CSCI 311 - Algorithms and Data Structures
Project 3

Assignment Date: Apr 15, 2024

Due Date: 11:59pm on May 07, 2024

Grace Date: 11:59pm on May 10, 2024

Introduction

In this project, we will be implementing a revised version of Dijkstra's algorithm and using it to solve a small variation on the shortest path problem. As in the first two projects, there is very little direction regarding how to design or structure your code. These decisions are, mostly, left to you. Make sure to start early and to stay organized. Note that this project is a little smaller than previous projects and is worth a total of 100 pts.

Submissions

a) Your C++ solution code ("project_3.cpp") for this project should be submitted on both **inginious** and **Canvas**.

Notes

There will be more than enough time in lab to discuss these problems in small groups and I highly encourage you to collaborate with one another outside of class. However, **you must write up your own solutions independently of one another**. **You also need to start with the skeleton file give on Canvas (rename it to "project_3.cpp").** Do not post solutions in any way.

The Problem

You are a software engineer at a nameless electric vehicle company working as part of a team on a tool for helping customers plan road trips. Charging stations are not yet as widely available as gas stations so, when determining a route from the starting point to the destination, the range of the vehicle must be considered. With summer approaching, your team is scrambling for a solution to this problem. Using your knowledge of graphs and shortest path algorithms, propose and implement an efficient algorithm for determining the shortest path between two cities paying attention to vehicle range and available charging stations.

Inputs/Outputs

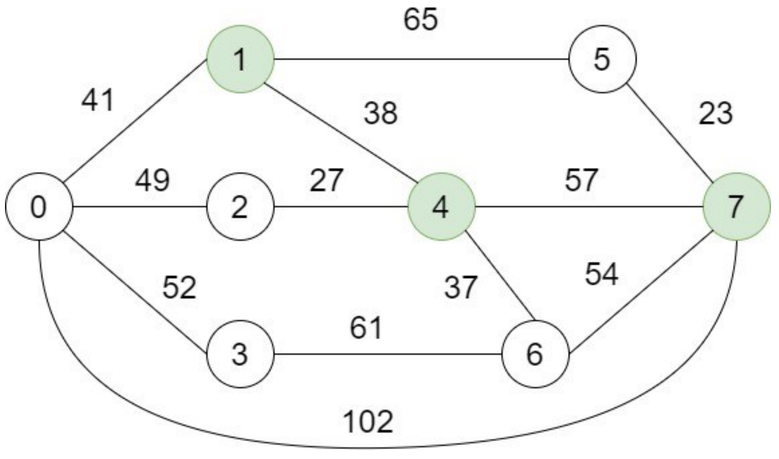
- Input will come from cin.
  - The first line will contain four space separated integers, n, m, c, and i.

    * n is the number of nodes in the graph.
    * m is the number of edges in the graph.
    * c is the maximum distance in miles the vehicle can travel on a full charge.
    * i is the range of the vehicle with its initial charge.
  - The second line contains a pair of integers, *start* and *end*, indicating the start and end nodes for the trip.
  - The following n lines contain pairs of space separated integers, v and s, representing nodes.
    * v is the name of a node. This will be an integer between 0 and $n-1$.
    * s is 1 if the node v has a charging station and 0 otherwise.
  - The following m lines each contain three space separated integers, u, v, and d, representing edges.
    * u and v are edge endpoints.
    * d is the distance in miles between nodes u and v.
- Print output to cout.
  - The output will have two lines.
  - The first line should be "Verify Path: 1" when a suitable path exists.
  - The second line should be the length of the trip followed by a colon, the *start* state, the sequence of charging stations visited on the trip, and the *end* state.
    * This sequence should be space separated, should start with *start*, and should end with *end*.
    * This sequence should represent a shortest path given the constraints. Ties will be broken between the paths with the same distance by the sum of the ids of the nodes along each path (in ascending order, which means the path of the smallest sum wins.)
  - If no suitable path exists, print "No suitable path from *start* to *end* exists" where *start* and *end* are node ids.

For example, the graph below corresponds to the input:

```
8 12 80 80
0 7
0 0
1 1
2 0
3 0
4 1
5 0
6 0
7 1
0 1 41
0 2 49
0 3 52
0 7 102
1 5 65
1 4 38
2 4 27
3 6 61
4 6 37
5 7 23
```

4 7 57
6 7 54



Output in this case should be:

```
Verify Path:  1
133: 0 4 7
```

Grading

- (30 pts) An implementation of Dijkstra's algorithm.
- (50 pts) A solution to the given problem, based on the percentage of test cases passed on inginious.
- (10 pts) A function bool verifyPath(Graph G, vector<int> path, int i, int c) which, given a graph, a vector of node ids representing a path from the start node to the destination, the initial charge of the vehicle, and the maximum charge of the vehicle, returns true if the vehiclecan make the trip following the given path and false otherwise.
- (10 pts) Coding style and readability.