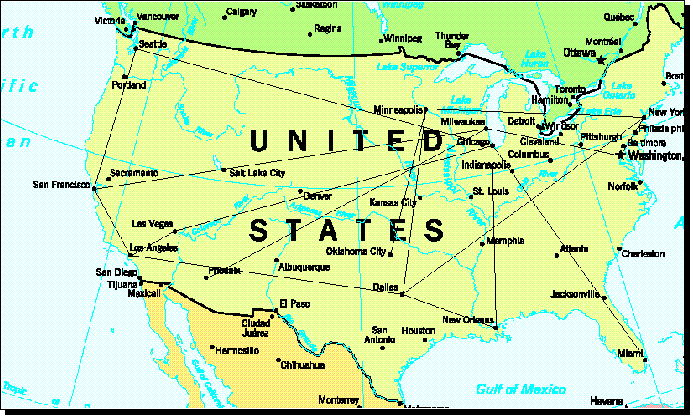
Problem

Write a program that determines the shortest distance between any two connected cities on the map below.



Specifications

1. Run the program from a menu that includes the following options:
   1. Show Connecting Cities
   2. Depth-first Traversal
   3. Breadth-first Traversal
   4. Shortest Path - Connections
   5. Shortest Path - Distance
   6. Adjacency Matrix
   7. Exit
2. To show connecting cites for each city on the map, show the city and each connected (adjacent) city and mileage to that city.
3. For shortest path connections, show the fewest number of connections on the path from the source city to the destination city.
4. For shortest path distance, show the shortest distance and the connecting cities on the path from the source city to the destination city.
5. Use an adjacency matrix to solve the problem given the following [table of distances](http://www.cs.slcc.edu/~bairdro/CS2420/TableOfDistances.html).

Admin

1. Grading
   * 0 points if your program does not compile.
   * +5 for comments, indentation and placement of {} per [Style Guide](http://www.cs.slcc.edu/style-guide.shtml).
   * +5 for each specification met.
2. Submission: An executable JAR file that also contains your .java source code files.

Make vertex of cities and vertex of distances the same as every row and column

String[] cities;

Double[][]weights;

Boolean[] visited;

Constructor

Public ShortestPath()

{

}

Adding edges to graph

Graph.addEdge(cedarCity, 105.0, vernal)

For(String city: cityList)

{

Graph.addVertex(new Vertix<String>(city))

}

**CS2420 Assignment 12 - Shortest Path**

**Table of Distances**

|  |
| --- |
|  |
|  | **SEA** | **SFR** | **LA** | **LV** | **PHO** | **OKC** | **DAL** | **MIN** | **MIL** | **CHI** | **NOR** | **NYC** | **WDC** | **MIA** |
| **SEA** | 0 | 808 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2060 | 0 | 0 | 0 | 0 |
| **SFR** | 808 | 0 | 414 | 0 | 0 | 0 | 0 | 0 | 2257 | 0 | 0 | 0 | 0 | 0 |
| **LA** | 0 | 414 | 0 | 272 | 0 | 0 | 1440 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **LV** | 0 | 0 | 272 | 0 | 0 | 0 | 0 | 0 | 0 | 1780 | 0 | 0 | 0 | 0 |
| **PHO** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1771 | 0 | 0 | 0 | 0 | 0 |
| **OKC** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 792 | 0 | 0 | 0 | 0 | 0 | 0 |
| **DAL** | 0 | 0 | 1440 | 0 | 0 | 0 | 0 | 949 | 0 | 0 | 571 | 1614 | 0 | 0 |
| **MIN** | 0 | 0 | 0 | 0 | 0 | 792 | 949 | 0 | 0 | 0 | 0 | 1217 | 0 | 0 |
| **MIL** | 0 | 2257 | 0 | 0 | 1771 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 811 | 0 |
| **CHI** | 2060 | 0 | 0 | 1780 | 0 | 0 | 0 | 0 | 0 | 0 | 948 | 0 | 0 | 1423 |
| **NOR** | 0 | 0 | 0 | 0 | 0 | 0 | 571 | 0 | 0 | 948 | 0 | 0 | 0 | 0 |
| **NYC** | 0 | 0 | 0 | 0 | 0 | 0 | 1614 | 1217 | 0 | 0 | 0 | 0 | 237 | 0 |
| **WDC** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 811 | 0 | 0 | 237 | 0 | 0 |
| **MIA** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1423 | 0 | 0 | 0 | 0 |

Above is an example of an adjacency matrix

Below is an example of an adjacency list

SEA → SFR →CHI

Shortest path logic example on 13.4

Breadth first and depth first traversal logic on 13.3

p. 678 may have a good example of a constructor

I’ve read through 13.4