CSE3080-3: Data Structures (Spring 2022)
Homework 3: Minimum Spanning Tree

Handed out: June 2, Due: June 21, 11:59PM (KST)
No Late Submission for this project

1. Introduction

In this project, we will implement **Kruskal's algorithm**, which calculates a minimum spanning tree of a given graph. We will write a program that reads a file which contains information of the graph and output the minimum spanning tree in the given format.

There can be many different implementation, but your implementation should have a time complexity of **O(e log n)**, where e is the number of edges and n is the number of vertices in the given graph.

As described in the lecture slides, you can achieve O(e log n) by using **min heap** and **disjoint set trees**.

2. Requirements (Read Carefully!)

(1) You should write a single C program, named **fp1**.

(2) The program takes one command-line argument, which is the input file name. For example, the user will run the program like this:

```
./fp1 input.txt
```

The first argument is the name of the input file.

(3) If the number of command-line arguments is not 1 (zero or more than 1), then you should print a usage string on the display and exit the program. The usage string looks like this:

```
usage: ./fp1 input_filename
```

(4) If the input file does not exist, your program should display an error message on the screen and terminate. The error message should be

```
The input file does not exist.
```

(5) The input file format is like the following. Here we assume the graph is an **undirected graph**.

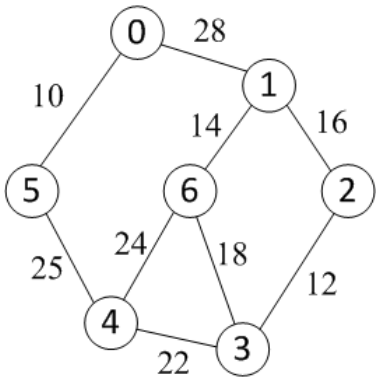
- The first line contains the number of vertices in the graph. We assume that the vertex number **starts from 0**. Thus, if there are 7 vertices in the graph, the vertex numbers are 0 to 6.

- The second line contains the number of edges in the graph.
- From the third line, each line contains three numbers, each separated by a space. The first two numbers are end points of an edge, and the third number is the weight of the edge.

An example input file looks like this.

```
7
9
0 1 28
0 5 10
1 2 16
1 6 14
2 3 12
3 4 22
3 6 18
4 5 25
4 6 24
```

The corresponding graph for this input will be:



(6) Your program should generate an output file named "fp1_result.txt". The format of the output file is as follows.

- From the first line, each line should print the edge that is included in the minimum spanning tree. Specifically, the line should contain three numbers: source vertex, destination vertex, and the weight. The edges should be printed in the ascending order of their weights.

- After you print all the edges included in the MST, then you should print two more lines. The next to last line should print the total cost of the minimum spanning tree. That is simply the sum of weights of all the edges included in the MST.

- The final line should say either "CONNECTED" or "DISCONNECTED". If the minimum spanning tree connects all vertices, then the line should print "CONNECTED". If the minimum spanning tree does not connect all vertices, then the line should print "DISCONNECTED".

An example output file for the input file described above is here. Note that the edges are printed in the ascending order of weights.

```
0 5 10
2 3 12
1 6 14
1 2 16
3 4 22
4 5 25
99
CONNECTED
```

(7) At the end of the program, you should print the following lines on the screen.

```
output written to fp1_result.txt.
running time: 0.073622 seconds
```

The actual running time printed will be different on each run. In order to print out the running time, use the function `clock()`.

Once you implement the program, compare the execution time of your program with the given binary file (fp1_example) using the given input file "input_large.txt". For this particular input file, the running time on the "input_large.txt" input file should be similar to the given binary file "fp1_example".

(8) Similar to other homework, you should write a Makefile. The TA will build your code by running "make". It should create a binary file, "fp1".

※ Test-run the given binary code and write your program so that it produces the same result as the given binary code.

※ Your implementation should support up to **10,000 vertices and 50,000,000 edges**.

3. Submission

You should submit your source codes and the Makefile. (You do not need to submit compiled binary files.) Combine your files into a zip file named cse3080_fp1_**20190001**.zip. The red numbers should be changed to **your student ID**. Submit your files on the cyber campus.

4. Evaluation Criteria

(1) Your program should compile without problem and produce the binary file.

(2) If the user does not give correct number of arguments, your program should print the usage message on the display and terminate.

(3) If the input file does not exist, your program should correctly print an error message on the display and terminate.

(4) When the user correctly executes the program with a command-line argument, your program should produce the result file “fp1_result.txt”.

(5) The output must be correct.

(6) The running time on the "input_large.txt" input file should be similar (on the Colab machine) to the given binary file "fp1_example". Points will be deducted if the running time of your program is much larger (more than five times) than the time taken by "fp1_example".

5. Notes

- You must write your own code. You may discuss ideas with other students but must not copy their work. Similarly, you can find ideas on the Internet, but must not copy the code from the sources obtained from the Internet.

- Our department uses a software that detects duplicate codes. Since the software uses an assembly code level detection, just changing the variable names will not bypass the software. Make sure you write your own code from the scratch. Then, you will have no problem.

- Explained earlier, but your program should compile and run on the **colab**. Make sure you check before you submit your work. The TA will download your code, run “make” and execute your binary files to check if they produce correct outputs.

6. Late Policy

Due to the grade submission deadline, no late turn-ins will be accepted for this project.