CSCI 2270 – Data Structures

Instructors: Zagrodzki, Ashraf

Due on Sunday, Nov 1, 11:59PM

Assignment 8 - Graph

OBJECTIVES

1. **Applications of Breadth First Traversal**

Overview

In this assignment, you will apply BFT for finding connected buildings in a graph.

Graph Class

Your code should implement graph traversal for buildings. A header file that lays out this graph can be found in **Graph.hpp** on Canvas. *As usual, do not modify the header file. You may implement helper functions in your .cpp file if you want as long as you don't add those functions to the Graph class.*

Your graph will utilize the following struct:

```
struct vertex;

struct adjVertex{
    vertex *v;
};

struct vertex{
    string name;
    bool visited = false;
    int distance = 0;
    vector<adjVertex> adj;

};
```

void addVertex(string name);

➔ Add new vertex 'name' to the graph.

void addEdge(string v1, string v2);

➔ Make a connection between v1 and v2.

void displayEdges();

➔ Display all the edges in the graph.

Format for printing:

If we create a graph with the following structure

```
graph.addVertex("ATLS");
graph.addVertex("EC");
graph.addVertex("AERO");

graph.addEdge("ATLS", "EC");
graph.addEdge("AERO", "EC");
```

We print the edges in the following manner.

```
ATLS --> EC
EC --> ATLS AERO
AERO --> EC
```

The order of vertices printed is the same as the order in which they were added to the graph. Similarly, the order of vertices to the right of "-->" sign is the same as the order in which the corresponding edge was added to the graph.

void breadthFirstTraverse(string sourceVertex);

➔ Breadth first traversal from sourceVertex. Format for printing:

```
// for the source vertex in the graph
cout<< "Starting vertex (root): " << vStart->name << "->";
// for other vertex traversed from source vertex with distance
cout << n->adj[x].v->name <<"("<< n->adj[x].v->distance <<")"<< " ";
```

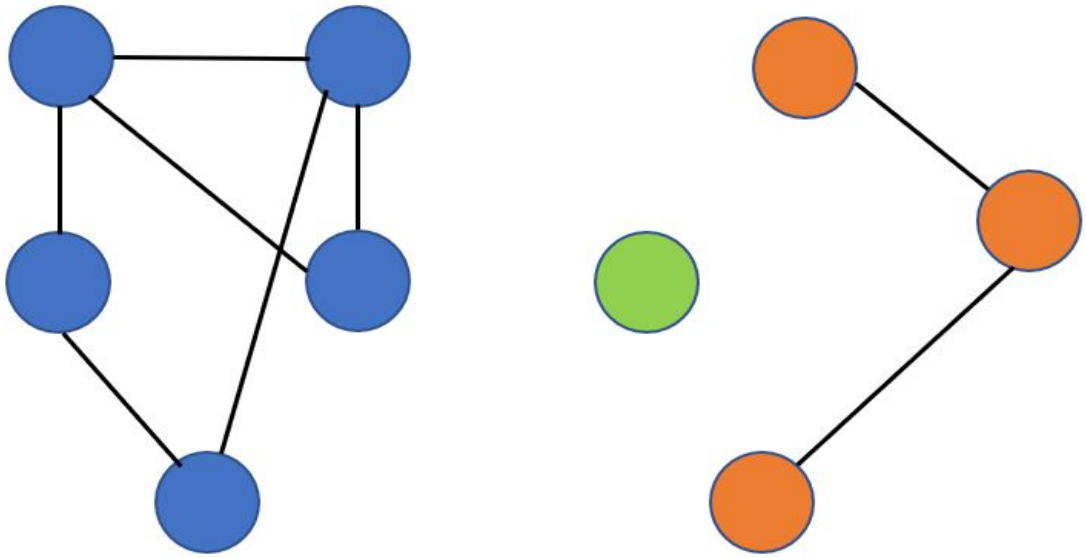
int getConnectedBuildings();

CSCI 2270 – Data Structures

Instructors: Zagrodzki, Ashraf

This method will provide the number of connected components in the graph i.e. the number of distinct subgraphs where no edges exist which connect vertices in two distinct subgraphs. In the following graph, the number of connected components is 3. The 3 components (distinct subgraphs) are highlighted below in different colors.

(If you are using BFT or DFT, re-define it as a helper function with no logging statements and call it in getConnectedComponents())



Please note that once you are done with your assignment on code runner you need to click on 'finish attempt' and the 'submit all and finish'. If you don't do this, you will not get graded."