Algorithms 2500 – Section 1A
First Project
Deadlines: 10/27 First report, 11/3 Second report

Greedy, Dynamic Programming and brute force for solving the Knapsack problem

Maximum points 100

General information

This project investigates and compares the performance of greedy, dynamic programming and brute force algorithms in solving different instances of the knapsack problem.

Maximum group size 3

Any programming language can be used for the implementation.

The knapsack problem

Consider a set of \( n \) elements \( A = \{ a_1, \ldots, a_n \} \), where each element \( a_i \) has value \( v_i > 0 \) and weight \( w_i > 0 \), and a knapsack with capacity \( W \). We need to select the set \( S^* \subseteq A \) that maximizes the value of the selected elements and incurs a weight less than \( W \), formally:

\[
S^* = \arg\max_{S \subseteq A} \sum_{a_i \in S} v_i \\
\text{s.t.} \quad \sum_{a_i \in S} w_i \leq W
\]

Implementation and evaluation

Three solutions for the knapsack problem should be implemented. The first solution should be based on the greedy approach, the second on Dynamic Programming and the last solution on brute force.

The brute force approach is a baseline approach which searches all possible subsets with a weight less than \( W \) and picks the one which provides maximum value.

The approaches should be tested on various inputs, with different size and heterogeneity. More in particular, an input size should be generated from a triple \( (n, \mu, \sigma^2) \) where:

- \( n \) is the number of elements in \( A \)
- \( \mu \) and \( \sigma^2 \) are the average and variance of a Gaussian distribution \( \mathcal{N}(\mu, \sigma^2) \), respectively.
- For each of the \( n \) elements, a random value and weight should be generated using the Gaussian distribution.
In order to generate random values from a Gaussian distribution existing packages and classes can be used. As an example for C++: http://www.cplusplus.com/reference/random/normal_distribution/

The evaluation should be carried out in terms of:

- Execution time
- Value of the solution

The above performance metrics should be investigated with respect to the size of the input and the variance of the distribution. As a result, at least four graphs should be generated to compare the two approaches:

- The trend of the execution time by increasing the size of the input (using fixed arbitrary $\mu$, $\sigma^2$)
- The trend of the value of the solution by increasing the size of the input (using fixed arbitrary $\mu$, $\sigma^2$)
- The trend of the execution time by increasing the variance $\sigma^2$ (using fixed arbitrary $\mu$ and $n$)
- The trend of the value of the solution by increasing the variance $\sigma^2$ (using fixed arbitrary $\mu$ and $n$)

The values mentioned as “arbitrary” should be selected in order to generate meaningful results and will be part of the evaluation.

Each point in the graph should be the average of the result for at least 10 inputs. For example, in order to measure the execution time for an input of size $n = 50$, it is necessary to generate at least 10 different inputs (all with the same fixed arbitrary $\mu$, $\sigma^2$). Then the execution time should be evaluated on each input, and the corresponding point in the graph will be the average of these values. An example of the expected graphs is shown in the following. Note that the values for the inputs size are arbitrary and meaningful values should be tested in the project (e.g. values for which the algorithms show a observable difference).
Reports

Two reports should be prepared. The first report does not require implementation and should include at least the following sections:

- Abstract: Summary of the following sections
- Introduction and motivation: Overview of the knapsack problem, formulation, history, relevance, complexity, relation to other problems, etc.
- Proposed solutions: Pseudo-code, description and complexity analysis of the proposed greedy, dynamic programming and brute force solution to the problem.
- Plan of experiments: Description of the methodology which is planned to use in the experiments, significance and expected results.

The second report requires implementation and describes the implementation and results. It should include at least the following sections:

- Abstract: Summary of the following sections and results
- Implementation: Description of the methodology used for implementing the proposed solutions (e.g. relevant classes and data structures, structure of the program, etc.).
- Experiments: Description of the experiments performed and obtained results. Results should be represented in a graphical form as well as described in the writing.
- Conclusions: Summary of the work and final considerations.
- Code: The code you have used, well commented and organized.