Professor Dr. Jens Vygen
Rudolf Scheifele

Programming Exercise 1

Exercise P.1. *Task:* Implement Edmonds' Cardinality Matching Algorithm as described in the lecture.

Usage: Your program should be called as follows:

```
program_name graph.dmx
```

Input: The argument `graph.dmx` is mandatory (i.e. your program should exit with an error message if it is not present), and it encodes the graph for which your program should find a maximum cardinality matching.

The file `graph.dmx` is given in DIMACS format, which is used to encode undirected graphs as follows: All lines beginning with a `c` are comments. Now, ignoring any comment-lines, to encode a graph G, the first line has the format

`p edge` n m

where $n = |V(G)|$ and $m = |E(G)|$. From this, $V(G)$ is implicitly identified with $\{1, \dots, n\}$. Note that vertex indices start with 1 in the DIMACS format. The following m lines have the format

`e` i j

representing that $\{i, j\} \in E(G)$. You can assume that n and m fit into an `integer` on the machine to be used for evaluation. Code for reading the input will be provided (see "Help" section below).

Output: Your program should return a maximum cardinality matching M in G by writing the complete DIMACS encoding of the subgraph $(V(G), M)$ to standard output.

Programming language: Your program must be written in C or C++, although the use of C++ is strongly encouraged. Program compilation and evaluation will be performed on Linux. Your program will be compiled with gcc-4.8.5 or gcc-7.3.0 using C++11 with the options `-pedantic -Wall -Wextra -Werror` enabled, i.e. all warnings are enabled and each remaining warning will lead to compilation failure. The default gcc version to be used is gcc-7.3.0, but you may also explicitly request gcc-4.8.5 to be used for compilation. Your submission must contain a bash script `compile.sh` which compiles your program (which for example may just call gcc or make directly using the mandatory compile options listed above). The standard library can be used as you wish, but no other libraries.

Algorithm evaluation: The algorithm is to be implemented as described in the lecture. The running time should be $\mathcal{O}(n^3)$.

Code evaluation: Your code must implement EDMONDS' CARDINALITY MATCHING ALGORITHM correctly. Running time in practice will also be evaluated, as well as the elegance, cleanness and organization of your code. Make sure to add good documentation and give the variables, functions and types meaningful names that make their role clear. Break your complicated functions into small simple ones, break your program into a few units etc. Of course, your program should not trigger undefined behavior. In particular, your program should be `valgrind`-clean, i.e. it should not leak memory and should not perform invalid operations on memory.

Help: On the website for the exercise classes you will find a C++ unit providing a simple graph class that you can use (if you wish). Code for reading the input files is also provided by means of a function that constructs the given graph class from an input file name. The website also contains a set of test instances for testing your code.

(25 points)

Deadline: Sunday, November 18, 23:59, via email to scheifele@or.uni-bonn.de. The websites for the lecture with all exercises and test instances can be found at:

`http://www.or.uni-bonn.de/lectures/ws18/coex.html`