# CSCI 3901 Assignment 3

Due date: 11:59pm Friday, October 4, 2019 in Brightspace

## Problem 1

### Goal

Work with graphs.

### Background

You are part of the city's accessibility office in charge of helping people navigate the city. Google maps doesn't exist yet. Your job is to tell people how to get from their current location to a destination in the city by walking

We describe the city by its intersections. Each intersection has (x, y) coordinates (both integers) to tell you where the intersection is. We identify a part of a road by telling you the coordinates of the two intersections at either end of the road.

When someone calls you, they will give you the coordinates of their current intersection and will give you the coordinates of their destination intersection. Your response should be the sequence of intersections to cross, using only city roads, to reach the destination in the shortest distance. That sequence will begin with the coordinates of their current location and will end with the coordinates of their destination intersection.

### Problem

Write a class called "HalifaxMap" that accepts city map data, a current location, and a destination and then prints the sequence of intersections to traverse to get to the destination.

The class has at least 3 methods:

- Boolean newIntersection( int x, int, y ) – record a new intersection for the city. Return true if we acknowledge the intersection and we didn't know about it before. Return false otherwise.
- Boolean defineRoad( int x1, int y1, int x2, int y2 ) -- record the existence of a road from (x1, y1) to (x2, y2) in the city. Return true if this road is a new one for the map and has been added to requests to consider. Return false otherwise.
- Void navigate( int x1, int y1, int x2, int y2 ) – print to the screen the sequence of intersection coordinates to follow to go from (x1, y1) to (x2, y2) in the shortest distance while staying on roads. Print "no path" if you can't get from the location to the destination.

Dijkstra's algorithm for shortest paths in graphs is one possible approach to solving this problem. You are welcome to use others.

### *Inputs*

All the inputs will be determined by the parameters used in calling your methods.

### *Outputs*

The navigate( ) method produces output to the screen. It produces a list of coordinates, one per line. The x and y coordinates are separated by a tab character. The first line is the current location of the person who is calling you. The last line is the location of the destination.

### *Assumptions*

You may assume that

- You do not need to worry about round off error in the lengths of the paths when determining the shortest path. You can safely round the length of any road to an integer.
- The roads are straight lines between the intersections. A road from (x1, y1) to (x2, y2) has length square_root( $(x2 – x1)^2 + (y2 – y1)^2$ )

### *Constraints*

- You may use any data structures from the Java Collection Framework.
- You may not use a library package to for your graph or for the algorithm to do matchings on your graph.
- If in doubt for testing, I will be running your program on bluenose.cs.dal.ca. Correct operation of your program shouldn't rely on any packages that aren't available on that system.

### *Notes*

- You will need a class to store each vertex of the graph. Following that, you will need to choose whether to store the graph as an adjacency matrix, as an adjacency list, or as an incidence matrix. Your best bet will likely be an adjacency list.

### *Marking scheme*

- Documentation (internal and external), program organization, clarity, modularity, style – 4 marks
- List of test cases for the problem – 3 marks
- Ability to create the graph from the inputs and to return appropriately in error conditions –3marks
- Ability to provide a path between two intersections – 10 marks