

Tema 1

Knapsack

```
# Ex 1 a)
def ex1_a():
    gMax = int(input("Introduceti gMax: ")) # gMax este K
    DP = [0] * (gMax + 1) # suma maxima <= cu i pe care o pot obtine cu
                          # obiectele de la 0 pana la interatia curenta
    obiecte = [int(x) for x in input().split()] # greutatile si valorile
                                                # obiectelor

    for i in range(len(obiecte)): # consideram obiectele de la 0 la i
        for j in range(gMax, 0,
                        -1): # parcurg descrescator greutatile pentru a
                            # putea actualiza dinamica pe un singur vector
            if j - obiecte[i] >= 0:
                DP[j] = max(DP[j], obiecte[i] + DP[j - obiecte[i]])

    print(DP[gMax])

# ex1_a()

# Ex 2 b)
def ex1_b():
    gMax = int(input("Introduceti gMax: ")) # gMax este K
    s = 0 # suma aproximativa
    nr = 0 # numarul curent
    while True:
        nr = int(input("Valoare/greutate obiect: "))
        if nr < 0: # asa semnalizam ca am terminat de citit
            break
        if nr <= gMax: # daca numarul citit este valid
            if s + nr <= gMax: # daca suma curenta + numarul curent nu
                              # depaseste gMax
                s += nr # adaugam la suma
            elif nr > s: # altfel luam numarul mai mare si apoi se va da
                        # break
                s = nr # pentru ca s + nr > gMax si nr > s
                        # atunci nr >= gMax / 2
        if s >= gMax / 2: # da break cand s depaseste gMax / 2 pentru ca
                        # OPT <= gMax => s >= OPT / 2
                        # => ALG 1/2 aproximativ
            break

    print(s)

ex1_b()
```

Load Balance

1. a) activități 20, 60, 60, 60

mașina 1: 60, 60

mașina 2: 60, 20

Obs! Pe acest exemplu OPT pune într-adevăr pe m_1 60, 60 și pe m_2 60, 20 $\Rightarrow OPT \cdot p = 120 \cdot 1,1 = 132 > 120 = ALG$
am găsit un exemplu pe care ALG e 1,1 aproximativ
 \Rightarrow este posibil (dar nu obligatoriu) ca ALG să fie 1,1 aprox.

b) Vom demonstra că nu se poate:

Passul 1: Pr. la absurd că OPT ar face o împărțire între cele două mașini cu diferență > 10 . În acest caz am putea să luăm o activitate cu timp cel mult 10 de pe mașina mai încărcată și să o punem pe cealaltă \Rightarrow am obține o împărțire între mai puțin decât OPT.

\Rightarrow CONTRADICȚIE!

dacă erau activități cu un timp de lucru ≤ 10 alg genera o dif ≤ 10 deci nu putea fi 1.1 aprox pt ca dif trebuia să fie minim 11 adică $1.1 \cdot 10$

3. Fie k indicele mașinii cu load maxim $\Rightarrow \text{ALG} = \text{Load}(k)$

Fie g ultima activitate adăugată pe mașina k . Fie $\text{Load}'(k)$, load-ul mașinii k fix înainte ca g să fie adăugat pe această mașină.

Dacă $g \leq m \Rightarrow g$ va fi pus pe o mașină goală $\Rightarrow \text{ALG} = \text{OPT}$. Altfel, $\text{ALG} = \text{Load}'(k) + t_g$ // din definiție

$$\leq \frac{1}{m} \left(\sum_{1 \leq i < g} t_i \right) + t_g \quad // \text{Cum}$$

// $\text{Load}'(k)$ este înainte de adăugarea lui g , și k este

// mașina cu încărcătura minimă, atunci $\text{Load}'(k) \leq \text{media}$

// mașinilor înainte de adăugarea lui g . Dacă ar fi fost mai mare

// atunci nu era mașina minimă $\Rightarrow \text{CONTRADICȚIE!}$

$$\leq \frac{1}{m} \left(\sum_{1 \leq i \leq n} t_i - t_g \right) + t_g$$

// Cum suma de mai sus nu îl conține pe t_g

// atunci este mai mică decât

$$// \text{suma tuturor activităților} = \frac{1}{m} \sum_{1 \leq i \leq n} t_i - \frac{1}{m} t_g + t_g$$

// fără t_g

$$= \frac{1}{m} \sum_{1 \leq i \leq n} t_i + t_g \left(1 - \frac{1}{m} \right)$$

$$\leq \frac{1}{m} \sum_{1 \leq i \leq n} t_i + \frac{t_m + t_{m+1}}{2} \left(1 - \frac{1}{m} \right)$$

// Cum suntem pe cazul $q > m$ atunci $t_q \leq \frac{t_m + t_{m+1}}{2}$
 // pt. că activitatea q se pune sigur pe o mașină
 // care nu este goală \Rightarrow se pune după activitatea m ,
 // care se pune pe ultima mașină goală, iar
 // activitățile sunt sortate în ordine desc.

$\leq OPT + \frac{1}{2} OPT(1 - \frac{1}{m})$ // Pentru primul
 // termen, OPT este sigur mai mare decât media
 // maximilor, pentru că este dat de mașina cu load-ul
 // cel mai mare, pt. al doilea termen $\frac{t_m + t_{m+1}}{2} \leq$
 // $\leq OPT$ pt. că există mai mult de $m+1$ activități

$$= OPT + \frac{1}{2} OPT - \frac{1}{2m} OPT$$

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

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

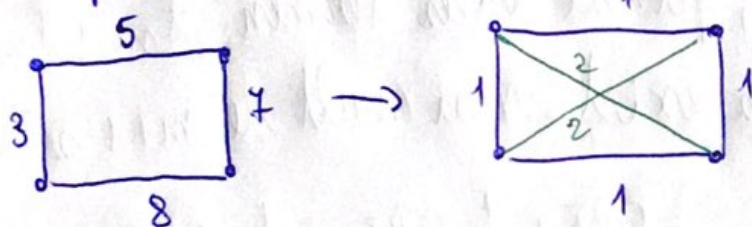
$$= \left(\frac{3}{2} - \frac{1}{2m} \right) OPT$$

TSP

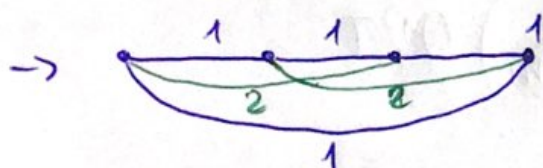
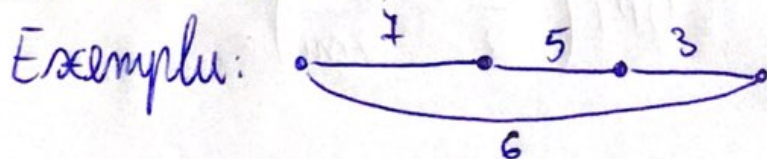
1) a) Pr. la absurd că \exists un algoritm polinomial ALG care
stie să găsească un ciclu hamiltonian de cost minim
într-un graf complet.

Fie G graful initial, construim G' graf complet
cu proprietatea că dacă în graful initial G , \exists muchie
indiferent de costul ei, în graful nostru G' vom
trasa muchie de cost 1, altfel de cost 2.

Exemplu:



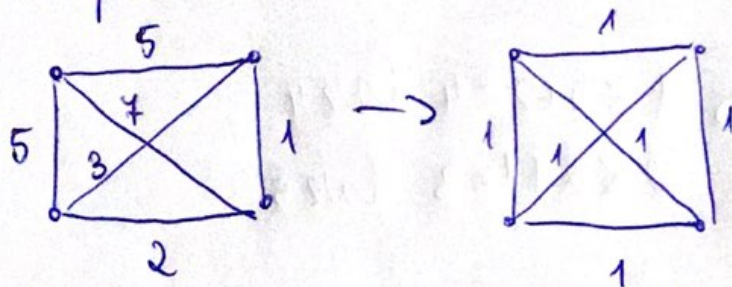
\Rightarrow Avem n noduri și muchii doar de costul 1 și
 $2 \Rightarrow ALG \geq n$ (ALG este costul minim al unui
ciclu hamiltonian)



Obs! Dacă $ALG = n$ înseamnă că s-au folosit doar
muchii de cost 1 \Rightarrow S-au folosit doar muchii care

\exists în G inițial $\Rightarrow G$ conține ciclu hamiltonian

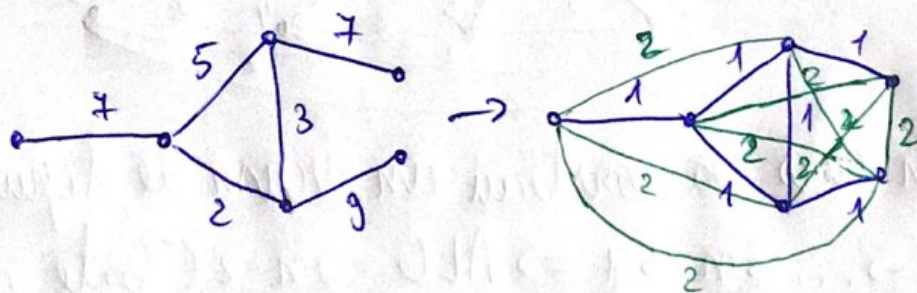
Exemplu:




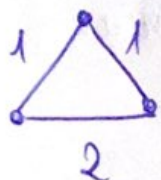
Analog dacă $ALG > n \Rightarrow G$ nu conține ciclu hamiltonian
 \Rightarrow Cum am presupus că \exists ALG polynomial \Rightarrow prin ALG se poate determina în timp polynomial dacă graful inițial G , conține un ciclu hamiltonian.

\Rightarrow CONTRADICȚIE pt. că știm că asta este o problemă NP-HARD

Exemplu:

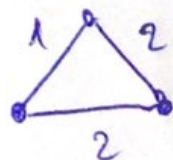


b)  $\Rightarrow 1 \leq 1+1=2$ Caz 1



$$\Rightarrow 1 \leq 1+2=3 \quad \text{Caz 2}$$

$$2 \leq 1+1=2 \quad \text{Caz 3}$$



$$\Rightarrow 1 \leq 2+2=4 \quad \text{Caz 4}$$

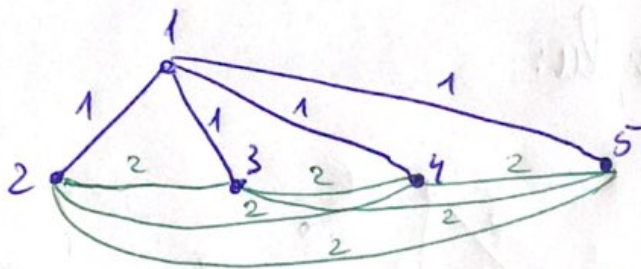
$$2 \leq 2+1=3 \quad \text{Caz 5}$$



$$\Rightarrow 2 \leq 2+2=4 \quad \text{Caz 6}$$

c) Construim grafel G astfel, un nod rădăcină conectat prin muchii de 1 la celelalte $n-1$ noduri frunză, apoi completăm cu muchii de 2 pînă G devine graf complet.

Exemplu:



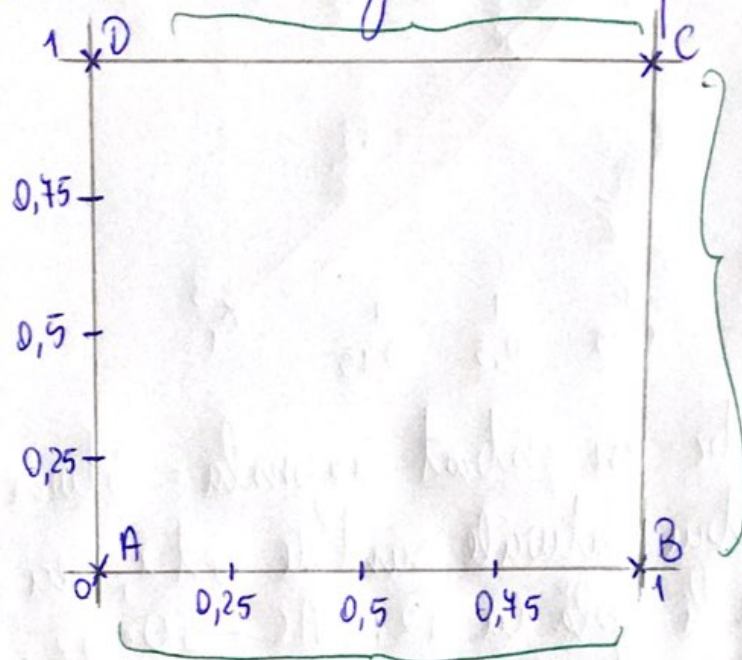
ALG din curs va construi un drum de tipul $1 \rightarrow 2 \rightarrow 3 \rightarrow \dots \rightarrow n \rightarrow 1 \Rightarrow ALG = 2n-2$ (Toate muchiile au costul 2 mai puțin muchia $1 \rightarrow 2$ și $n \rightarrow 1$ care au costul 1)

$$\Rightarrow 2n-2 > \frac{3}{2}n \quad \forall n \geq 5$$

\Rightarrow Pt. această instanță ALG nu este $\frac{3}{2}$ aproximativ

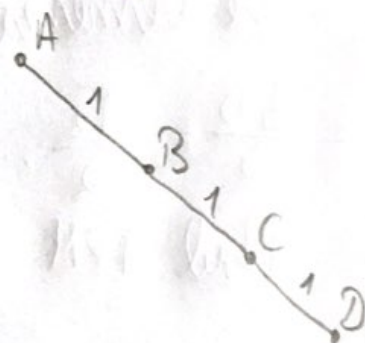
BONUS

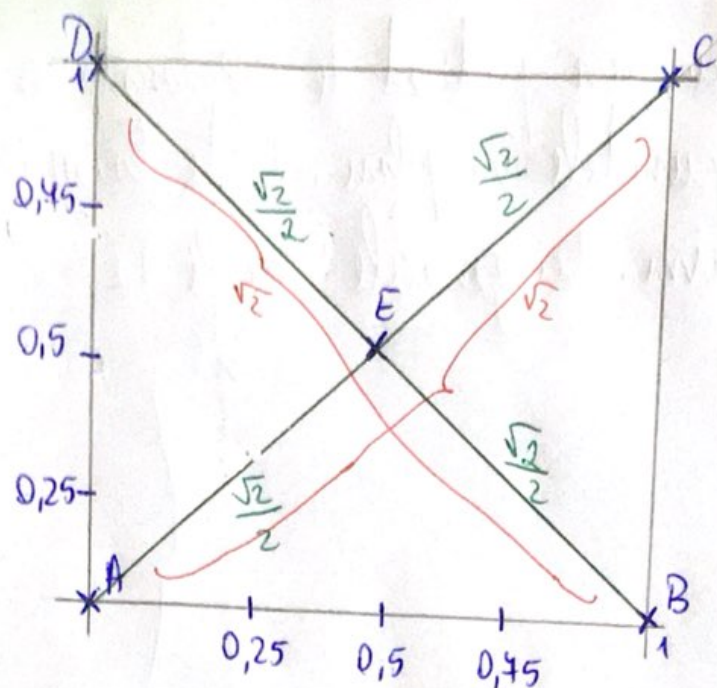
2) a) Vom construi un pătrat de coordonate $(0,0); (0,1); (1,0); (1,1)$ - punctele în plan. Toate laturile vor fi de cost minim. Diagonalele vor fi $\sqrt{2}$.



APM-ul pentru acest pătrat, are costul 3.

Acum vom adăuga un
noul punct în pătrat, E ,
de coordonate $(\frac{1}{2}, \frac{1}{2})$.

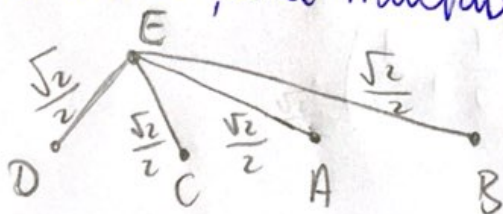




Știm că într-un pătrat diagonala = latura $\sqrt{2}$, în cazul nostru laturile sunt de cost 1, deci diagonala va fi egală cu $\sqrt{2}$. ($AC = BD = \sqrt{2}$, diagonale în pătratul nostru).

$$\text{De asemenea } AE = BE = CE = DE = \frac{\text{diagonala}}{2} = \frac{AC}{2} = \frac{BD}{2} = \frac{\sqrt{2}}{2}.$$

Noul APM va conține muchiile AE, EB, CE și ED .



Deci costul noului APM este $\kappa \cdot \frac{\sqrt{2}}{2} = 2\sqrt{2} \approx 2,8284$.
În concluzie, putem afirma că dacă adăugăm

punctul de coordonate $(\frac{1}{2}, \frac{1}{2})$ pentru pătratul nostru initial de coordonate $(0,0); (0,1); (1,0)$ și $(1,1)$, vom obține un APM de cost mai mic.

b) Vom folosi următorul algoritm (ALG):

- Găsim APM pentru $P \cup Q$.
- Salvăm nodurile din parcurgerea în preordine (RSD - rădăcină, stânga, dreapta)
- După aceea vom șterge toate nodurile din Q din parcurgere
- La final vom avea toate nodurile adiacente rămase din parcurgerea în preordine, obținând astfel un lant.

Știm din lema 2 din Curs 3, că algoritmul ALG creează un lant X cu $L(X) \leq 2 L(Y)$, unde $L(Z)$ este suma lungimilor muchiilor din graful Z .

Din modul în care funcționează o parcurgere în preordine, fiecare muchie din APM va fi luată în calcul de maxim 2 ori:

- * Când vom intra în subarborele acelei muchii
- * Când ieșim din subarbore.

Ștergerea unor noduri din parcurgere, nu va crește lungimea muchiilor parcurse (din regula triunghiului).

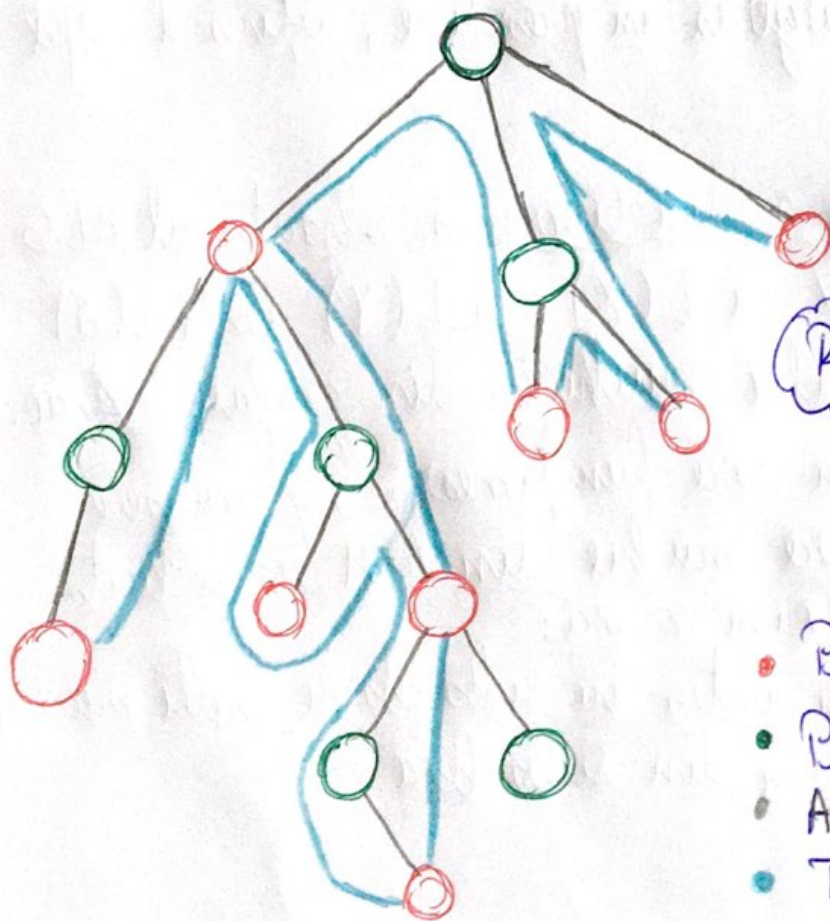
$\Rightarrow L(X) \leq L(E) = 2 \cdot L(APM)$, unde E este parcurgerea euloriană a APM-ului

$$\Rightarrow L(X) \leq 2 \cdot L(APM)$$

Algoritmul nostru ALG ne va feri un arbore de cost cel mult $2 \cdot L(APM)$.

În concluzie, costul oricărui APM al lui P va fi cel mult costul arborelui generat.

$$L(T) \leq L(ALG) \leq L(APM)$$



Reducerea unui APM

- Puncte din P
- Puncte din Q
- APM ($P \cup Q$)
- T

Vertex Cover

a) $C = \{C_1, C_2, \dots, C_m\}$ unde

$$C_1 = \{x_1, x_2, x_3\}$$

$$C_2 = \{x_1, x_4, x_5\}$$

\vdots

$$C_m = \{x_1, x_{n-1}, x_n\}$$

OPT = 1 ($x_1 = \text{true}$, restul de x_2, x_3, \dots, x_n sunt false), dar ALG poate alege pe worst case x_2, x_4, x_6 până la $x_{n-1} \Rightarrow \text{ALG} = m \Rightarrow \text{ALG}$ m aproximativ

b) Greedy-3CNF(C, X)

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 aleator $C_j \in C$

4: Pentru fiecare x_i din C_j marcăm x_i true.
(marcăm toate variabilele din C_j true)

5: Eliminăm din C toate predicatele ce
conțin cel puțin una din cele 3 variabile

6: return X

Eie OPT multimea variabilelor true din cardinalul minim pt. rezolvarea problemei. Eie C' multimea tuturor clauzelor selectate la pasul 3 de algoritm. Observăm că aceste clauze sunt disjuncte între ele pt. că altfel ar fi fost eliminate la o iteratie anterioară a algoritmului.

$$\Rightarrow \text{Card } C' = \frac{1}{3} \text{ Card } X \text{ (fiecare clauză conține 3 variabile disjuncte)}$$

Obs 2! Cum $C' \subseteq C$ din lemma 1 din curs \Rightarrow

$\Rightarrow \text{Card OPT} \geq \text{Card } C'$ // o variabilă din OPT ar
// putea să taie doar o clauză din C' pt. că este
// o submulțime de variabile disjuncte

$$\text{Card OPT} \geq \text{Card } C' = \frac{1}{3} \text{ Card } X$$

$$\Rightarrow \text{Card OPT} \geq \frac{1}{3} \text{ Card } X$$

$$\Rightarrow \text{Card } X \leq 3 \text{ Card OPT}$$

\Rightarrow Algoritm 3 aproximativ

c) Fie $X = \{x_1, x_2, \dots, x_n\}$ o multime de variabile booleene si C formula in forma 3CNF.

Restricții: 1) $0 \leq Y_i \leq 1 \quad \forall i \in \overline{1, n}, Y_i \in \mathbb{R}$

2) Pt un $C_i = \{x_\alpha, x_\beta, x_\gamma\}$ avem proprietatea ca $x_\alpha + x_\beta + x_\gamma \geq 1$

3) $1 \geq x_i \geq 0, x_i \in \mathbb{Z}$

Trebuie să minimizăm $\sum_{1 \leq i \leq n} x_i$

d) Fie $f: X \rightarrow \mathbb{R}^+, f(x_i) = Y_i$, o pondere, $0 \leq Y_i \leq 1$ număr real.

Dacă $Y_i \geq \frac{1}{3} \Rightarrow x_i = \text{true} \in S$ // S este multimea
// de variabile alese ca true
altfel $x_i = \text{false}$

$x_\alpha + x_\beta + x_\gamma \geq 1 \Rightarrow$ cel puțin unul dintre

$Y_\alpha, Y_\beta, Y_\gamma \geq \frac{1}{3}$

ALG este 3 aproximativ

Justificare:

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

$$= 3 \sum_{1 \leq i \leq n} f(x_i) \cdot Y_i \leq 3 \cdot \text{OPT}$$

Popescu Paulina Robertta Karlos
GRUPA 231