Computer Science 431
Algorithms
The College of Saint Rose
Spring 2015

Problem Set 4: Brute Force and Decrease and Conquer

Due: 11:59 PM, Thursday, March 12, 2015

You may work alone or in groups of size 2 or 3 on this assignment. Only one submission per group is needed.

Programming Task: Insertion Sort

Extend your sorting algorithm comparison program from the previous problem set to include options to use insertion sort. (5 points)

Use the extended code to perform an empirical analysis of insertion sort and add it to your document describing your empirical studies. (6 points)

Note: the code, empirical results, and analysis for bubble sort and selection sort that you submitted previously will not be graded. Instead, your grade for that part of the previous problem set will be determined by your updated submission. So fix it up!

Programming Tasks: Working with Highway Data

You worked a bit with the graph data representing highways in your first problem set. We will now return to that to start building a program in which you will implement several algorithms over the course of the rest of the semester.

Start with the skeleton of the program in `/home/cs431/maps/Mapping.java`. As you can see, this program reads in a `.gra` file, then enters an interactive loop where the user can perform a variety of operations on the graph.

Your tasks for this problem set:

- Add your code that builds the graph into the `Mapping` constructor. If your previous code did not work, a working version can be provided for you.
- Print out, in a nice format, a list of all waypoints. This is the `listPlaces` method in the starter code. (3 points)
- Print out, in a nice format, a list of all connections. This is the `listConnections` method in the starter code. (3 points)
- Print out the northernmost and southernmost latitudes, the easternmost and westernmost longitudes among waypoints in the graph, the shortest and longest waypoint names, the lengths of the shortest and longest road segments, and the average road segment length in the

graph. Implement this as a new command `Stats`. Do **not** remember these in variables when you are reading the file and creating the graph. Compute them from your graph structure when the command is issued. In the case of ties for the longest and/or shortest names, your command should print all waypoint names of the extreme length. (12 points)

For example, for the `dc-all.gra` graph, my `Stats` command prints:

```
Lat,Lng extents: (38.792435,-77.070508) to (38.984333,-76.934123)
Shortest waypoint names:
I-395@7
I-395@9
I-295@3
I-395@3
I-295@2
I-395@4
I-395@2
I-395@8
I-295@1
I-395@5
Longest waypoint names:
DC295@I-295/695&I-295@4&I-695@I-295/295
Connection lengths: shortest 0.0665295151, longest 1.69070, average 0.65215
```

- Implement new commands `PrintDepthFirst` and `PrintBreadthFirst` that print out the depth-first and breadth-first traversals of the graph starting at a given `Waypoint` label.

  For example, for the `dc-all.gra` graph, my `PrintDepthFirst` command with a starting point of `US1@DC/MD` prints:

```
0: US1@DC/MD (38.935078,-76.963005)
1: US1@MonAve (38.924661,-76.985664)
2: US1@CapSt (38.916882,-77.009182)
3: US1@RhoIslAve_W (38.913142,-77.019997)
4: US1@US50_E&US50@US1_N&US1AltWas@US1_S (38.903458,-77.019525)
5: US50/US1AltWas@I-395 (38.90486,-77.015705)
6: US50/US1AltWas@BreRd (38.913008,-76.992016)
7: US50@US1Alt_N&US1AltWas@US50_E (38.917249,-76.972146)
8: US1AltWas@DC/MD (38.930637,-76.957253)
9: US50@SouDakAve (38.918184,-76.954594)
10: US50@DC/MD (38.917917,-76.941805)
11: US1/US50@ConAve (38.891968,-77.021542)
12: US1_S/US50_W (38.891968,-77.031627)
13: US1@MaiAve (38.88365,-77.032099)
14: I-395@1&US1@I-395 (38.879249,-77.036197)
15: I-395@2 (38.879107,-77.033601)
16: I-395@3 (38.882314,-77.027882)
17: I-395@4 (38.882402,-77.024009)
```

```
18: I-395@5 (38.882515,-77.019176)
19: I-395@7 (38.882815,-77.01254)
20: I-395@8 (38.885496,-77.01269)
21: I-395@9 (38.894757,-77.014182)
22: I-395@10 (38.897813,-77.014187)
23: I-395@US50 (38.90491,-77.016081)
24: I-395/US1@VA/DC (38.873929,-77.043343)
25: I-66/US50 (38.893471,-77.053385)
26: I-66@EStExpy (38.895909,-77.053256)
27: I-66@US29 (38.902489,-77.055831)
28: I-66/US50@VA/DC (38.891834,-77.06479)
```

and my `PrintBreadthFirst` command with a starting point of `US1@DC/MD` prints:

```
0: US1@DC/MD (38.935078,-76.963005)
1: US1@MonAve (38.924661,-76.985664)
2: US1@CapSt (38.916882,-77.009182)
3: US1@RhoIslAve_W (38.913142,-77.019997)
4: US1@US50_E&US50@US1_N&US1AltWas@US1_S (38.903458,-77.019525)
5: US1/US50@ConAve (38.891968,-77.021542)
6: US50/US1AltWas@I-395 (38.90486,-77.015705)
7: US1_S/US50_W (38.891968,-77.031627)
8: US50/US1AltWas@BreRd (38.913008,-76.992016)
9: I-66/US50 (38.893471,-77.053385)
10: US1@MaiAve (38.88365,-77.032099)
11: US50@US1Alt_N&US1AltWas@US50_E (38.917249,-76.972146)
12: I-66/US50@VA/DC (38.891834,-77.06479)
13: I-66@EStExpy (38.895909,-77.053256)
14: I-395@1&US1@I-395 (38.879249,-77.036197)
15: US50@SouDakAve (38.918184,-76.954594)
16: US1AltWas@DC/MD (38.930637,-76.957253)
17: I-66@US29 (38.902489,-77.055831)
18: I-395/US1@VA/DC (38.873929,-77.043343)
19: I-395@2 (38.879107,-77.033601)
20: US50@DC/MD (38.917917,-76.941805)
21: I-395@3 (38.882314,-77.027882)
22: I-395@4 (38.882402,-77.024009)
23: I-395@5 (38.882515,-77.019176)
24: I-395@7 (38.882815,-77.01254)
25: I-395@8 (38.885496,-77.01269)
26: I-395@9 (38.894757,-77.014182)
27: I-395@10 (38.897813,-77.014187)
28: I-395@US50 (38.90491,-77.016081)
```

Implementation of these commands is worth 12 points.

Written Problems

1. Levitin Exercise 3.3.3, p. 113 (5 points)
2. Levitin Exercise 3.5.1, p. 128 (3 points)
3. Levitin Exercise 3.5.4, p. 128 (3 points)
4. Levitin Exercise 4.1.9, p. 137 (2 points)
5. Levitin Exercise 4.4.8, p. 156-157 (6 points)

Submitting

Before 11:59 PM, Thursday, March 12, 2015, submit your problem set for grading. To complete the submission, upload an archive (a `.7z` or `.zip`) file containing all required files using Submission Box at `http://sb.teresco.org` under assignment "PS4".

Grading

This assignment is worth 60 points, which are distributed as follows:

Feature	Value	Score
Insertion sort implementation	5	
Insertion sort section of empirical analysis	6	
`listPlaces` method in `Mapping`	3	
`listConnections` method in `Mapping`	3	
`Stats` command in `Mapping`	12	
`PrintDepthFirst` and `PrintBreadthFirst` commands	12	
Exercise 3.3.3	5	
Exercise 3.5.1	3	
Exercise 3.5.4	3	
Exercise 4.1.9	2	
Exercise 4.4.8	6	
Total	60	