HW8 Grading Policy & Some Clarifications

April 26, 2001

Grading

Here is how the program for computing shortest paths was graded by Bo:

- The program was tested on
  g.Dijkstra("NJJERSEY CITY N", "PASCRANTON SSW", output);
  the "test: ****ms" written on your report is the time your program took to run on my machine. Most results are well under 4s. (Whether your algorithm stops when the destination is extracted from the priority queue doesn't change the order of the running time.) 4 points were deducted for an incorrect result, 10 points were deducted if algorithm didn't work, and 2 points were deducted if the running time was greater than 10s.
- Adjacency Lists should be used for representing the graph,
- A binary heap should be used for the priority queue.
- You should keep track of the position of the vertex in the heap, so that Decrease-Key can be done in $O(log|V|)$. Searching through the whole heap to find the vertex whose key you want to decrease doesn't make sense. That means that you are not taking advantage of the heap structure; that search alone would take $O(|V|)$. 4 points were deducted if you did this badly.
- The correct result for connected component is 56. 4 points were deducted for an incorrect result
- BFS is used to compute connected components, since the biggest connected component has 91665 nodes. If you use DFS, it would go into that many levels of recursion before exiting. This would need a very large stack. If you tried DFS, you were likely to get a StackOverflow Exception for this assignment. On the other hand, for BFS, the max size for the queue only 709 for the biggest component, and each element of the queue only needs to hold the infomation you need. For DFS, for each step in recursion, lots of information is stored. So recursive functions, though clean in style, can be dangerous when you go too deep. Note that the problems is not that we have a big graph. If each connected component were small, DFS won't have too many levels of recursion. 1-2 points were taken off for not giving an adequate reason for choosing BFS over DFS.

- I was loose in grading the running time analysis, since I didn't make it clear in the handout. Some of you gave me the actual running time, some gave an analysis in the big O notation, some even plotted a graph of running time vs. graph size...

Notes posted in newgroup

In case you didn't read the newsgroup and were confused ...

- The first parameter of the constructor for Graph might be confusing for some of you. It's supposed to be the first nodeID you want to include in your graph. I think I didn't mention that the nodeID is roughly ordered by states, so nodes in the NY state have nodeID between NY_NODE_BEGIN, NY_NODE_END (I give the value of them in test.java). So if you want to be able to construct a subgraph for one state so that you can test your algorithm on a smaller data, you can make use of that parameter. but I will only test your program with Graph g = new Graph(1, NUM_NODES, "road.dat"). Thus, you may choose to ignore the first parameter in your program.

- All the node IDs in the example of the handout are off by 1. The reason is that the node ID in the data file starts with 1, while in my program, since I used that infomation as the index to the array, I started with 0. Thus, each one is off by 1, and I didn't change it back when I printed them out ...