

Shortest Paths in a Network

Compiler and Platform

Compiler : Java compiler(javac)
Java version : JDK 1.8.0
Programming : Java
Platform : Eclipse IDE

Program Description

This program initially builds graph based on input file. Note that input file should have format < vertex1 > < vertex2 > cost. The program computes shortest path based on Dijkstra Algorithm. . Dijkstra Algorithm is computed based on takes two input values. One is the Source Node value and other is the destination Node value and computes the shortest path based on the weight or cost between the nodes. We can also obtain all reachable vertices from each vertex using DFS Algorithm. You can enter the commands or get output in a file via command mentioned below.

Project Files-

Graph.java –

Graph Class initializes and handles all the operations related to the graph. It has Vertex class which keeps information about vertices, Edge class to keep information about edges, MinHeap class which defines the functions of Minimum Binary Heap and Path Class which stores path names. It has main method which reads the input file and executes queries.

And it has following functions –

- boolean processQuery(Scanner in, Graph g) --> Executes queries.
 - void initializeGraph(String s, String d, double c, boolean st) --> Build an undirected graph.
 - void addEdge(String v, String w) → Add additional edge
 - void deleteEdge(String v, String w) → Delete the edge
 - void edgedown (String v, String w) → Make edge unavailable
 - void edgeup (String v, String w) → Make edge available
 - void printPath(String w) → Print path after alg is run
 - void unweighted(String s) → Single-source unweighted
 - boolean isEdgeDown(String source, String dest) --> Find if edge is down
 - void vertexUp (String v) → Make vertex available
 - void vertexDown (String v) → Make vertex unavailable
 - void Dijkstra (String src) → Find shortest path from source
 - void printPath (String dest) → Print path from source to dest.
 - void DFS()
 - void DFS_Visit(String node) → Iterate through all adjacent edges.
-

Shortest Paths in a Network

Data Structures used –

- 1) *TreeMap* - Vertices information is stored. TreeMap is preferred over HashMap because it orders vertex names in alphabetical order.
- 2) *ArrayList* – Maintains the information about list of neighbouring /adjacent edges.
- 3) *Boolean* - Is used to store the availability of a vertex.
- 4) *LinkedList* – Is used to store information about down edges. Two vertices of an edge will be stored as linked list.
- 5) *TreeSet* – Is used to store information about reachable nodes in Reachable Algo. TreeSet is mainly used because we can iterate easily inside this structure.

Two classes have been created to store information about Vertex and Edges.

Execution

Unzip the file

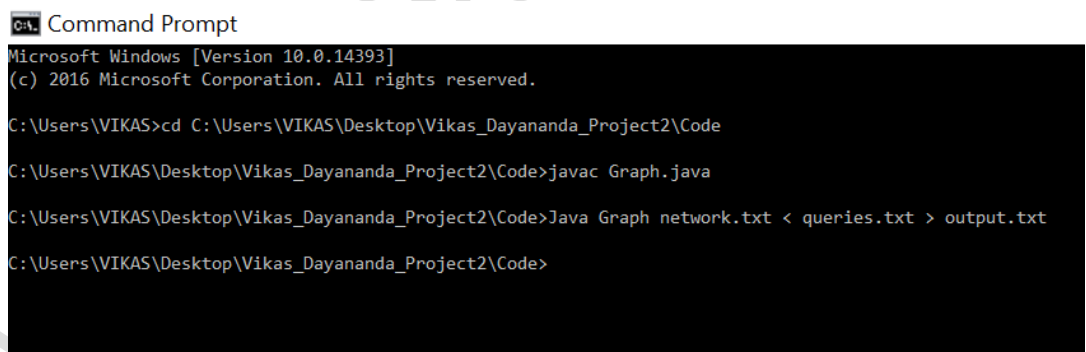
Open Command Prompt

Change Location to the code folder

Enter following Commands

Javac Graph.java

Java Graph network.txt < queries.txt > output.txt



```
C:\> Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\VIKAS>cd C:\Users\VIKAS\Desktop\Vikas_Dayananda_Project2\Code
C:\Users\VIKAS\Desktop\Vikas_Dayananda_Project2\Code>javac Graph.java
C:\Users\VIKAS\Desktop\Vikas_Dayananda_Project2\Code>Java Graph network.txt < queries.txt > output.txt
C:\Users\VIKAS\Desktop\Vikas_Dayananda_Project2\Code>
```

Limitations-

The program works well with all the input graphs as long as it follows the format. Edges with negative cost cannot be used because Dijkstra Algorithm does not consider negative edges. I also tested with some large graphs and code worked well.

By,

Vikas Dayananda

vdayanan@uncc.edu

800969865