Gregory Jerian

2/11/16

Period 4

Dijkstra’s Algorithm Readme

Introduction

This program calculates the shortest path between two nodes on a graph. To do this, it utilizes Dijkstra’s Algorithm, a fancy algorithm created by a guy called Dijkstra who was apparently really smart.

How this fulfills the expectation

This assignments fulfills the expectation of using Dijkstra’s algorithm to find the shortest path between two nodes. The graph is represented by a HashMap that maps Strings to HashMaps that map Strings to Integers. In this way, both connected nodes and distances are represented for each node in the graph. The user cannot access most of the code as it has private visibility modifiers, but can access the public HashMap path, showing the previous node in the path from the starting node to a certain node for each node in the graph. Also, the user can access the Shorty interface, allowing them to call Shorty.next(), which gives them the shortest path between a starting node (implicit in a Dijkstra object) and an end node (a String entered by the user).

Explanation of current errors

There are no errors, amazingly. <https://www.youtube.com/watch?v=3GwjfUFyY6M>

Overview of code

There are two classes (Graph and Dijkstra) and an interface (Shorty). The Graph class is used mainly for testing, as it creates a graph and passes it into the Dijkstra class. The Shorty interface is called by the user to get the shortest path. The code that actually applies the Dijkstra algorithm and finds the shortest path is all in the Dijkstra class.

In the Dijkstra class, the first thing that happens is that all four ArrayLists from the powerpoint are initialized. These include the nodes, the selected set (k), the distances to the start node (d), and the previous nodes (prev). Then, the distances between the current (starting) node and all the nodes it is directly connected to are found. After these are recorded in the d ArrayList, the program sets the node that has the shortest distance to the starting node as the NEW current node. It repeats the process stated above, finding all the nodes directly connected to the NEW current node, and finding their distances to the start node. It does this over and over, and if it finds a path to a certain node that is shorter than the one already recorded in the d ArrayList, then it replaces that distance with the shorter one. Finally, it updates the previous node, replacing it with the new one. Once the program has iterated through all nodes as the current node, the ArrayLists should be filled up and the shortest paths found. The user can test this by calling Shorty.next(), which will give the shortest path to a certain end node using the path HashMap, which maps each node to the one before it.

Discussion of major challenges

I didn’t encounter too many major challenges. At first it was hard understanding how to use the HashMaps but I figured out how to use iterators to go through them eventually. Also, some of the logic was weird with replacing a previous path with a new, shorter one, but I figured it out eventually.

Acknowledgements

Thanks to Mr. Kuszmaul for helping me with the code. Also, thanks to the “powerpoint stolen from the internet” that was posted on Schoology.