Căruțaru Adriana Ștefania
Grupa 243

Algoritmi Avansați

Tema 1. Algoritmi Aproximativi

I Knapsack

1. a)
```
f = open("date.in")
def citire():
    K = int(f.readline())
    S = []
    i = f.readline()
    while i:
        ob = int(i)
        S.append(ob)
        i = f.readline()
    return K, S, len(S)

K, S, n = citire()
matriceSol = [[0 for i in range(K+1)] for j in range(n+1)]
for i in range(1, n+1):
    for j in range(1, K+1):
        if j - S[i-1] >= 0:
            matriceSol[i][j] = max(matriceSol[i-1][j], S[i-1] +
                                   matriceSol[i-1][j-S[i-1]])
        else:
            matriceSol[i][j] = matriceSol[i-1][j]
print("Valoarea obiectelă din rucsac este", matriceSol[n][K])
```

```
* afișare obiecte
i = n
j = K
while i > 0:
    if matriceSol[i-1][j] = matrice[i][j]:
        i -= 1
    else:
        print(i, end = " ")
        j -= S[i-1]
        i -= 1
f.close()
```

b)

```
f = open("date.in")
K = int(f.readline())
suma = 0
for ob in f.readline().split():
    ob = int(ob)
    suma += ob
    if suma > K:
        suma -= ob
        if ob > suma:
            suma = ob
print(suma)
f.close()
```

Demonstrație 1/2 - aproximativ

$K \geq OPT$

Fie suma = suma la pasul i

obi = obiectul de la pasul i el nu mai poate fi adăugat la suma

Se disting 2 cazuri

Cazul 1: Nu există un astfel de obiect obi. Atunci

suma = OPT = suma tuturor obiectelor

Cazul 2: Prin adăugarea obiectului obi suma depășește K, adică suma + obi > K

Există 2 posibilități:

1. suma < K/2 => obi > K/2 > suma

În această situație îi dăm sumei valoarea lui obi > K/2 >= OPT/2

2. suma >= K/2 >= OPT/2 => obi < K/2 < suma

În această situație suma rămâne la fel

Luând în considerare ce am demonstrat mai sus avem

OPT/2 <= suma <= OPT

II Load Balance

b. ALG1 = 2-aproximativ ; ALG2 = 4-aproximativ

a) Fie OPT soluția optimă

ALG1 = 2-aproximativ => OPT <= ALG1 <= 2 OPT /·2

2OPT <= 2ALG1 <= 4OPT

Alegem J un input cu proprietatea că ALG2(J) = 4OPT. (*)

ALG2(J) = 4OPT
2ALG1(J) <= 4OPT
/=> ALG2(J) >= 2ALG1(J) => Există un J cu proprietatea enunțată

(*) ALG2 = 4-aproximativ => OPT <= ALG2 <= 4OPT

b) OPT <= ALG2(i) <= 4OPT /·2

2OPT <= 2ALG2(i) <= 8OPT

Cum ALG1 este 2-aproximativ => ALG1(J) <= 2OPT ∀ J input.

2ALG2(J) >= 2OPT
2OPT >= ALG1(J)
/=> ∄ J input a.î. ALG1(J) > 2·ALG2(J)

3. Fie o mulțime de n activități sortate descrescător după timpul de procesare $\Rightarrow t_1 \geq t_2 \geq t_3 \geq \dots \geq t_{n-1} \geq t_n$.

Din lema 1 din Curs 2 avem că $OPT \geq \max\left(\frac{1}{m}\sum_{1\leq j\leq n} t_j, \max\{t_j \mid 1\leq j\leq n\}\right)$

(În cazul de față avem $\max\{t_j \mid 1\leq j\leq n\} = t_1$) $\Rightarrow$

$\Rightarrow OPT \geq \max\left(\frac{1}{m}\cdot\sum_{1\leq j\leq n} t_j, t_1\right)$

Din lema 3 din Curs 2 avem că dacă $n > m$ atunci $OPT \geq t_{m+1} + t_{m+1}$, unde m este numărul de mașini

Fie k indicele mașinii cu load maxim în urma algoritmului $\Rightarrow$ avem $ALG = load(k)$.

Considerăm q ultima activitate a mașinii cu indicele k.

Considerăm $load(k) = load$-ul mașinii k după ce au fost distribuite primele $q-1$ activități, $\Rightarrow ALG = load(k) + t_q$

$$load(k) = \frac{1}{m}\sum_{1\leq i\leq m} load(i) = \frac{1}{m}\sum_{1\leq i\leq q} t_j = \frac{1}{m}\left(\sum_{1\leq j\leq n} t_j - t_q\right) \leq$$

$$\leq \frac{1}{m}\sum_{1\leq j\leq n} t_j - \frac{1}{m}t_q \leq OPT - \frac{1}{m}t_q$$

Cazul I: $q \leq m \Rightarrow$ activitatea q va fi asignată unei mașini goale $\Rightarrow ALG = t_q \leq t_1 \leq OPT \Rightarrow ALG = OPT$

Cazul II: $q > m \Rightarrow ALG = load(k) + t_q \leq OPT - \frac{1}{m}t_q + t_q = OPT + \left(1 - \frac{1}{m}\right)t_q$

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

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

$\Rightarrow$ Algoritmul GAP $\frac{3}{2} - \frac{1}{2m}$ - aproximativ

III TSP

1. Considerăm că graful TSP conține doar ponderile 1 sau 2.

a) Presupunem prin absurd că problema nu este NP-hard.

Considerăm G un graf nepondereat ⇒ determinarea unui ciclu Hamiltonian în G este o problemă NP-hard.

Fie următorul algoritm:

Pas 1: Construim un nou graf G' complet din toate muchiile lui G (acestea vor avea costul 1 în G'), și completăm cu muchii de cost 2 până G' este graf complet.

Pas 2: Aplicăm algoritmul pentru TSP pe G' și obținem traseul de cost minim.

Conform ipotezei, algoritmul se execută în timp polinomial.

În continuare, dacă costul traseului este egal cu numărul de muchii din G, putem afirma că G conține un ciclu hamiltonian. Existența ciclului și componentele acestuia au fost determinate în timp polinomial ⇒ ipoteza este falsă ⇒ problema este NP-hard.

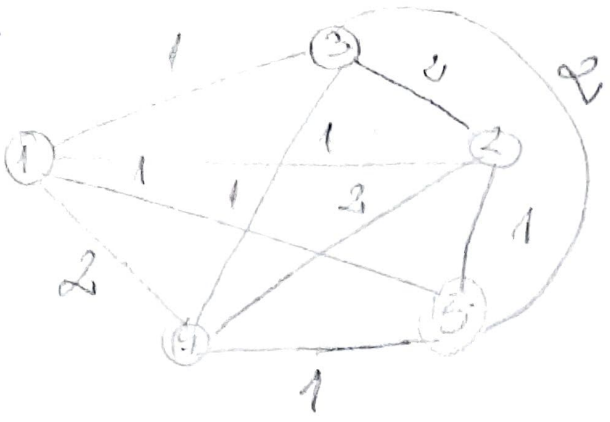
b) Distingem 4 cazuri:

1. 1, 1, 1 ⇒ 1+1 ≥ 1 Adevărat
2. 1, 1, 2 ⇒ 1+1 ≥ 2
   1+2 ≥ 1
   2+1 ≥ 1
   ⇒ Adevărat
3. 1, 2, 2 ⇒ 1+2 ≥ 2
   2+1 ≥ 2
   2+2 ≥ 1
   ⇒ Adevărat
4. 2, 2, 2 ⇒ 2+2 ≥ 2
   2+2 ≥ 2
   2+2 ≥ 2
   ⇒ Adevărat.

⇒ Pondrile satisfac inegalitatea triunghiului

c)

G:



OPT 1, 3, 4, 5, 2, 1

$\Rightarrow OPT = 1 + 1 + 1 + 1 + 1 = 5$

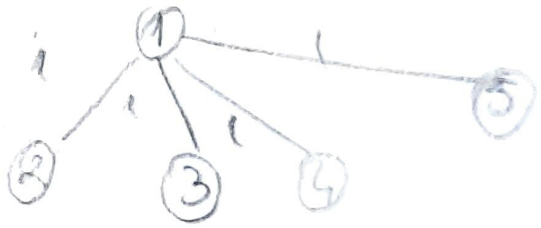
ALG: 1, 2, 3, 4, 5, 1

$\Rightarrow ALG = 1 + 2 + 2 + 2 + 1 = 8$

$\Rightarrow 8 \not\leq \frac{3}{2} \cdot 5 = 7,5$

$\Rightarrow ALG \not\leq \frac{3}{2} \cdot OPT$

MST:



$\Rightarrow$ Algoritmul nu este $\frac{3}{2}$-aproximativ

IV Vertex cover

a) Considerăm o formulă de forma $\bigwedge_{x_k \in X, \{x_i, x_j\}, i \neq j, i,j,k \leq n} (x_i \vee x_j \vee x_k)$, adică $x_i, x_j \in X$ sunt 2 variabile fixate și x_k ia restul valorilor din X în afară de x_i și x_j. Astfel, formula este formată din $n-2$ predicate. La pasul 3 al algoritmului se alege un predicat aleator. În final vor fi alese toate predicatele. La pasul 4 se alege aleator o variabilă x din predicat. La fiecare alegere aleatoare se poate ca algoritmul Greedy să aleagă mereu variabila x_k, astfel efectuând $n-1$ operații de ștergere până se vor șterge toate toate predicatele. Dacă la prima iterație ar fi ales x_i sau x_j atunci s-ar fi eliminat toate predicatele.

Astfel, considerăm $OPT = 1$ și $ALG = n-2 \Rightarrow OPT \leq ALG \leq (n-2)OPT$

Conform celor prezentate anterior, algoritmul Greedy-3SAT este $(n-2)$-aproximativ, în funcție de numărul de variabile din mulțimea X.

b) 1: $C' = \{c_1, \dots, c_m\}$ mulțimea de predicate
$X = \{x_1, \dots, x_n\}$ mulțimea de variabile

2: cât timp $C' \neq \emptyset$ execută

3: alegem aleat. un $c_j \in C$

4: fie x_i, x_j, x_k variabilele din c_j

5: $x_i \leftarrow$ true, $x_j \leftarrow$ true, $x_k \leftarrow$ true

6: eliminăm din C' toate predicatele ce conțin x_i, x_j sau x_k

7: return X

Justificare:

La fiecare pas, în C' vom avea doar predicatele care încă nu au fost evaluate cu true. În final, C' este vidă $\Rightarrow$ nu vor fi predicate ce nu fi fost evaluate cu true.

Considerăm C^* mulțimea de predicate selectate la pasul 3. Când adăugăm un predicat în C^*, toate celelalte ~~predicate din C~~ ~~vor~~ predicate din C' care conțin ~~același~~ măcar una dintre variabilele predicatului curent vor fi șterse și nu vor mai fi niciodată adăugate în C^*. Astfel putem afirma că mulțimea C^* este o mulțime de predicate disjuncte (nu au variabile comune).

Considerăm ALG soluția furnizată de algoritmul Greedy și OPT soluția optimă.

Astfel, cum fiecare predicat din C^* conține exact 3 variabile
$\Rightarrow |ALG| = 3|C^*| \leq 3|OPT| \Rightarrow$ algoritmul Greedy este 3-aproximativ.

c) Fie $X = \{x_1, x_2, \dots, x_n\}$ cu proprietatea că dacă $x_i = 1$ atunci x_i este considerat true. Dacă $x_i = 0$ atunci x_i este considerat fals și dorește minimizarea $\sum_{i=1}^{n} x_i$

Constrângerile:

1. Pentru orice predicat de forma $(x_i \vee x_j \vee x_k)$ trebuie ca $x_i + x_j + x_k \geq 1$
2. $\forall i = \overline{1,n}$ avem $0 \leq x_i \leq 1$

d) Soluție problemă programare liniară

Rezolv problema în varianta cu numere reale și dacă $x_i \geq \frac{1}{3}$ atunci rotunjesc x_i la 1 și implicit x_i va fi considerat true, altfel x_i va avea valoarea 0 și va fi considerat fals.

Prin prima constrângere de la subpunctul anterior ne asigurăm că pentru orice predicat C măcar una dintre variabilele sale este 1 (adică true). Astfel o conjuncție care are măcar un termen true va fi evaluată la true. Astfel, pentru un predicat $(x_i \vee x_j \vee x_k)$, avem x_i sau x_j sau $x_k \geq \frac{1}{3}$, deci măcar uneia dintre aceste variabile i se va atribui valoarea de true.

Considerăm OPT soluția optimă a problemei și ALG soluția descrisă anterior.

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

$\Rightarrow$ soluția prezentată anterior este un algoritm 3-aproximativ.