Algoritmi aproximativi

Knapsack

1. Fie S un șir de nr nat $\{s_1, \dots, s_n\}$ și k un nr nat, $k \in \overline{1,n}$.

a) Scrieți un alg pseudo-polinomial care găsește suma maximă, dar care să fie $\leq k$, ce poate fi formată din elem din S (nr întregi poz, luate cel mult o sg dată). Indicați compl. spațiu/timp și just de ce e corect (soluție optimă).

```
def suma_max(k, v):
    valori = set()
    valori.add(0)
    total = 0
    for nr in v:
        copie = set(valori)
        for val in valori:
            if val + nr <= k:
                valori.add(val + nr)
                total = max(total, val + nr)
    return total.
```

- Complexitate timp
  - ↳ În cel mai rău caz, <u>valori</u> poate conține toate val dintre 0 și $k \Rightarrow$ len(valori) $\leq k+1$
  - ↳ Iterăm prin n numere, pentru fiecare iterăm prin <u>valori</u> deja construit

  $O(n \cdot k)$

- Complexitate spațiu
  - ↳ valori e un set care în cel mai rău caz are dim $O(k)$
  - ↳ copie - un set inițializat local de max size $O(k)$, dar fiind local, obiectul este eliberat de garbage collector după iterație

  $O(k)$

Alg prez. constr toate sumele posibile $\leq k$, iar alegerea celei mai mari face soluția optimă $\Rightarrow ALG(I) = OPT(I)$

6) Scrieți un alg. aproximativ care calculează o sumă cel puțin pe jumătate de mare ca cea optimă, dar rulează în timp $O(n)$ și complexitate spațiu $O(1)$.

```
~~import sys~~
~~k = int(input())~~
~~total = 0~~
~~for line in sys.stdin:~~
~~    val = int(line)~~
~~    total = total + val~~
~~    if( total > k):~~
~~        total - val~~
~~        total = total - val~~
~~    elif (total <= k):~~
~~        to~~
```

```
import sys
k = int(input())
total = 0
for line in sys.stdin:
    val = int(line)
    if(val + total <= k):
        total = total + val
    else if(total < val)
        total = val
print total.
```

- Compl Timp: $O(n)$ (parcurgerea, citirea elem)
- Compl Spațiu: $O(1)$ (am declarat 3 variabile)

* Un alg ALG pt o prob de maximizare s.m. ρ-aprox pt o valoare $\rho < 1$ dacă $ALG(I) \geq \rho \cdot OPT(I)$

Vrem să arătăm $ALG \geq \frac{1}{2} OPT$

Avem 2 cazuri:

$\boxed{1}$ cel în care am $s_i, i \in \overline{1,n}$ cu $s_i \geq \frac{OPT}{2}$. ALG garantat va găsi un răsp mai mare sau egal cu s_i. (considerând cazul în care avem un sg elem pt răspuns)

$\boxed{2}$ pp că suma ajunge $\leq \frac{1}{2} OPT$, urm elem fiind $\geq k - \frac{k}{2}$, sau $\geq \frac{k}{2}$, caz în care suma e înlocuită de acest elem

$R: Max(M, G)$
- G → greedy sumă
- M → cel mai mare $\leq k$

$$OPT \leq G + M \Rightarrow \frac{OPT}{2} \leq \frac{G+M}{2} \leq R$$

Load Balance

1. Fie o iterație a prob. Load Balance pentru 2 mașini. Un student propune un alg de rezolvare și susține că acesta este 1.1 aprox. El rulează alg pe un set de n activități și obține o încărcătură de 80 pe una din mașini, resp 120 pe cealaltă. Este posibil ca factorul lui de aprox să fie corect..

a) .. ținând cont că rez obținut anterior a fost făcut pe un set de activități, fiecare cu timpul max 100?

Pentru setul de valori: ~~{10 20, 30, 60, 90}~~ {30, 90, 60, 20}

Soluție optimă: $Max(\{60, 30\}, \{90, 20\}) = 110$ (OPT

M_1: | 60 | 30 | |

M_2: | 90 | 20 |

Algoritmul nostru: $Max(\{60, 20\}, \{30, 90\}) = 120$

(pp că alg este înp la jumate și să facem un FCFS a primului seg pe o mașină, și celălalt pe cealaltă).

M_1: | 30 | 90 |

M_2: | 60 | 20 |

$ALG = 120 < OPT \cdot 1.1 \Rightarrow$

$\Rightarrow$ e posibil să fie 1.1 aprox.

b) .. ținând cont că rez obținut anterior a fost făcut pe un set de activ. cu timp cel mult 10

$A = \{a_1, a_2, \dots a_m\}$, ~~$max(A) \leq 10$~~ $a_i \leq 10$, $i = \overline{1,m}$

Pt $\forall a_i$, alg va aloca activitatea mașinii mai puțin încărcate la mom resp, a.î. diferența nu poate depăși $max(A) = 10$

Pt ca ALG să fie 1.1 aprox $\Rightarrow OPT \cdot 1.1 > ALG$

~~Fie mașina 1 cu $M_1 \leq 110$ și mașina 2 cu $M_2 \geq 110$~~

~~Fie $\sum_{i \in M_1} i \geq \sum_{i \in M_2} i$~~

Pentru a fi un alg 1.1. aprox, diferența dintre cele 2 mașini trb să fie cel puțin $\leq 1.1 \cdot 10 = 11 \Rightarrow$ imposibil având în vedere că e balansat.

Worst case: $m_2 \to 95$, $m_1 \to 105$ $\Bigr\} \Rightarrow 105 = \frac{120}{x} \Rightarrow x = 1.14 > 1.1$.

2. Fie ALG_1 și ALG_2 2 algoritmi de rezolvare pt aceeași problemă de minimizare. ALG_1 este un alg 2-aproximativ, resp ALG_2 este un alg 4-aproximativ. Stabiliți val de adevăr a urm prop, dând o scurtă just.

a) Există cu siguranță un input I pt care

$$ALG_2(I) \geq 2 \cdot ALG_1(I)$$

$$\left.\begin{array}{l} ALG_1(I) \leq 2 \cdot OPT(I) \\ ALG_2(I) \leq 4 \cdot OPT(I) \end{array}\right\} \Rightarrow \left\{\begin{array}{l} \frac{ALG_1(I)}{2} \leq OPT(I) \\ \frac{ALG_2(I)}{4} \leq OPT(I) \end{array}\right.$$

~~În cele mai rele cazuri:~~ $\cancel{ALG_1 = 2OPT}$ ~~și~~ $\cancel{ALG_2 = 4OPT}$ ~~(General)~~

Fie urm problemă: se citește un nr x

↳ Se generează lista $v: \{|x|, |x|+1, |x|+2, |x|+3, |x|+4\}$

↳ Se cere un nr din listă cât mai mic, mai mare decât $|x|$

Alg optim: $OPT = v[1] = |x|+1$

Fie Alg1 care dă output la al 3 lea element $\Rightarrow v[3] = |x|+2$

Fie Alg2 care dă output la al 4-lea element $\Rightarrow v[4] = |x|+3$

!! Un alg pt o prob de minimizare s.m. ρ-aprox, pt o val $\rho > 1$ dacă $ALG(I) \leq \rho OPT(I)$ pt orice intrare I

Verific alg:

ALG1: $|x|+2 \leq 2(|x|+1) \Leftrightarrow |x|+2 \leq 2|x|+2 \Leftrightarrow 0 \leq |x|$ adv $\Rightarrow$ 2-aprox

ALG2: $|x|+3 \leq 4(|x|+1) \Leftrightarrow |x|+3 \leq 4|x|+4 \Leftrightarrow -1 \leq 3|x|$ $\Rightarrow$ adv $\Rightarrow$ 4-aprox

$$\begin{aligned} ALG_2(I) \geq 2 ALG_1(I) &\Leftrightarrow |x|+3 \geq 2(|x|+2) \\ &\Leftrightarrow |x|+3 \geq 2|x|+4 \\ &\Leftrightarrow -1 \geq |x| \text{ fals} \Rightarrow \end{aligned}$$

$\Rightarrow$ nu există input încât $ALG_2(I) \geq 2ALG_1(I)$

b) Nu există niciun input I pt care

$ALG_1(I) \geq 2 \cdot ALG_2(I)$

$$\left.\begin{array}{l} OPT \leq ALG_1 \leq 2 \cdot OPT \\ OPT \leq ALG_2 \leq 4 \cdot OPT \end{array}\right\} OPT \leq ALG_1 \leq 2 \cdot OPT \leq 2 \cdot ALG_2 \Rightarrow$$

$\Rightarrow ALG_1 \leq 2 \cdot ALG_2$, pt orice input $\Rightarrow$

$\Rightarrow$ deci pt niciun input nu avem $ALG_1 > 2 \cdot ALG_2$

Mai există de tratat doar cazul $ALG_1 = 2ALG2$, care intră în condiția $ALG_1 \leq 2ALG_2$

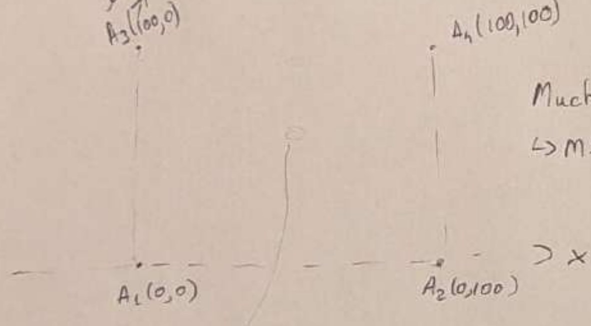
↳ Condiția este ~~adevărată~~ falsă, puteam avea o probl care are egalitatea ce satisface inegalitatea $\Rightarrow$ Nu este obligatoriu să nu existe niciun input.

Travelling Salesman Problem

Fie P o mulțime de puncte în plan. Fie T MST-ul construit pe baza punctelor din P. Uneori adăugând și alte puncte pe lângă cele din P, putem obține un MST cu cost mai mic. Un asemenea arbore, construit prin adăugare de noduri s.n. Steiner Tree. Algoritmii pt calcularea de ST-uri sunt de obicei NP-hard.

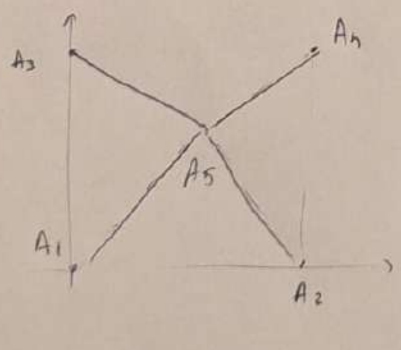
a) Ar. că există cazuri în care adăugând un punct $q \notin P$, obțin un MST pt $P \cup \{q\}$ cu un cost mai mic decât T.

Fie $P = \{A_1(0,0), A_2(0,100), A_3(100,0), A_n(100,100)\}$



Muchii - $\{(A_3, A_1), (A_1, A_2), (A_2, A_n)\}$
$\Rightarrow$ MST cu $\boxed{\text{cost} : 300}$

Dacă aș adăuga punctul $\{q\} = A_5 = \{(x=50, y=50)\}$



$\frac{diag}{2} = \frac{\sqrt{200}}{2} = \frac{10\sqrt{2}}{2} = 5\sqrt{2}$

$\Rightarrow$ Obținem un nou MST cu muchiile: $\{(A_3, A_5), (A_5, A_n), (A_5, A_2), (A_1, A_5)\}$

Cost: $4 \cdot 5\sqrt{2} = 20\sqrt{2} < 300$

R: Am obținut un nou MST pt un $q \notin P$ adăugat, unde MST e format/construit cu un cost mai mic decât T.

6) Fie Q o mulțime de puncte în plan, disjunctă față de P. Ar. că T este de cel mult 2 ori mai mare ca și cost față de MST-ul pt $P \cup Q$. Altfel spus, odată ce avem un MST pentru P, putem îmbunătăți rezultatul adăugând alte puncte, dar niciodată cu mai mult de un factor de 2.

- Ar. $Cost(T) \leq 2 \cdot Cost(T')$, unde $T' = MST(P \cup Q)$

Considerăm $MST(P \cup Q)$ arborele Steiner optim
Dublăm muchiile sale $\Rightarrow$ gf devine eulerian $\Rightarrow$ există un ciclu eulerian care parcurge fiecare muchie exact o dată

ex.

$$\begin{array}{ccccc} A_1 & - & Q & - & A_2 \\ & & / & & \\ & & A_3 & & \end{array} \Rightarrow \begin{array}{ccccc} A_1 & \overset{1}{\underset{6}{=}} & Q & \overset{2}{\underset{3}{=}} & A_2 \\ & & /\!/_{5,4} & & \\ & & A_3 & & \end{array}$$

$Cost(\text{ciclu eulerian}) = 2 \cdot Cost(Mst(P \cup Q))$

Eliminăm punctele din Q, înlocuind drumurile $A \to Q \to A_1$ în $A \to A_1$
Cum suntem în plan, sigur $\Rightarrow d(A, A_1) < d(A, Q) + d(Q, A_1) \Rightarrow$
$\Rightarrow$ nu are cum să obținem un cost mai mare
În loc de ciclul hamiltonian $A_1 \to Q \to A_2 \to Q \to A_3 \to Q \to A$ obț. $A \to A_1 \to A_3 \to A$
$Cost(\text{ciclu curent}) \leq 2 \cdot Cost(MST(P \cup Q))$

Dacă elimin o muchie, ciclu dispare și obțin un drum hamiltonian $\Rightarrow Cost(MST(P)) \leq Cos(dr.\ ham.) \leq 2 \cdot Cost(MST(P \cup Q))$,
MST(P) fiind cel mai ieftin arbore în P $\Rightarrow$

$$\Rightarrow Cost(MST(P)) \leq 2 \cdot Cost(MST(P \cup Q))$$

Vertex Cover

Fie $X = \{x_1, x_2, .. x_m\}$ o mulțime de booleene. Numim formulă booleană (peste X) în Conjunctive Normal Form (CNF) o expresie de forma $C_1 \wedge C_2 \wedge .. C_m$, unde fiecare predicat (clauză) C_i este o disjuncție a unui nr. de variabile (e alcătuit din mai multe variabile cu simbolul „v").

Exemplu: $(x_1 \vee x_3 \vee x_n) \wedge (x_2 \vee x_3 \vee x_7) \wedge (x_1 \vee x_5 \vee x_6) \wedge (x_2 \vee x_5 \vee x_7)$

Evident că orice astfel de expresie va fi evaluată ca true dacă toate din lui X iau val true. Ne interesează să aflăm nr de elem din X care tb să fie true, a.î.. toată expresia să fie true.

Fie urm alg pt problema de mai sus în varianta în care fiecare clauză are exact 3 var. (3CNF

Greedy-3CNF

1. Fie $C = \{C_1, .. C_m\}$, $X = \{x_1, .. x_m\}$
2. Cât timp $C \neq \emptyset$ execută:
   (a) alegem aleator $C_j \in C$
   (b) Fie x_i una din var din C_j
   (c) x_i = true
   (d) Eliminăm ~~to~~ din C toate pred de îl conțin pe x_i
3. Soluția constă din variabilele pe care le-am setat ca true pe parcursul execuției.

(a) Factorul de aproximare (worst case)

Fie $I_m = \wedge C_i$, unde $C_i = (X_1 \vee X_i \vee X_i)$ pt $i = \overline{1,m}$

OPT: Ar fi să ia algoritmul direct pe X_i din C_1, și astfel obț. soluția unică care șterge tot.

Cel mai rău: Ar fi să ia pe rând C_i ~~din~~ unde $i = \overline{2,m}$, iar x_i ales să ~~fie~~ fie X_1, astfel soluția va consta la final din toate soluțiile $\{x_1, x_2 ... x_m\}$ = m soluții

În worst case, alg dat este m-aproximativ

b) Modificați alg a.î. să fie 3-aproximativ.

1. Fie $C = \{C_1, .. C_m\}$, $X = \{x_1, .. x_n\}$
2. Cât timp $C \neq \emptyset$ execută
   (a) Alegem aleator $C_j \in C$
   (b) Pentru fiecare x_i din C_j
   (c) x_i = true
   (d) Eliminăm din C toate prod care conțin x_i-urile
3. Soluția constă din variabilele pe care le-am setat ca true pe parcurs.

ALG aplicat pe exemplul anterior, orice $\forall C_j$ ar nimeri, toate C_j conține X_1, astfel eliminând toate valorile din C_j resp, eliminăm toate celelalte C_i, $i = \overline{1,n} \neq C_j$, acestea conținându-le pe $\overline{x}_i$

Astfel, ALG este 3-aproximativ

---

c) Reformulați problema de mai sus sub forma unei prob. de programare liniară (cu nr reale)

~~1) Fie $C = \{C_1, .. C_m\}$, $X = \{x_1, .. x_m\}$~~
~~2) Cât~~

- Fie $C = \{C_1, .. C_m\}$, $X = \{x_1, .. x_n\}$
- Pentru fiecare booleană $x_i \in [0;1]$
- Vrem să minimizăm nr total de var true
- V este conceput cu + => o clauză $(x_a \vee x_b \vee x_c) \Rightarrow x_a + x_b + x_c \geq 1$

Pt exemplul furnizat:

$$(x_1 \vee x_3 \vee x_n) \wedge (x_2 \vee x_3 \vee x_7) \wedge (x_1 \vee x_5 \vee x_6) \wedge (x_2 \vee x_5 \vee x_7)$$

obținem instanța:
$$\begin{cases} x_1 + x_3 + x_n \geq 1 \\ x_2 + x_3 + x_7 \geq 1 \\ x_1 + x_5 + x_6 \geq 1 \\ x_2 + x_5 + x_7 \geq 1 \end{cases}$$

Obs sol optimă: X_1 și X_5
ec liniară pe care vr. să o minimiz: $S_{min} = x_1 + ... x_7 \Rightarrow$ alegem x_1 și x_5 true și restul val false $\Rightarrow$
$\Rightarrow$ OPTIM