Project 2

Dijkstra's Algorithm Implementation

CS 241 Winter 2017, Dr. Hao Ji

Taylor Thurlow

tjthurlow@cpp.edu

10 March 2017

Section 1: Project description

The program is an implementation of a graph structure using vertices and edges. Using Dijkstra's algorithm and a priority queue, we can traverse the tree and determine the shortest path between on vertex and any other vertex. The program provides a simple command-line based interface which gives the user the option to add an edge (road), determine the shortest distance between two vertices (cities), and remove an edge (road). All data is populated from the provided data files when the program is executed.

Section 2: Project specification

Implement a graph data structure for a practical application.

Input: Two files - one contains city data and the other contains road data.

- city.dat: This file contains information about cities, where each line has 5 attributes: *City Number*, *City_Code* (2 letters), *Full_City_Name*, *Population*, and *Elevation*.
- road.dat: This file contains information about roads, where each line has 3 attributes: *From_City*, *To_City*, and *Distance*. Note that all roads are assumed to be one-way.

Output: A menu driven system which has the following options.

- Read the original data files and store the data to appropriate data structures.
- Let the user of this program enter a City Code and your program should print out the city information (the whole record).
- Find the connection between two cities.
    - The user will be asked to enter two City Codes. The program finds the shortest distance between the two cities.
- Insert a road (edge) between two cities

    - The user will be asked to enter two City Codes and its Distance. Note that if a pair of City Codes already exists or if the City Code doesn't exist, print out a warning message.
- Delete a road (edge)
    - The user will be asked to enter two City Codes for a road. Note that if the road entered doesn't exist, print out a warning message.
- Exit.

Section 3: Testing methodology

The majority of the work for this project is in the implementation of Dijkstra's algorithm. Even though the algorithm itself isn't necessarily complicated, there are a lot of seemingly simple functions required that can be hard to understand and write. There is a LOT of code written before it becomes reasonable to start testing anything, just because there are so many small components to it. Aside from testing the obvious required input types, the important thing to check for is that the program doesn't get stuck in a loop looking for a city that doesn't exist. Building the command line interface wasn't difficult but took some time and thought - it's easy to mess up the parsing of the input.

Section 4: Lessons learned

Dijkstra's algorithm seems easy enough on paper, but there are a lot of small steps that are easily 'ignored' when doing the steps in class. The data structures themselves were not that complicated given our experience with them in previous projects and lectures. The biggest hurdle was writing the code for Dijkstra's without testing - in hindsight I should have stopped and figured out a way to test that what I had written was functioning.

Section 5: Analysis of output

Maps were used for the graph structure, and Dijkstra's algorithm is implemented with a priority queue. Priority queues are implemented with binary heaps. In general, the total time complexity of a binary heap in this case is $O((|V| + |E|) \log V)$. The map structure is not as clear, though the Java specifications do specify a removal time complexity of $O(1)$ and a value access complexity of $O(n)$. This data structure seemed to fit my application perfectly, and I had almost no issues with its implementation.

```
Command: Q

City code: LV

12 LV LEE VINING 8390 5983

Command: D

City codes: CH PM

The minimum distance between CHINO HILLS and POMONA is 143

through the route: CH, PR, TR, PM

Command: I

City codes and distance: GG BO 100

You have inserted a road from GARDEN GROVE to BOSSTOWN with a

distance of 100.

Command: R

City codes: KV MP

This road does not exist.

Command: E

Exiting...
```