

# EECS 2510 – Non-Linear Data Structures and Programming in C++

Lab 5 – Due Sunday, December 12, 2021 at 11:59 PM

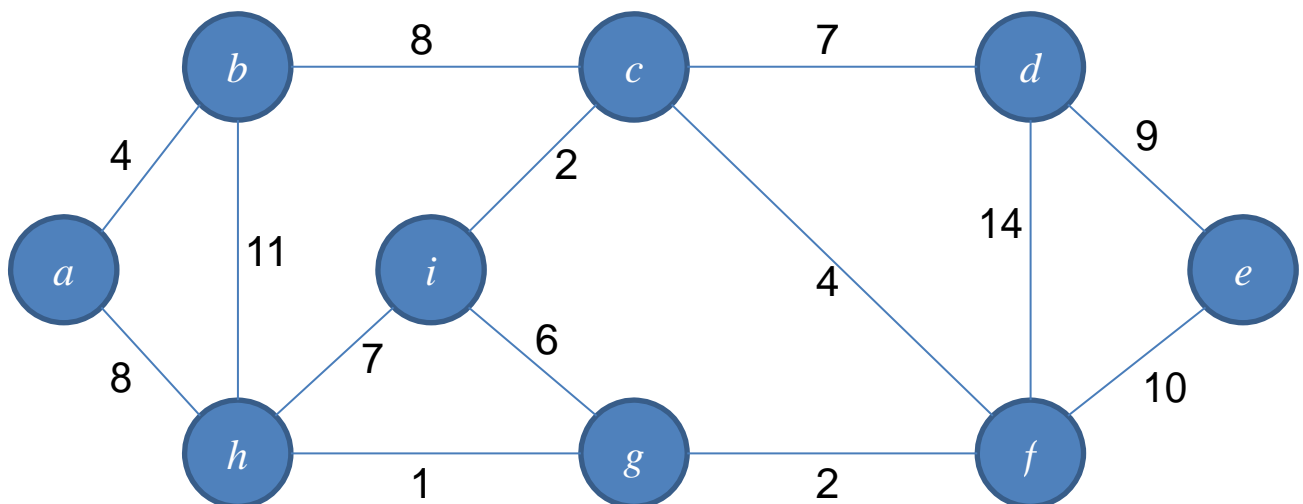
For this programming lab, you are to implement both of the algorithms we covered for producing minimum spanning trees for a weighted, non-directed graph – Kruskal’s algorithm, and Prim’s algorithm (see Chapter 23).

Your input will be in the form of a text file. The first line in the file will be an integer  $N$ . Following the line with  $N$  will be  $N$  *more* lines, each with the name of a node. Node names will be one- or two-character strings, one per line. These will be á la Excel’s column headers

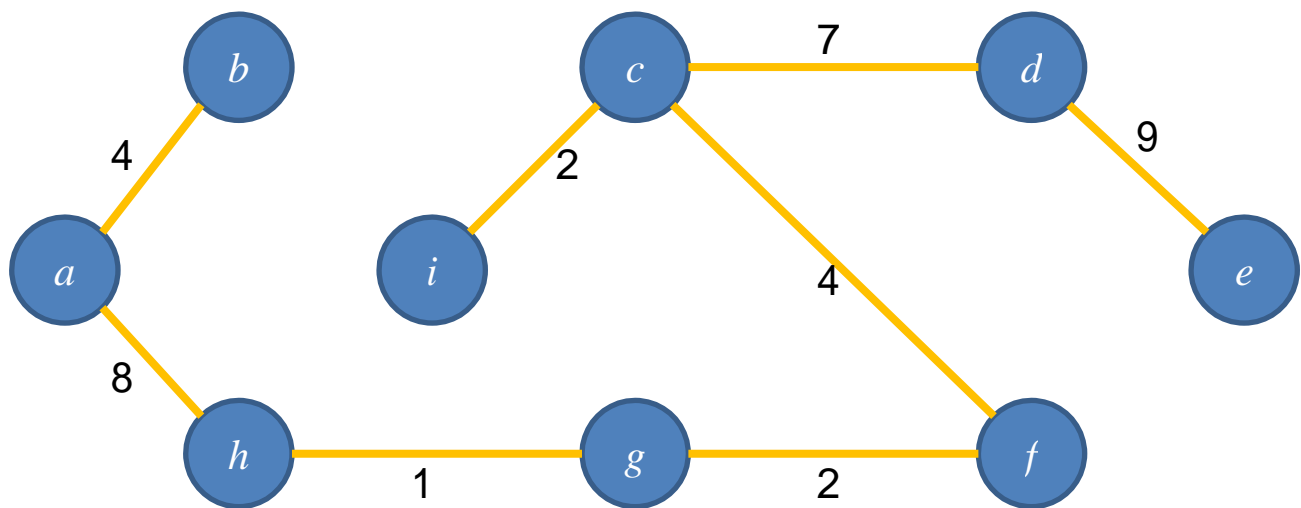
Following the node names will be an adjacency matrix consisting of  $N$  rows of  $N$  items per row. Each will be a positive edge weight (potentially, a floating point value, so count on using `double` as the data type for your adjacency matrix). The items on each row will be space-delimited, so the `>>` stream operator will work for you to do the file input. This isn’t supposed to be an ugly parsing project!

Your program is to read the node names and the adjacency matrix and then compute a minimum-weight spanning tree for the graph, producing two sets of output, one each for Kruskal and another for Prim. The output for each algorithm will consist of exactly  $N+1$  lines: One line to tell which algorithm the output is for (“Kruskal” or “Prim”), a double (the MST’s weight), followed by the  $N - 1$  edges that make up the tree, one-per-line. The edges will be specified in your output (see below for an example) as `node1<dash>node2<colon><space>weight`, with the edges sorted alphabetically by node1, and then by node2. Node1 will be alphabetically less than node2. A line containing 32 dashes will separate the output from Kruskal’s algorithm from the output of Prim’s

Below is the graph for which we walked through both algorithms in class:



After running through Kruskal's algorithm, we had this tree, which has a weight of 37:



The input file for this graph would look like this:

```

9
a
b
c
d
e
f
g
h
i
0 4 0 0 0 0 0 8 0
4 0 8 0 0 0 0 11 0
0 8 0 7 0 4 0 0 2
0 0 7 0 9 14 0 0 0
0 0 0 9 0 10 0 0 0
0 0 4 14 10 0 2 0 0
0 0 0 0 0 2 0 1 6
8 11 0 0 0 0 1 0 7
0 0 2 0 0 0 6 7 0

```

Kruskal's output for this graph would be:

```

37
a-b: 4
a-h: 8
c-d: 7
c-f: 4
c-i: 2
d-e: 9
f-g: 2
g-h: 1

```

Note that the edges are named by the alphabetically-lower vertex first (we will never refer to edge d-c; rather, it would be edge c-d).

The list of edges is sorted by the first vertex (a, a, c, c, c, d, f, and g above), and then by the second vertex (a-b precedes a-h, and c-d precedes c-f, which precedes c-i).

I suggest you try your program on several graphs that exercise (exorcise?) various possibilities. Work them out on paper, and then build a text file containing their representation. Use the test case(s) to prove to yourself that your program works properly. I will test your program with a graph I have pre-created.

**IF YOU RUN INTO PROBLEMS, AND HAVE MADE A GOOD ATTEMPT AT SOLVING THEM, E-MAIL ME, RATHER THAN LOSING TIME LOST IN THE WILDERNESS. DON'T WAIT TO GET STARTED ON THIS. "I WAS BUSY WITH MY OTHER CLASSES" WILL NOT BE A VALID EXCUSE. THIS IS END-OF-SEMESTER CRUNCH TIME FOR ALL OF US. THERE WILL NOT BE ANY EXTENSIONS OF ANY KIND FOR THIS ASSIGNMENT. I AM GIVING YOU ALL THE TIME I CAN AND STILL LEAVE TIME TO GET GRADES COMPUTED AND SUBMITTED TO THE REGISTRAR BY THE UNIVERSITY DEADLINE.**

To turn in your program, use 7-Zip ([www.7-zip.org](http://www.7-zip.org)) to create a compressed archive of your entire Visual Studio workspace (don't just send me your source and/or .exe files). Submit your 7-Zip archive to Blackboard.

As always, your code is to be yours and yours alone. Do not use code from any other source (including each other and the Internet), even as a reference. Failure to write this code from scratch will be considered a breach of academic integrity, and will be dealt with most severely.

Employ good programming habits. The Allman style for braces is strongly encouraged. DOCUMENT YOUR CODE. Some of you have been very lax about comments this semester. Blocks of code (and functions / methods) should have a block comment introducing what you're about to do (and how you're about to do it), and the individual steps should have enough line comments to walk someone new to your code through your thought process. You don't have to write a textbook explaining the algorithm, but for someone even vaguely with the algorithm, they should be able to read your code and understand the steps. Your comments should be a tutorial for the uninitiated; not just an outline for those already familiar with your code

You will need a priority queue for this program. I suggest you implement it with a heap (and all that entails – see Chapter 6). You may NOT use any STL classes that implement a queue, priority queue, set, hash, map, etc – those are for you to code. A heap and some linked lists should be all you need.

Your set operations (for Kruskal) may be readily implemented as a linked list of linked lists (just a suggestion). If you want to work that way, feel free. If there's some other data structure you would like to use, that's up to you, too, but you are required to stick to C++'s built-in primitives, plus whatever you can build out of them (no template objects like vectors or collections to do the work FOR you).

Name your program SPAN.exe, and have the file name it reads the graph information from be the lone command-line parameter. Therefore, you may start your program from the command line with something like this:

**SPAN D:\GraphData.txt**