Algorithm Engineering Challenge

Sebastian Österlund

November 15, 2016

The deadline for the project is December 16th 2016 23:59 CEST

1 Introduction

In this challenge you will take on a hard problem. This project is meant to be done in a group of **three** persons, and will count for 30% of your grade. You will have about 4 weeks to complete this project. The requirements for the deliverable are stated in Section 3.

You can work in groups of up to three participants. Each group has to hand in *a report* and *code*.

- Find a group. Working on your own is also perfectly fine.
- Do a literature study, that is, read everything you can find about the problem and summarize the results in your report, for example, in a section on previous work.
- Develop an own algorithm for your problem. Implement it, experiment with the benchmark instances and aim at making it as performant as possible. Performance is with respect to solution quality or running time or a combination of the two. Describe the intuition behind your algorithm and motivate your design decisions and engineering steps in your report.

2 Problem Description

The VU Alchemy Lab is doing research into creating gold from other materials. For their experiments they need a large number of different compounds.

In their search for gold they have been able to create a number of machines that magically convert one compound into another. Since the results of their research has not been that successful yet, the funding is running out.

Buying the materials required for their research is quite expensive. Luckily they have a stockpile of a bunch of materials, and can get some of the materials quite affordably (i.e. the price is negligible) depending on the demand for that material. Since they have created machines that convert one compound into another, they should be able to synthesise any expensive material.

To be able to continue their research, the researchers need to be able to synthesise any compound given any other compound; they simply have to build the machines that can synthesise that compound.

Building such a machine is not free. Every machine has a cost associated with it. However, when such a machine has been built, it can be used indefinitely. Your task is to build a set of machines such that every compound can be synthesised from any given compound, while aiming for the lowest possible cost of building the machines.

2.1 Input specifications

The input file will contain a catalogue of available machines. Each line of the file represents a particular machine model. The format of each line is:

```
<machine name>  <orig compound>  <new compound>  <price>
```

Machine names start with the character 'M' (without the quotes) and then a non-zero padded integer. Examples of machine names are 'M1', 'M2', 'M13', etc.

Compound names start with the character 'C' (without the quotes) and then a non-zero padded integer. Examples of compound names are 'C4', 'C6', 'C24', etc.

Prices are simply non-zero padded integers with no commas or periods or unit designations (assume all prices are in EUR). Examples of prices are '987334', '13948295', etc.

Each field in the input file is white space separated using spaces or tab characters. There may or may not be additional white space after the last entry.
Example input file:

```
M1      C1      C2      277317
M2      C2      C1      26247
M3      C1      C3      478726
M4      C3      C1      930382
M5      C2      C3      370287
M6      C3      C2      112344
```

2.2 Output specifications

Your output must be exactly the following. Your program must first output the sum price of all the machines to be purchased (do not insert commas, periods, or any units) as a simple integer, followed by a new line.

Your program should then output the numbers of the machines purchased in ascending order, but omitting the 'M' designations. Each machine number must be separated by a single space character. There should be no additional space character after the last machine. Instead, the program should output a new line after the last machine number.
Example output (there should be a newline at the end of each line):

```
617317
2 3 6
```

3 Requirements

3.1 Code

- Hand in one **zip** containing all your code as well as your documentation.
- A script in the top-level that prepares (`prepare.sh`) everything on Ubuntu 16.04. You are allowed to install packages using *apt*. The script will be run with root privileges.
- Script in top-level that runs (`run.sh <input file>`) your program with the specified input file.

We have provided an example skeleton for Java and for C, found at `https://github.com/VU-Programming`. You are free to use any programming languages and/or libraries you like.

3.2 Project Report

We expect you to submit a project report (about 4 pages) discussing your solution in detail. Discuss in detail your algorithm as well as the implementation details. You should also discuss any obstacles you encountered during implementation.

A typical report could be structured like this

- Introduction (What is the problem? Why is it important?)
- Previous work (summary of literature)
- Own contribution (algorithm, design, intuition, engineering, relation to previous work, implementation issues)
- Computational results (show some benchmarks for different inputs)
- Discussion (what have you done, any interesting observations you made, what would be cool to do now with the knowledge you gained in this project)

4 Further info

The grade will be determined by

- the quality of the report
- the quality of the code
- the originality of the algorithm engineering ideas
- the performance of the code on the benchmark instances
- the ability to follow the rules and to meet the deadline. We set 0.5 grade point penalty per day after the deadline.

These criteria will be evaluated partly in an **interview** with **all** members of the group. Different grades within a group are possible in case of unbalanced work division.

4.1 How to submit

Info on how to submit your solution will be posted later.

4.2 Challenge scoreboard

To create a more competitive environment, we have set up a scoreboard. By submitting your solution, a score will be calculated based on the performance of your program. In this way, you can see how your solution stacks up against other group's algorithms. The group that has the best score will win a prize!