Oprea Tudor
grupa 241

26.05.2023

Algoritmi Aproximativi

Knapsack

1. a)

```
#include <iostream>
#include <set>
using namespace std;

set <double> sume;

int main()
{  double nr_obiecte, k, maxi = -1;
   cin>> nr_obiecte;
   cin>> k;
   for (int i = 1; i <= nr_obiecte; i++)
      { double obiect;
        cin>> obiect;
          for (auto ind: sume)
             if ( ind + obiect <= k)
                sume.insert (ind + obiect);
                  if (ind + obiect > maxi)
                     maxi = ind + obiect;
                  }
      }
   cout << maxi;

   return 0;

}
```

Justificare corectitudine:

Algoritmul furnizează suma maximă a elementelor din S, verificând toate perechile posibile de elemente (check), facându-se în precizarea că suma $\leq k$.

Complexitate: TIMP $O(n \cdot k)$ ↳ n = nr. de check

SPAȚIU $O(2^n)$ → pe worst case în care ar verifica fiecare cu fiecare (cea mai mare combinație)

b)
```
#include <fstream>
using namespace std;
#include <fstream>
ifstream f("intrare.in");

int main()
{ double k, x, suma;

    suma=0;
    f>>k;
    while(f>>x)
      if (suma+x<=k)
          suma=suma+x;
      else if (suma<x)
          suma=x;

    }
    cout<<suma;
    f.close();
    return 0;
}
```

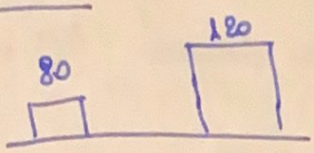
TIMP: $O(n)$

SPAȚIU: $O(1)$ ⇝ reține o singură sumă

↳ citește toate numerele, o singură dată și nu mai face altă parcurgere

Pe măsură ce citim numerele, verif. dacă le putem

Load Balance

1. 

1.1 aprox.

a) activitățile au un timp de lucru de maxim 100

⇒ da, poate fi 1.1 aproximată

exemplu: dacă am avea trei activități cu

$t_1 = 45$
$t_2 = 80$ ⇒ algoritmul ar da 80, 120
$t_3 = 75$ dar în OPT ar da la fel

⇒ OPT = ALG = 120

⇒ factorul de aproximare = 1

1.1 > 1 ⇒ algoritmul propus poate fi 1.1 aprox.

b) activitățile au timp de lucru maxim 10

Nu se poate, intuitiv deoarece dacă greutățile sunt < 10, toate cele de pe la 80 la 120 vor fi distribuite în mod egal pe cele două mașini, deci soluția OPT va da un rezultat cu totul diferit ⇒ ALG nu poate fi 1.1 aproximată (este mult mai departe).

Demonstrație formală: Fie mașinile 1 și 2 cu load-urile $\mathcal{L}_1$ și $\mathcal{L}_2$.

presupunem că $|\mathcal{L}_2 - \mathcal{L}_1| > 10$

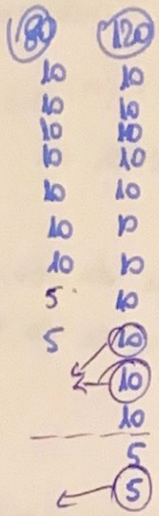
dar $t_{activitate} \leq 10$ ⇒ putem muta o activitate de pe mașina 2 pe mașina 1 (sau invers) pentru a schimba care ar fi mașina cu load-ul maxim

⇒ max $(\mathcal{L}_2, \mathcal{L}_1) <$ OPT dar NU SE POATE, ar fi mai bun decât optim.

⇒ $|\mathcal{L}_2 - \mathcal{L}_1| < 10$

Pentru 80 și 120, max $(\mathcal{L}_1, \mathcal{L}_2) = OPT \leq 105$

$\Rightarrow$ Alg. nu poate fi 1.1 aproximativ deoarece $\frac{ALG}{OPT} = \frac{120}{105} > 1.1$



100 și 100 dacă se mută așa

105 și 105 (mai multe nu se poate)

3. Avem activitățile sortate descrescător $\rightsquigarrow$ curs 8, slide 43

LEMA 3: Pentru mulțimea de activități preprocesate $t_1, t_2, \dots t_m$ cu proprietatea $t_1 \geq t_2 \geq \dots \geq t_m$, dacă $n > m$, atunci $OPT \geq t_m + t_{m+1}$.

- fie M = mașina cu load-ul maxim la finalul aplicării alg $\Rightarrow$ ALG = load(M) și J = task-ul asignat ultima dată (implicit pe mașina M)
- introducem notația load' $\rightsquigarrow$ load'(i) = load-ul mașinii i înainte de asignarea task-ului J.

$\Rightarrow$ **ALG = load'(M) + t_J**

LEMA 1: curs 8, slide 32 $\Rightarrow$ $OPT \geq \max\left\{\frac{1}{m}\sum_{j=1}^{n} t_j, \max\{t_j \mid 1 \leq j \leq m\}\right\}$

$\Rightarrow OPT \geq \frac{1}{m}\sum_{j=1}^{n} t_j$

$\Rightarrow load'(M) \leq \frac{1}{m}\sum_{i=1}^{m} load'(i) = \frac{1}{m}\sum_{1 \leq j \leq J} t_j \leq \frac{1}{m}\sum_{1 \leq j \leq n} (t_j - t_J)$

$\leq \frac{1}{m}\sum_{j=1}^{n} t_j - \frac{1}{m} t_J \leq OPT - \frac{1}{m} \cdot t_J$

⇒ pot exista două cazuri

1. J se poate plasa pe o nouă mașină ⇒ ALG = OPT = J

2. J nu are loc pe o mașină goală ⇒

$$ALG = load'(M) + t_J \leq OPT - \frac{1}{m} \cdot t_J + t_J$$

$$\| $$

$$OPT + \left(1 - \frac{1}{m}\right) t_J \leq$$

$$\leq OPT + \left(1 - \frac{1}{m}\right) \cdot \frac{1}{2} (t_m + t_{m+1})$$

$$\leq OPT + \left(\frac{1}{2} - \frac{1}{2m}\right) OPT \leq$$

$$\leq OPT \left(\frac{3}{2} - \frac{1}{2m}\right)$$

⇒ factorul de aproximare poate fi îmbunătățit la $\left(\frac{3}{2} - \frac{1}{2m}\right)$.

Travelling Salesman Problem

1. TSP cu toate muchiile cu ponderea 1 sau 2.

a) Presupunem prin red. la absurd că probl. nu mai este NP-hard în acest caz.

Fie G (graf) neponderat ⟶ pentru G, problema găsirii unui ciclu hamiltonian este NP-hard.

Construim G' din G, după regulile:

- nodurile din G cu ponderea 1 se păstrează în G'
- G' va conține noduri suplimentare de ponderea 2 până când devine graf complet.

Dacă am rula TSP pe graful G' am obține costul total N, deoarece am alege mereu muchiile de cost 1, cele preluate din G, iar pe restul pur și simplu le-am ignora.

⇒ deci ∃ un ciclu hamiltonian în graful G

⇒ problema presupusă non-NP-hard a furnizat o soluție NP-hard ⇒ ↯ ⇒ problema rămâne NP-hard pentru aceste circumstanțe

b) Caz 1: toate laturile egale ($a = b = c$)

1.1. $a = b = c = 1$ ⇒ se satisface inegalitatea $1+1 \geq 1$

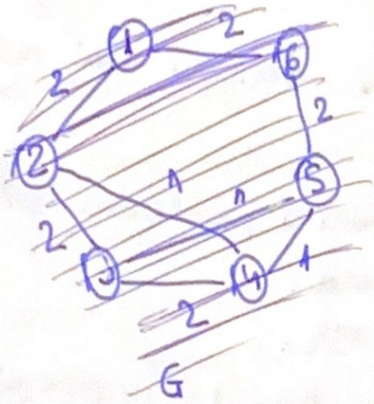
1.2. $a = b = c = 2$ ⇒ se satisface inegalitatea $2+2 \geq 2$

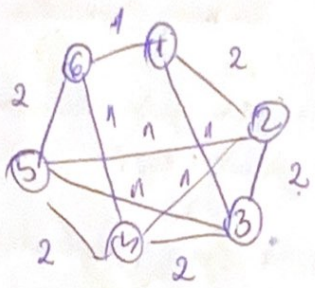
Caz 2: $a = b = 1$, $c = 2$ $\quad a + c \geq b \quad (1+2 \geq 1)$ Adev.

Caz 3: $a = b = 2$, $c = 1$ $\quad 2+1 \geq 2$ Adev

⇒ Se respectă regula triunghiului în aceste situații cu aceste ponderi

c) putem incerca un contraexemplu





Pentru graful G, TSP ar oferi soluția

1 2 3 4 5 6 1 cu un cost

c = 2+2+2+2+2+1 = 11

dar soluția optimă este OPT

1 3 5 2 4 6 1 cu un cost

c = 1+1+1+1+1+1 = 6

$\frac{3}{2} \cdot 6 \geq 11$ ($6 = OPT$, $11 = ALG$) $\Rightarrow$ algoritmul nu este $\frac{3}{2}$ aproximativ pentru această problemă

Vertex Cover

a) Worst case va fi dacă vom avea grupuri de variabile de forma $(x_{i_1}, x_{i_2}, x_{i_3}, x_k)$ și se alege mereu x_k (sau altă variabilă care se găsește doar într-un termen). În acest caz, algoritmul 3CNF va face n transformări (pentru n termeni) ale lui x din false în true (deoarece celălalt termen va fi true deja).

OPT = 1 ; ALG = (n-1)

⇒ Algoritmul este (n-1) aproximativ.

b)

```
cât timp C ≠ 0 execută
    alege aleator c din C
        pentru i ∈ var(c):
            i ← true
    eliminăm toate predicatele ce conțin i-uri (variabilele) din C
```

Am considerat $C = \{c_1, c_2, \dots c_n\}$ mulțimea predicatelor și $X = \{x_1, x_2 \dots x_m\}$ mulțimea variabilelor.

Dem: Fie Cd = mult. de predicatelor disjuncte.

- OPT ≥ |Cd|, deoarece OPT(=) toate eval. să fie "true".
- ALG va scoate din posibilele variabile de modificat, la fiecare pas, un număr de 3 variabile ⇒

ALG ≤ 3·OPT ⇒ ALG este 3 aproximativ.

c) Sub forma unei probleme de programare liniară, enunțul ar fi astfel:

Fie mulțimea variabilelor $X = \{x_1, x_2, \dots x_n\}$, unde $x_i = 1$, dacă x_i a fost true sau 0 în caz contrar. Constrângeri:

C_1: orice predicat C_i cu variabilele x_a x_b x_c are proprietatea că $x_a + x_b + x_c \geq 1$ (adică minim una din ele este "true")

C_2: Orice variabilă x_a, $x_a \leq 1$

d) Soluție:

- $x_a \leq 1$
- dacă $x_i \geq \frac{1}{3} \rightarrow x_i \leftarrow$ "true"

altfel $x_i \leftarrow$ "false"

Demonstrație:

$$ALG = \sum_{i=1}^{n} Eval(x_i) \leq \sum_{i=1}^{n} Eval(x_i) \cdot 3x_i = 3\sum_{i=1}^{n} Eval(x_i) \cdot x_i$$

unde $Eval(x_i) = \begin{cases} 1, & x_i \geq \frac{1}{3} \\ 0, & \text{altfel} \end{cases}$

$\leq 3OPT$

$\Rightarrow ALG \leq 3OPT$