Students:

This content is controlled by your instructor, and is not zyBooks content. Direct questions or concerns about this content to your instructor. If you have any technical issues with the zyLab submission system, use the **Trouble with lab** button at the bottom of the lab.

10.35 Programming Assignment 5 - Graph, BFS, Graphviz (BST, Summer 21 B)

Due Saturday Aug. 28, 2021 at 11:59 PM

Collaboration Policy Reminder

You may not use code from any source (another student, a book, online, etc.) within your solution to this PROGRAM. In fact, you may not even look at another student's solution or partial solution to this PROGRAM to use as a guide or otherwise. You also may not allow another student to look at any part of your solution to this assignment. You should get help on this assignment by coming to the instructors' office hours or by posting questions on Slack. Do not post your code on Slack.

Source files

You can download all of the source files for this assignment on the Canvas Programming Assignment 5 link on Canvas.

Problem Definition

In this assignment you will build a graph (graph is unweighted and directed) from an input file specification. Once you have correctly generated the graph, your program will perform a Breadth First Search (BFS) on an instance of a graph setting each vertex's distance value. Finally, your program will output the BFS distance annotated graph as a .dot file named **graph.dot** that can be viewed in the Graphviz web viewer. http://webgraphviz.com/

Tasks

- Read a graph specification from a file. Construct the corresponding graph in memory. The graph must be encoded using adjacency lists. You may use STL list or vector classes where needed.
- Run Breadth First Search (BFS) on the graph representation (adjacency lists) and

Graph Structure

A framework organization of your program's files/classes is given below. Substitute additional STL data structures as needed. For instance, you may use an STL pair and/or map. You may implement each class in a single file (all member functions inline).

Graph.H

- **vertices** - A vector containing all of the vertices in the graph. The start vertex is the first node in the adjacency list.
- **void build_graph(ifstream& ifs)** - Reads the graph input file and instantiates a graph object. Invoke from within an overloaded Graph constructor which has has an ifstream parameter. Call this overloaded Graph constructor from within your main function.
- **void bfs()** - Breadth First Search. Via a BFS traversal, this function should assign each individual vertex's distance to the number of hops that the vertex is from the start vertex. The start vertex (label) is the **first node listed in the input file input.txt**.
- **void output_graph()** - writes the graph object to an output file named **graph.dot** that can be viewed in Graphviz.

Graph.H

```
/*
 * cs10c_sum21
 * Graph.H
 */


#ifndef GRAPH_H_
#define GRAPH_H_

#include <vector>
#include <list>
#include <queue>
#include <fstream>
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include "Vertex.H"
```

```
}
    void output_graph(){ /* implement output_graph */ }
    void bfs(){ /* implement bfs */ }
private:
    vector<Vertex> vertices;
};

#endif /* GRAPH_H_ */
```

Vertex.H

- label - Individual vertex's label.
- int distance - Distance the vertex is from the start vertex. The graph is unordered so all distances between two adjacent vertices are 1.
- neighbors - An STL list of the vertices adjacent to an individual graph vertex instance. The integer value is the neighboring vertex's position in the graph object's member vector vertices.
- (Optional) use a vector of neighbors - Can be used as a substitute for an STL list to store references to an individual vertex's neighbors.

Vertex.H

```
/*
 * cs10c_sum21
 * Vertex.H
 */

#ifndef VERTEX_H_
#define VERTEX_H_

#include <string>
#include <list>
```

- The first line is the number of nodes N (**8** in the input.txt below) .
- The second line is the number of edges M (**20** in the input.txt below).
- Next comes the node labels in string format (**Lodi, Nile, Mondavi, Rivoli, Pyramids, Zurich, Marengo, Friedland** in the input.txt below). There will be **8** in total.
- The first node label, that follows directly the number of edges, is the source node for BFS (**Lodi** in the input.txt below).
- Finally the edges are listed. For each line, the source node is on the left and the destination node is on the right. For example, **Nile Rivoli** indicates that there is an outgoing edge from **Nile** that connects as an ingoing directed edge to **Rivoli**.

Let's look at an example node, **Nile**. We can see that **Nile** is on the left hand side of an edge below exactly twice: **Nile Rivoli** and **Nile Marengo**. This indicates that **Nile** has *two* outgoing edges or an out-degree of 2. Additionally **Nile** appears on the right hand side of an edge exactly *four* times: **Mondavi Nile**, **Zurich Nile**, **Rivoli Nile** and **Friedland Nile**. This indicates that Nile has 4 incoming edges or an in-degree of 4.

Sample Input File, input.txt

```
8
20
Lodi
Nile
Mondavi
Rivoli
Pyramids
Zurich
Marengo
```

Sample graph.dot file

Below is a very simple **graph.dot** file that represents the **input.txt** file above. Your **graph.dot** file outputted by your program must add the distance of each node to the graph. To see this file rendered in Graphviz right-click and open in new tab here.

```
/* Digraph
 * assn5
 * graph.dot
 * cs010_21sumB
 */

digraph G {

  // nodes
  Lodi;
  Nile;
  Mondavi;
  Rivoli;
  Pyramids;
  Zurich;
```

Below is the main.cpp your program will be tested with. You can download a copy of it below.

```
/*
 * cs10c_sum21
 * main.cpp
 */


#include "Graph.H"

int main(int argc, char* argv[]) {

    if(argc != 2) {
      cerr << "Usage error: expected executable  input" << endl;
      exit(1);
    }
```

included in a shortest path from the source node to each node that is reachable are clearly denoted and distinguishable from those edges not included in shortest path tree. Graphviz has an attribute style that can be set to bold or dashed, e.g., style=bold, style=dotted. **Extra credit may be awarded for particularly well structured and stellar appearing graphs drawn with Graphviz.**

NOTE: Your program will be run with the following command **$./a.out input.txt**

Turn-in Instructions and Demos

Submit all of your source code, a test main.cpp, in a single .zip file named assn5.zip to

Trouble with lab?