COMP 340 **Analysis of Algorithms**

Problem Set 6

Assigned: April 6, 2018 *Due: April 20, 2018*

0. [Optional] Your quiz next week will come from TADM chapter 6. It will have one question (possibly with 2 parts) which will be one of: 6-9, 6-13, or 6-17, so I suggest trying these problems. Note that the website's solution for 6-13 is unnecessarily complicated (and I'm not even sure it's correct).

1. Implement an algorithm to solve the bandwidth problem. This problem takes as input a graph G with n vertices and m edges. The goal is to find a permutation of vertices on the line which minimizes the maximum length of any edge. In other words, we need to a mapping f that assigns each of the n vertices a unique number from 1 to n. The cost of each edge (u, v) is then $|f(u) - f(v)|$, and we seek to minimize the total cost of all edges.

   Since this problem is NP-complete (which we will learn about later), it's quite unlikely you will find a polynomial-time (worst-case) algorithm. You can solve this problem with backtracking, and the goal of this assignment is to find a practical algorithm by pruning the backtracking search tree. However, you are not actually required to use backtracking: if you have some other technique in mind, go for it. But your program must always produce the correct answer.

   Note: You must do this assignment on your own — no partners/teams!

   You can use any programming language and computer you want to. The only requirement is that I should be able to run the program on the server (possibly with instructions provided by you).

Tips

- If you don't understand the problem, you will not produce a working program. Ask for help!
- Test input files are linked to on the course webpage. Test on smaller files first. Try creating your own input files, too. The link to input files includes a `README` file describing them and a `RESULTS` file with answers to them.
- Get your program working first! The first thing you'll need to do is read in a graph from a file. Skiena has C code for this, but similar code exists online for many different languages.
- Don't optimize prematurely. Pruning is the most important thing. Only after you've spent some time on this is it worth it to worry about things like whether iteration or recursion is faster.
- Think about what data structures you're using. What operations are necessary? Which operations are used more often than others? You're probably better off keeping it simple.
- Learn to use a profiler. Which one will depend on which language you're using and preference. Humans are bad at knowing where their programs spend their time — a profile tells you what is the slowest part.
- The input files are named `g-X-n-m`, where `X` is the type of graph (see the README file with the data files, although you may not need to worry about this detail), n is the number of vertices, and m is the number of edges. The first line in each file is n, the second is m, and the rest of the lines contain edges (two vertices), one per line.

Grading

Email me

1. Your source code
2. A brief description of your algorithm and any interesting optimizations you found
3. The resulting output from running your program on some sample input files.
4. The largest test file (in terms of the number of vertices) that your program could finish in one minute or less. Use the command line program `time` to do this, using the “real” clock time.

I will run your programs on the server, make sure they are correct, and time them. If your program does not produce correct answers, you will get a zero. So get it working first. You will get plenty of points for having a correct program, but full points will be awarded based on how fast your program is and the optimizations it makes.