TCSS 342A Project 3
Graphs and Shortest Paths

Group Number: K
Members: Raymond, David, Walker
Date: 03 / 09 / 2017

Part 1: Graph Representation

- Used a HashMap from a **Vertex** to an **ArrayList<Edge>**
    - Each Vertex maps to a list of its adjacent edges
    - Hash maps (and java.util.HashMap) time complexities are:
        - Average case O(1)
        - Worst case O(n)
    - Benefits:
        - Easy and quick to find a vertex's edges
            - Avoids adding duplicate vertices
        - Space complexity is O(|E|) - much better than an adjacency matrix

- Stored boolean *known*, Vertex *prev*, and int *weight* (used for a PriorityQueue) as fields in the Vertex class
    - all used by Dijkstra’s algorithm

Part 2: Dijkstra’s algorithm

- Vertex implements the Comparable interface
    - Compares vertices based on the distance variable associated with that vertex (prioritizes smallness)
    - Distance variables are set to Integer.MAX_VALUE before every shortestPath() call.

    - Vertex being Comparable allows a PriorityQueue to be used in shortestPath() in order to prioritize the smallest distances

How you tested your code

Our test code for the shortest path can be split in 2 categories

- Valid Input:
- -First we tested that the graph can find a very basic path from on to another node. Then we ran the test again with more and more nodes.
- -Second we can a couple test to make sure the graph can select a path of with more nodes and less weight.
- -Lastly we checked to make sure the the graph will return a
- with just a single node and a weight of 0 when the same
- node is passed twice.
- Invalid Input:
- -When an unreachable node is entered as the destination does the shortest path return null?
- -Does shortest path return null when one or both nodes are null?
- - Does it return null when one or both nodes are not part of the graph?

Real work application - Finding Paths Between Movies Via Filming Location

Getting program-readable data...

a) Raw movie database file is downloaded from IMDB.com
b) Database is truncated to movies starting with 'A ' because there are so many movies.
c) File is parsed and a digraph is created from movie and location data.
d) Edge and vertex files read by program as normal.

Notes

- It is technically an undirected graph because (movie1, movie2) implies the inverse
- It is also technically an unweighted shortest path problem (all edges have the same cost)
    - So, this isn't the most efficient way to solve this problem
    - But it's a good way to test our program

Results

```
Vertices size: 6516
Edges size: 501442
Start vertex? A Christmas Story (1983)
Destination vertex? A Day in the Warsaw Ghetto: A Birthday Trip in Hell (1991)
Path is [A Christmas Story (1983),
        <<<connected by Cleveland, Ohio, USA>>>
A Film About Races (2009),
        <<<connected by Los Angeles, California, USA>>>
A Brief Sketch (2002) (V),
        <<<connected by Hollywood, Los Angeles, California, USA>>>
A Recipe for Love (2010),
        <<<connected by Warsaw, Mazowieckie, Poland>>>
A Day in the Warsaw Ghetto: A Birthday Trip in Hell (1991)]
Weight is 4
```

Program successfully processes 6,516 movies with 501,442 edges.

```
A Walk with Nigel (2010)
A Year in the Life (2009) }
1   London, England, UK
A Walk with Nigel (2010)
A Year of Your Love (2009)
1   London, England, UK
A Walk with Nigel (2010)
A doppia faccia (1969)
1   London, England, UK
A Walk with Nigel (2010)
A mi me gusta (2008)
1   London, England, UK
A Walk with Nigel (2010)
A to Z: The Life Cycle of a Car (1997)
1   London, England, UK
```

Modified edge file format that adds an edge description which allows the program to show which location linked each movie pair.

Conclusion

Division of labor

Walker

– Basic graph and Dijkstra algorithm implementation

- Bonus section

Raymond

- Wrote Test class and refined shortestPath

David

- Debugged and refined MyGraph methods
- Exception Handling