DIMA OANA-TEODORA 241

# TEMA 1

## Knapsack (2p):

1) $S = \{ s_1, s_2, \dots, s_n \}$

$K \in \mathbb{N}, K \geq s_i, i = \overline{1,n}$

a) - profitul unui obiect este egal cu greutatea acestuia
- algoritmul este același ca la Knapsack 0/1
- pseudo-polinomial $O(n * K)$
- nu mai folosesc matrice ⇒ optimizez spațiul auxiliar ⇒ $O(K)$ pt spațiu

DP[i] = profitul ghiozdanului cu capacitatea "i", $i = \overline{1,K}$

DP[K] = suma maximă cerută

Cod: → PYTHON

```
def KnapSack(K, S, n):
    DP = [0] * (K+1)
    for i in range(n):
        for j in range(K, S[i], -1):
            DP[j] = max(DP[j],
                        S[i] + DP[j - S[i]])

    return DP[K]


# citire
f = open("in.txt")
n = (int)(f.readline())
```

```
S = []
for i in range(n):
    S.append ( (int) (f.readline()))

  K = (int) (f.readline)
  print ( "S max= ", KnapSack (K, S, n))
```

b) Algoritm aproximativ care calculează o sumă cel puțin pe jumătate de mare ca cea optimă

Complexitate timp: O(n)

Complexitate spațiu: O(1)

Idee: adun într-o variabilă sum și în cazul în care depășesc valoarea K, stochez în sumă maximul dintre suma actuală și elementul curent.

Cod: → PYTHON

```
f = open ("in.txt")
n = (int) (f.readline())
K = (int) (f.readline())
sum = 0
for i in range (n):
    x = (int) (f.readline())
    sum += x
    if sum > K:
        print ( max (sum - x, x))
        break
```

Load Balance (max 3p)

1) a) timpul de lucru de cel mult 100, alg 1.1 aprox?

Exemplu care susține ~~[illegible]~~ ipoteza

$n = 3$ activități

2 au timpul de 60 fiecare $\Rightarrow$ $2 \times 60 = 120$

$60 < 100$

1 are timpul de 80, $80 < 100$

împreună $80 + 120 = 200$

$\Rightarrow$ alg optim se încadrează în 1.1 aprox.

b) timpul de lucru de cel mult 10, alg 1.1 aprox?

Contradicție

Dacă avem activități de cel mult 10 $\Rightarrow$ diferența maximă este 10.

Ip că avem o configurație optimă unde diferența este mai mare ca 10 $\Rightarrow$ mașina mai încărcată are mai mult de o activitate $\rightarrow$ iau o activitate de aici și o pun pe mașina mai liberă $\Rightarrow$ soluție mai bună ca cea optimă

cazul optim $\rightarrow$ mașina cea mai încărcată $= 105$

$\rightarrow$ mașina cealaltă $= 95$

timpul maxim este 120 $\Rightarrow$ $\frac{120}{105} \geq 1.1$ $\Rightarrow$

$\Rightarrow$ alg nu are un factor de aprox de 1.1 pe cazul propus

~~3) [illegible]~~

3) Ordered - Scheduling Algorithm

+ ~~pr~~ preprocesare cu care sortăm descrescător activitățile după timpul de desfășurare.

Lower bound → mașina care lucrează cel mai mult va lucra cel puțin cât cel mai costisitor job (cel puțin cât media mașinilor)

$$\frac{3}{2} \longrightarrow \frac{3}{2} - \frac{1}{2m}$$

<u>Idee</u>: îl schimb pe $t_q$ în $(t_m + t_{m+1})$ load maxim

Fie K indicele mașinii cu load maxim

Fie q ultimul job adăugat mașinii K

Fie load'(M) - load-ul mașinii M după ce am asignat primele q-1 joburi, dar nu și jobul K.

$$load'(K) \le \frac{1}{m} \sum_{1 \le i \le m} load'(i) = \frac{1}{m} \sum_{1 \le j \le q} t_j < \frac{1}{m} \sum_{1 \le j \le m} t_j \le LB \le OPT$$

<u>Lema 3</u>: $OPT \ge t_m + t_{m+1}$

$$load'(K) + t_q \le \frac{1}{m} \sum_{1 \le i \le m} (-t_q) + t_q) \le \frac{1}{m} \sum_{1 \le j \le m} t_j - \frac{1}{m}\left(\frac{1}{2}(t_m + t_{m+1})\right) +$$

$$+ \frac{1}{2}(t_m + t_{m+1}) \le \frac{1}{m} \sum_{1 \le j \le m} (t_j - t_q) + \frac{1}{2}\underbrace{(t_m + t_{m+1})}_{OPT} \le$$

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

~~$\frac{1}{m} \sum t_j = OPT$~~

~~$t_q = \frac{1}{2m} OPT$~~

TSP (max 2p):

1) Varianta TSP unde toate muchiile au ponderea 1 sau 2.

a) problema rămâne NP-hard pt aceste instanțe

→ reduc problema TSP la o problemă NP-hard cunoscută și anume problema ciclului hamiltonian.

→ grafic $G=(V,E)$ pt ciclul hamiltonian
grafic $G'=(V',E') + K=$ cost maxim, pt TSP

→ construire $G'$ urmând pașii:

- adaug costul $c(e)=1$ pt toate $m \in E$
- conectez marginile rămase din $E'$ care nu sunt prezente în $G$ având costul $c(e')=2$
- $K=N$

⇒ construire graf nou $G'$ în timp polinomial, convertind $G$ într-un graf complet $G'$

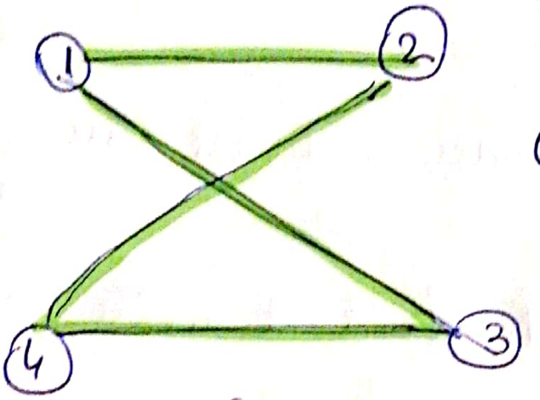
Dem: Pp că $G$ are un ciclu hamiltonian
Acum ~~#~~ toate $v \in V$ formează TSP de cost $N$
(deoarece $c(e)=1$ ~~$\forall e \in E$~~)

+ ciclu ⇒ se întoarce în vârful din care a plecat

Pp că $G'$ conține TSP de cost $N$
TSP traversează toate vârfurile grafului întorcându-se în vârful original.
Niciun vârf nu este ~~ext~~ exclus din graf ⇒

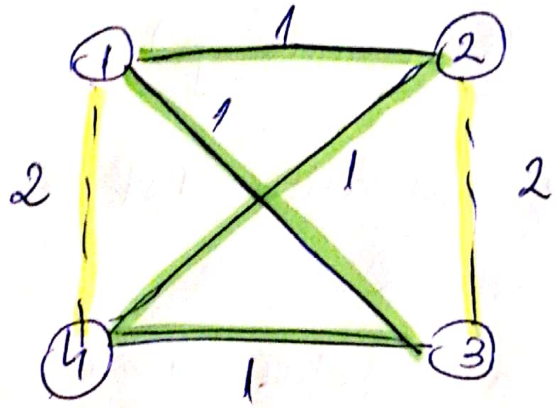
⇒ folosește toate $m \in E$ de cost $c(m)=1$ ⇒ ciclu hamiltonian în $G$

$N = 4$

$G =$ 

Ciclu hamiltonian:

1 2 4 3 1

$G' =$ 


TSP:

1 2 4 3 1

Cost = 4 = N

~~a) Regula triunghiului~~

~~$\forall a, b, c \quad a + b \geq c$~~

~~Verific toate combinațiile de muchii:~~

~~$a = 1, b = 1, c = 1$~~

~~$1 + 1 \geq 2$ (A)~~

~~$a = 1, b = 1, c = 2$~~

~~$1 + 1 \geq 2, \; 2 + 1 \geq 1, \; 1 + 2 \geq 1$ (A)~~

~~$a = 1$~~

b) Inegalitatea triunghiului $\forall u, v, w \in V$

$$\underbrace{c(u, w)}_{a} \leq \underbrace{c(u, v)}_{b} + \underbrace{c(v, w)}_{c}$$

Verific pe cazuri:

$a \leq b + c, \; \forall a, b, c \in \{1, 2\}$

$1 + 1 \geq 1 \Rightarrow 2 \geq 1$

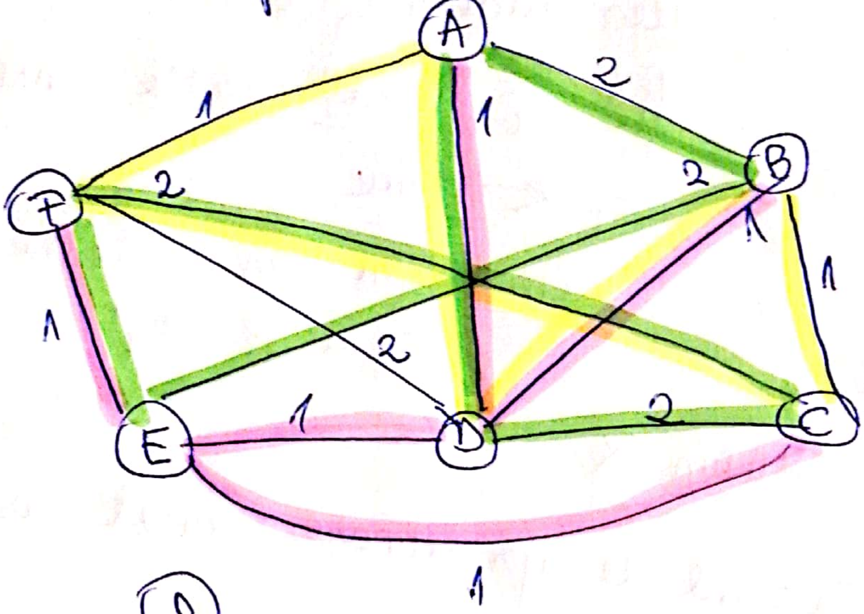
$1 + 1 \geq 2 \Rightarrow 2 \geq 2$

$1 + 2 \geq 2 \Rightarrow 3 \geq 2$

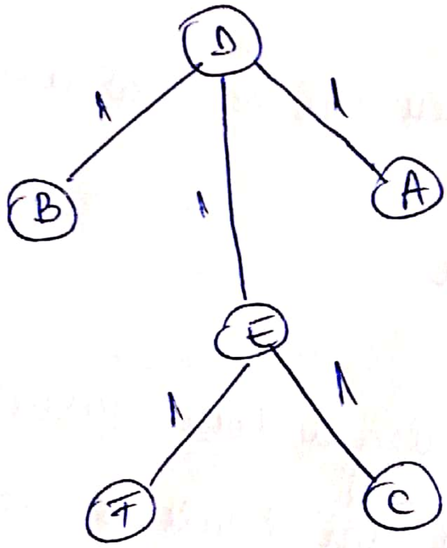
$2 + 2 \geq 2 \Rightarrow 4 \geq 2$

c) Aleg un contraexemplu

Fie G =



MST al G:



Rădăcina: D

D → A → B → E →

→ F → C → D ⇒

⇒ cost: 10

găsit de alg dat

Drum optim:

A → D → B → C → F → A ⇒ cost: 6

$\frac{10}{6} \geq \frac{3}{2} \iff 1,66 \geq 1,5$

Vertex cover (max 2p)

Greedy - 3CNF(C, X)

$C = \{c_1, \dots, c_m\}$

$X = \{x_1, \dots, x_n\}$

cât timp $C \neq \emptyset$ execută

aleg aleator $c_j \in C$

fie $x_i$ una dintre variabilele din $c_j$

$x_i \leftarrow$ true

eliminăm din C toate predicatele ce-l conțin pe $x_i$

return X

a) Factorul de aproximare (worst case)

- algoritmul alege mereu prima variabilă din cele m grupuri
- worst case: $n \geq m$
- soluție de cost $= m$
- soluție optimă $\rightarrow$ aleg a doua variabilă dintr-un grup care se regăsește în toate celelalte grupuri

$(x_1, x_{m+1}) \vee (x_2, x_{m+1}) \ldots \vee (x_m, x_{m+1})$

$\rightarrow$ soluție de cost $= 1$

$$ALG = \sum_{1 \leq i \leq m} f(v_i) \begin{cases} 1 \mid x_i \geq \frac{1}{m} \\ 0 \mid x_i < \frac{1}{m} \end{cases} \leq \sum_{1 \leq i \leq m} f(v_i) \cdot \cancel{m} x_i \leq$$

$$\leq m \sum_{1 \leq i \leq m} f(v_i) \cdot x_i \leq m\, OPT \Rightarrow ALG \leq m\, OPT$$

b) $C = \{c_1, \dots, c_m\}$

$X = \{x_1, \dots, x_n\}$

cât timp $C \neq \emptyset$ execută:

aleg $c_j \in C$

$x_i, x_j, x_k \leftarrow c_j$

$x_i, x_j, x_k \leftarrow$ true

elimin din $C$ toate predicatele care conțin $x_i, x_j, x_k$

returnez $X$

c) $X = \{x_1, x_2, \dots, x_n\}$

$$x_i = \begin{cases} 1, & x_i = \text{true} \\ 0, & x_i = \text{false} \end{cases}$$

Minimizăm: $\sum\limits_{1 \le i \le n} f(x_i)\, x_i$

Constrângeri:

pt $\forall x_i$, $x_i \le 1$

pt $\forall c_i$ $(x_i, x_j, x_k)$ avem: $x_i + x_j + x_k \ge 1$

d) 1/0 programare liniară

$x_i \le 1$

și $x_i \ge \frac{1}{3} \Longrightarrow x_i = \text{true}$

$$ALG = \sum_{1 \le i \le n} \begin{cases} 1, & x_i \ge \frac{1}{3} \\ 0, & x_i < \frac{1}{3} \end{cases} \le \sum_{1 \le i \le n} f(x_i)\, 3x_i = 3 \sum_{1 \le i \le n} f(x_i)\, x_i \le$$

$\le 3OPT$

Concluzia: $ALG \le 3OPT$