(11)

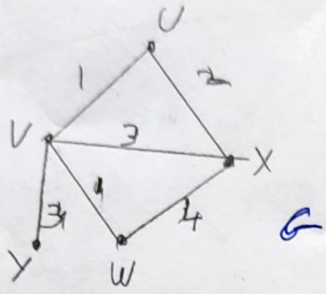
Although the edge weight of every edge is increased by R, the shortest path spanning tree T remains the same.

∴ R is increased on edge weight of every edge in G.

∴ The least weighted edges of G still remain the least weighted edges

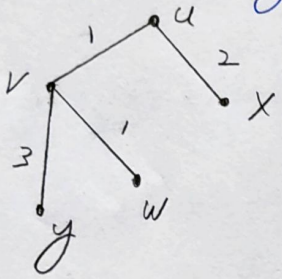
∴ There is no change in T.

example :-



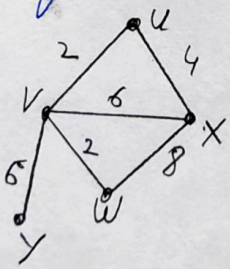
G

Shortest path spanning tree T from G is

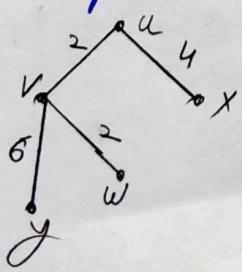


let k be increment on edge weight of every edge of G

let k = 2



Now the shortest path spanning tree is



(12) Let G be an undirected connected graph but not a Tree

Then G must contain a cycle C.

Suppose C consists of k nodes as

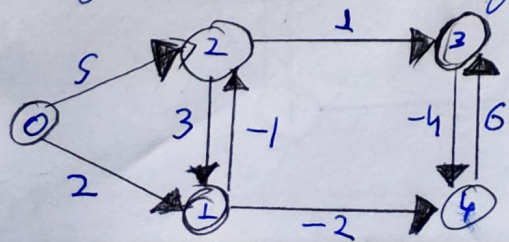
$V_1 \rightarrow V_2 \rightarrow \cdots \rightarrow V_k \rightarrow V_1$

Now in DFS tree, node $V_1, V_2 \cdots V_k$ will all be on the same path from root to a leaf.

But, in BFS tree, node $V_1, V_2 \cdots V_k$ will form atleast two ~~leaves~~ branches, branching from the node first visited.

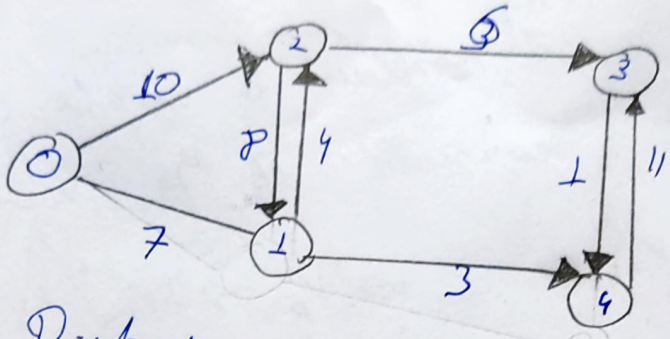
∴ BFS and DFS will produce the same tree T if and only if $G == T$.

(13) Let G be a directed graph with negative edge ~~costs~~ costs for some edge



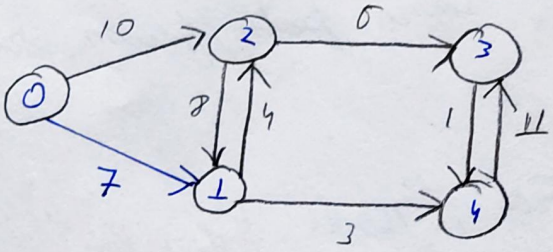
The most minimum cost is −4

So lets add 5 to every node

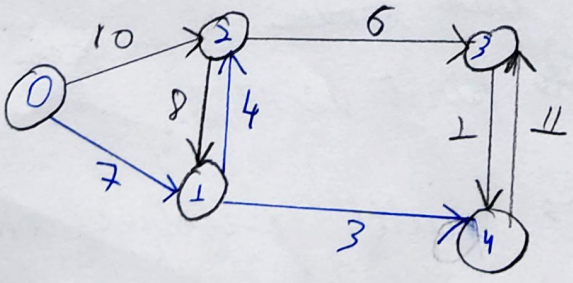


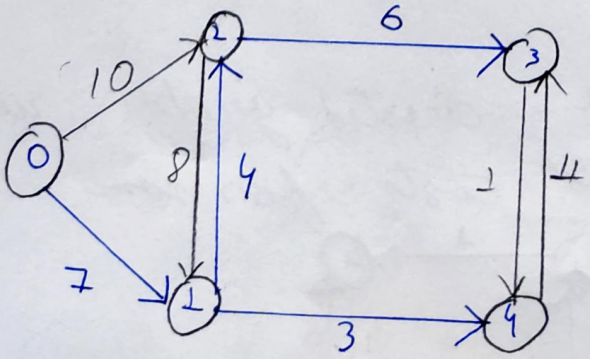
Applying Dijkstra's Algorithm.

Starting with node 0



$\begin{pmatrix} 0 & \cdot \\ 0 & \end{pmatrix}$ 10 > 7

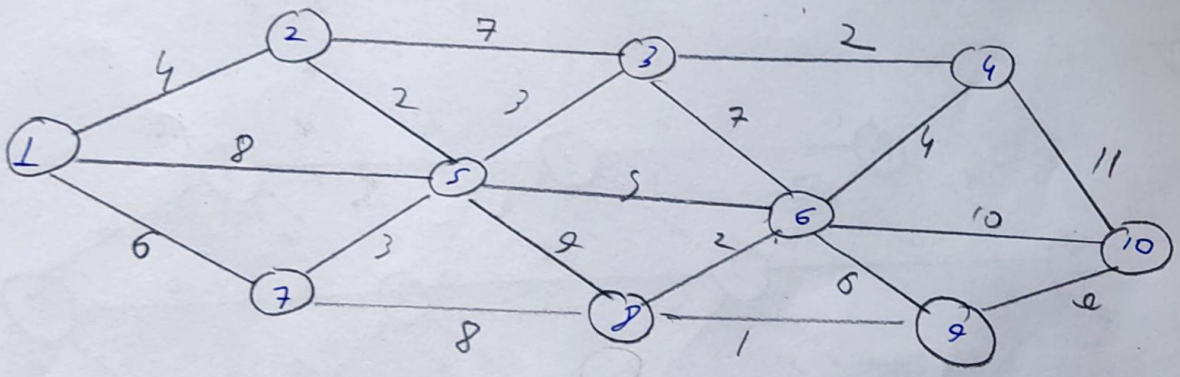




Dijkstra's Algorithm fails on a graph with negative weighted edges.

Since, Dijkstra's Algorithm follows greedy approach. It doesnot reconsider a visited node even if shorter path exists.

(14)

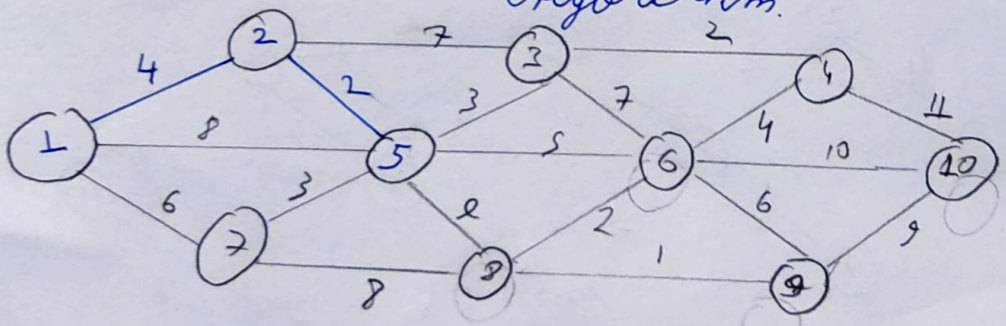


Let k be the ~~number of edges in the~~ shortest path ~~spanning~~ tree from source to the farthest node

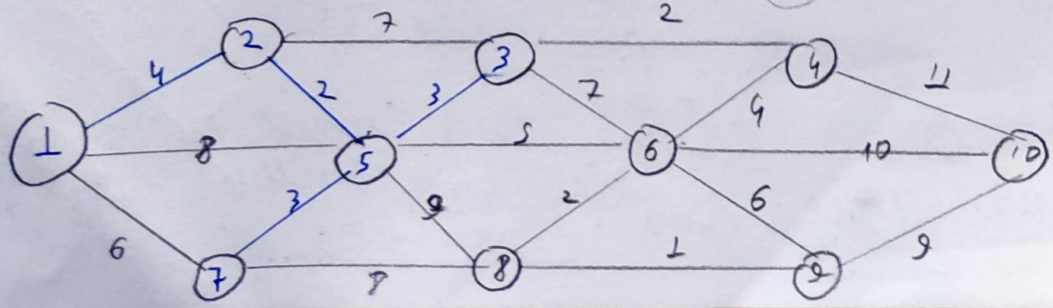
$\therefore k \leq |E|$

Applying Dijkstra's Algorithm.

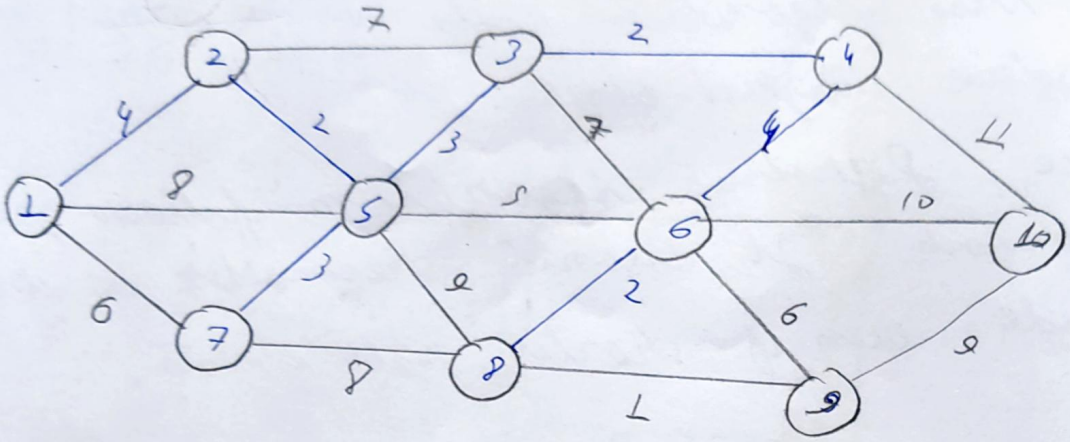
I)



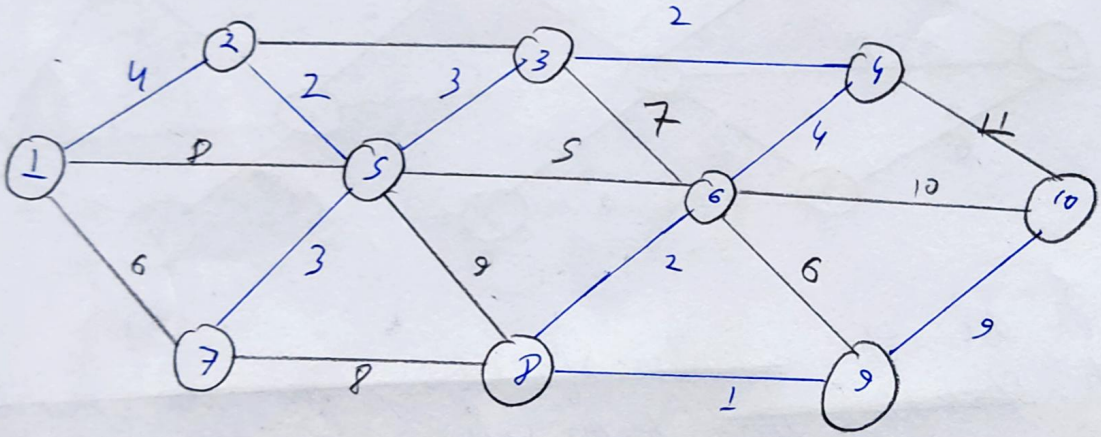
II)



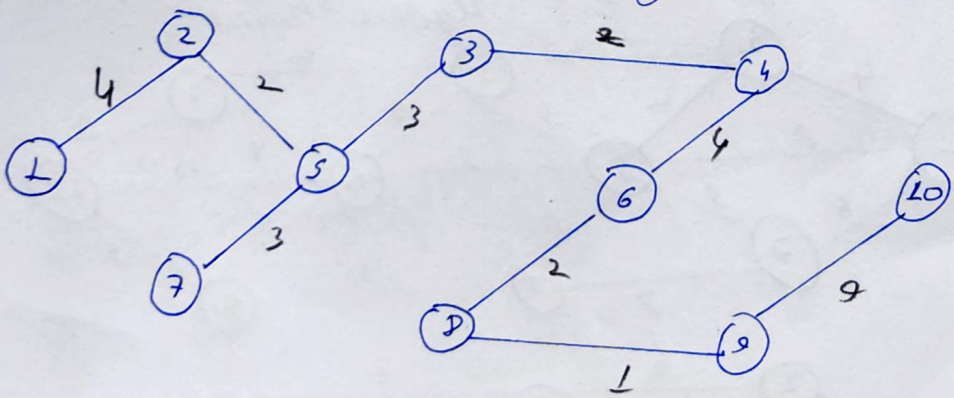
III



IV



∴ our shortest path spanning tree is



∴ $k = 6$ (∴ our root is 5 & farthest node is 10)

our tree has 4 colors.

(15) Kruskal's Algorithm

```
1.  A ← ∅
2.  for each vertex v ∈ V[G]
3.      do Create-Set(v)
4.  sort the edges of E by nondecreasing weight w
5.  for each edge(u,v) ∈ E in order by nondecreasing weights
6.      if Find-Set(u) ≠ Find-Set(v)
7.          A ← A ∪ {(u,v)}
8.          Union(u,v)
9.  return A;
```

Correctness of Kruskal's Algorithm

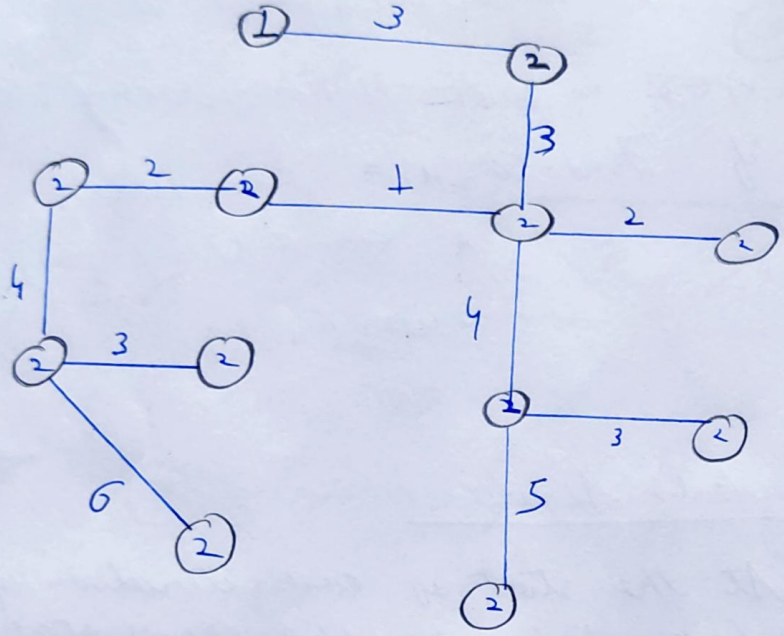
Loop invariant: At the start of each iteration of for loop at line 2, the array A[0 : v(G)] does not contain vertex v.

Initialization: before the first iteration, the array A[0 : v(G)] is empty.
So, the sub array does not contain ~~value~~ vertex v.

Maintainance: At line 3, the loop creates a partition set for every vertex in V(G).
∴ v ∈ V(G), there exists a partition set for vertex v.

Termination: The loop terminates when all the vertex of graph G has been assigned a particular set.

The algorithm selects the edges with least weights and connect those two vertex. by the end, we get a Minimum Spanning tree



∴ This is our Minimum spanning tree

steps :-

1. connect all vertex with edge w = 1
2. connect all vertex with edge w = 2
3. connect all vertex with edge w = 3
4. connect all those vertex with edge weight 4 & not forming a cycle.
5. connect all those vertex with weight 5 and 6 & not forming any cycle
6. Stop when all the edges are connected. We got our tree