Departamento de Informática, PUC-Rio
Prof. Thibaut Vidal

PROJETO E ANÁLISE DE ALGORITMOS (INF 2926)

Implementation Work – Deadline 12/12/2016

1 General Description

This work requires to develop some algorithms for the knapsack problem and some of its variants, analyze their complexity, their possible optimality, and compare with the measured computational time on some benchmark instances.

The work should be returned by e-mail to vidalt@inf.puc-rio.br before the deadline, with the title "INF2926 – project – your matricula(s)" and should contain the following elements:

1. The source code of the algorithm, along with a document which provides all necessary instructions for compilation and execution;
2. A ZIP file containing the solutions produced by the algorithm in the specified format;
3. A report **of maximum 6 pages**, which answers to the questions and analyses listed below.

The following restrictions are imposed for the project:

- The algorithm should be developed in ANSI C or C++, Java or Python.
- The project can be done as a single student, or by group of two.
- The better the complexity, the better the grade.
- Any copy of the source code or report of another group will automatically lead to a grade of 0.
- A late delivery of the work (after 23:59:59 on the 12/12/2016) will automatically lead to a grade of 0.

Evaluation: The project will count for **a total of 3 points**, which are part of the second evaluation of INF 2926.

Benchmark Instances and Solution Format: To allow the computational evaluation of the algorithm, a set of benchmark instances for the problem, as well as specifications for the INPUT and OUTPUT format will be made available at the address: `http://www-di.inf.puc-rio.br/~vidalt/courses/2016.2-INF2926/2016.2-INF2926.html`.

2 Questions

1. (1pt) **On the fractional knapsack problem.** We are given a list of items $i \in \{1, \dots, n\}$, each one characterized by a non-negative weight w_i and value p_i. We also consider a knapsack of capacity B. The goal of the fractional knapsack problem is to select a subset of items, in order to maximize their total value, and such that their total weight does not exceed the capacity B. Only in this question, we allow to use a fraction (e.g., 10%) of an item i.

   - Describe the pseudo-code of a greedy algorithm for this problem, and prove its optimality.
   - Present the data structures used for its implementation and analyze the computational complexity of the algorithm.
   - Implement the algorithm, and verify if its theoretical complexity is attained, in practice, on the benchmark instances provided with the project.

2. (1pt) **Moving on towards integer optimization.** We consider the same problem as previously, but we are not allowed anymore to take a fraction of an item.

   - Does the previous greedy algorithm still return optimal solutions in all cases. If Yes, prove it. If No, provide a counter-example.
   - If the greedy algorithm does not guarantee optimality, provide an alternative **optimal** algorithm and analyze its computational complexity.
   - Implement the proposed algorithm, and verify if its theoretical complexity is attained, in practice, on the benchmark instances provided with the project.

3. (1pt) **Adding some spice to the problem.** The company who hired your services forgot to mention that some of these items contain chemical, toxic or corrosive products.

   To catch up with this little omission, you are now given additional information on possible incompatibilities of products within the knapsack, in the form of graph of conflicts $G = (V, E)$. This graph contains an edge (i, j) in the case where the two products i and j are forbidden to be carried together in the knapsack. Your goal is still to maximize the total value of the items contained in the knapsack, but without violating any conflict constraint.

   - Propose a greedy heuristic for this problem. This algorithm does not need to systematically return an optimal solution.

- Describe the data structures used for its implementation and analyze the computational complexity of the algorithm.
- Implement the algorithm, and verify if its theoretical complexity is attained, in practice, on the benchmark instances provided with the project.

(Bonus 1pt:) Report the value of the solutions obtained by the algorithm in a table. The three groups with the best solutions for this problem will be granted an extra bonus point.