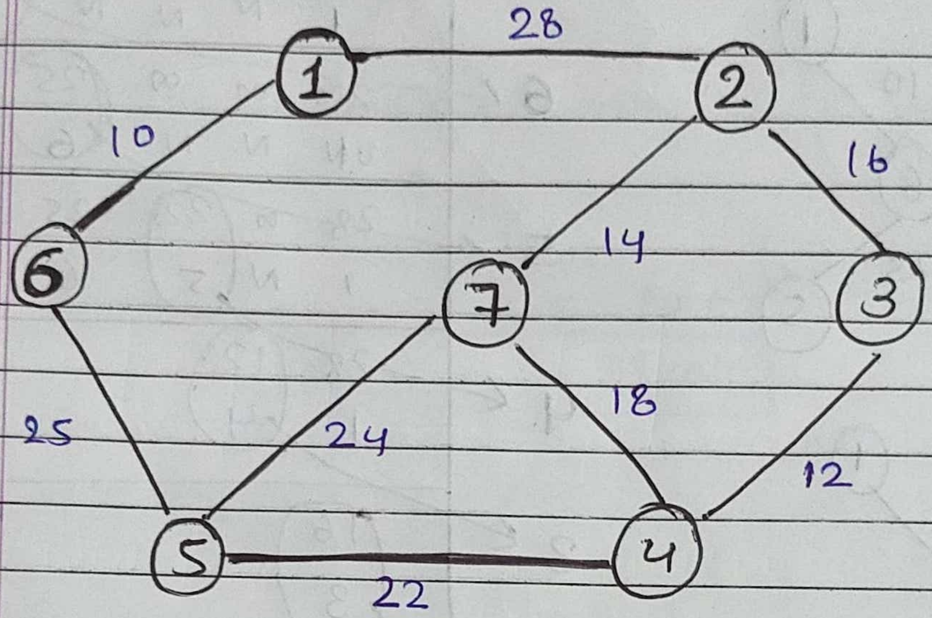
Date:
Page:

Assignment 2

Que1 What do you mean by minimum spanning Tree? Write and Explain algorithm for minimum spanning tree for a given graph using Prim's and Kruskal's both with source node 1.



A minimum spanning Tree is a subset of Edges of an undirected, weighted graph that connects all vertices together without any cycles and has minimum possible total edge weight.

Kruskal's Algorithm

Step 1 Sort all the Edges of a graph in non-decreasing order By weight.

Step 2 Take the minimum weight Edge and add 2 MST (Minimum spanning Tree).

Step 3 Take the next minimum weight Edge and add 2 MST

Step 4 Take the next minimum weight Edge and add 2 MST if cycle is not formed.

Step 5 Repeat Step 4.

Implementation Algorithm

MST_KRUSKAL (G, w)

{g = graph
w = weightage
A = set

1. $A = \phi$
2. for each vertex $v \in G.V$.
3. Make-set (v)
4. sort the Edges of G.E into non-decreasing order by weight w.
5. for Each Edge $(u, v) \in G.E$, in non-decreasing order By weight.
6. If find-set(u) $\neq$ find-set(v)
7. $A = A \cup \{u, v\}$
8. UNION (u, v)
9. return (A)

Prim's Algorithm

```
MST-PRIM (G, w, r)
for Each V ∈ G.V
    u.key = ∞
    u.π = NIL
    r.key = 0
    Q = G.V
    while Q ≠ 0
    u = Extract - Min(Q)
    for Each V ∈ G. Adj [V]
    if V ∈ Q and w (u, v) < V. Key
        V. π = u
    V. Key = w (u,v)
```

Step1 find the adjacent of root node and take minimum from them.

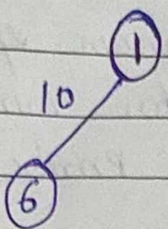
Step2 find the adjacent of new node and take minimum from them and previous.

Step3 find the adjacent of new node and take minimum from them and previous if cycle is not formed

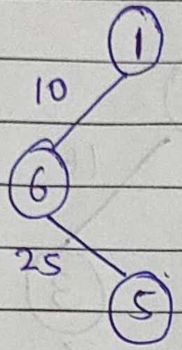
Step4 Repeat the step 3.

Using prim's Algorithm

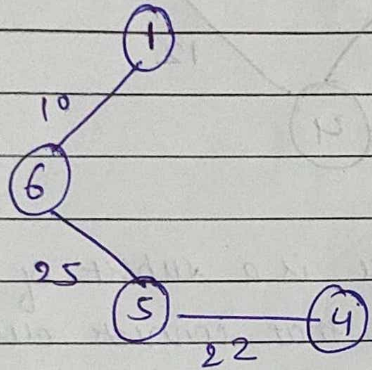
Step1



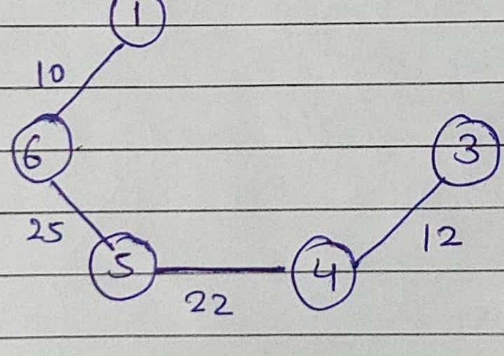
Step2



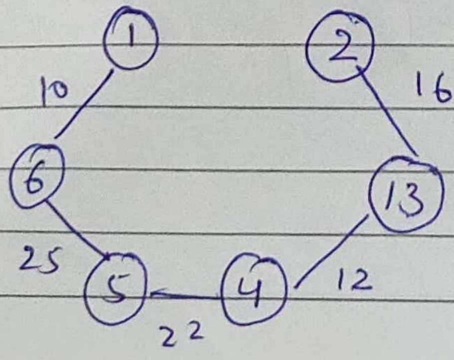
Step3



Step4

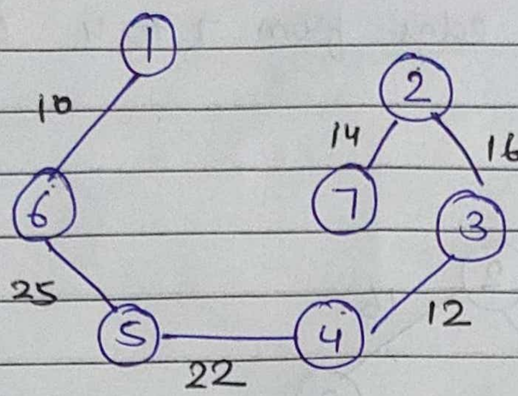


Step5



	1	2	3	4	5	6	7
	0	∞	∞	∞	∞	∞	∞
		N	N	N	N	N	N
1		28	∞	∞	∞	10	∞
		1	N	N	N	1	N
6		200	∞	∞	25	10	∞
		0N	N	N	6	01	N
5		28	∞	22	25	10	24
		1	N	5	6	1	5
4		28	12				18
		1	4				4
3		16					18
		3					4
2	28						14
	2						2
7			18	24			
			7	7			

Step6



Cost of Minimum spanning Tree = sum of all Edge weight

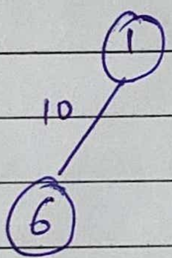
$$= 10 + 25 + 22 + 12 + 16 + 14$$

$$\boxed{\text{Cost} = 99 \text{ units}}$$

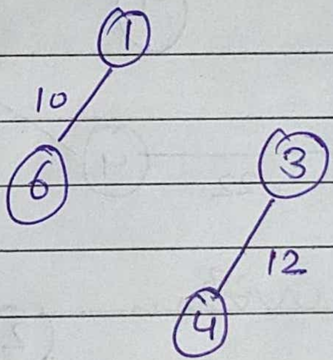
Using Kruskal Algorithm

We draw edges with increasing order of their cost.

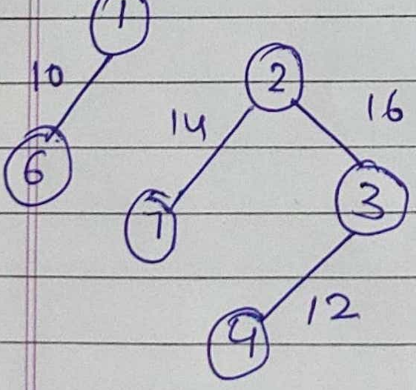
Step1



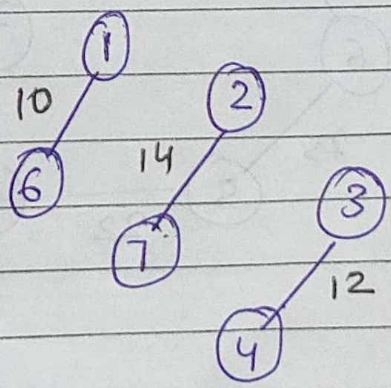
Step2



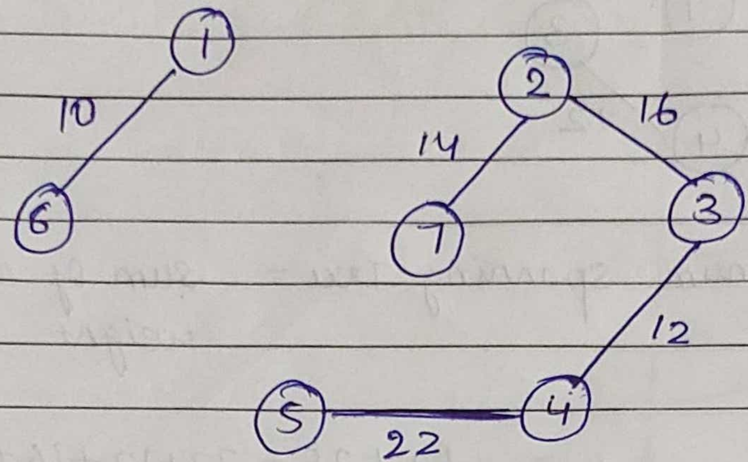
Step4



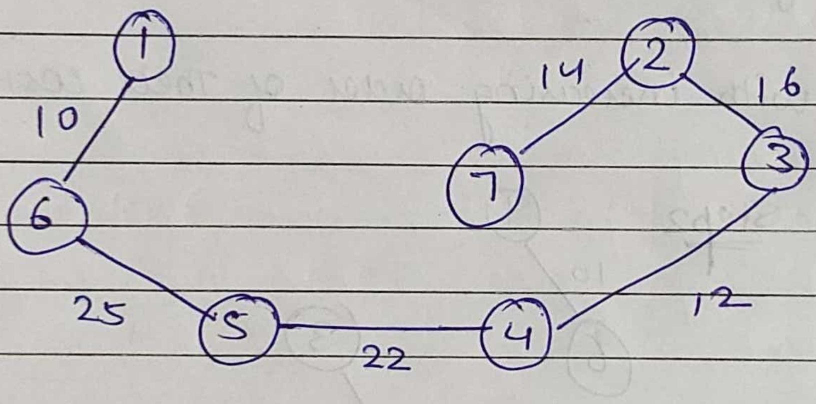
Step3



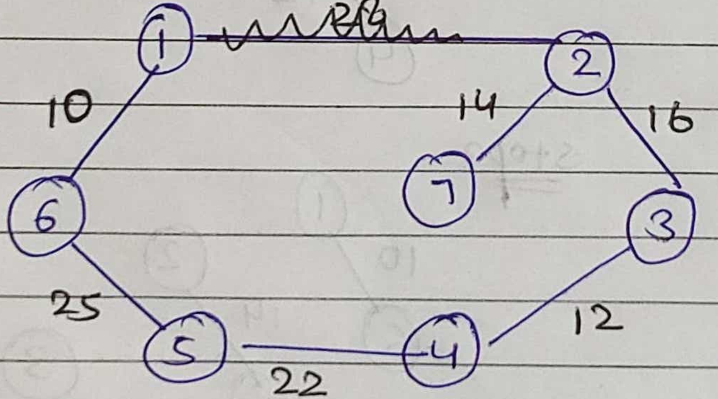
Step 5 We will not connect edge from 7 to 4 Because it is forming a loop



Step 6 Again we can't connect 5 to 7 it is forming loop



Step 7



Date:

Page:

Qw 2

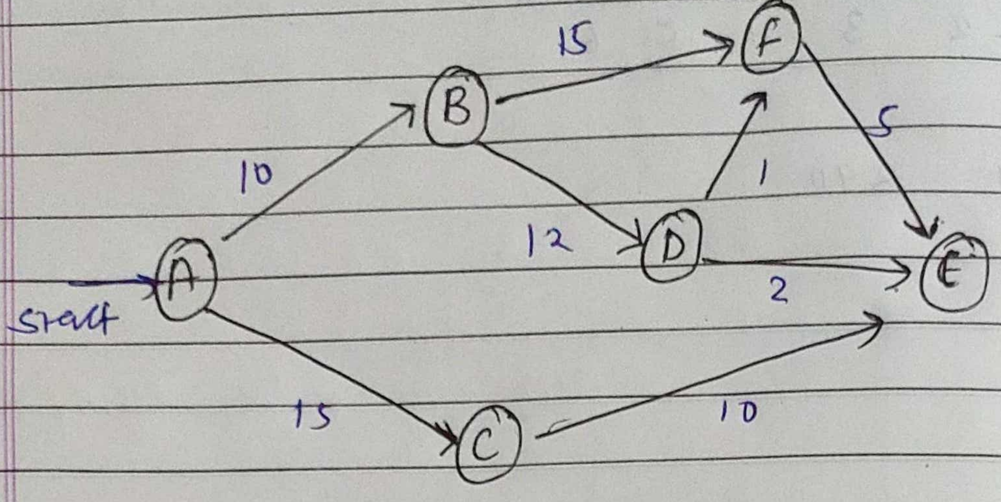
$\boxed{\text{Dijkstra's } (G, w, s)}$

Initialise - single source (G, s)
$P = \phi$
$Q = G.V$
while $Q \neq \phi$
u = Extract - Min(Q)
$P = P \cup \{u\}$
for each vertex $v \in G.Adj[u]$
Relax (u, v, w)

Initialise - single source (G, S)
1. for Each vertex $v \in G.V$
2. $v.d = \infty$
3. $v.\pi = NIL$
4. $s.d = 0$

Relax (u, v, w)
If $v.d > u.d + w(u,v)$
$v.d = u.d + w(u,v)$
$v.\pi = u$

Qu2



Step 1 Initialize single source & Q = {A, B, C, D, E, F}

	A	B	C	D	E	F
A	0	∞ N	∞ N	∞ N	∞ N	∞ N
B		10 A	15 A	∞ N	∞ N	∞ N
C			15 A	22 B	∞ N	25 B
D				22 B	25 C	25 B
F					24 D	23 D
E					24 D	

A) A → B → C → D → F → E

B) 24

C) A → B → D → E

Date:
Page:

Que 3 **Bellman-ford Algorithm**

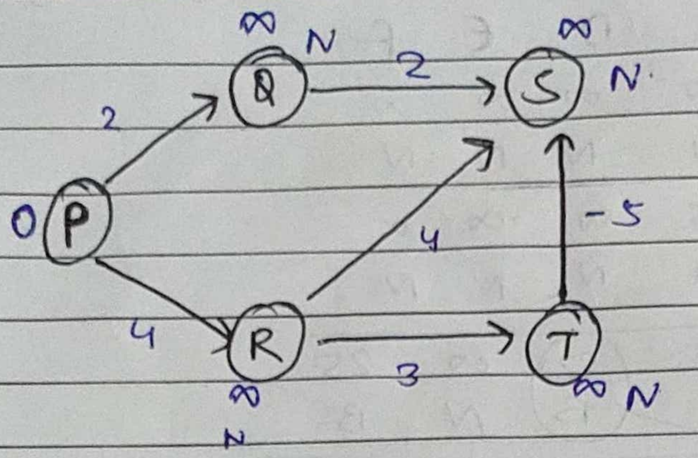
```
BF (G, w, s)
Initialise-single source (G, s)
for i = 1 to |G.V| - 1
for Each Edge (u, v) ∈ G.E
    Relax (u, v, w)
  for Each Edge (u, v) ∈ G.E
   If v.d > u.d + w(u, v)
    return false
   return True
```

```
Relax (u, v, w)
If v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
```

```
Initialise-Single-source (G, s)
 for Each Vertex v ∈ G.V
     v.d = ∞
     s.d = 0
     v.π = NIL
```

Step1

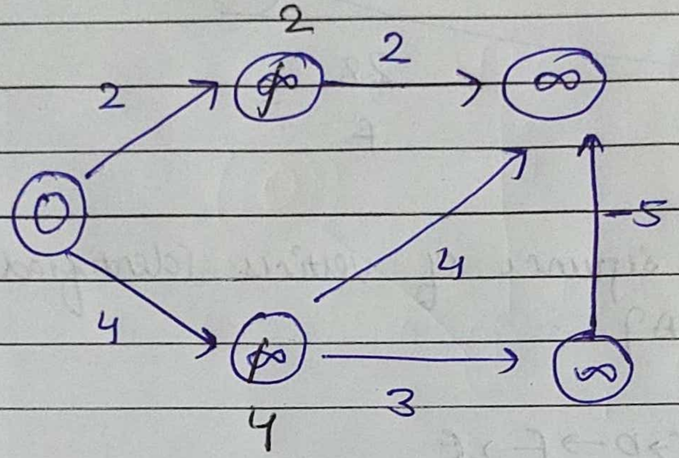
Qu 3



(P-Q) (P-R) (Q-S)
(R-S) (R-T) (T-S)

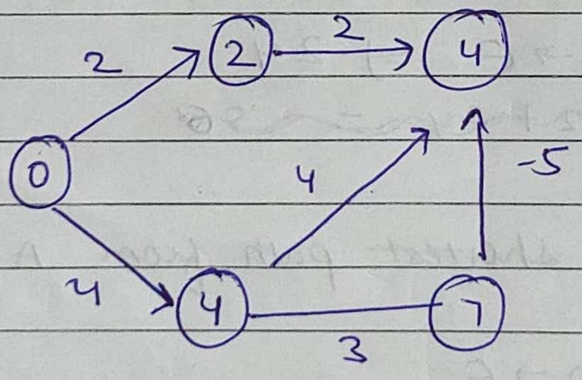
Step2 Select the source vertex with path value 0 and other vertices as Infinity

Round 1



This process is followed n-1 times where n is the total number of vertices

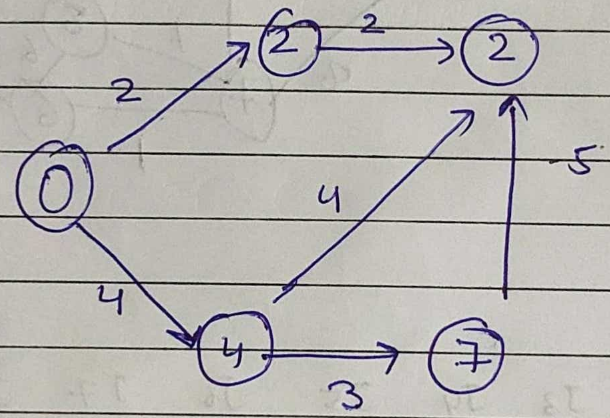
Step3 (P-Q) (P-R) (Q-S) (R-S) (R-T) (T-S)



Round 2

Therefore after n-1 iterations we find our path length and we can check if the graph has a negative cycle or not

	Q	R	S	T
0	∞	∞	∞	∞
0	2	4	∞	∞
0	2	4	4	7
0	2	4	2	7
0	2	4	2	7



Since weight of vertices aren't changing ie above round will be our final output But in case of program this will run (V-1) ie (5-1) => 4 times

(A)

(B) 2

(C) P→R→T→S