CS 245 - Optional Assignment Fall, 2018

Optional Assignment - Graphs & Minimum Spanning Trees

The goal of this assignment is to practice the implementation of graphs, specifically focused on an algorithm for determining a minimum spanning tree for a graph. For this assignment, the correctness of the data structures is measured by how well your implementation conforms to the given specification.

This is an optional assignment. You are not required to submit a response for this assignment. See “Grading” for how this assignment may raise your grade.

Background

In class, we discussed the graph data structure as one which contains vertices and edges connecting pairs of vertices. Of the implementations we discussed, this assignment requires an weighted, undirected graph. Only three graph functions are required, as shown described in Table 1 below and defined by the Graph.java interface. (See below.) While the size of the graphs used in this assignment as well as the wording may suggest the use of an adjacency matrix, you are free to implement your graph using an adjacency list if you wish.

Function Signature	**Usage / Notes**
<<constructor>> (int vertices)	Constructs and returns a graph with the number of vertices passed as the argument. Vertices have IDs, numbered 0, 1, …, vertices-1. No edges exist between vertices at instantiation.
void addEdge (int v1, int v2, int w)	Adds an undirected edge or weight w between two vertices.
int getEdge (int v1, int v2)	Returns the weight of the edge between vertices v1 and v2. May return 0 or a negative number if such an edge does not exist.
int createSpanningTree ()	This function performs the following: a) Creates the minimum spanning tree from the source graph. b) Removes any edges in the graph which are not in the minimum spanning tree. c) Returns the weight of the minimum spanning tree.

Table 1: Graph Functions

Requirements (Process)

Requirement 1: Get the files you need:

1. Download this zip file from https://drive.google.com/file/d/1Q3O0Jj70ZJHTqAhvTRrBUMhNgt5MZHfB/view?usp=sharing
2. Unzip these files in the appropriate (source) directory. When unzipped, you should see two files: `OptioinalAssignmentTest.java` and `Graph.java`

Requirement 2: Add to the code in order to make it run. Specifically, your implementation must implement a class called `GraphAdjMatrix.java`[1]. This implementation must contain at least the functions listed in Table 1.

When the implementation is correct, the output will indicate a starting point for the implementation grade.

Grading

The assignment generates text ("`Starting point for this assignment`") based on tests of the functions priority queue functions. Assuming your implementation of the minimum spanning tree also produces the correct output, your starting point for the grade is as indicated. Your final grade may be adjusted based on additional items including, but not limited to, code quality and adherence to the above requirements.

The grade for this assignment *may* be used to replace your lowest practice assignment grade or your lowest take-home assignment grade. This assignment will not lower your course grade. (If the grade for this assignment would result in an overall lower course grade, it will be ignored and your grade will be unaffected.)

Submission

You are required to submit one item for this assignment: Java class `GraphAdjMatrix.java` to complete the two requirements above. You may submit this class in one of two ways; either: A) create an archive (tar or zip) and submit that archive to Canvas; or B) Place the class on GitHub and submit the URL to this GitHub repository on Canvas.

[1] As discussed above, you are free to have this class use an adjacency list, despite the name.