Northeastern University
College of Engineering
Department of Electrical & Computer Engineering

EECE 2560: Fundamentals of Engineering Algorithms

Spring 2020 - Homework 6

Instructions

- For programming problems:
    1. 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.
    2. 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.
    3. At the beginning of your source code files write your full name, students ID, and any special compiling/running instruction (if any).
    4. Test your code on the COE Linux server before submitting it:
        a. If your program does not compile in the COE server due to incompatible text encoding format, then before uploading your source code file to the server make sure it 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>

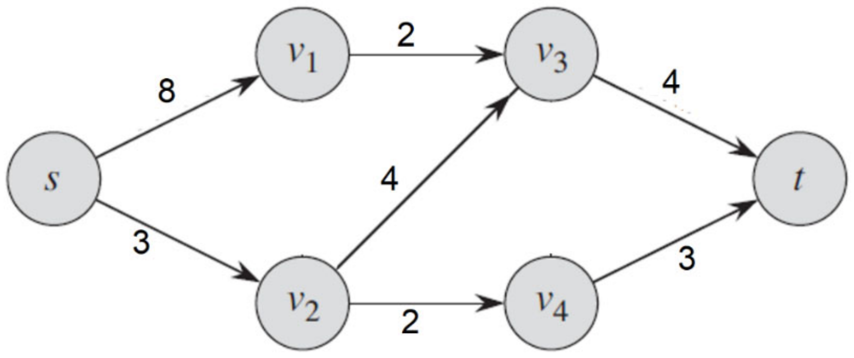
- Submit the following to the homework assignment page on Blackboard:
    1. Your homework report developed by a word processor (no handwritten or drawn contents are acceptable) and submitted as one PDF file. The report includes the following (depending on the assignment contents):
        a. Answers to the non-programming problems.
        b. A summary of your approach to solve the programming problems.
        c. A summary of the skills you acquired and challenges you faced by implementing the programs.
        d. Your recommendations of extension to the programming problems.
        e. The screen shots of the sample run(s) of your program(s)
    2. 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.

Note: Yon can submit multiple attempts for this homework, however, only your last submitted attempt will be graded.

Problem 1 (20 Points)

a) What is the minimum cut of the flow from ***s*** to ***t*** in the following flow network?



b) Apply the Ford-Fulkerson Algorithm on the above network to find the maximum flow from ***s*** to ***t***. Draw all residual networks needed to apply the algorithm

Problem 2 (80 Points)

Write a program that 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 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. Each vertex in your graph should represent the main entrance of each 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.
4. 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
5. Only add edges to the vertex's direct neighbors on the map (i.e., do not include non-essential edges). Also, no need to consider the internal paths inside any building.
6. Create a text file to store your graph data, which includes: the number of vertices followed by the data of the graph edges. For each edge, provide its <vertex1> <vertex2> <distance>
7. Write your C++ program so that it starts by reading the graph text file you created in the previous step and then implement this graph using either an adjacency-list or an adjacency-~~matrix.~~
8. 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 shortest path between these two buildings.
9. 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;
      }
```