \neq C \Rightarrow$ "NU" cu o prob. ~~de~~ $p \geq \frac{1}{2}$

~~NU => sigur nu~~ ~~DA => sigur da~~ ~~DA~~ 2

~~DA => ?~~ ~~NU => ?~~ rulăm de m.m. ori

=> un sg. NU înseamnă

$Pr[r_i = 1] = 1/2$

$A \times (Br) = Cr$ ret DA $\qquad O(n^2)$

altf. ret NU

OBS! Dacă $A \times B \neq C \Rightarrow Pr[A \times (Br) = Cr] \geq \frac{1}{2}$

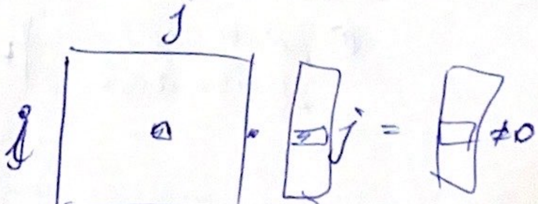
dem: fie $D = A \cdot B - C$

din ip $AB \neq C$ deci $D \neq O_{(n,n)}$ => ∃ vectori r a.î. $D \cdot r \neq 0$ și $\neq 0$ multitudine de asţ. de vectori

$Pr[D \cdot r \neq 0] \geq \frac{1}{2}$

∀ r a.î. că $Dr = 0$, ∃ r' a.î. $D \cdot r' \neq 0$

$D \neq 0 \Rightarrow \exists\, D[i][j] \neq 0$



pt. $r' = r + v$

$v = [0, 0, \underset{j}{1}, 0]$ => r' va ≠ r cu r[j] modif. în 1

$Dr' = D(r + v)$

↳ $\underbrace{D \cdot r}_{\text{nul}} + \underbrace{D \cdot v}_{\text{nenul}}$

$\Rightarrow P[D \cdot r = 0, D \neq 0] < \frac{1}{2}$

Alt ex: cu bancnota de 100$ în sală SAU orice altă căutare.

2. QUICK SORT (ALG. LAS VEGAS)

↳ sortarea unui cuv. în rusă

(C.A.R. HOARE, MOSCOVA, 1959)

worst $O(n^2)$

prob. ~ $O(n \log n)$

pivotul ales este minimul sau maximul

DIVIDE

→ pivot   SEE A

($<p$) , ($>p$) , ($=p$)

L less   G greater   E equal

CONQUER: SORT. REC L, G

COMBINARE — ? nu există :)

recurs

$T(0) + T(n-1)$ toți

$1 + n-1+1 \quad n$

$\to n^2$

Cel mai bun pivot?

Găsirea medianei

Timp?

Asimptotic liniar → găs. med → recurs.

$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) + \Theta(n)$

$\underline{T(n) = \Theta(n \log_2 n)}$ merem // poate fi mai rău decât basic (nu se găsește mereu liniar mediana)

relaxăm situația: alegem aleator pivotul

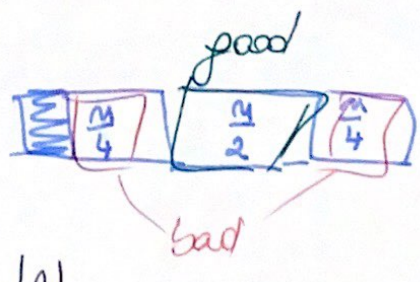
Paranoid quicksort

repetă Alege x

construiește L, G, E în fct de x

până când $|L| \le 3/4 |A|$ și $|G| \le 3/4 |A|$

repetăm recursiv și pe L și pe G



→ Alegem un pivot bun cu o prob. de $\frac{1}{2}$. // în medie 2 selecții

23.05.2023

(13)

Curs AA

săpt. 13

Structuri randomizate de date : Skip Lists, Bloom Filters

I Skip Lists

- Listă dublu înlănț.

14 ↔ 23 ↔ 24

căutare : $O(n)$ ... am vrea $\log(n)$

14 → 34 → 72

14 → 23 ↔ 34 ↔ 42 ↔ 50 ↔ 59 ↔ 66 ↔ 72 ↔ 80

~ uniform distrib.

sup.	n	$\sqrt{n}$
inf.	n^2	n

n noduri

↓

bucăți de câte n noduri

$n^{\frac{1}{\lg k}} \cdot \lg k$

$L_1 + \frac{L_2}{L_1} = 2\sqrt{n}$

$\sqrt{n}$

$\sqrt{n}\ \sqrt{n}$

$k\sqrt[k]{n}$ pentru k nivele

sau

$\lg k \sqrt[\lg k]{n} = 2 \lg n$

Inserare : vedem de mai inserăm pe un nivel prin a da cu banul

Ștergere : de pe cel mai de sus nivel, în jos

Th: Căutarea se face "probabil" în timp logaritmic $O(\lg n)$

nr. nivele (A), nr. pași orizontali (B)

$\Rightarrow Pr(A \& B) = 1 - O(1/n^c)$

II Bloom Filters

Cum reținem ce avem într-un arr.

Opțiune : mai multe fct. de hash.

• Bloom Filters

$S = \{ 1, 3, 5, 6 \}$

NAIV → iterare prin listă

caut în toată lista : sp. $O(n)$, caut $O(n)$, inser $O(n)$

• vector caracteristic

i:	0	1	2	3	4	5	6	7	8	9
$v[i]$:	0	1	0	1	0	1	1	0	0	

~~sp $O(1)$~~ caut $O(1)$

inserare $O(1)$

spațiu $\underbrace{O(\max(S))}$

OPTIMAL

$m = -\frac{n \ln P}{(\ln 2)^2}$

m = lung. arului

$\frac{m}{n} \ln 2$ ← k = nr hash-uri

n = nr. el. inserate

→ P = prob. de false positive

$P = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k$

• hash map : inser.

$A[hash[x]] = x$

ins. $O(1)$
caut $O(1)$

$y \neq x$ cu $hash[x] = hash[y]$ → :(

îl inserăm lângă (pe căsuțe urm.)

VREM SĂ FIE CÂT MAI MARE

Altă sol:

1	2	3	4	5	6	7	8	9	10	11	12
0	1	1	0	1	0	1	0	0	0	1	0

$h_1, h_2, h_3 : D \to \{1, ..., 12\}$

h_1("pug") = 2
h_2("pug") = 11
h_3("pug") = 3

h_1(cat) = 5
h_2(cat) = 7
h_3(cat) = 11

h_1(bird) = 2
h_2(bird) = 5
h_3(bird) = 9

find nu se află

h_1(fish) = 11
h_2(fish) = 5
h_3(fish) = 2 } false positive

și nu prea merge ștergerea

NU → clar nu
DA → poate fi fals pozitiv
P cât mai mică