# Seminar 1 -grupa 233

**Problema rucsacului**

Fie doi vectori de dimensiune *n* , primul reprezentand greutatea unui obiect, al doilea reprezentand valoarea unui obiect, si un “rucsac” de capacitate totala W. Sa se determine care valoarea maxima a obiectelor ce pot fi incarcate in rucsac in limita capacitatii acestuia.

1. Varianta discrete

* Un obiect poate fi “taiat”, la nevoie

**Solutie:**Sortam obiectele crescator dupa raportul greutate/valoare. Cat timp nu am deposit capacitatea rucsacului, adaug, in ordine, obiectele in rucsac. Datorita obiect incarcat se poate depasi capacitatea totala a rucsacului, caz in care fractionam obiectul.

W=50 – capacitatea rucscului;  
Obiecte (value/weight): [20/5; 30/10; 100/50; 90/30; 75/40]

Sortez obiectele: [**20/5**; **90/30**; **30/10**; ***100/50***; 75/40]

Sol: 20+90+30+100\*5/50= **150**

Complexitate: este data de sortare: O(n log n)

Algoritm de tip greedy – in constructia solutiei optime aleg mereu elemental cel mai avantajos.   
Caracteristici: Timpi de lucru cel mult polinomiali. Dar, necesita justificare pentru corectitudine.   
  
**Corectitudinea algoritmului de mai sus:**  
Reducere la absurd + exchange argument  
  
Fie G solutia data de algoritmul greedy (adica obiectele selectate de catre algoritm)  
Fie O – solutia optima pentru problema. Fie elementele lui O sortate crescator dupa greutate/valoare (adica in acelasi stil ca G). Deoarece G si O sunt solutii diferite, cu siguranta va exista un *i* astfel incat G[i]=/=O[i].   
greutate(G[i])/valoare(G[i]) <= greutate(O[i])/valoare(O[i]) deoarece algoritmul greedy mereu selecteaza elementele cu raportul minim (de la mic la mare).  
2 cazuri:  
a) daca greutate(G[i])/valoare(G[i]) < greutate(O[i])/valoare(O[i]) – atunci inlocuiesc (eventual partial) obiectul O[i] cu obiectul G[i] si obtin o solutie mai buna decat O! Contradictie!  
b) daca greutate(G[i])/valoare(G[i]) = greutate(O[i])/valoare(O[i]) – pot inlocui ca in pasul de mai sus. Solutia nu se imbunatateste, dar dispare diferenta dintre G si O la pasul current.   
Referitor la pasul b) – daca alegem O sa fie o solutie optima care difera de G “cel mai tarziu”. In cazul acesta apare contradictie, deoarece diferenta dintre O si G dispare dupa exchange (eventual va fi mutata mai tarziu).

1. **Problema rucsacului varianta 1/0**

“1/0” – adica un obiect ori este adaugat in rucsac in intregime ori este lasat pe dinafara. Nu mai pot “taia” obiectele.

W=50;  
O (val/w): [60/10; 100/20; 120/30)  
Daca as aplica acelasi algoritm ca mai sus (dar cu restrictia ca nu pot taia ultimul obiect)  
As obtine valoarea totala 160:   
Solutia optima: 220!  
Evident nu mai merge aceiasi idee.

*Spoiler*: Algoritmul in forma aceasta nu ofera solutia optima, dar este un algoritm ½ aproximativ.   
  
Solutie pt problema in varianta 2?

Prima solutie ar fi una naiva: verificarea tuturor posibilitatilor si alegerea celei mai profitabile.

Altfel spus: la fiecare pas ‘*i’* merg recursive atat pe firul ca obiectul *i* este adaugat la solutie cat si pe firul ca acel obiect este omis de la solutie. Aleg pe cea mai buna dintre cele doua variante.  
Cod sursa C++: <https://onlinegdb.com/HkgC_RDbO>

Complexitate timp: O(2^n) – foarte rau! Deoarece la fiecare pas “thread-ul” se bifurca in doua, respective se “rasneste” redundant la aceiasi sub-problema de mai multe ori.   
   
  
Varianta a doua:  
Folosim un tablou bidimensional DP[][] cu semnificatia ca DP[i][j] reprezinta valoarea maxima ce o pot obtine folosing obiecte din intervalul [0,i-1] fara depasi greutatea totala j.   
**DP[n][W]** – va fi solutia pentru problema noastra.

Formula de calcul pentru DP:

DP[i][0]=DP[0][j]=0 pt orice i si j;   
pt I,j>0  
DP[i][j]=max(DP[i-1][j],DP[i-1][j-w[i-1]]+val[i-1]);

De implementat la laborator!

Complexitate?  
timp: O(nxW) – pseudo-polinomial.   
spatiu: O(nxW)

Nu toate problemele se pot rezolva pseudo-polinomial!  
Daca weight-urile nu ar fi fost numere intregi, map-area DP nu arm ai fi fost posibila! Deci nu am fi avut un algoritm pseudo-polinomial corect!  
  
spatiu: O(nxW) – se poate imbunatati la un O(2xW)