SIENAcollege
Computer Science

Computer Science 385
Design and Analysis of Algorithms
Siena College
Spring 2018

Problem Set 1

Due: start of class, Friday, February 2, 2018

You may work alone or in a group of size 2 or 3 on this assignment. However, in order to make sure you learn the material and are well-prepared for the exams, you should work through tasks together or do them on your own before discussing them with your partner(s), should you choose to work in a group. In particular, the "you do these and I'll do these" approach is sure to leave you unprepared for the exams.

Getting Set Up

You will receive an email with the link to follow to set up your GitHub repository, which will be named `ps1-yourgitname`, for this problem set. One member of the group should follow the link to set up the repository on GitHub, then that person should email the instructor with the other group members' GitHub usernames so they can be granted access. This will allow all members of the group to clone the repository and commit and push changes to the origin on GitHub.

Submitting

Your submission requires that all required code deliverables are committed and pushed to the master for your repository's origin on GitHub. For this assignment, please answer the two written questions in your repository's `README.md` file. If you see everything you intend to submit when you visit your repository's page on GitHub, you're set.

Only one submission per group is needed. Please be sure that all group members' names are in all source code files and are clearly listed in the repository's `README.md` file.

Programming Task: A New Graph Method

In an undirected graph, such as the `HighwayGraph` from Lab 2, the *degree* of a vertex is the number of adjacent (undirected) edges.

1. Your repository includes a fresh copy of the `HighwayGraph.java` starter you used for Lab 2.

2. Add a new method to the `Vertex` class that computes the degree of the vertex with a given number. So the method call added in `main`:

```
int d = g.vertices[12].degree();
```

would compute and return the degree of vertex 12 and store it in `d`. (4 points)

Of course, the above could be a good test, but you do not need to include such tests in your final submission.

3. Remove the `System.out.println(g);` line from the given `main` method.

4. Add code in `main` that computes and prints a table of the number of vertices of each degree in the graph. To do this efficiently in terms of memory, do **not** store lists of the vertices of every degree. You only need the counts, so only store the counts. Further, your solution should work for any maximum degree but should be memory efficient. That is, your storage should be $O(maxdegree)$, not $O(|V|)$ or $O(|E|)$. (7 points)

5. Add code in `main` that finds the largest degree of all the waypoints (vertices) in the graph, and then prints out a line containing the name of the input graph file, the largest degree, and lines for each of the vertex labels and their coordinates which have that largest degree. So here, you **will** need to store a list of vertices, but only store it for the largest degree, again so we are not wasting memory on things that are not needed. (6 points)

You should download and test with several of the graphs (including some larger ones with thousands of vertices) on the METAL web site at `http://tm.teresco.org/graphs/`. Your code will be tested on a wide variety of graph inputs. You can check correctness on small graphs by drawing them by hand or looking at them in HDX.

The reference solution produces the following output for the region graph for the state of Missouri:

```
Graph MO-region.tmg. |V| = 4776, |E| = 5428
Vertex degree counts:
0: 0
1: 162
2: 3479
3: 811
4: 317
5: 7
All vertices of degree 5
I-35BLCam/US36_E/US36BusCam/US69 (39.754482,-94.234371)
US36_W/US36BusCla/MO110Han/MO151 (39.752481,-92.258055)
US24_E/US24BusPar/MO15 (39.493634,-92.000027)
I-435(71)/I-49/I-470/US50/US71 (38.94142,-94.53685)
MO100@TukBlvd&US66HisStL@MO100&US66HisMan@US66His_E (38.618748,-90.201691)
I-55(93)/I-55BLCap/US61/MO74 (37.260118,-89.566941)
US65_N/US65BusOza/MO14 (37.023593,-93.228714)
```

Programming Task: Generating Example Arrays

In future problem sets, you will be asked to perform empirical analysis on the sorting algorithms we will be studying. To do this, you will need to generate input arrays for the sorting algorithms. In order to test the best, worst, and average cases of some of these algorithms, you will need to generate input arrays with various characteristics. For simplicity, we will sort arrays of `int`.

- an array filled with n random values within a given range
- an array filled with n values sorted in ascending order
- an array filled with n values sorted in descending order
- an array filled with n values "nearly" sorted in ascending order

You may use any programming language. If you use Java, implement it within a class `IntArrayGenerator` that includes `static` methods to generate and return arrays of `int` with each of the above characteristics, and a `main` method that tests these methods for various values of n and ranges. Be sure you can achieve similar functionality if you choose a different language. It is important that you generate these efficiently – for example, you should not generate sorted input by generating random input then sorting it. Generate it in sorted order right from the start. (12 points)

Written Questions

For the questions below, consider the graph representations discussed in class for storing directed graphs.

Suppose that the amount of memory required by the adjacency matrix graph representation is exactly $\frac{|V|^2}{8}$ bytes. (This assumes each Boolean value in the array is stored as a single bit.) In addition, assume the amount of memory required by the adjacency lists graph representation is exactly $16|V| + 16|E|$ bytes[1].

? Question 1:

If you have a graph with 1,000,000 vertices and each vertex has 4 outgoing edges, exactly how much memory in gigabytes is needed to store the graph using each representation? For each representation, can it fit into 16GB of main memory which is typical of a PC today? (3 points)

[1] Justification for why $16|V| + 16|E|$ is a reasonable ballpark estimate for the adjacency lists representation from class: It counts 4 bytes for each reference in the `vertices[]` array, 12 bytes for each `Vertex` object (4 bytes for the `head` reference plus 8 bytes needed by Java to store housekeeping information about each `Vertex` object), and 16 bytes per `Edge` object (4 bytes for the integer `dst`, 4 bytes for the `next` reference, and 8 bytes to store Java housekeeping information about each `Edge` object).

Question 2:

Graphs that have only a few number of edges per vertex are known as sparse graphs. A graph with a high number of edges per vertex is said to be dense. Suppose you have a complete directed graph of 1,000,000 vertices meaning that every vertex has a directed edge to all the other vertices. This is the "densest" graph there is! Exactly how much memory is needed to store the graph using each representation? Express your answer in gigabytes. (3 points)