Tema Algoritmi Aproximativi

Knapsack:

* A)

*if* \_\_name\_\_ == '\_\_main\_\_':  
 unordered\_list = [2, 3, 4, 5, 1, 6, 8]  
 max\_weight = 8  
 weights = {0}  
 total = 0  
 *for* w *in* unordered\_list:  
 temp = weights.copy()  
 *while* temp:  
 val = temp.pop()  
 *if* val + w <= max\_weight:  
 weights.add(val + w)  
 total = max(total, val + w)  
 print(total)

Algoritmul este pseudo polinomial deoarece complexitatea sa este O(n\*m), unde:

* n = numarul de obiecte
* m = capacitatea rucsacului

m = capacitatea rucsacului deoarece in cel mai rau caz putem aveam toate numerele de la 0 la max\_weight in set-ul nostru

* B)

*if* \_\_name\_\_ == '\_\_main\_\_':  
 unordered\_list = [2, 3, 5, 1, 4]  
 max\_weight = 10 # var 1  
 total = 0 # var 2  
 *for* val *in* unordered\_list: # var 3  
 total += val  
 *if* total > max\_weight:  
 print(max(total - val, val))  
 *break*

Solutia este fezabila deoarece ‘total’ nu depaseste capacitatea rucsacului, iar orice obiect este capacitatea rucsacului.

Complexitate: O(n)

Algoritmul este ½ aproximativ.

Demonstratie:

Load Balance:

* 1)
* A)

Avand un set de activitati cu timp de lucru de maxim 100, luam ca exemplu o posibila desfasurare a algoritmului, anume punem 80 prima data pe o masina, si dupa pe cea de a doua 60, 60. In acest caz algoritmul nu este doar 1,1 aproximativ, ci este si cel optim deci afirmatia nu contrazice aproximarea.

* B)

Avand un set de activitati cu timp de lucru de maxim 10, scopul unui algoritm load balance fiind acela de a imparti activitatile pe cele doua masini cat mai efficient posibil si mereu alegand masina urmatoarea cea cu costul minim pana la acel moment, vom avea o diferenta de maxim 10 intre cele doua masini. Vom lua ‘worst case scenario’ pentru un input cu restrictiile noaste:

* 3)

Solutie: