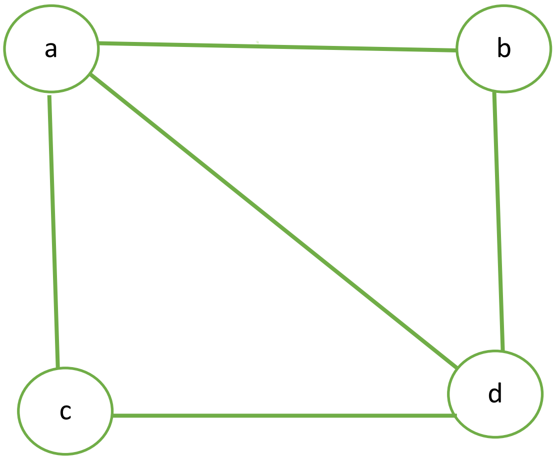
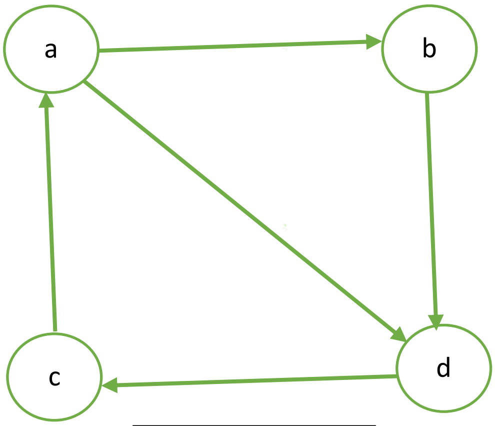


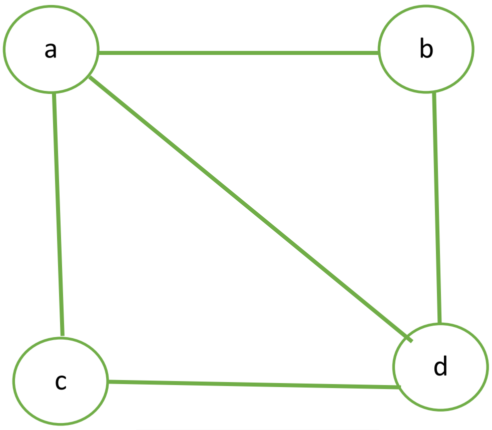
Here a, b, c, d are nodes and the lines between them are edges.

There are different types of graphs like

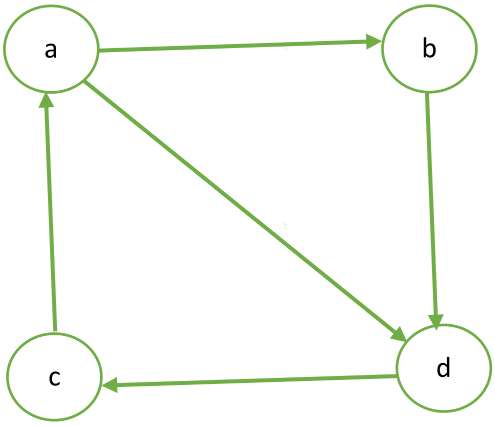
1. Direct and Indirect Graphs
2. Weighted and non-weighted graphs



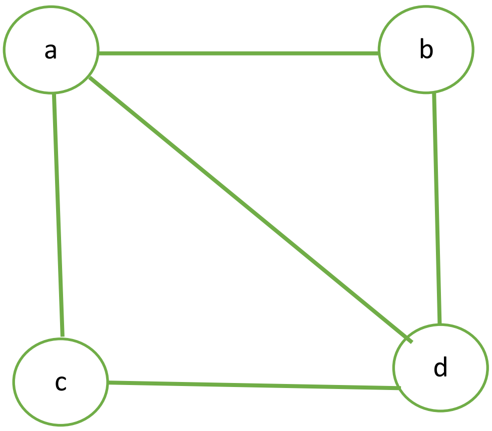
Directed Graph



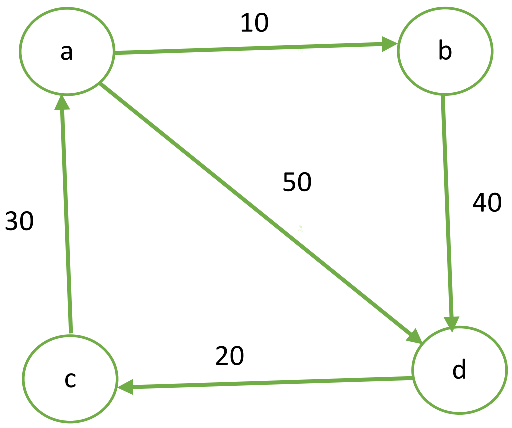
Indirected Graph



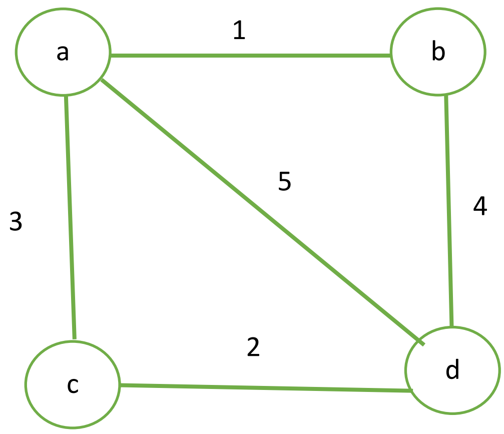
Non - Weighted Directed Graph



Non - Weighted Indirected Graph



Weighted Directed Graph



Weighted Indirected Graph

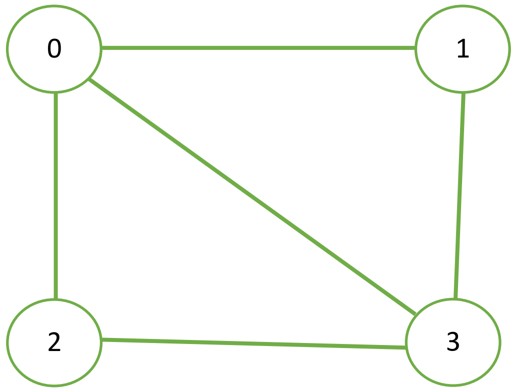
We can represent graphs using matrices

	0	1	2	3
0	0	1	1	1
1	1	0	0	1
2	1	0	0	1
3	1	1	1	0

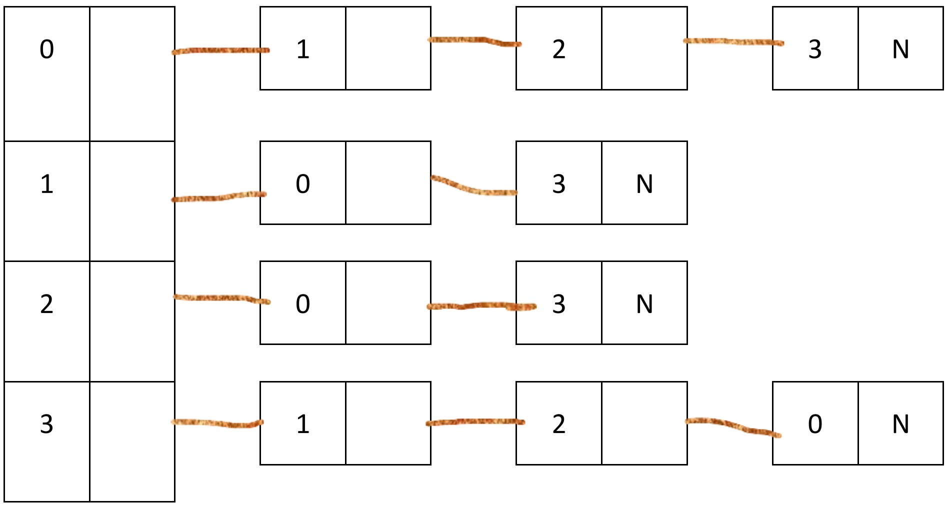
Keep 1's in place of these numbers

0 1
1 3
2 3
0 2
0 3

These are edges



Adjacency Representation of Graphs (Adjacency List) :-



Edges

0 1
1 3
2 3
0 2
0 3

Adjacency List

0	**1, 2, 3**
1	**0, 3**
2	**0, 3**
3	**0, 1, 2**

There are two different ways to traverse through the graph :

BFS - Breadth First Search

- It works on Queue.

DFS - Depth First Search

- It works on Stack

We have a visited array initially with filled will all '0's when we visit that particular node it's value is changed to '1' and we will not consider it when we traverse through the queue and find them in the adjacency lists.

Suppose we have nodes like a, b, c, d, e, f, g then, our initial visited array will be like this with an empty queue

0	0	0	0	0	0	0
a	b	c	d	e	f	g
Initial Visited array						

Empty queue						

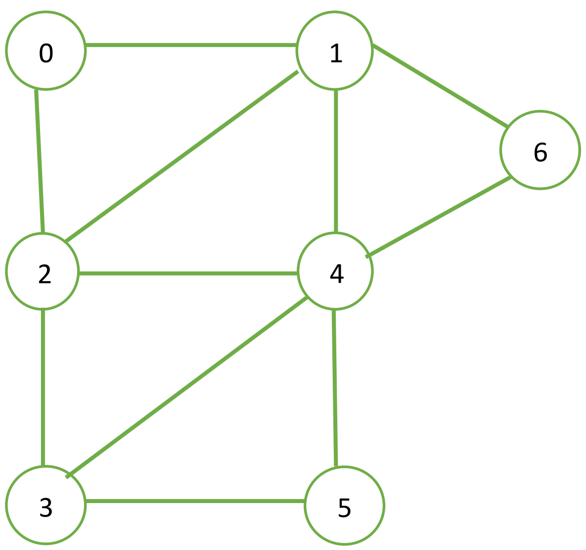
Suppose we have queue filled with a, d, g then, our visited array will be like this

1	0	0	1	0	0	1
a	b	c	d	e	f	g
Visited array						

a	d	g				
queue						

Here 0 means unvisited and 1 means visited.

BFS - Breadth First Search :-



	0	1	2	3	4	5	6
0	0	1	1	0	0	0	0
1	1	0	1	0	1	0	1
2	1	1	0	1	1	0	0
3	0	0	1	0	1	1	0
4	0	1	1	1	0	1	1
5	0	0	0	1	1	0	0
6	0	1	0	0	1	0	0

Edges

0 1
0 2
1 2
1 4
1 6
2 3
2 4
3 4
3 5
4 5
4 6

Adjacency List

0	1, 2
1	0, 2, 4, 6
2	0, 1, 3, 4
3	2, 4, 5
4	1, 2, 3, 5, 6
5	3, 4
6	1, 4

There is a visited array and queue of size 7 since there are 7 nodes in the above graph

0	0	0	0	0	0	0
0	1	2	3	4	5	6

Visited array

Queue

Since there it is a graph we can start traversing from any of the nodes of a graph

Here I am taking my initial node as 0.

1,2 is the adjacency list of 0, So we place **0,1,2** in our queue and change the value of 0,1,2 in visited array from 0 to 1

1	1	1	0	0	0	0
0	1	2	3	4	5	6

Visited array

0	1	2				

Queue

Adjacency List

0	**1, 2**
1	**0, 2, 4, 6**
2	**0, 1, 3, 4**
3	**2, 4, 5**
4	**1, 2, 3, 5, 6**
5	**3, 4**
6	**1, 4**

Now we have to go for adjacency list of 1 since it is after 0 in the queue by ignoring the already **visited values 0,2** and **insert 4, 6** as they are not visited and change the value of 4,6 in visited array to 1.

1	1	1	0	1	0	1
0	1	2	3	4	5	6

Visited array

0	1	2	4	6		

Queue

Adjacency List

0	**1, 2**
1	**0, 2, 4, 6**
2	**0, 1, 3, 4**
3	**2, 4, 5**
4	**1, 2, 3, 5, 6**
5	**3, 4**
6	**1, 4**

Now we have to go for adjacency list of 2 since it is after 1 in the queue by ignoring the already **visited values 0,1,4** and **insert 3** as it is not visited and change value of 3 in visited array to 1.

1	1	1	1	1	0	1
0	1	2	3	4	5	6

Visited array

0	1	2	4	6	3	

Queue

Adjacency List

0	**1, 2**
1	**0, 2, 4, 6**
2	**0, 1, 3, 4**
3	**2, 4, 5**
4	**1, 2, 3, 5, 6**
5	**3, 4**
6	**1, 4**

Now we have to go for adjacency list of 4 since it is after 2 in the queue by ignoring the already **visited values 1,2,3,6** and **insert 5** as it is not visited and change value of 5 in visited array to 1.

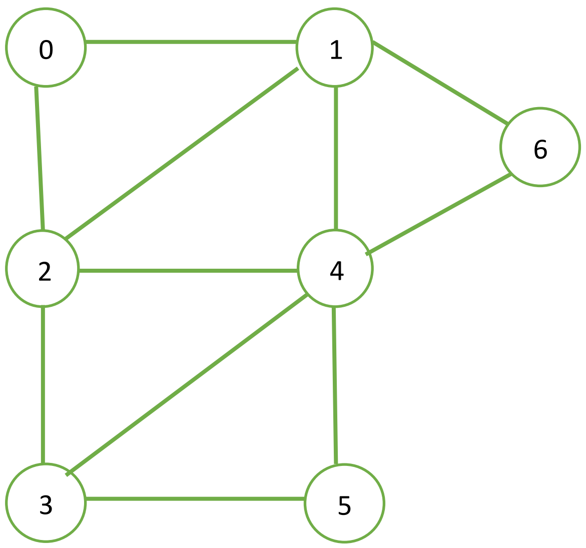
1	1	1	1	1	1	1
0	1	2	3	4	5	6

Visited array

0	1	2	4	6	3	5

Queue

The traversal of graph from 0 is 0, 1, 2, 4, 6, 3, 5



Adjacency List

0	1, 2
1	0, 2, 4, 6
2	0, 1, 3, 4
3	2, 4, 5
4	1, 2, 3, 5, 6
5	3, 4
6	1, 4

Edges

0 1
0 2
1 2
1 4
1 6
2 3
2 4
3 4
3 5
4 5
4 6

Now I am considering 3 as my initial node for the same tree.

0	0	0	0	0	0	0
0	1	2	3	4	5	6
Visited array						

Queue						

1. Check the adjacency list of 3.

0	0	1	1	1	1	0
0	1	2	3	4	5	6
Visited array						

3	2	4	5			
Queue						

2. Check the adjacency list of 2. visited values 3,4 insert 0,1

1	1	1	1	1	1	0
0	1	2	3	4	5	6
Visited array						

3	2	4	5	0	1	
Queue						

3. Check the adjacency list of 4. visited values 1,2,3,5 insert 6

1	1	1	1	1	1	0
0	1	2	3	4	5	6
Visited array						

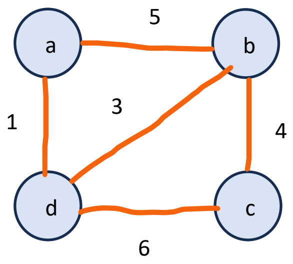
3	2	4	5	0	1	6
Queue						

The traversal of graph from 3 is 3,2,4,5,0,1,6.

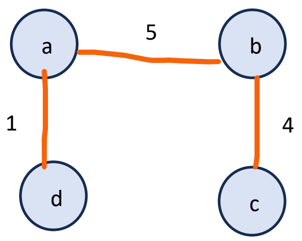
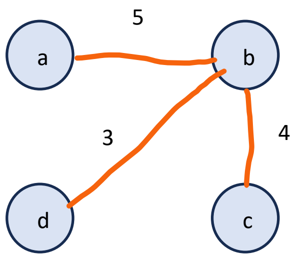
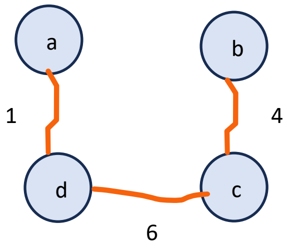
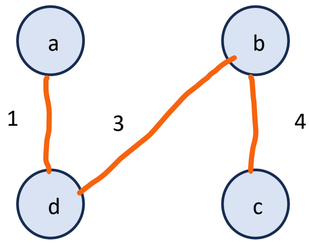
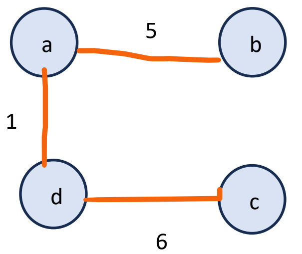
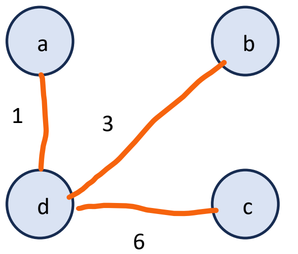
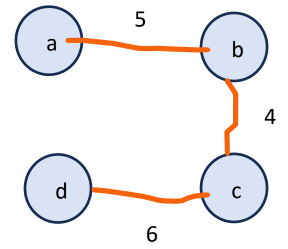
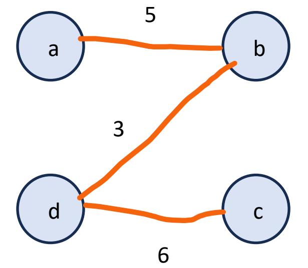
Minimum Spanning Tree (MST) :-

We can find the minimum spanning Tree of a weighted graph by using n nodes and n-1 edges of a graph.

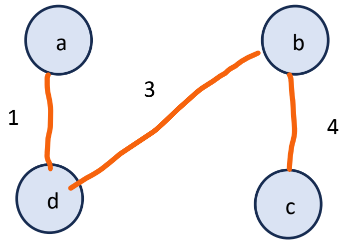
Suppose we have the following graph



Some of the possible spanning trees for above graph are the following

			
Weight = 1+5+4 = 10	Weight = 5+3+4 = 12	Weight = 1+6+4 = 11	Weight = 1+3+4 = 8
			
Weight = 5+1+6 = 12	Weight = 1+3+6 = 10	Weight = 5+4+6 = 15	Weight = 5+3+6 = 14

From the above minimum weight we calculated is 8. Therefore the minimum spaning tree of given graph is



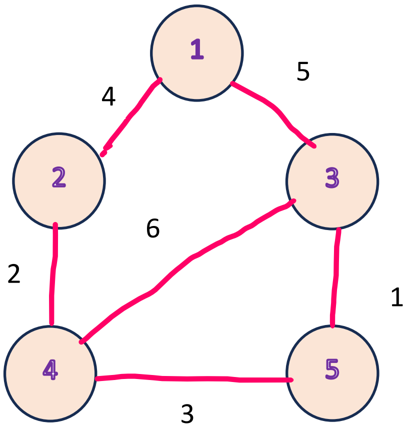
We can find the minimum spanning trees of a graph by using two algorithms

1. Prim’s Algorithm
2. Krushkal’s Algorithm

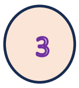
Prim’s Algorithm :-

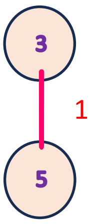
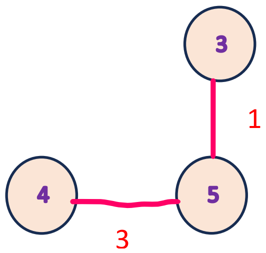
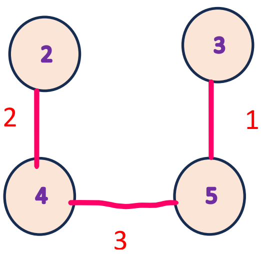
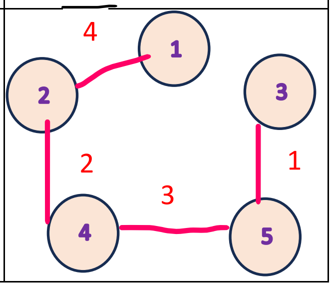
a. In this algorithm, First we have to find the minimum weighted edge and it’s edge ended nodes.
b. We can use of those two node’s to check all possible connected nodes to that node.
c. Then, select the minimum weighted edge to form in the next step in graph and remove it from our checking list.
d. Each time you check, you ignore all the possible cycle fomations.

Let our graph be



Minimum weight in the above graph is 1 i.e; between 3 and 5.

Let 3 be our initial node 

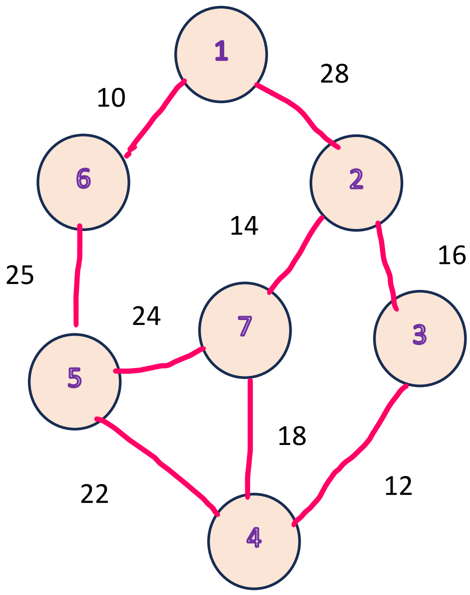
1) 3 – 4 -> 6 3 – 5 -> 1 3 – 1 -> 5		2) 3 – 4 -> 6 3 – 1 -> 5 5 – 4 -> 3	
3) 3 – 4 -> 6 3 – 1 -> 5 4 – 2 -> 2		4) 3 – 4 -> 6 3 – 1 -> 5 2 – 1 -> 4	

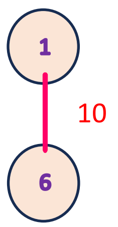
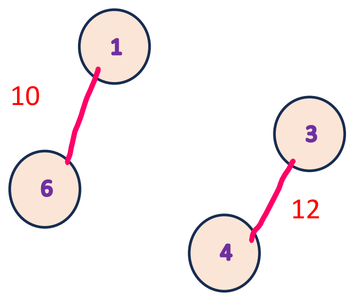
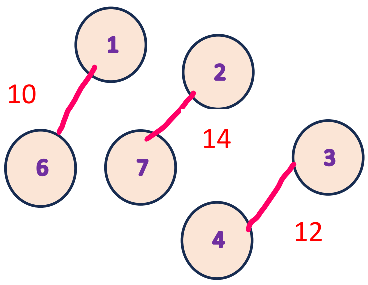
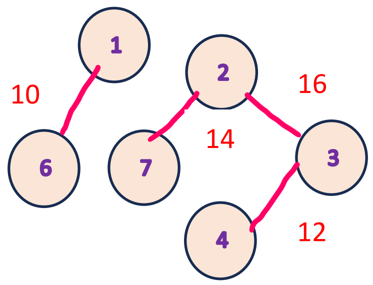
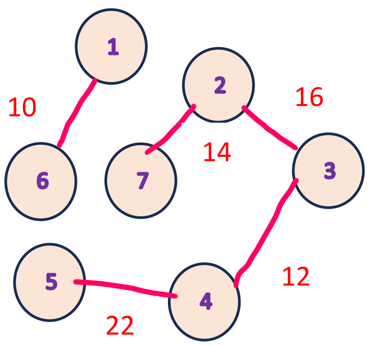
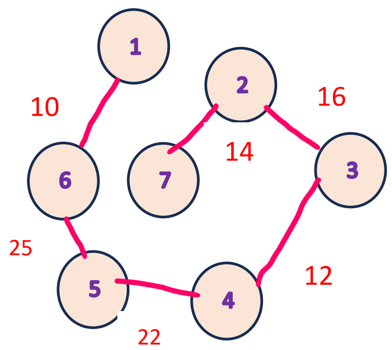
Here In step 4, we formed minimum spanning tree with weight 10 by using all 5 nodes and 4 edges

Kruskal’s Algorithm :-

a. In this algorithm, First we have to find the minimum weighted edge and it’s edge ended nodes.
b. We have to write all possible edges in the increasing order of weights.
c. Then, select the minimum weighted edge to form in the next step in graph and remove it from our checking list.
d. Each time you check, you ignore all the possible cycle fomations.

Let our graph be



<table>
<tr>
<td>1)
1 – 6 -> 10
3 – 4 -> 12
2 – 7 -> 14
2 – 3 -> 16
4 – 7 -> 18
4 – 5 -> 22
5 – 7 -> 24
5 – 6 -> 25
1 – 2 -> 28</td>
<td></td>
<td>2)
3 – 4 -> 12
2 – 7 -> 14
2 – 3 -> 16
4 – 7 -> 18
4 – 5 -> 22
5 – 7 -> 24
5 – 6 -> 25
1 – 2 -> 28</td>
<td></td>
</tr>
<tr>
<td>3)
2 – 7 -> 14
2 – 3 -> 16
4 – 7 -> 18
4 – 5 -> 22
5 – 7 -> 24
5 – 6 -> 25
1 – 2 -> 28</td>
<td></td>
<td>4)
2 – 3 -> 16
4 – 7 -> 18
4 – 5 -> 22
5 – 7 -> 24
5 – 6 -> 25
1 – 2 -> 28</td>
<td></td>
</tr>
<tr>
<td>5)
4 – 7 -> 18
4 – 5 -> 22
5 – 7 -> 24
5 – 6 -> 25
1 – 2 -> 28</td>
<td></td>
<td>6)
5 – 7 -> 24
5 – 6 -> 25
1 – 2 -> 28</td>
<td></td>
</tr>
</table>

In step 5 and 6, (4 – 7 -> 18) and (5 – 7 -> 24) forms a cycle. So I ignored and removed them and continued with the next minimum weight.

Here In step 6, We formed the minimum spanning tree with weight 99 by using all 7 nodes and 6 edges .

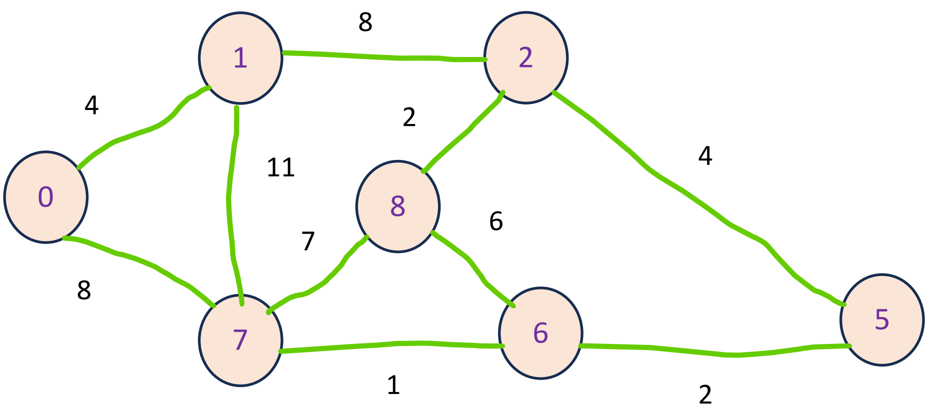
We can find the shortest path trees of a graph by using two algorithms

1. Dijkstra's Algorithm
2. Warshall's Algorithm

Dijkstra's Algorithm :-

a. In this everytime, we check the node, weight, status of a node.
b. First select any node and take it as initial node with weight '0' and it's status is taken as permanent(P) where remaining nodes weight is taken as infinite and temporary(T) status.
c. Now we want to check it's all possible edges and select minimum one from that where weight of the edge is minimum of (previous edge, new edge). It's weight is calculated from the initial edge.
d. Ignore the cyclic formations.

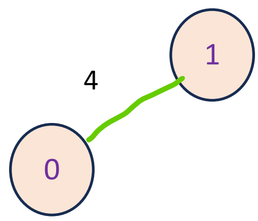
Let our graph be



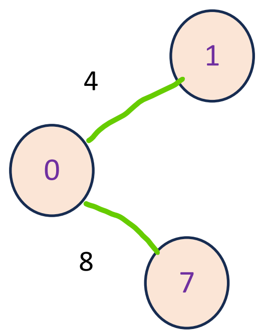
And consider 0 as initial node

<table>
<tr>
<td>For vertex 0
<table>
<tr><th>V</th><th>W</th><th>S</th></tr>
<tr><td>0</td><td>0</td><td>P</td></tr>
<tr><td>1</td><td>inf</td><td>T</td></tr>
<tr><td>2</td><td>inf</td><td>T</td></tr>
<tr><td>5</td><td>inf</td><td>T</td></tr>
<tr><td>6</td><td>inf</td><td>T</td></tr>
<tr><td>7</td><td>inf</td><td>T</td></tr>
<tr><td>8</td><td>inf</td><td>T</td></tr>
</table>
</td>
<td>Check edges of 0
0 -> 1 =>
min(inf,4) = 4

0 -> 7 =>
min(inf,8) = 8

min(4,8) = 4
So 1 node will be created with 0 and make 1 as permanent.</td>
<td>
</td>
</tr>
<tr>
<td>For vertex 1
<table>
<tr><th>V</th><th>W</th><th>S</th></tr>
<tr><td>0</td><td>0</td><td>P</td></tr>
<tr><td>1</td><td>4</td><td>P</td></tr>
<tr><td>2</td><td>inf</td><td>T</td></tr>
<tr><td>5</td><td>inf</td><td>T</td></tr>
<tr><td>6</td><td>inf</td><td>T</td></tr>
<tr><td>7</td><td>8</td><td>T</td></tr>
<tr><td>8</td><td>inf</td><td>T</td></tr>
</table>
</td>
<td>Check edges of 1
1 -> 7 =>
min(8,15) = 8

1 -> 2 =>
min(inf,12) = 12

min(8,12) = 8
So 7 node will be created with 0 and make 7 as permanent.</td>
<td>
</td>
</tr>
</table>

For vertex 7

V	W	S
0	0	P
1	4	P
2	12	T
5	inf	T
6	inf	T
7	8	P
8	inf	T

Check edges of 7

7 -> 6 =>

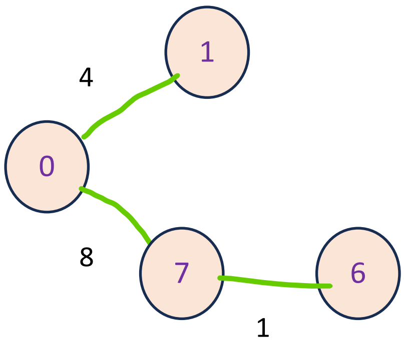
min(inf,9) = 9

7 -> 8 =>

min(inf,15) = 15

min(9,15) = 9

So 6 node will be created with 7 and make 6 as permanent.



For vertex 6

V	W	S
0	0	P
1	4	P
2	12	T
5	inf	T
6	9	P
7	8	P
8	15	T

Check edges of 6

6 -> 8 =>

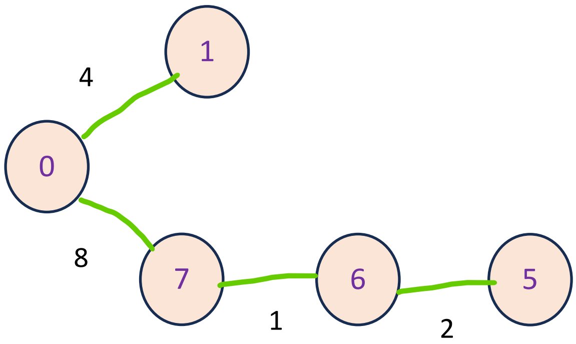
min(15,15) = 15

6 -> 5 =>

min(inf,11) = 11

min(15,11) = 11

So 5 node will be created with 6 and make 5 as permanent.



For vertex 5

V	W	S
0	0	P
1	4	P
2	12	T
5	11	P
6	9	P
7	8	P
8	15	T

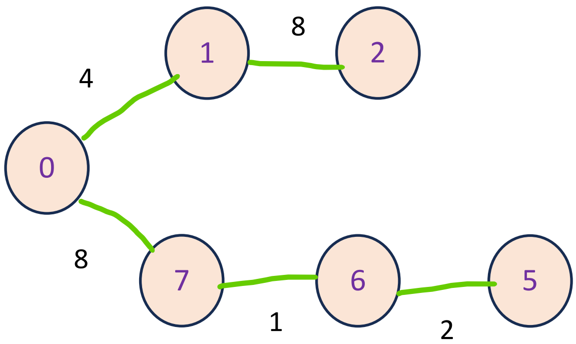
Check edges of 5

5 -> 2 =>

min(12,15) = 12

min(12) = 12

So 2 node will be created with 1 and make 2 as permanent.



For vertex 2

V	W	S
0	0	P
1	4	P
2	12	P
5	11	P
6	9	P
7	8	P
8	15	T

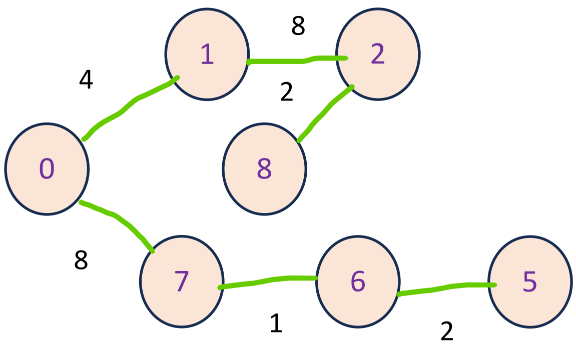
Check edges of 2

2 -> 8 =>

min(15,14) = 14

min(14) = 14

So 8 node will be created with 2 and make 8 as permanent.



V	W	S	5	15	P
0	0	P	6	13	P
1	4	P	7	8	P
2	12	P	8	14	P

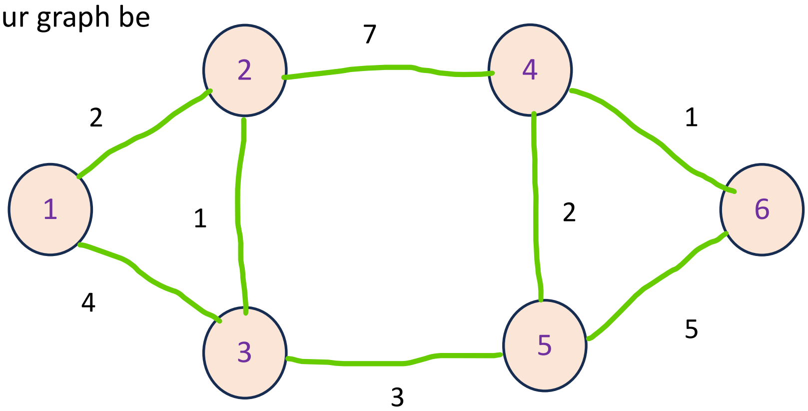
Warshall's Algorithm :-

a. In this algorithm we work with adjacency matrices of a graph.
b. First, we write the adjacency matrix of graph and take it as A_0 graph.
c. Then write n adjacency matrixes using the formula

$$A_k(i, j) = \min(A_{k-1}(i, j), A_{k-1}(i, k) + A_{k-1}(k, j)$$

d. Finally formed matrix will be the adjacency matrix of required shortest path tree.

Let our graph be



A_0	1	2	3	4	5	6
1	0	2	4	inf	inf	inf
2	2	0	1	7	inf	inf
3	4	1	0	inf	3	inf
4	inf	7	inf	0	2	1
5	inf	inf	3	2	0	5
6	inf	inf	inf	1	5	0

1 – 2 -> 2
1 – 3 -> 4
2 – 4 -> 7
2 – 3 -> 1
3 – 5 -> 3
4 – 5 -> 2
4 – 6 -> 1
5 – 6 -> 5

A_1	1	2	3	4	5	6
1	0	2	4	inf	inf	inf
2	2	0	1	7	inf	inf
3	4	1	0	inf	3	inf
4	inf	7	inf	0	2	1
5	inf	inf	3	2	0	5
6	inf	inf	inf	1	5	0

$A_1(2,3) = \min(1, 2 + 4 = 1$
$A_1(2,4) = \min(7, 2 + inf) = 7$
$A_1(2,5) = \min(inf, 2 + inf) = inf$
$A_1(2,6) = \min(inf, 2 + inf) = inf$
$A_1(3,2) = \min(1, 4 + 2) = 1$
$A_1(3,4) = \min(inf, 4 + inf) = inf$
$A_1(3,5) = \min(3, 4 + inf) = 3$
$A_1(3,6) = \min(inf, 4 + inf) = inf$
$A_1(4,2) = \min(7, inf + 2) = 7$
$A_1(4,3) = \min(inf, inf + 4) = inf$

$A_1(4,5) = \min(2, inf + inf) = 2$
$A_1(4,6) = \min(1, inf + inf) = 1$
$A_1(5,2) = \min(inf, inf + 2 = inf$
$A_1(5,3) = \min(3, inf + 4) = 3$
$A_1(5,4) = \min(2, inf + inf) = 2$
$A_1(5,6) = \min(5, inf + inf) = 5$
$A_1(6,2) = \min(inf, inf + 2) = inf$
$A_1(6,3) = \min(inf, inf + 4) = inf$
$A_1(6,4) = \min(1, inf + inf) = 1$
$A_1(6,5) = \min(5, inf + inf) = 1$

A_2	1	2	3	4	5	6
1	0	2	3	9	inf	inf
2	2	0	1	7	inf	inf
3	3	1	0	8	3	inf
4	9	7	8	0	2	1
5	inf	inf	3	2	0	5
6	inf	inf	inf	1	5	0

$A_2(1,3) = \min(4, 2 + 1) = 3$
$A_2(1,4) = \min(\text{inf}, 2 + 7) = 9$
$A_2(1,5) = \min(\text{inf}, 2 + \text{inf}) = \text{inf}$
$A_2(1,6) = \min(\text{inf}, 2 + \text{inf}) = \text{inf}$
$A_2(3,1) = \min(4, 1 + 2) = 3$
$A_2(3,4) = \min(\text{inf}, 1 + 7) = 8$
$A_2(3,5) = \min(3, 1 + \text{inf}) = 3$
$A_2(3,6) = \min(\text{inf}, 1 + \text{inf}) = \text{inf}$
$A_2(4,1) = \min(\text{inf}, 7 + 2) = 9$
$A_2(4,3) = \min(\text{inf}, 7 + 1) = 8$

$A_2(4,5) = \min(2, 7 + \text{inf}) = 2$
$A_2(4,6) = \min(1, 7 + \text{inf}) = 1$
$A_2(5,1) = \min(\text{inf}, \text{inf} + 2) = \text{inf}$
$A_2(5,3) = \min(3, \text{inf} + 1) = 3$
$A_2(5,4) = \min(2, \text{inf} + 7) = 2$
$A_2(5,6) = \min(5, \text{inf} + \text{inf}) = 5$
$A_2(6,1) = \min(\text{inf}, \text{inf} + 2) = \text{inf}$
$A_2(6,3) = \min(\text{inf}, \text{inf} + 1) = \text{inf}$
$A_2(6,4) = \min(1, \text{inf} + 7) = 1$
$A_2(6,5) = \min(5, \text{inf} + \text{inf}) = 5$

A_3	1	2	3	4	5	6
1	0	2	3	9	6	inf
2	2	0	1	7	4	inf
3	3	1	0	8	3	inf
4	9	7	8	0	2	1
5	6	4	3	2	0	5
6	inf	inf	inf	1	5	0

$A_3(1,2) = \min(2, 3 + 1) = 2$
$A_3(1,4) = \min(9, 3 + 8) = 9$
$A_3(1,5) = \min(\text{inf}, 3 + 3) = 6$
$A_3(1,6) = \min(\text{inf}, 3 + \text{inf}) = \text{inf}$
$A_3(2,1) = \min(2, 1 + 3) = 2$
$A_3(2,4) = \min(7, 1 + 8) = 7$
$A_3(2,5) = \min(\text{inf}, 1 + 3) = 4$
$A_3(2,6) = \min(\text{inf}, 1 + \text{inf}) = \text{inf}$
$A_3(4,1) = \min(9, 8 + 3) = 9$
$A_3(4,2) = \min(7, 8 + 1) = 7$

$A_3(4,5) = \min(2, 8 + 3) = 2$
$A_3(4,6) = \min(1, 8 + \text{inf}) = 1$
$A_3(5,1) = \min(\text{inf}, 3 + 3) = 6$
$A_3(5,2) = \min(\text{inf}, 3 + 1) = 4$
$A_3(5,4) = \min(2, 3 + 8) = 2$
$A_3(5,6) = \min(5, 3 + \text{inf}) = 5$
$A_3(6,1) = \min(\text{inf}, \text{inf} + 3) = \text{inf}$
$A_3(6,2) = \min(\text{inf}, \text{inf} + 1) = \text{inf}$
$A_3(6,4) = \min(1, \text{inf} + 8) = 1$
$A_3(6,5) = \min(5, \text{inf} + 3) = 5$

A_4	1	2	3	4	5	6
1	0	2	3	9	6	10
2	2	0	1	7	4	8
3	3	1	0	8	3	9
4	9	7	8	0	2	1
5	6	4	3	2	0	3
6	10	8	9	1	3	0

$A_4(1,2) = \min(2, 9 + 7) = 2$
$A_4(1,3) = \min(3, 9 + 8) = 3$
$A_4(1,5) = \min(6, 9 + 2) = 6$
$A_4(1,6) = \min(\text{inf}, 9 + 1) = 10$
$A_4(2,1) = \min(2, 7 + 9) = 2$
$A_4(2,3) = \min(1, 7 + 8) = 1$
$A_4(2,5) = \min(4, 7 + 2) = 4$
$A_4(2,6) = \min(\text{inf}, 7 + 1) = 8$
$A_4(3,1) = \min(3, 8 + 9) = 3$
$A_4(3,2) = \min(1, 8 + 7) = 1$

$A_4(3,5) = \min(3, 8 + 2) = 3$
$A_4(3,6) = \min(\text{inf}, 8 + 1) = 9$
$A_4(5,1) = \min(6, 2 + 9) = 6$
$A_4(5,2) = \min(4, 2 + 7) = 4$
$A_4(5,3) = \min(3, 2 + 8) = 3$
$A_4(5,6) = \min(5, 2 + 1) = 3$
$A_4(6,1) = \min(\text{inf}, 1 + 9) = 10$
$A_4(6,2) = \min(\text{inf}, 1+ 7) = 8$
$A_4(6,3) = \min(\text{inf}, 1+ 8) = 9$
$A_4(6,5) = \min(5, 1 + 2) = 3$

A_5	1	2	3	4	5	6
1	0	2	3	8	6	9
2	2	0	1	6	4	7
3	3	1	0	5	3	6
4	8	6	5	0	2	1
5	6	4	3	2	0	3
6	9	7	6	1	3	0

$A_5(1,2) = \min(2, 6 + 4) = 2$
$A_5(1,3) = \min(3, 6 + 3) = 3$
$A_5(1,4) = \min(9, 6 + 2) = 8$
$A_5(1,6) = \min(10, 6 + 3) = 9$
$A_5(2,1) = \min(2, 4 + 6) = 2$
$A_5(2,3) = \min(1, 4 + 3) = 1$
$A_5(2,4) = \min(7, 4 + 2) = 6$
$A_5(2,6) = \min(8, 4 + 3) = 7$
$A_5(3,1) = \min(3, 3 + 6) = 3$
$A_5(3,2) = \min(1, 3 + 4) = 1$

$A_5(3,4) = \min(8, 3 + 2) = 5$
$A_5(3,6) = \min(9, 3 + 3) = 6$
$A_5(4,1) = \min(9, 2 + 6) = 8$
$A_5(4,2) = \min(7, 2 + 4) = 6$
$A_5(4,3) = \min(8, 2 + 3) = 5$
$A_5(4,6) = \min(1, 2 + 3) = 1$
$A_5(6,1) = \min(10, 3 + 6) = 9$
$A_5(6,2) = \min(8, 3 + 4) = 7$
$A_5(6,3) = \min(9, 3 + 3) = 6$
$A_5(6,4) = \min(1, 3+ 2) = 1$

A_6	1	2	3	4	5	6
1	0	2	3	8	6	9
2	2	0	1	6	4	7
3	3	1	0	5	3	6
4	8	6	5	0	2	1
5	6	4	3	2	0	3
6	9	7	6	1	3	0

$A_6(1,2) = \min(2, 9 + 7) = 2$
$A_6(1,3) = \min(3, 9 + 6) = 3$
$A_6(1,4) = \min(8, 9 + 1) = 8$
$A_6(1,5) = \min(6, 9 + 3) = 6$
$A_6(2,1) = \min(2, 7 + 9) = 2$
$A_6(2,3) = \min(1, 7 + 6) = 1$
$A_6(2,4) = \min(6, 7 + 1) = 6$
$A_6(2,5) = \min(4, 7 + 3) = 4$
$A_6(3,1) = \min(3, 6 + 9) = 3$
$A_6(3,2) = \min(1, 6 + 7) = 1$

$A_6(3,4) = \min(5, 6 + 1) = 5$
$A_6(3,5) = \min(3, 6 + 3) = 3$
$A_6(4,1) = \min(8, 1 + 9) = 8$
$A_6(4,2) = \min(6, 1 + 7) = 6$
$A_6(4,3) = \min(5, 1 + 6) = 5$
$A_6(4,5) = \min(2, 1 + 3) = 2$
$A_6(5,1) = \min(6, 3 + 9) = 6$
$A_6(5,2) = \min(4, 3 + 7) = 4$
$A_6(5,3) = \min(3, 3 + 6) = 3$
$A_6(5,4) = \min(2, 3 + 1) = 2$

Therefore, The required shortest path graph is