In this lab you will experiment with three algorithm design techniques applied to the solution of variations of the knapsack problem discussed in class.

1. Implement a backtracking algorithm that solves the optimization 0-1 knapsack problem. Instead of deciding whether we can take items worth a predefined amount of money, as described in class, in this version of the problem you need to find the highest-value load that can fit in the knapsack.

2. Assume now that the thief has at his disposal an unlimited number of items of each type. Implement a dynamic programming algorithm, based on the minimum coin method to allow the thief to find the highest-value load that can fit in the knapsack.

3. Implement a greedy algorithm that solves the optimization continuous knapsack problem. This problem is identical the 0-1 knapsack, except that in this case we can take fractions of items. For example, if we take $\frac{3}{4}$ of an item that has value 2 and weight 3, the value of the fraction would be $\frac{3}{2}$ and its weight would be $\frac{9}{4}$.

Compare the running times of the algorithms for various parameter values and, as usual, write a report describing your results. Given the little time available, a demo will not be required, thus it is very important that your report accurately describes your work.