Home Work 4.

Ques. 1. Design a data structure that has the following properties (assume n elements in the data structure, and that the data structure properties need to be preserved at the end of each operation):

- Find median takes O(1) time
- Insert takes O(log n) time.

Do the following:

1. Describe how your data structure will work.
2. Give algorithms that implement the Find-Median() and Insert() functions.

Ans. Consider a array which is sorted.
Form a max-heap from the first half of the array.
Second half of the array should be changed to min-heap.

Algorithm:

// Initialize length of max heap to zero.
// Initialize length of min-heap to zero
len_max_heap = len_min_heap = 0

1. Find_Median():
   if len_max_heap > len_min_heap:
      return root of max_heap.

elif len_min_heap > len_max_heap:
    return root of min_heap

else:
    return (root of max heap + root of min_heap) / 2.

2. Insert (new_element):
    if new_element < max_heap_root:
        Insert new_element to max_heap.
        length_max_heap++;
    else:
        Insert new_element to min_heap;
        len_min_heap++;

    if ~~length~~ len_min_heap - len_max_heap > 1:
        extract root element from min heap and insert into max_heap.
        len_min_heap--;
        len_max_heap++;
    elif len_max_heap - len_min_heap > 1:
        extract root of max_heap and insert into ~~max~~ min_heap.
        len_max_heap--;
        len_min_heap++;

→ Since root of ~~max~~ heap represents median, extracting it takes O(1) time.
→ Assuming array is sorted, insertion takes O(log n) time.

Ques. 2. Let us say that a graph $G=(V,E)$ is a near tree if it is connected and has most $n+k$ edges, where $n=|V|$ and k is a constant. Give an algorithm with running time $O(n)$ that takes a near tree G with cost on its edges, and returns a minimum spanning tree of G. You may assume that all edge costs are distinct.

Ans. $n=|V|$

Algorithm:

Perform a DFS on the graph $k+1$ times.

check this in each dfs

if a cycle is detected:
Remove the edge with maximum value on this cycle.

→ The graph will still remain connected if a edge with maximum weight is removed from a cycle.

→ edges in minimum spanning tree $= |V|-1$.

→ Run time of dfs $= O(|V|+|E|)$
here edges $= n+k = |V|+k$
(∵ at most it can have $n+k$ edges.

hence running time $O(n)$.
(∵ k is constant.

Ques 3. You are given a minimum spanning tree T in the graph $G = (V, E)$. Suppose we remove an edge from G creating a new graph G_1. Assuming that G_1 is still connected, devise a linear time algorithm to find a MST in G_1.

Ans. Algorithm:

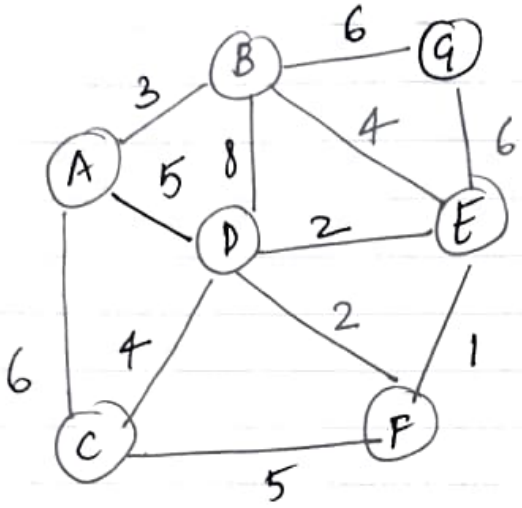
If Check if the weight of the removed edge is more than each weight in T.
MST in G_1 remains T.
(∵ largest edge in cycle can't be in MST).

If If the removed edge is not present in T,
run dfs from any vertex in G_1 to check if the graph is connected.
MST remains T.
// Running dfs takes $O(n)$ time

If Say a edge u–v is removed.
len1 = Run dfs from u and check if the graph is connected. —(1)
len2 = Run dfs from v and check if the graph is connected —(2)
If len1 > len2:
graph returned by (2) is MST in G_1.
// len1 and len2 are sum of edge costs.

→ dfs takes linear time

Quest. Considering the following graph G:



1. In graph G, if we use Kruskal's Algorithm to find the MST, what is the third edge added to the solution? Select all correct answers.
   a. E-F
   b. D-E
   c. A-B ✓
   d. C-F
   e. D-F

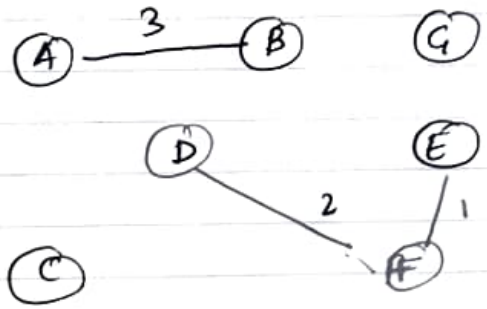
2. In graph G, if we use Prim's Algorithm to find MST starting at A, what is the second edge added to the solution?
   a. B-G
   b. B-E ✓
   c. D-E
   d. A-D
   e. E-F

3. What is the cost of the MST in the Graph?

a. 18
b. 19
C. 20 ✓
d. 21
e. 22

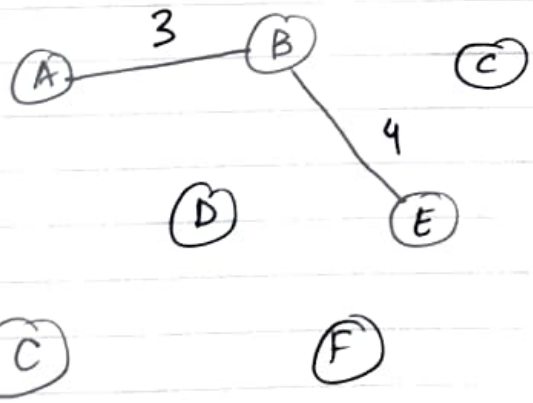
Ans. 1.) C. A – B.
2.) b. B – E
3.) c. 20

1.)



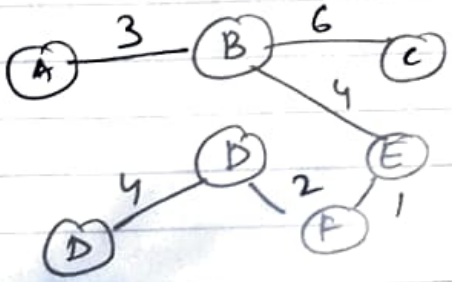
iteration
1 - EF
2 - DF
3 - AB
(∵ DE will form cycle, it won't be added)

2.)



Iteration
1 - AB
2 - BE

3.)



3 + 6 + 4 + 1 + 4 + 2
20

Ques 5. A new startup FastRoute wants to route information along a path in a communication network, represented as a graph. Each vertex represents a router and each edge a wire between routers. The wires are weighted by the maximum bandwidth they can support. FastRoute comes to you and asks you to develop an algorithm to find the path with maximum bandwidth from any source s to any destination t. As you would expect, the bandwidth of a path is the minimum of the bandwidths of the edges on that path; the minimum edge is the bottleneck. Explain how to modify Dijkstra's algorithm to do this.

Ans Algorithm:

Initially $S = \{s\}$ and $d(s) = 0$
for all other nodes $d(u) = \infty$
while $S \neq V$
Select a node $v \notin S$ with at least one edge from S for which
$d(v) = \max_{e=(u,v) : u \in S} (\min(d(u), l(e)), d(u))$
Add v to S
end while

→ this should be maximum

→ Since we have to find the edge with maximum weight of minimum-weight edge in the path so that we get maximum bandwidth.

Ques 6. A network of n servers under your supervision is modeled as an undirected graph $G=(V,E)$ where a vertex in the graph correspond to a server in the network and an edge models a link between the two servers corresponding to its incident vertices. Assume G is connected. Each edge is labeled with a positive integer that represents the cost of maintaining the link it models. Further, there is one server S (call its corresponding vertex as S) that is not reliable and likely to fail. Due to a budget cut, you decide to remove a subset of the links while still ensuring connectivity. That is, you decide to remove a subset of E so that the remaining graph is a Spanning tree. Further, to ensure that the failure of S does not affect the rest of the network, you also require that S is connected to exactly one other vertex in the remaining graph. Design an algorithm that given G and the edge costs efficiently decides if it is possible to remove a subset of E, such that the remaining graph is a spanning tree where S is connected to exactly one other vertex and (if possible) finds a solution that minimizes the sum of maintenance costs of the remaining edges.

Ans. Connected graph
Server S is not reliable, likely to fail.

Algorithm :-

- Say the Graph $G = (V, E)$.
- Remove the faulty server S and call the new graph G_1. Remove all edges adjacent to G_s in G.
- Find a minimum spanning tree using Prims on G_1. Call this tree G_1'.
- Check all edges of S in G, add the minimum cost edge to G_1'.
  ↳ Say l
- If no minimum spanning tree is found on G_1, return the error message that cannot find minimum spanning tree.

Ques7. Given a connected graph $G=(V,E)$ with positive edge weights. In V, s and t are two nodes for shortest path computation, prove or disprove with explanation:

1. If all edge weights are unique, then there is a single shortest path between any two node in V.

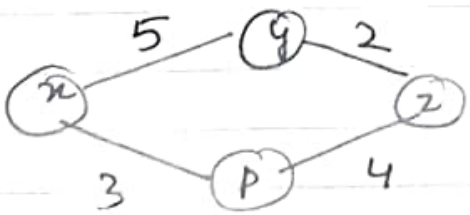
2. If each edge's weight is increased by k, the shortest path cost between s and t will increased by a multiple of k.

3. If the weight of some edge e decreases by k, then the shortest path cost between s and t will decrease by at most k.

4. If each edge's weight is replaced by its square, i.e., w to w^2. then the shortest path between s and t will be the same as before but with different costs.

Ans. 1. False.
2. False
3. False
4. False.

1.



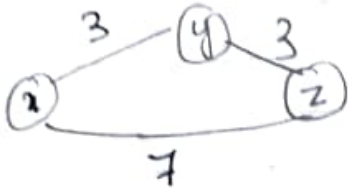
Two shortest paths:

$x - y - z \rightarrow 5+2 \Rightarrow 7$

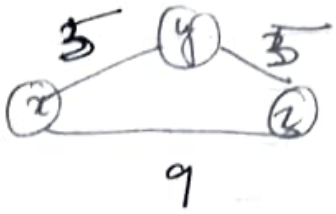
$x - p - z \rightarrow 3+4 \Rightarrow 7$

False

2.



$(k=2)$
$+2$ →



Shortest path $x-z$
$3+3$
$=6$

False

Shortest path $x-z$
is 9.

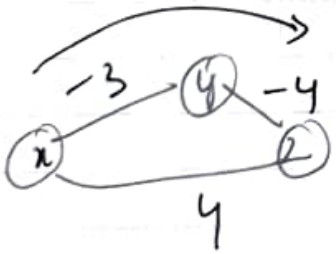
$\rightarrow 9-6=\underline{3}$.

3 is not a multiple of 2.

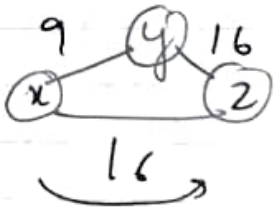
3. If sum of edge weights become negative, Dijkstra's will end up in infinite loop and hence will not decrease by at most k.

False

4.



square →



Shortest path $x-z$
$(-3)+(-4)$
$=(-7)$

False

Shortest path $x-z$
is 16

Shortest path from $x-z$ has changed.

Ques8. Consider a directed, weighted graph G where all edge weights are positive. You have one star, which allows you to change the weight of any one edge to zero. In other words, you may change the weight of any one edge to zero. Propose an efficient method based on Dijkstra's algorithm to find a lowest-cost path from node s to node t, given that you may set one edge weight to zero.

Ans. Algorithm:

- Find shortest path from source vertex s to all vertex.
- Find shortest path from t to all other vertices after reversing edge.
- Iterate over all edges in G, and store the distance from source to destination (Including that edge) say $x - y$.
  $s - x - y - t$.
- Find the path with minimum length and set the value of that edge say (x', y') to 0.
- Now $s - x' - y' - t$ is the required path.
- This algorithm requires 2 invocation of dijkstras and hence the time complexity remain same. $O(E \log V)$