COL-106 Assignment 6
For non-AGP students

General Objective

In this assignment, you will be working on computing shortest paths in a graph with turn penalties.

Problem Statement

A road network can be modeled as a graph, where each junction is modeled as a vertex and each road segment as a weighted edge. The weight of an edge represents the cost incurred in traveling through the corresponding road segment. Additionally, there is a cost associated with every turn taken while traversing the network. For example, every left turn incurs a cost c_{left}, every right turn incurs a cost c_{right}. You are required to find the minimum cost path from node A to node B, using Dijkstra's shortest path algorithm.

Dijkstra's algorithm cannot incorporate turn costs directly, since the turn costs are dynamic, which violate the invariant of the algorithm. To incorporate turn costs, a pseudo-dual of the graph can be created, modelling road segments as nodes and turns between two consecutive nodes as edges [1]. Refer to [1] for details on creation of the pseudo-dual graph. In this pseudo-dual graph, Dijkstra's algorithm can be applied directly to compute the shortest paths.

We will consider directed graphs of the form of rectangular grids of size $H \times W$, that allow three directions of movement (backward turns are not allowed). Accordingly, there will be three fixed turn costs c_{left}, c_{right} and $c_{forward}$.

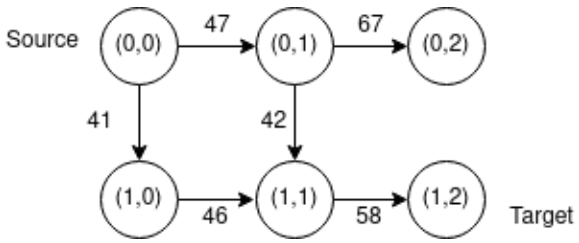

Figure 1

Figure 1 shows a 2×3 grid, where you are required to find the shortest path from node $(0, 0)$ to node $(1, 2)$. Say the turn costs are given by $c_{left} = 8$, $c_{right} = 1$ and $c_{forward} = 0$. Going from $(0, 0) - (0, 1) - (1, 1)$ in this grid will incur a cost of $c_{forward} + 47 + c_{right} + 42$, assuming the first step is always taken in a forward direction.

Figure 2 shows the corresponding pseudo-dual graph, constructed as per [1]. The graph contains a node for every edge in the original graph that does not terminate in the Source

node. Additionally, we define the source node of the dual graph as $((-1,-1),(0,0))$ corresponding to an edge from an imaginary node $(-1,-1)$ to the source node in the original graph. You must follow the same terminology in your code.

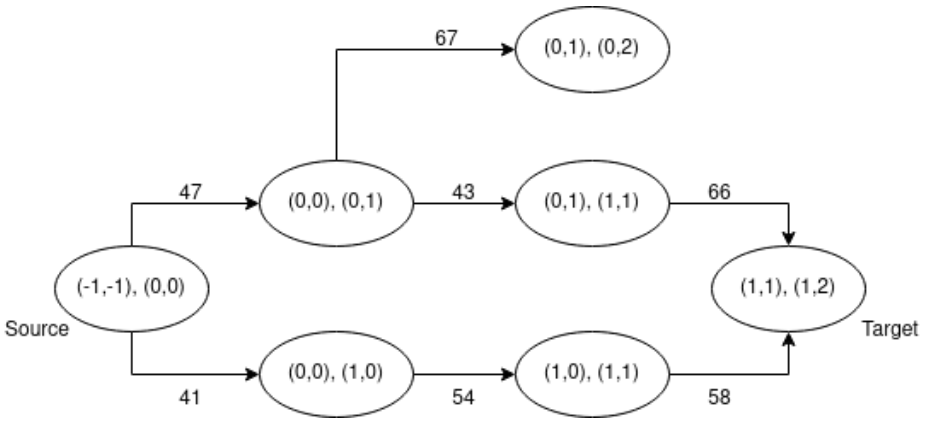

Figure 2

Instructions

1. Given a graph and turn costs, your task is to

   (a) Construct the pseudo-dual graph.

   (b) Run Dijkstra's algorithm on the pseudo-dual graph to find the shortest path bewteen a pair of nodes.

   Your code will be evaluated both on the intermediate dual graph, and the final path obtained. Construction of dual graph and implementation of Dijkstra's on the dual graph carry weightage of 50% each.

2. We have provided you the skeleton code containing interfaces and class files. The `Digraph` class loads graph data from a csv file and returns an adjacency list for the graph. You need to add your code in the respective functions in the `ShortestPathFinder.java` only. Do not modify the other files. You have to adhere to the interfaces provided.

3. Please read the comments in the skeleton code carefully and don't call or modify the mentioned methods in your code as this may break things while we will evaluate your submissions and you may get 0 grade due to that.

4. We have also provided sample input files and a driver code (`checker.sh`) to check your implementation on 3 mandatory test cases. Each test case is evaluated on both the dual graph construction and Dijkstra's algorithm.

5. Do not put any print statements while submitting the code. Make sure to remove all print statements in your implementation before submitting to avoid side effects.

6. The driver file, checker files and interface files will be replaced with the original ones.

7. Any sort of plagiarism will be dealt with strictly as per course policy.

References

[1] Tom Caldwell. On finding minimum routes in a network with turn penalties. *Communications of the ACM*, 4(2):107–108, 1961.