Northeastern University
College of Engineering
Department of Electrical & Computer Engineering

EECE7205: Fundamentals of Computer Engineering

Homework 5

Instructions

- For programming problems:
  - Your code must be well commented on 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 or to use advanced data structures, like vectors. Only basic library functions (like string functions) and data structures (like arrays).
  - At the beginning of your source code files write your full name, students ID, and any special compiling/running instruction (if any).
  - Make sure your compile and run using a standard GCC compiler (available on the CoE Linux server) before submitting the source code file(s) (do not submit any compiled/executable files):
    - If your program does not compile with a GCC compiler due to incompatible text encoding format, then make sure the program is saved with the encoding Unicode (UTF-8).
    - Compile using g++ -std=c++11 <filename>

- **Deliverables**: 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):
    - Answers to the non-programming problems that show all the details of the steps you follow to reach these answers.
    - A summary of your approach to solving programming problems.
    - 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. Include only 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 they are distributed evenly within that area.
3. The vertices in your graph that represent the buildings correspond to 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. Assign **sequential numbers** (0, 1, 2, … etc.) to all your graph vertices.
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 number representing a building or an intersection. You will need another text file to map every one of your chosen building numbers to the building character code as shown at the bottom of the above map (e.g., SH for Shillman Hall). This mapping text file should include in its first line, the number of buildings in your graph, which in our case is 16.
5. In your program use the graph array sequential indices (i.e., 0,1, 2, ...) so that an array index corresponds to the same vertex number stored in the graph text file.
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 files you created and store the graph data in an adjacency-matrix data structures. Also read the mapping text file to an appropriate array to map the buildings character codes to their vertices numbers in your graph. 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.
9. When it runs, the program reads from a user the start and destination buildings (using the buildings character code). The program uses Dijkstra's algorithm to provide the user with the sequence of the buildings character codes and intersections numbers over the shortest path between the start and destination. Optionally, give the intersections some meaningful codes, like the buildings, and hence you will need to include these codes in the mapping text file.

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, building character codes, and optionally the intersections character codes. 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 files that have your graph data and vertices mappings.

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;
	}
```