Northeastern University
College of Engineering
Department of Electrical & Computer Engineering

EECE7205: Fundamentals of Computer Engineering

Spring 2022 - Homework 5

Instructions

- For programming problems:
  - Your code must be well commented by explaining what the lines of your program do. Have at least one comment for every 4 lines of code.
  - You are **not** allowed to use any advanced C++ library unless it is clearly allowed by the problem. For example, you cannot use a library function to sort a list of data if the problem is asking you to implement an algorithm to sort the list.
  - At the beginning of your source code files write your full name, students ID, and any special compiling/running instruction (if any).
  - Your code must compile and tested with a standard GCC compiler (available in the CoE Linux server). before submitting the source code file(s) (do not submit any compiled/executable files):
    a. If your program does not compile with a GCC compiler due to incompatible text encoding format, then make sure the program is saved with Encoding Unicode (UTF-8). In visual studio, Save As -> Click on the arrow next to Save -> Save with Encoding -> Yes -> Unicode (UTF-8) -> Ok
    b. Compile using g++ -std=c++11 <filename>

- Submit the following to the homework assignment page on Canvas:
  - Your homework report developed by a word processor and submitted as <u>one</u> PDF file. For answers that require drawing and if it is difficult on you to use a drawing application, which is preferred, you can neatly hand draw the answer, scan it, and include it into your report. The report includes the following (depending on the assignment contents):
    a. Answers to the non-programming problems that show all the details of the steps you follow to reach these answers.
    b. A summary of your approach to solve the programming problems.
    c. The screen shots of the sample run(s) of your program(s).
  - Your well-commented programs source code files (i.e., the .cc or .cpp files).

  Do NOT submit any files (e.g., the PDF report file and the source code files) as a compressed (zipped) package. Rather, upload each file individually.

<u>Note</u>: You can submit multiple attempts for this homework, however, only what you submit in the last attempt will be graded. This means all required files must be included in this last submission attempt.

Problem 1 (100 Points)

Write a program that will help us find the shortest path between any two buildings in the Northeastern University Boston campus. Utilize the Dijkstra's Algorithm to implement the program.

To carry out this task, do the following:

1. Using any drawing software, design an undirected (i.e., roads are bidirectional) graph using the campus map available at: https://www.northeastern.edu/campusmap/printable/campusmap15.pdf
2. Only include 16 of the buildings in the area of the campus that is located inside *Huntington Avenue*, *Ruggles Street*, *Massachusetts Avenue*, and the *Orange Line* railroad. Select your buildings so that you have four buildings from each one of the following numbering range: 20's, 30's, 40's, and 50s.
3. The vertices in your graph that represent the buildings correspond to the main entrances of these building in the map. If the main entrance is not known, use the center point of the building. You will need also to add vertices to represent roads intersections. You need enough intersection vertices so that none of the generated shortest paths will guide the user to walk on a non-pedestrian area.
4. Create a text file to store your graph data, which includes: the total number of vertices followed by the data of the graph edges. For each edge, provide its <vertex1> <vertex2> <distance>. Here a vertex is entered as a building or an intersection number. Provide your own numbers to the intersections making sure that they are different than the used building numbers.
5. In your program use sequential indices (i.e., 0,1, 2, ...) to represent the vertices in the algorithm. However, use an appropriate data structure to map your program indices to the user-friendly building and intersections numbers as stored in the text file. Building numbers are the one shown in the map so that the user can only interface with your program using these numbers.
6. The edges in your graph represent the roads between the vertices. The weight of an edge is determined by the distance between its two vertices. These distances can be measured utilizing Google map as explained in: https://support.google.com/maps/answer/1628031
7. Only add edges to the vertex's direct neighbors on the map. A direct neighbor of vertex is another vertex on the map that represents a building or an intersection and can be reached without passing by any other building or intersection. Also, no need to consider the internal paths inside any building. Generally, you can utilize Google maps to help you identify which vertices/edges to include and which ones that should not be included.
8. Write your C++ program so that it starts by reading the graph text file you created and map the vertices user-friendly numbers to vertices indices. Do not hard code the graph vertices/edges in your program as your program should work with any graph by only changing the text file contents. Implement the graph using either an adjacency-list or an adjacency-matrix.
9. When it runs, your program will read from a user the start and destination buildings (using the campus map numbers to refer to these buildings). The program will then use Dijkstra's algorithm to provide the user with the sequence of the buildings numbers and intersections over the shortest path between the start and destination.

10. Verify your program shortest path results with the corresponding results given by Google maps for the “walking” directions.

In your homework report, include the following:

- The drawing of the undirected graph you created in step (1). Have the graph labeled with the vertices numbers that match the numbers on the campus map. Also, have the edges labeled with the distances you calculated from google map.
- In addition to including the screen shots of at least two sample runs of your program, include copies of the graph you created in step (1) on which you need to highlight the shortest path generated by your sample runs. Also, include screen shots of the Google map walking directions correspond to your sample runs.
- Make sure to submit with your homework the text file that has your graph data.

Programming Hints

- The following C++ code reads integers from a text file and stores them in an array:

```
#include <iostream>
#include <fstream>
#include <stdexcept>
#define maxints 100
using namespace std;

int main() {
	ifstream inf;
	int count = 0;
	int x;
	int list[maxints];

	inf.open("c:\\temp\\ints.txt");
	if (inf.fail())
	{
		cerr << "Error: Could not open input file\n";
		exit(1);
	}
	//activate the exception handling of inf stream
	inf.exceptions(std::ifstream::failbit | std::ifstream::badbit);

	while (count < maxints) { //keep reading until reading maxints or
				  //until a reading failure is caught.
		try { inf >> x; }
		//Check for reading failure due to end of file or
		// due to reading a non-integer value from the file.
		catch (std::ifstream::failure e) {
			break;
		}
		list[count++]=x;
	}
		for(int i = 0; i < count; i++)
			cout << list[i] << endl;

		inf.close(); //Close the file at the end of your program.
		return 0;
	}
```