Gheorghe Andrei, Tema 1 Algoritmi Avansați

Knapsack:

1. a)

```
fin = open('inputP1', 'r')
k = int(fin.readline())
v = [int(x) for x in fin.readline().split()]
n = len(v) + 1
dp = [[0] * (k+1) for i in range(n)]

for i in range(n):
      dp[i][0] = 1


for i in range(1, n):
      for j in range(1, k+1):
      if j < v[i-1]:
            dp[i][j] = dp[i-1][j]
      else:
            dp[i][j] = 1 if dp[i-1][j-v[i-1]] == 1 else dp[i-1][j]

for i in range(k, 0, -1):
      if dp[n-1][i] == 1:
      print(i)
      break
```

b)

```
fin = open('inputP1', 'r')
k = int(fin.readline())

mx = 0

for x in fin.readline().split():
      x = int(x)

      mx += x
      if mx < k:
      mx -= x

      mx = max(mx, x)
print(mx)
```

Load Balance:

1. a) Da, este posibil ca factorul de aproximare sa fie corect
   Ex: pentru un input de 3 activități cu cu timpii 50, 70 si 80 => OPT = ALG = 120 => un factor de aproximare de 1, 1.1 > 1 => algoritmul poate fi 1.1 aproximativ
   b) Nu, algoritmul propus nu poate fi 1.1 aproximativ
   Notez loadul celor 2 masini cu L(a), respectiv L(b)
   Timpul de lucru al activităților este de cel mult 10 => În cazul optim |L(b) - L(a)| <= 10 (dem: presupun ca |L(b) - L(a)| > 10, dar timpul de lucru al activităților este de cel mult 10, atunci înseamnă ca pot muta o activitate de pe masina cu load-ul mai mare pe masina cu load-ul mai mic astfel max(L(b),L(a)) < OPT => contradicție) => max(L(b),L(a))=OPT<=105 => algoritmul nu poate fi 1.1 aproximativ deoarece ALG/OPT=120/105>1.1
2. ...
3.

$Fie\ k\ ma\text{sin}a\ cu\ incarcatura\ maxima\ si\ j\ ultimul\ job\ adaugat\ la\ k$

$$1.\ \frac{1}{m}\sum_{p=1}^{n} t_p \ \le\ max\left(\frac{1}{m}\sum_{p=1}^{n} t_p, t_{max}\right) \ \le\ OPT$$

$$2. Daca\ n > m,\ OPT \ge t_m + t_{m+1}$$

$$Daca\ j \le m$$

$$ALG = Load\left(M_k\right) \ =\ t_j \ \le\ t_{max} \ \le OPT$$

$$Daca\ j > m$$

$$ALG = Load\left(M_k\right) \ \le\ t_j + \frac{1}{m}\sum_{p=1}^{n} t_p - \frac{1}{m}\cdot t_j \le OPT + \frac{m-1}{m}\cdot t_j(\,din\ 1) \le OPT + \frac{m-1}{m}\cdot\frac{t_m + t_{m+1}}{2}$$

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

=> factorul de aproximare poate fi îmbunătățit la 3/2-1/2m

TSP:

1. a) Presupunem ca exista un algoritm optim în timp polinomial astfel incat sa putem rezolva TSP pt orice graf complet cu ponderi egale cu 1 sau 2.
   Știm ca HC-P este NPC.
   Fie un graf G oarecare pe care vrem sa rezolvăm HC-P. Din graful G construiește graful G’ astfel:
   V(G’)=V(G’)
   toate muchiile din G se găsesc și în G’ cu costul 1.
   Adaug muchii în G’ până când acesta devine graf complet, iar ponderea fiecărei muchii va fi 2
   Am presupus ca algoritmul oferă o soluție optimă.
   Dacă G are ciclu hamiltonian atunci ALG oferă o soluție la TSP de cost exact n.
   Altfel, algoritmul nu poate oferi o soluție mai mica decat n+1>n => exista un algoritm

optim în timp polinomial astfel incat sa putem rezolva TSP ⇔ P=NP => problema este NP-hard

b) Cazul 1: toate laturile sunt egale cu 1 sau 2 => triunghi echilateral, se respecta regula triunghiului

Cazul 2:

$L_1 = 2,\ L_2 = L_3 = 1$

$L_1 \geq L_2 \geq L_3 => 2 \geq 2$

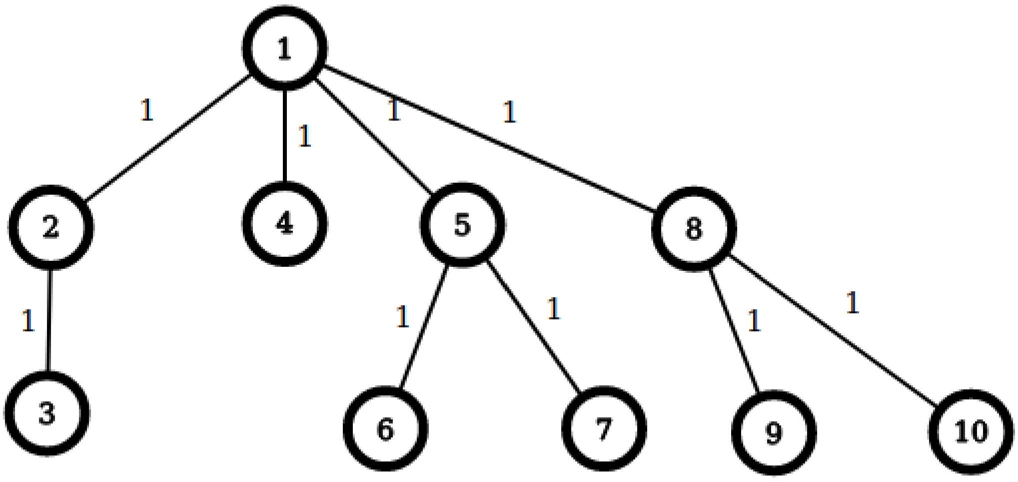
Cazul 3:

$L_1 = L_2 = 2,\ L_3 = 1$

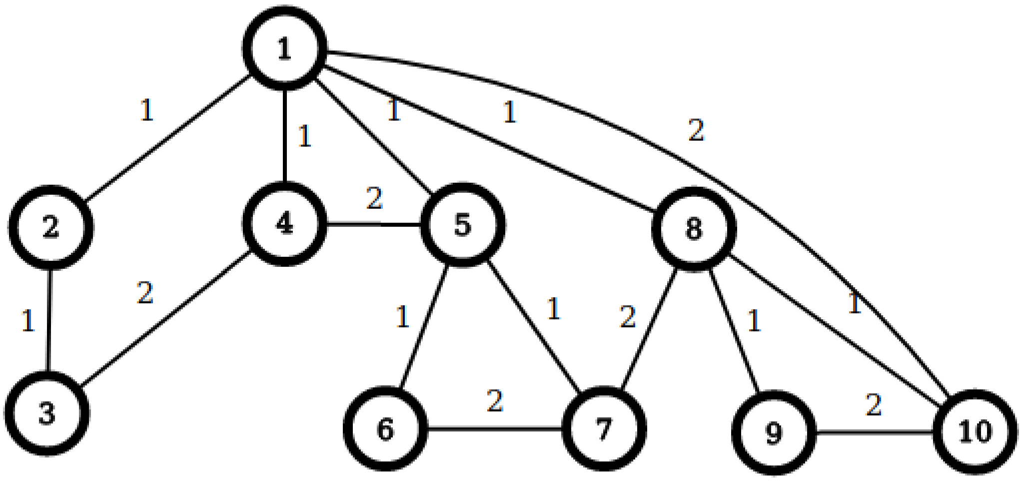
$L_1 \geq L_2 \geq L_3 => 3 \geq 2$

=> se respecta regula triunghiului

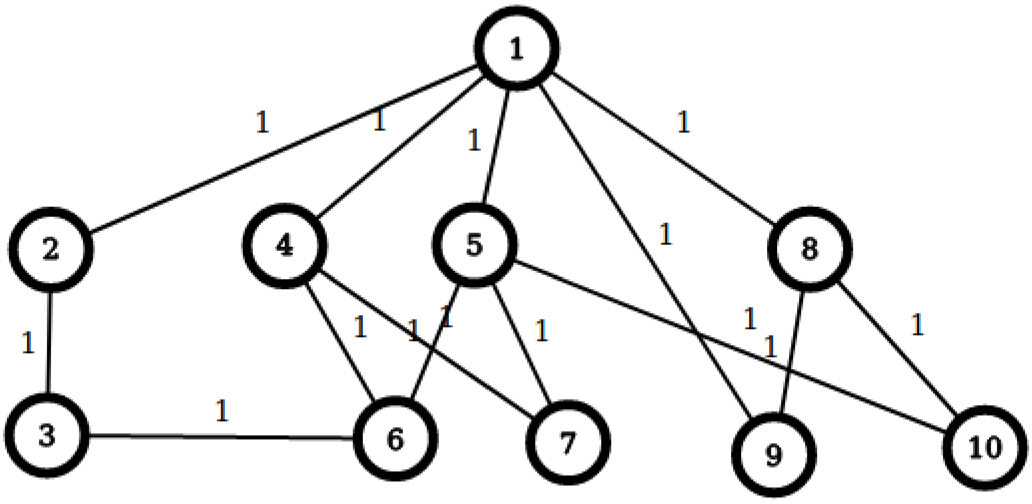
c) Fie un graf G cu următorul MST



Algoritmul propus în curs poate propune soluția 1 2 3 4 5 6 7 8 9 10 1 cu cost total 16



Dar soluția optimă este 1 2 3 6 4 7 5 10 8 9 1 cu cost total 10



16/10>3/2 => algoritmul nu este 3/2-aproximativ

Vertex Cover:

1. a) Factorul de aproximare este de ordinul n

   Ex: $(\mathbf{x_1} \lor \mathbf{x_2} \lor \mathbf{x_3}) \land (\mathbf{x_1} \lor \mathbf{x_3} \lor \mathbf{x_4}) \land \ldots \land (\mathbf{x_1} \lor \mathbf{x_{n-1}} \lor \mathbf{x_n})$

   OPT = 1 (e nevoie doar sa facem pe x1 <- true) dar algoritmul dat poate genera o soluție egala cu n-1

   b) 1: $C = \{C_1, \ldots, C_m\}$ mulțimea de predicate, $X = \{x_1, \ldots, x_n\}$ - mulțime de variabile

   2: cât timp $C \neq \varnothing$ execută

   3: Alegem aleator $(x_i\ x_j\ x_k) \in C$.

   4: $x_i \leftarrow$ true.

   5: $x_j \leftarrow$ true.

   6: $x_k \leftarrow$ true.

   7: Eliminăm din C toate predicatele ce îl conțin pe x_i, x_j, x_k.

   8: return X

   Fie $C = \{C_1, \ldots, C_m\}$ mulțimea de predicate, $X = \{x_1, \ldots, x_n\}$ - mulțime de variabile și OPT numărul minim de elemente din X care trebuie să aibă valoarea true astfel încât toată expresia să fie true. Fie $E' \subset X$ o mulțime de predicate cu elemente disjuncte.

   Fie S - o “acoperire” pentru expresia data. Deoarece E’ este mulțime de predicate cu elemente disjuncte atunci orice variabila din S poate “acoperi” doar un singur predicat din E’ => |S| >= |E’| pentru orice acoperire S. => OPT>=|E’|

   Fie E* - mulțimea de predicate selectate la linia 3 a algoritmului

   E* este o mulțime de predicate cu elemente disjuncte (toate predicatele ce conțin variabile care se găsesc în predicatul selectat la pasul curent sunt eliminate)

OPT>=|E*|

3OPT>=3|E*|=**ALG => algoritmul este 3-aproximativ**

c) C = {C_1, . . . , C_m} mulțimea de predicate, X = {x_1, . . . , x_n} - mulțime de variabile și A = {c_1 , … , c_n} unde c_i este 1 dacă variabila x_i este egala cu true în soluția aleasă, 0 altfel

Trebuie minimizata suma

$$\sum_{i=1}^{n} f\left(x_i\right) \cdot c_i$$

Constrangeri:

Pentru fiecare predicat (x_i x_j x_k) din C, $c_i + c_j + c_k$ >= 1 și 0<=c_i<=1 pentru orice i

d) Dacă c_i >= ⅓ adaug x_i ← true, altfel x_i ← false

$c_i + c_j + c_k$ >= 1 => pentru orice predicat (x_i x_j x_k) măcar unul dintre c_i,c_j,c_k>=⅓ => cel puțin o variabila va avea valoarea true pentru orice predicat => toate predicatele vor avea valoarea true

$$ALG = \sum_{1 \le i \le n} f\left(x_i\right) \cdot \begin{cases} 1, \ c_i \ge \frac{1}{3} \\ 0, \ altfel \end{cases} \le \sum_{1 \le i \le n} f\left(x_i\right) \cdot 3 \cdot c_i \le 3 \cdot \sum_{1 \le i \le n} f\left(x_i\right) \cdot c_i \le \ 3 \cdot OPT$$

=> algoritmul este 3-aproximativ