CS 584/684 Spring 2017 Homework 7 – due noon, Wednesday, May 24 2017

Your solutions to problems 1-3 should be type-set in LaTeX and submitted in both `.tex` and `.pdf` form, with file names `hw7.tex` and `hw7.pdf`. These two files, plus any additional source files invoked from your `.tex` file (such as pictures), should be bundled together into a single `.zip` file named *your-last-name*`-tex-hw7.zip`. Your code for problem 4 should be submitted in a single, separate `.zip` file named *your-last-name*`-code-hw7.zip` with contents as described below in the description of problem 4.

Submit by emailing to `hamialex@pdx.edu` including the zip file as a separate attachment and including "`CS584 HW7`" in the subject line.

All algorithms must be accompanied by proofs of correctness and of running time.

1. Do CLRS exercise 24.3-4.

2. As is well-known, America's highway infrastructure is crumbling. Yet travel must continue. Suppose you are given a map of U.S. cities and roads connecting them that shows, for every road segment, the probability of traveling down that segment *safely*, i.e. without destroying your axle in a pothole, falling into a river due to a broken bridge, etc. Design and analyze an algorithm to determine the safest route from Portland to your preferred summer vacation spot.

More formally, suppose you are given a directed graph $G = (V, E)$, where every edge e has an independent safety probability $p(e)$. The safety of a path is the product of the safety probabilities of its edges. Design and analyze an algorithm to determine the safest path from a given start vertex s to a given target vertex t.

3. Do CLRS exercises 25.3-4 and 25.3-5.

4. Implement Borůvka's MST algorithm, based on the description in the lecture notes. Your implementation should run in $O(|E| \lg |V|)$ time. Note: Remember to use adjacency lists.

Your program should take one command line argument, which is the name of an input file. The format of that file will be as follows:

- First line contains a number C of test cases in the file, where $0 \leq C \leq 10^6$.
- Then come C test cases, each describing a weighted connected graph, and consisting of:
  - A single line containing the number of vertices V, where $2 \leq V \leq 10^3$.
  - A single line containing the number of edges E, where $1 \leq E \leq 10^6$.
  - For each edge $i : 1 \leq i \leq E$, a single line describing edge e_i by three space-separated numbers $v_1\ v_2\ w$, where v_1 and v_2 are the vertex numbers of the edge's endpoints, with $1 \leq v_1 < v_2 \leq V$, and w is an integer edge weight, with $0 \leq w \leq 10^6$.

  The described graph can be assumed to be connected, i.e. forall $v : 1 \leq v \leq V$ there is at least one edge whose endpoint is v. Furthermore, you may assume that all weights are globally distinct.

Your program should output (to `stdout`) one line for each test case, of the form "`Case i:` es" where es is a space-separated list of edge numbers for the edges forming the MST, in increasing order by edge number.

Example Input:

```
2
3
3
1 2 100
1 3 200
2 3 300
7
12
1 2 4
2 3 26
2 4 14
1 4 12
3 4 30
3 6 16
4 6 3
1 5 18
4 5 2
5 6 10
6 7 5
5 7 8
```

Corresponding Example Output:

```
Case 1: 1 2
Case 2: 1 4 6 7 9 11
```

Warning: If your output format is not correct (even spacing), you will get no credit; this problem will be graded by doing a `diff` against a standard output file.

Place your code (one or more source files) together with a `Makefile` in a fresh subdirectory with the name *your-last-name*`-code-hw7`. Then create a single `zip` archive with the name *your-last-name*`-code-hw7.zip` containing just that directory and its contents. It should be possible to build an executable file called `hw7` and test it on an input file `/path/to/foo` by the following steps:

1. `unzip` *your-last-name*`-code-hw7.zip`
2. `cd` *your-last-name*`-code-hw7`
3. `make`
4. `./hw7 /path/to/foo`