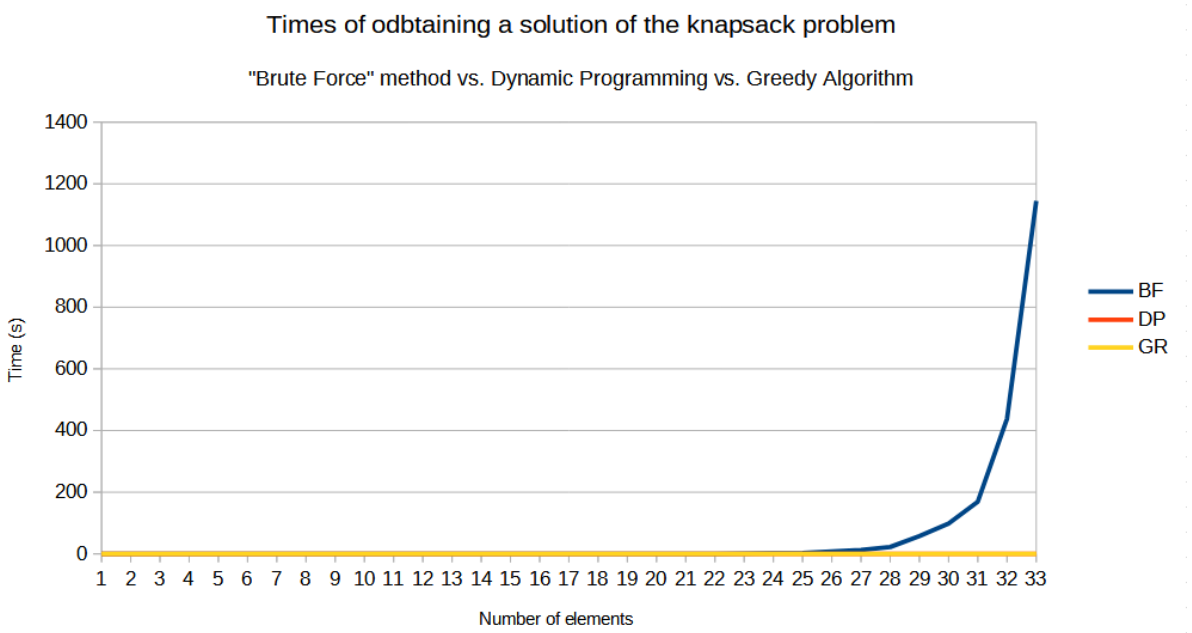
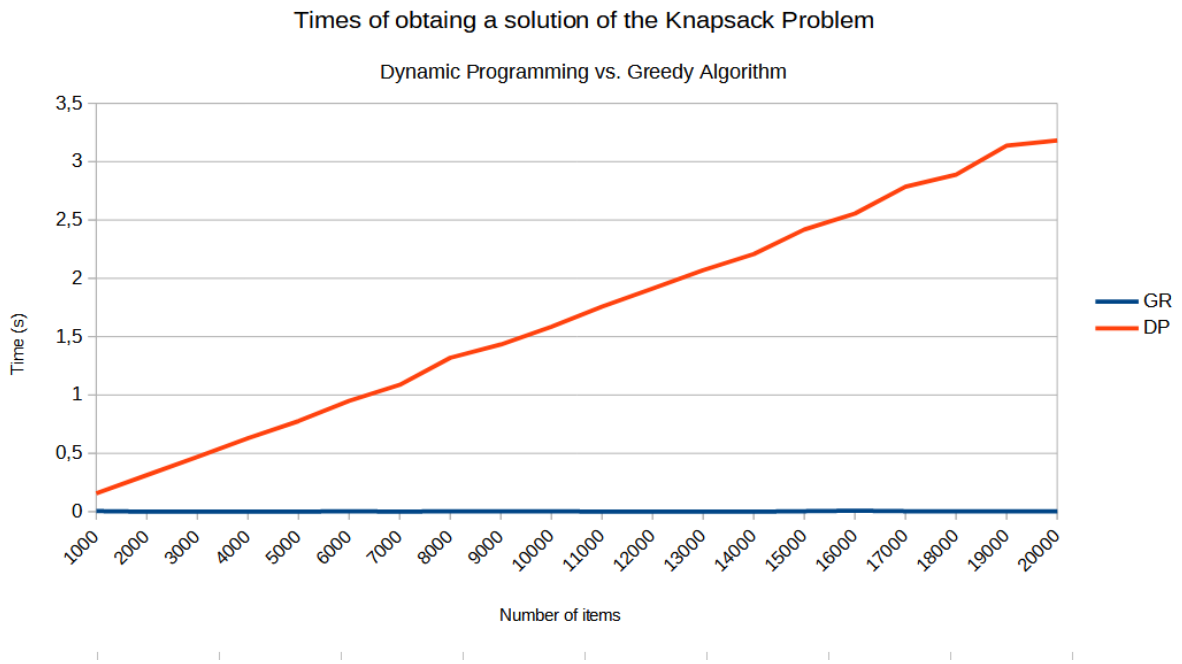
Bogna Kilanowska 148252

First, we compare three methods: Brute Force, Greedy and Dynamic. We prepared a file with 33 groups of items. The first group contains only 1 item, the second one 2 items, up to the 33rd group which contains 33 items. Each item has a random weight from 1 to 10 and a random value from 1 to 100. For the capacity equals 20, we measured the time of obtaining the solution.

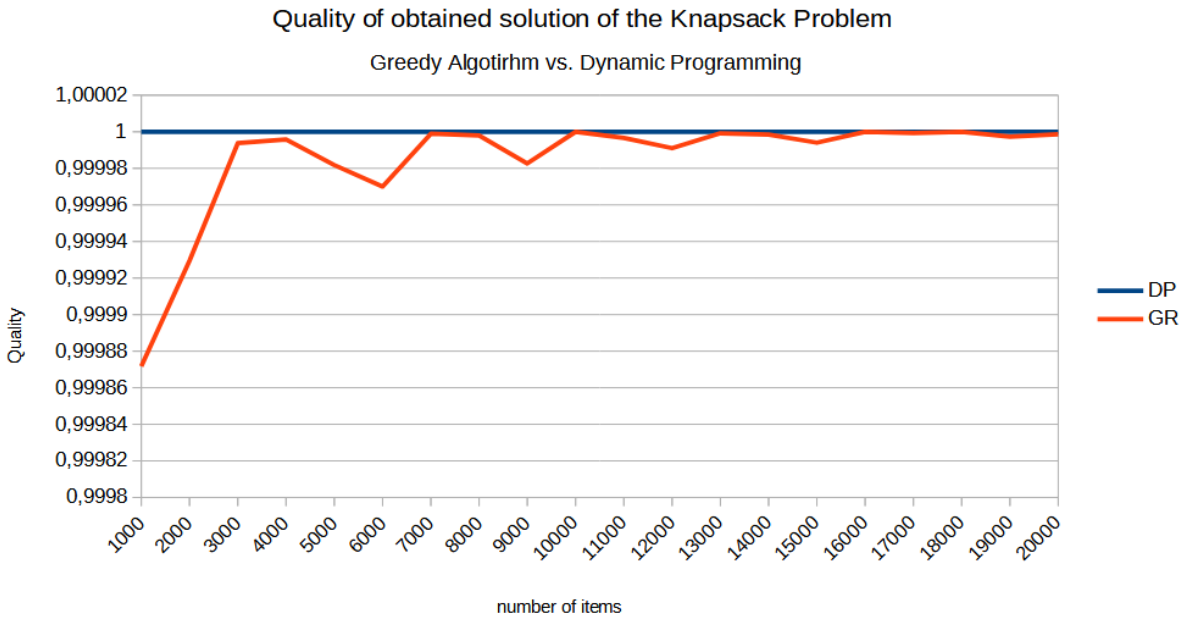


As we can see, the Brute Force algorithm has a very high time complexity, which equals O(2^n).
In the Brute Force method, we try each combination of items we want to pack to the knapsack, therefore we must check 2^n combinations.
The Brute Force method is the exhaustive method of searching for the solution. The biggest advantage of such a method is the guarantee of finding the optimal solution.
The biggest disadvantage of this algorithm is its time complexity. We cannot run the program for bigger instances, because the time of waiting for a solution is too long.

To see how Dynamic Programming and Greedy algorithms behave, we had to run a program for bigger data. We prepared a file with 20 groups of items. The first group contains 1000 items, the second one 2000 items, up to the 20th group which contains 20000 items. Each item has a random weight from 1 to 50 and a random value from 1 to 1000. For the capacity equals 1000, we measured the time of obtaining the solution.

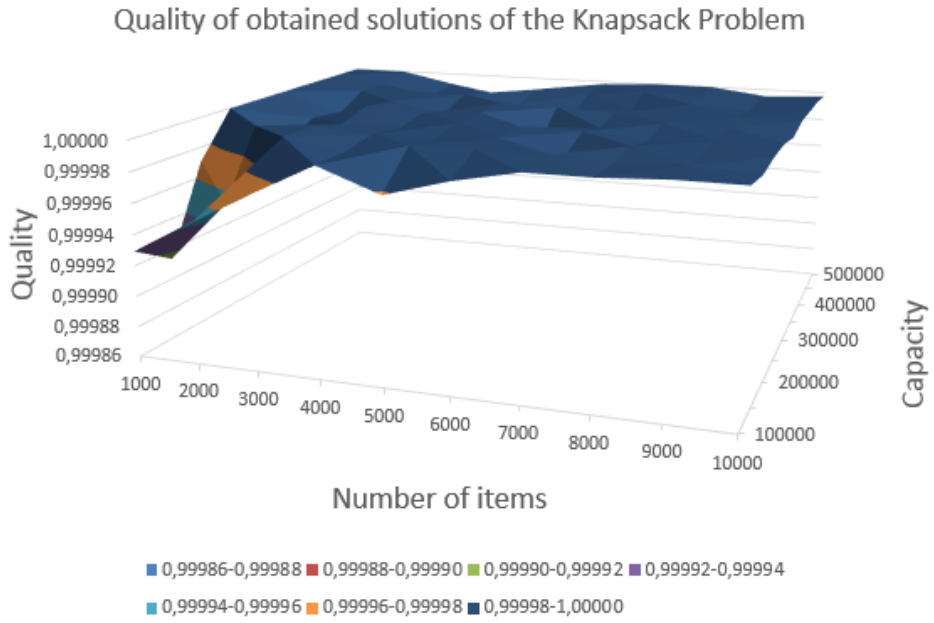


As we can see, the Dynamic Programming method has a higher time complexity than the Greedy method. But in fact, we cannot compare the Greedy algorithm to other methods of solving the Knapsack Problem, why? Because of the quality of solutions obtained by this method.

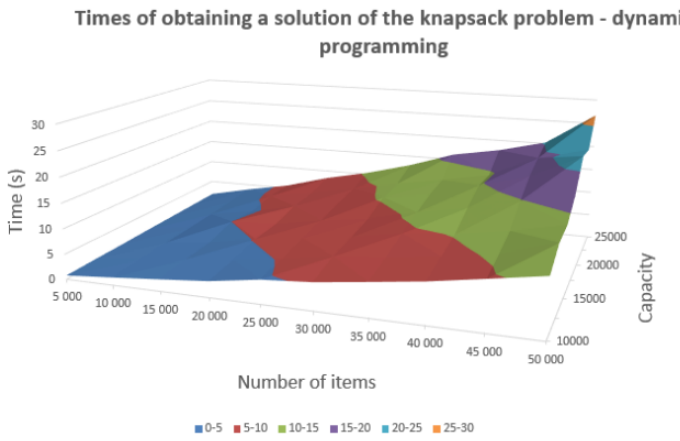


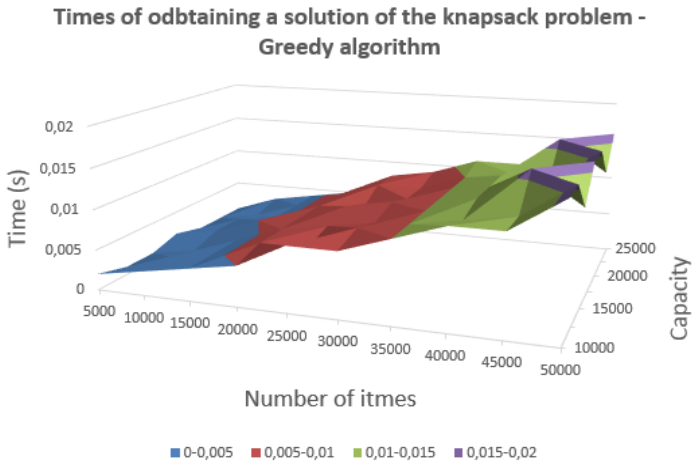
From the line chart we can read that while the Dynamic Programming method always gives the optimal solutions, the Greedy algorithm often gives solutions that are only close to the optimal ones. The quality of solutions grows with the growth of the number of items.

We also prepared for a different set of data the 3-dimensional chart which shows that the quality of solutions obtained is lower when the number of items and the capacity of the Knapsack is lower.



The last 2 charts were prepared for the number of items growing from 5000 up to 50000, each with the weight <1;50) and the value <1;100), for capacities 10000, 12500, …, 25000.





The chart dedicated to the Dynamic Programming method shows that the time complexity of this algorithm depends both on the number of items (n) and capacity of the knapsack (c) and equals O(n*c).

The time complexity of the Greedy Algorithm depends mostly on the number of items. It is caused by the fact that in this method in the first step all items are sorted by their ratio of value to the weight. We decided to use the Heap Sort method of sorting all the elements because it has fixed time complexity equals O(n*logn).

The complexity of the second step, in which the algorithm picks items with the best ratio, equals O(n). O(n) < O(n*logn), therefore the complexity of the Greedy method for searching the solution of the Knapsack Problem is O(n*logn).

The Knapsack Problem belongs to the NP-complete class.

To make a conclusion, we made a simple table:

Method	Brute Force	Greedy	Dynamic Programming
Optimal solution	Always	Not always	Always
Time complexity	O(2^n)	O(n*logn)	O(n*c)

When it is necessary to find the optimal solution, the Dynamic Programming method seems to be the best one. When the short time of obtaining the solution is more crucial than finding the best solution, the Greedy method is more appropriate.