CS 218 DATA STRUCTURES
ASSIGNMENT 6 & 7
SECTION B, C, D, and G
Fall 2020

DUE: 1st Jan 2020

NOTE: Late submissions will not be accepted

TO SUBMIT: Commented and well-written structured code in C++ on the google classroom. An undocumented code will be assigned a zero.

PROBLEM BACKGROUND

During the Pandemic CoVID, the classes and labs in universities are mostly conducted online. The load on the internet service and Local area network (LAN) has significantly increased. It is observed in NUCES that our LAN is not adequately designed, and it is not using the available resources to its fullest. As you people are familiar with the fact that the NUCES Lahore campus consists of various buildings so the LAN needs to connect the classrooms, labs, faculty offices in these buildings efficiently.

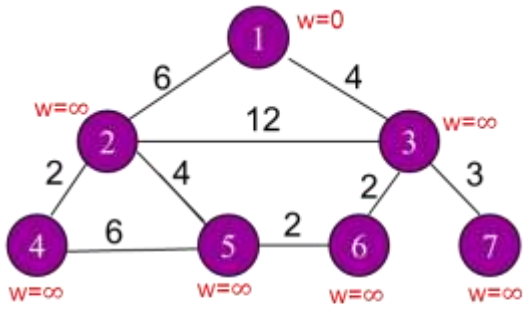
We have modeled this problem to a graph, where nodes represent computers in classrooms, labs, and offices and the networking wires that connect them are represented by edges. The edges have weight that indicates the length of the wire needed to connect two nodes. Your task is to take this graph as input and design a LAN network, FLAN that covers all the nodes with minimum wire length and do so efficiently.

Input
1. Input a **weighted undirected graph** G(V, E). 2. A vertex v in V represents a computer in NUCES. Each computer has a unique ID and location (office, lab or classroom). As it is a weighted graph so each vertex has a weight that is initialized to a max int it can hold. 3. The edge has a weight that indicates the length of the wire needed to connect two computers.

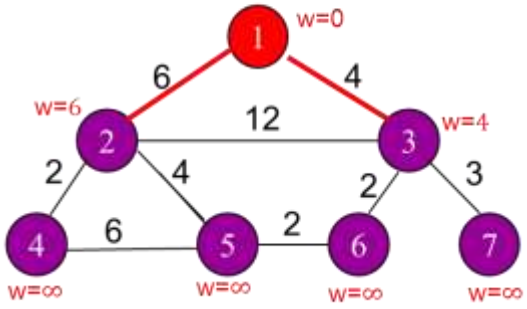
Your task is to take a graph as input and design a LAN network, FLAN that covers all the nodes and do so efficiently using the following idea

1. Input graph G(V,E), in an adjacency list representation from a file
2. Select a vertex v of Graph G randomly and sets its weight = 0
3. Build a MinHeap on V,vertices using weight of the vertices as key in heap.
4. While MinHeap is not empty, do
    a. ExtractMin vertex v from MinHeap
    b. For each adjacent vertex **u** of v
        i. If **u** is in MinHeap and its weight is more than weight of edge (v,u), then
            - update the weight of u with weight of edge (v,u) by using decreaseKey function in MinHeap so that its position in the heap is properly updated.

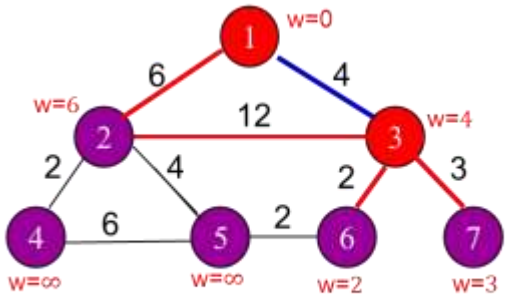
Example:



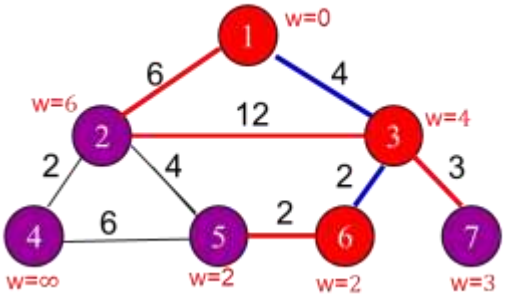
Vertex 1 is selected, W(1)=0
Weight of all other vertices = ∞
MinHeap contain all vertices



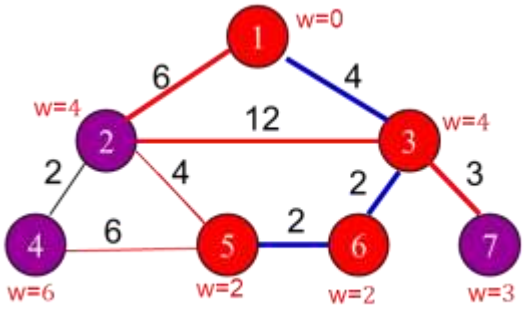
Remove MinVertex v=1 from heap
For each adjacent vertex **u** of v update weight in graph and Minheap



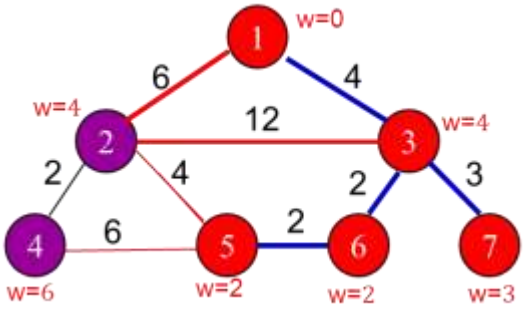
Remove Min weight Vertex v from heap
For each adjacent vertex **u** of v=3 in minheap update weight if necessary



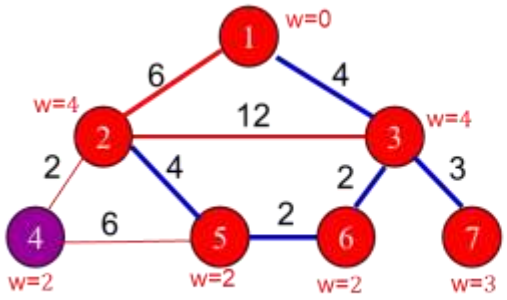
Extract Min weight Vertex v from heap
For each adjacent vertex **u** of v=6 in minheap update weight if necessary



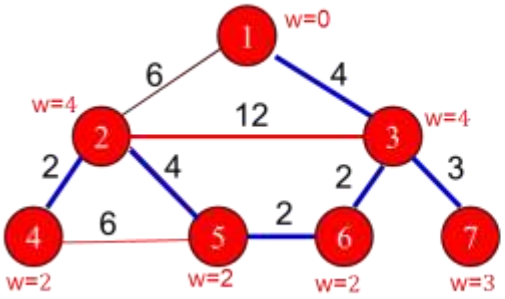
Extract Min weight Vertex v from heap
For each adjacent vertex **u** of v=5 in minheap update weight if necessary



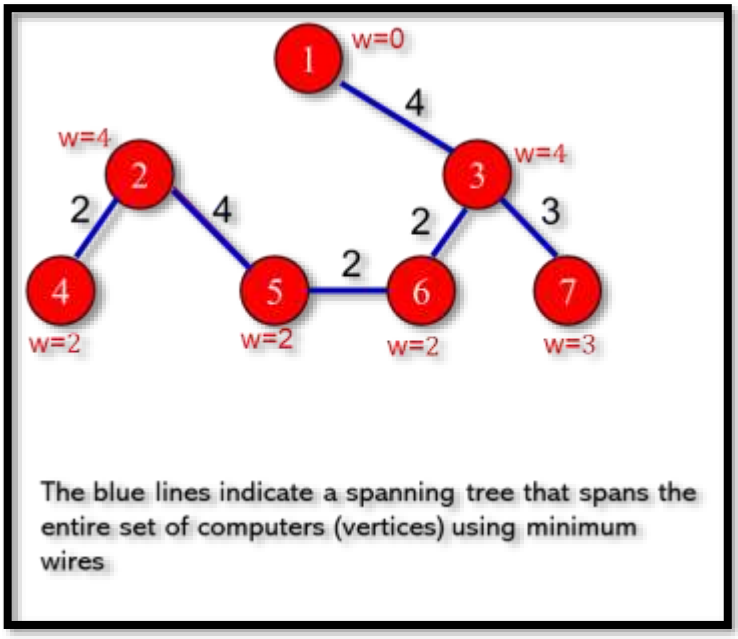
Extract Min weight Vertex v from heap
For each adjacent vertex **u** of v=7 in minheap update weight if necessary



Extract Min weight Vertex v from heap
For each adjacent vertex **u** of v=2 in minheap update weight if necessary



Extract Min weight Vertex v from heap
For each adjacent vertex **u** of v=4 in minheap update weight if necessary



Input File Format

Total number of vertices
Vertex info in the form (vertex ID, type)
Total number of edges
edge info (starting vertex id, ending vertex id), weight

```
7
1, office1
2, lab1
3, lab 4
4, lab 9
5, lab 10
6, office2
7, office6
9
1,3 4
1,2 6
2,4 2
2,5 4
2,3 12
3,6 2
3,7 3
4,5 6
5,6 2
```

Assignment Requirements

Create a class for graph node
Create a class for the graph
- Use adjacency list representation to maintain edge information
- Use STD list for adjacency list representation

Build a Minheap class with functions

1. Constructor
2. Destructor
3. BuildHeap
4. ExtractMin
5. DecreaseKey

Keep information about the position of each vertex in the Heap. You can use an array of size |V| and store the index of the place where vertex exists in the heap

Create a Menu in the main for the following functionality

1. Input a graph from a file
2. Save a graph in a file
3. Add a vertex in existing graph G
4. Add an edge in existing graph G
5. Print the set of edges that covers all the vertices using the above algorithm (for above example you will print the blue edges.)