Tema 1 Algoritmi Avansați

Bouruc Petru-Liviu

March 25, 2021

Knapsack

a)

```
#include <iostream>
#include <set>

using namespace std;

set<int> sume;

int main()
{
    int n, k, Max = 0;
    cin >> n >> k;
    sume.insert(0);
    for(int i = 1, x; i <= n; ++i)
    {
        cin >> x;
        for(auto j : sume)
            if(j + x <= k)
            {
                sume.insert(j + x);
                Max = max(Max, j+x);
            }
    }
    cout << Max;
    return 0;
}
```

b)

```
#include <iostream>
#include <fstream>

using namespace std;
```

```
ifstream fin("date.in");

int main()
{
    int k, sol = 0, x;
    fin >> k;
    while(fin >> x)
        if(sol + x <= k) sol += x;
        else sol = max(sol, x);
    cout << sol;
    return 0;
}
```

Ideea de rezolvare a acestui subpunct este urmatoarea: in timp ce citim fiecare numar incercam sa il adunam la suma, fara a depasi valoarea k. La un moment dat vom avea o suma mai mica decat $k/2$ si vom incerca sa adunam la aceasta suma un numar x, putem avea 2 cazuri:

- daca suma va fi mai mare decat k atunci cu siguranta x va fi cel putin $k/2$, astfel il putem lua, respectand conditia
- daca suma este mai mica decat k, o pastram si trecem mai departe

Load Balance

2)

Pentru cei 2 algoritmi, avem relatiile: $\begin{cases} OPT \leq ALG1 \leq 2OPT \ (1) \\ OPT \leq ALG2 \leq 4OPT \ (2) \end{cases}$

a)

Inmultind (1) cu 2, obtinem $2OPT \leq 2ALG1 \leq 4OPT$ (3)

Fie un input I astfel incat $ALG2(I) = 4OPT$ (upper-bound pentru ALG2) (4).

Din (3) si (4) $\implies$ exista un algoritm I pentru care $ALG2(I) \geq 2ALG1(I)$

b)

Inmultind (2) cu 2, obtinem $2OPT \leq 2ALG2 \leq 8OPT$ (5)

Fie un input I astfel incat $ALG1(I) = 2OPT$ (upper-bound pentru ALG1) (6).

Din (5) si (6) $\implies$ nu exista un algoritm I pentru care $ALG1(I) > 2ALG2(I)$

3)

Problema poate fi rescrisa sub forma: avem o multime de activitati t_1, t_1, ..., t_1 astfel incat $t_1 \geq t_2 \geq ... \geq t_n$ (curs 2, slide 43).

Lema 3: Daca $n > m \implies OPT \geq t_m + t_{m+1}$

Fie k - masina cu load-ul cel mai mare la sfarsitul asignarii (ALG = load(k))
Fie J - ultimul job asignat masinii k
Notam cu load'(i) - loadul masinii i dupa ce s-au asignat primele j-1 activitati $\implies$ ALG = load'(k) + t_j

$Lema\ 1\ :\ OPT \geq max\{\frac{1}{m}\sum_{1\leq j\leq n} t_j, max\{t_j|1 \leq j \leq n\}\} \implies OPT \geq \frac{1}{m}\sum_{1\leq j\leq n} t_j$
Asadar,
$load'(k) \leq \frac{1}{m}\sum_{1\leq i\leq m} load'(i) = \frac{1}{m}\sum_{1\leq j<J} t_j \leq \frac{1}{m}(\sum_{1\leq j\leq n} t_j - t_J) \leq \frac{1}{m}\sum_{1\leq j\leq n} t_j - \frac{1}{m}t_J \leq OPT - \frac{1}{m}t_q$

In momentul de fata, avem 2 cazuri:

Cazul I)

Jobul J poate fi programat pe o masina goala astfel incat sa nu se depaseasca OPT, deci ALG = OPT.

Cazul II)

Altfel,
$ALG = load'(k) + t_J \leq OPT - \frac{1}{m}t_J + t_J = OPT + (1 - \frac{1}{m})t_J \leq OPT + (1 - \frac{1}{m}) \cdot \frac{1}{2}(t_m + t_{m+1}) \leq OPT + (\frac{1}{2} - \frac{1}{2m})OPT \leq (\frac{3}{2} - \frac{1}{2m})OPT$

TSP

1)

a)

Presupunem prin reducere la abusrd ca problema nu este NP-Hard.

Fie un graf oarecare G', neponderat. Stim ca problema gasirii unui ciclu hamiltonian intr-un astfel de graf este o problema NP-Hard. Asadar, din G construiesc din graful G, astfel:

- toate muchiile din G se gasesc si in G' cu costul 1
- adaugam muchii in G' pana cand acesta devine graf complet, ponderea fiecarei muchii fiind 2

Rulam algoritmul TSP pentru graful G'. Daca acesta ne ofera un raspuns de cost total N, inseamna ca muchiile luate in calcul provin din graful G (cele de cost 1), deci exista un ciclu hamiltonian in graful G, deci o problema pe care am considerat-o la inceput a $nu\,fi\,NP-Hard$ ne-a furnizat o solutie la o problema pe care o stim a fi NP-Hard $\implies$ contradictie

b)

Pentru oricare 3 muchii, valorile ponderilor pentru acestea pot apartine urmatoarelor multimi:

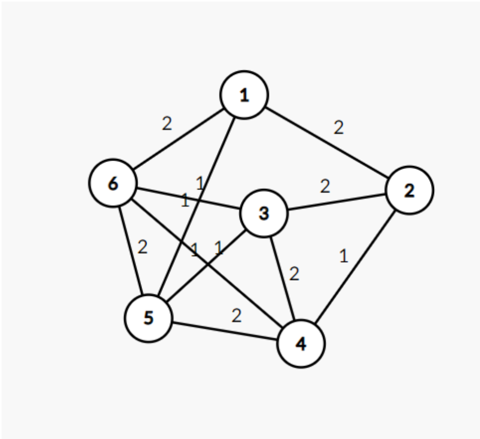
- 1, 1, 1 (1 + 1 ≥ 1)
- 1, 1, 2 (1 + 1 ≥ 2)
- 1, 2, 2 (1 + 2 ≥ 2)
- 2, 2, 2 (2 + 2 ≥ 2)

Asadar, inegalitatea triunghiului este satisfacuta pentru oricare caz.

c)

Pentru a verifica, putem gasi un contraexemplu.

Fie urmatorul graf G:



Algoritmul TSP genereaza drumul: 1, 2, 3, 4, 5, 6, 1 de cost total 12
Ciclul hamiltonian de cost optim: 1, 2, 4, 6, 3, 5, 1 de cost total 7
$\frac{3}{2} \cdot 7 < 12 \implies$ algoritmul nu este 3/2 aproximativ pentru aceasta problema

Vertex Cover

a)

Cel mai defavorabil caz ar fi atunci cand fiecare clauza este de forma (x_1, x_1, x_i) cu $i \neq 1$, iar in urma algoritmului este ales mereu aleator a 3-a variabila, x_i. Astfel, pentru a produce un rezultat valid, algoritmul poate produce (n-1) operatii, in timp ce solutia optima este doar de o operatie. Deci algoritmul este (n-1)-aproximativ.

b)

Putem modifica algoritmul astfel:

1. C = C1, C2, ..., Cn multimea de predicate, X = x1, x2, ..., xn multimea de variabile
2. cat timp C ≠ 0 executa
3. alegem aleator Ci ∈ C
4. fie x1, x2, x3 variabilele din Ci
5. x1 <- true; x2 <- true; x3 <- true
6. eliminam din C toate predicatele ce contin x1, x2, x3

Fie C' multimea predicatelor disjuncte. Deoarece OPT presupune ca fiecare evaluare sa fie "true" atunci |OPT| ≥ |C'|. Cum de fiecare data cand alegem un predicat aleator, "scapam" de trei variabile pe care nu le vom mai lua in calcul ulterior, inseamna ca algoritmul nostru ALG2 ≤ 3OPT.

c)

Fie $X = \{x_1, x_2, ..., x_n\}$ cu proprietatea ca $x_i = 1$ daca variabila x_i a fost setata ca "true", altfel $x_i = 0$.

Vrem sa minimizam $\sum_{1 \leq i \leq n} f(x_i) \cdot x_i$

Constrangerile sunt:

- pentru orice predicat C_i format din x_i, x_j, x_k sa avem $x_i + x_j + x_k \geq 1$
- pentru orice variabila x_i sa avem $x_i \leq 1$

d)

Vom modela problema sub forma uneia de tipul 1/0 linear programming. Astfel, cum $x_i \leq 1$, daca $x_i \geq \frac{1}{3}$ atunci vom "lua" variabila x_i (o vom seta pe "true"), altfel nu.

$$ALG = \sum_{1 \leq i \leq n} f(x_i) \begin{cases} 1, x_i \geq \frac{1}{3} \\ 0, x_i < \frac{1}{3} \end{cases} \leq \sum_{1 \leq i \leq n} f(x_i) \cdot 3x_i = 3 \sum_{1 \leq i \leq n} f(x_i) \cdot x_i \leq 3OPT$$