# Seminar 1 -grupa 232

**Problema Rucsacului:**Avem n obiecte, caracterizate prin valoare si greutate, si un rucsac de capacitate W.

1. Varianta continua a problemei:

Sa se selecteze un numar de obiecte astfel incat greutatea totala sa nu depaseasca W. Putem alege sa includem doar fractiuni de obiect.

Solutia:

* Sortam obiectele crescator dupa greutate/valoare. Iterez prin lista de obiecte ordonate; Cat timp mai exista loc in rucsac adaug obiectul current. Daca adaugarea obiectului face ca greutatea totala sa depaseasca W atunci pe ultimul obiect il “tai” astfel incat greutatea totala sa fie egala cu W.

W=50  
Obiecte (val/greutate): [30/10; 50/25; 100/40; 15/3; 25/5]

Sortez obiectele: [**25/5**; **15/3**; **30/10**; ***100/40***; 50/25]

zW=0

Sol: 25+15+30+100\*32/40= 25+15+30+80= **150**  
  
Complexitate: O(n log n) – data de sortare;

Corectitudine:  
*Reducere la absurd + “exchange Argument”*

Fie G – arrayul de obiecte selectate de catre algoritmul greedy iar O array-ul de obiecte selectate intr-o solutie optima. Presupunem prin absurd ca O=/=G. (Subliniem faptul ca obiectele pot fi fractionate)  
  
Ordonam obiectele din O dupa acelasi criteriu ca obiectele din G.   
evident va exista un indice *‘j’* astfel incat G[j]=/=O[j] -> prima diferenta intre O si G

greutate(G[j])/valoare(G[j]) <= greutate(O[j])/valoare(O[j])

cazul 1: “<” interschimb in O pe obiectul O[j] cu obiectul G[j] (sau o fractie din acesta) si obtinem ceva “mai optim” decat O. Contradictie!  
cazul 2: “=”. Pot face exchange-ul pentru a “tranforma” O in G.   
Sau, ca sa fie evidenta o contradictie. Alegem O – solutia optima care difera “cel mai tarziu” de G. Eu reusesc sa o fac pe O sa “semene” maim ult cu G facand un “exchange”, inseamna ca O nu era optima cea mai asemanatoare cu G. Contradictie!

2) Varianta 1/0 a problemei rucsacului: un obiect poate fi ori adaugat complet in rucsac ori lasat pe dinafara, dar nu poate fi fractionat.

W= 50

Obiectele: [60/10; 100/20; 120/30]  
Solutie Greedy: 160

Solutie Optima: 220

Spoiler alert: desi algoritmul de tip greedy nu mai imi furnizeaza solutia optima, el este inca un algoritm 0,5 aproximativ pt varianta 1/0 Knapsack Problem.

Solutia 1 pentru 1/0 Knapsack problem

* Merg pe idea ca pe oricare dintre obiecte il pot select sau nu si bifurc cele doua cazuri. (metoda recursiva)  
  Cod sursa C++: https://onlinegdb.com/HJVoGx\_b\_  
  Complexitate: O(2^n) Pt fiecare obiect se bifurca lantul de calcule. Repetam aceiasi sub-problema de mai multe ori.
* Solutia 2 pentru 1/0 Knapsack problem:  
  Programare dinamica!  
    
  Cand abordati o problema folosind PD idea centrala este “ce ar trebui sa stiu deja ca sa aflu solutia pentru problema mea?”

“care este base case pentru problema noastra?”

Problema: “Alegand din obiectele indexate de la 0 la n-1, sa se obtina o valoare maxima respectand restrictia de greutate totala W”

Problema generala: “Alegand din obiectele indexate de la 0 la i-1, sa se obtina o valoare maxima respectand restrictia de greutate totala j”

Pot reprezenta raspunsurile pt problema generala folosind un tablou bidimensional.  
DP[i][j] = profitul maxim obtinut prin alegerea din primele i obiecte fara sa depasesc greutatea totala j.

DP[n][W] – aici se va gasi solutia problemei noastre!

DP[i][0]=DP[0][j]=0 pt orice i, j;

Pt i&j>0: DP[i][j]=max(DP[i-1][j], val[i-1]+DP[i-1][j-w[i-1]])  
Complexitate timp O(nxW) – complexitatea este polinomiala dar nu depinde de lungimea intrarii. In acest caz se numeste **pseudopolinomiala**.

Complexitate spatiu? O(nxW) – Se poate mai bine! Tinem minted oar ultimele 2 linii ale matricei: cea completata la pasul anterior si cea pe care o completam in present. Aducem complexitatea spatiu la O(2xW)  
Observatie: Algoritmul pseudo-polinomial “merge” doar in cazul in care greutatile sunt numere intregi.