Rafał Stachowiak conclusions

In this exercise I have developed three ways to solve Knapsack problem. Brute force, dynamic and greedy solution.

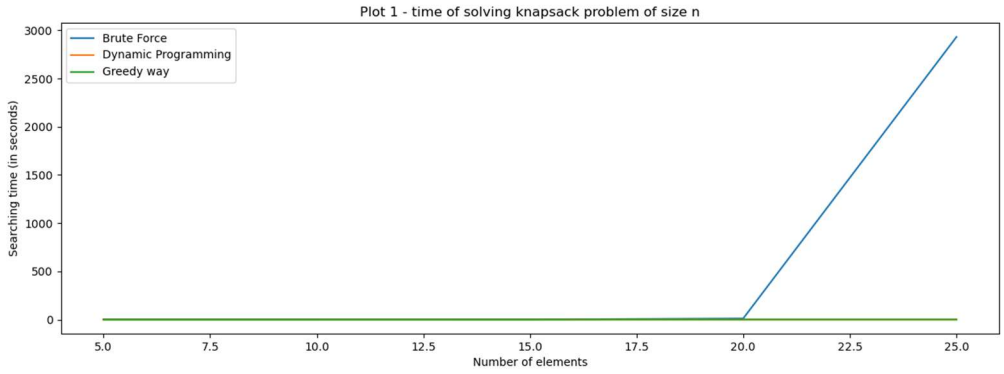
There is no doubts that the best and always optimal solution is generated by the Brute force approach. This comes with a price though. The complexity $O(2^n)$ is killing the program – this is exponential time algorithm.

The dynamic programming has complexity O(n*B) where B is capacity. This is pseudo-polynomial time algorithm. It is more faster but it has no guarantee to give the optimal solution.

The greedy way of solving knapsack problem is only an approximation of a optimal solution. In my program I am using the ratio between value and weight as a most significant factor to look for.

Time of solving greedy algorithm and dynamic programming approach is low in comparison to enormously high time of brute force algorithm.

Here is the plot; unfortunately after running it for 30 elements I've got a memory error so the limit here is 25.



To conclude
Brute force – always optimal solution
Dynamic programming solution – optimal or slightly worse (often the value is the same but the weight is slightly bigger)

Greedy solution – good approximation but it is hard to get always optimal solution

Knapsack problem:

Generally decision problem "Can a value of at least V be achieved without exceeding the weight W?" is NP-complete because there is no polynomial time algorithm to solve it.

The optimisation problem though is NP-Hard so its resolution is at least as difficult as the decision problem, and there is no known polynomial algorithm which can tell, given a solution, whether it is optimal

Dynamic programming is an example of pseudo-polynomial time algorithm

Approximation by greedy algorithm is a polynomial-time algorithm