Prof. Dr. Knut Reinert,
Prof. Dr. Alexander Bockmayr,
René Rahn

October 15, 2014

Algorithms - Programming Exercices

WS 2014/15

Exercise 1
Due date: 29.10.2014 until 1 pm.

Shortest paths

1. Implement *Dijkstra's algorithm* on a graph $G = (V, E)$ using a minimum priority queue. You can find the pseudocode of the algorithm and some additional explanation in the svn `https://svn.imp.fu-berlin.de/agbio/algorithms/WS14/materials/exercise1/`. You can use the `std::priority_queue` provided by the STL (`http://www.cplusplus.com/reference/queue/priority_queue/`). Additional test data is accessible from the svn (`https://svn.imp.fu-berlin.de/agbio/algorithms/WS14/data/`).
2. The program shall be submitted as a single source file with the name `exercise1.cpp` in your group folder of the svn. (Note: make sure all group members have read/write access to that folder.)
3. Together with your program you have to provide a short application note (1-2 DIN-A4 pages; pdf) describing the implementation and an evaluation of the runtimes.
4. The due date is Wednesday 29th of October 2014 until 1 pm at the latest. All versions submitted later than this time stamp won't be assest.
5. The program shall run on a linux pool machine (see the wiki for additional information `http://www.mi.fu-berlin.de/w/ABI/ProgrammingExercisesWS14`).
6. The compiler flags shall be set to `-pedantic -Wall -ansi -O3`.

Requirements of your program:

- Input:
    1. The graph $G = (V, E)$ in form of the *Trivial Graph Format* (TGF) (see below for more information).
    2. The id of the source node s (integer).

3. The id of the target node t (integer).

- Your program shall be called as follows:
  `./exercise1 graph.tgf sourceId targetId`
- The program shall output the length of the shortest path in the first line followed by the identifiers of the nodes from s to t in the second line and all separated by a single whitespace, e.g.,
  ```
  host$ ./exercise1 test.tgf 0 5
  678.9
  0 1 2 3 4 5
  ```
- If no valid path exists the program shall output the single information `no path`, e.g.,
  ```
  host$ ./exercise1 test.tgf 5 0
  no path
  ```

As input format we use the Trivial Graph Format (TGF) which is defined as follows:

```
0 someString0
1 someString1
2 someString2
3 someString3
#
0 1 weight
1 2 weight
1 3 weight
3 2 weight
```

First all node identifier (int) and associated node names (arbitrary strings - we limit to single words) are listed. After a single line containing only a single # symbol all directed edges are listed with their weights (double).

IMPORTANT: The generated output must be in the form as described above. Don't use any other formatting and don't print any debug information in the final version of your submitted program!
You can score 4 pts.. 3 pts. for the program and 1 pt. for the application note.
3 pts. = The program compiles and runs successfully with no errors.
2 pts. = The program compiles and contains minor errors.
1 pt. = The program compiles and contains some critical errors.
0 pts. = The program doesn't compile or does not meet the requirements.