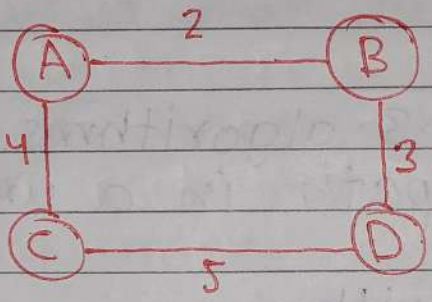
[Minimum Spanning Tree]

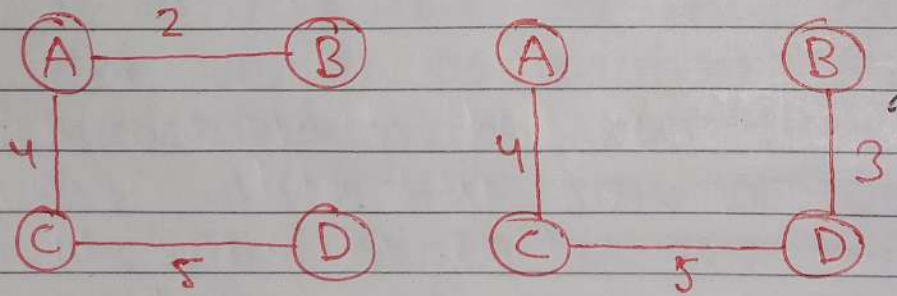
The sum of weights of edges of different spanning trees of a graph may be different.

A spanning tree of graph G whose sum of weights is minimum minimum among all spanning trees of G is called Minimum Spanning Tree of G.



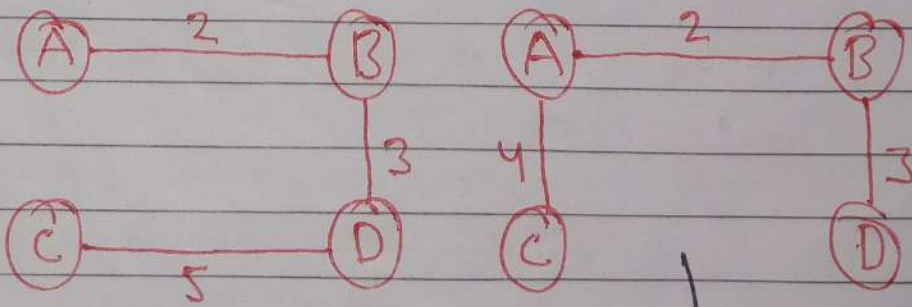
if no. of vertices is V, than no. of edges must be V-1.

a) Graph G



→ weight is 12.

weight is 11



weight is 10

weight is 9

Here the tree with weight 9 is the minimum spanning tree. If there are duplicate weights in the graph than more than one spanning trees are possible; and if all weights are uniaue then there will be only one minimum spanning tree.

Minimum Spanning Tree gives us the most economical way of connecting all the vertices in a graph.

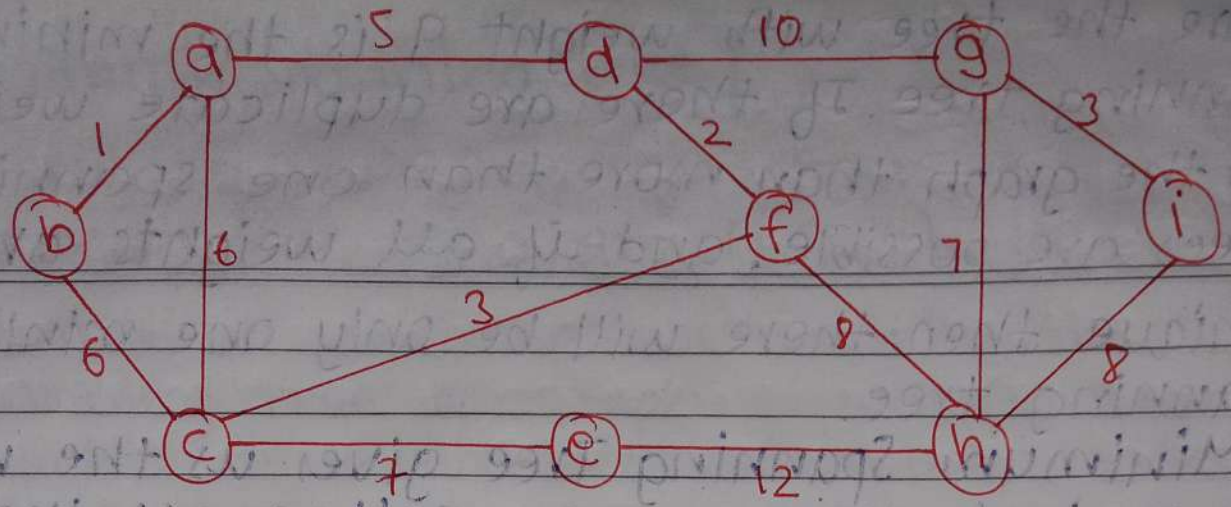
There are many ways for creating minimum spanning tree but the most famous methods are Prim's Algorithm and Kruskal's Algorithm.

- Both of these methods use the Greedy Approach.

* Prim's Algorithm :- We use Prim's Algorithm to find minimum cost spanning tree.

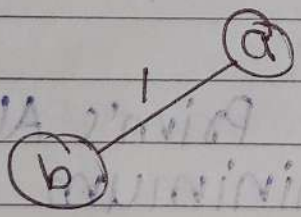
Prim's Algorithm me hum kisi bhi arbitrary verter se start krege, or us node se hme ye check krna h ke kis kis node pe jump krskte hai, or jis node pe kam cost ayegi uspe jump krege.

Prim's Algorithm always connected way me hi kaam krta hai. (Kruskal's Algorithm at the end connected ho jata hai but in between disconnected hota hai).

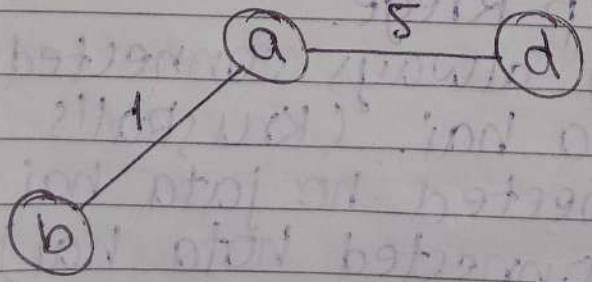


* Create minimum spanning tree using Prim's Algorithm :- (start from b)

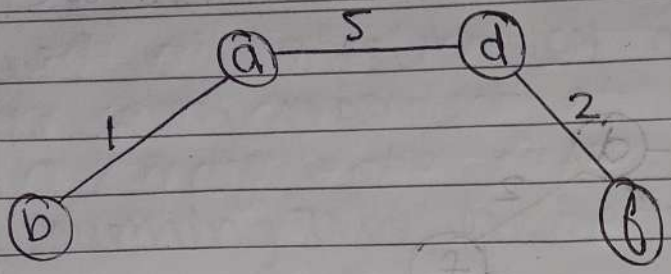
vertex b se start krege, b se hum 2 vertex pe ja skte hai, i.e., a and c. Dono me se a pe jane ki cost kamm hai. so, we choose a vertex.



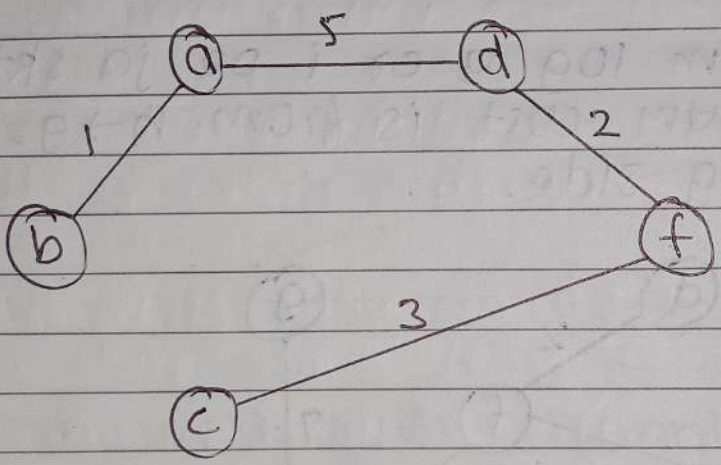
Now, we check that a se hum kis kis node pe ja skte hai, i.e., d and c. Dono me se d pe jane ki cost kamm hai. so, we choose d vertex.



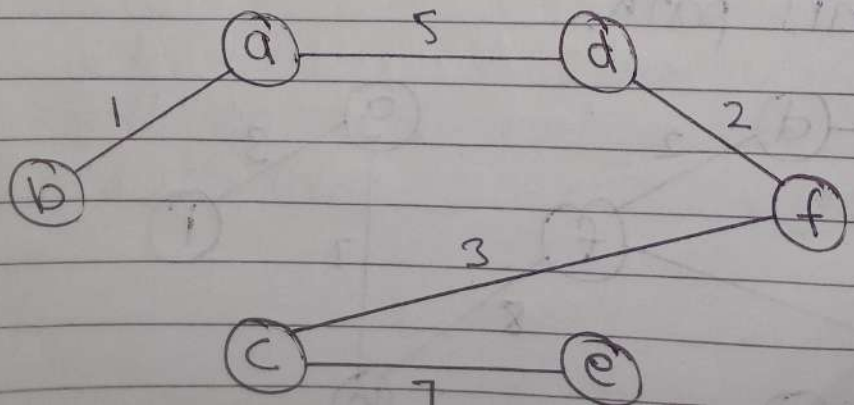
Now, we check that d se hum kis vertex pe jump krr skte hai, i.e., g & f. Dono me se f pe jane ki cost kam hai. So, we choose f vertex.



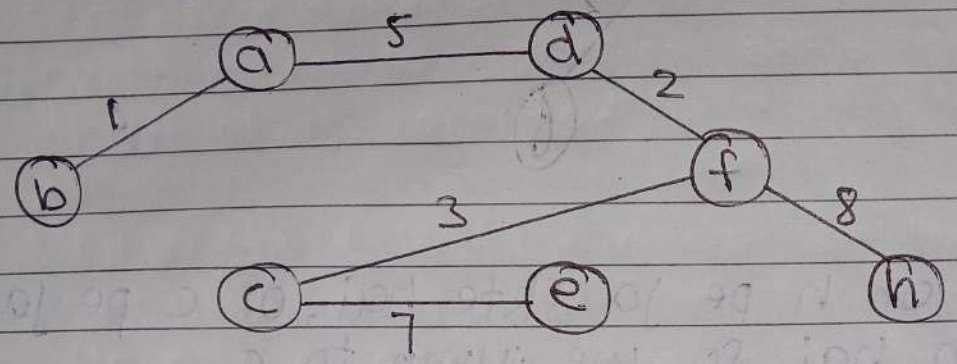
f se hum c or h pe ja skte hai, or c pe jane ki cost kam hai. So, we jump to c.



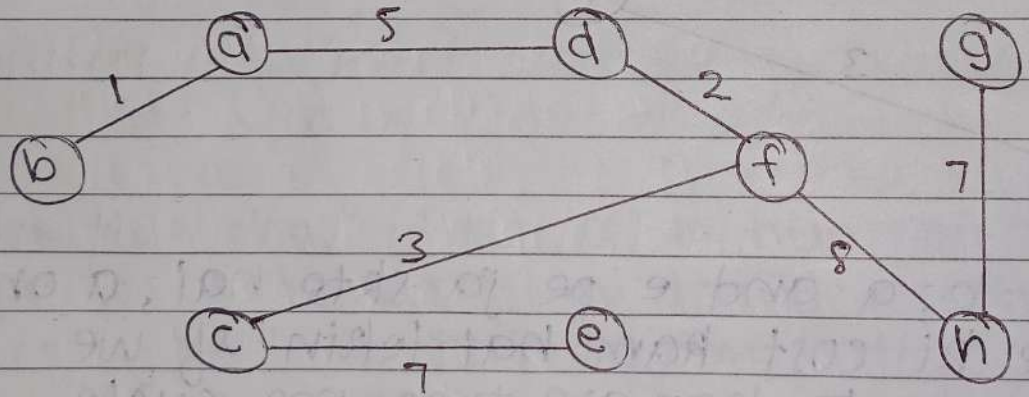
c se hum b, a and e pe ja skte hai, a or b pe jane ki cost kam hai lekin if we choose a or b to hamare tree me cycle create ho rhi hai, and the rule of spanni -ng tree is that there is no cycle in the tree. So, we have to select vertex e.



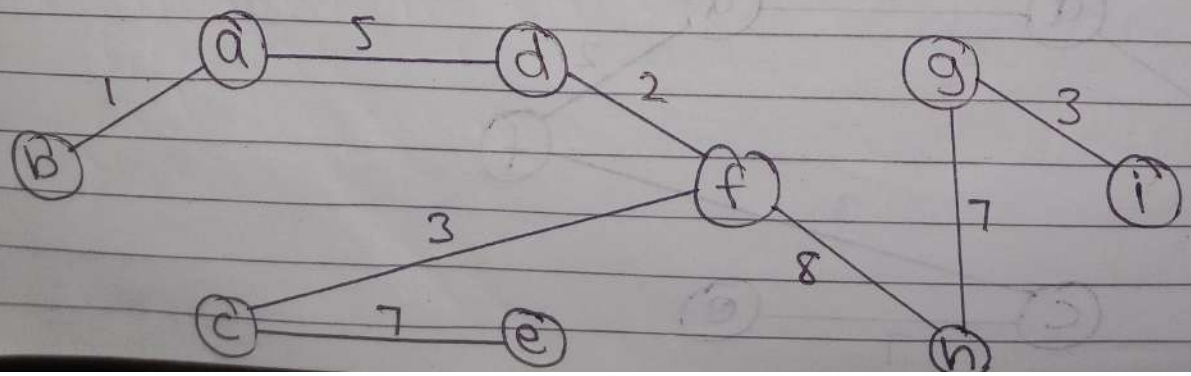
Now, if we want to go to h vertex then there is two ways, one from e and one from f. Now, we have to check the minimum one. So, the minimum one is f vertex. So we choose f → h path.



Now, h vertex is hum log g or i pe ja skte hai and the minimum cost is from h → g. So, we goes on the g side.



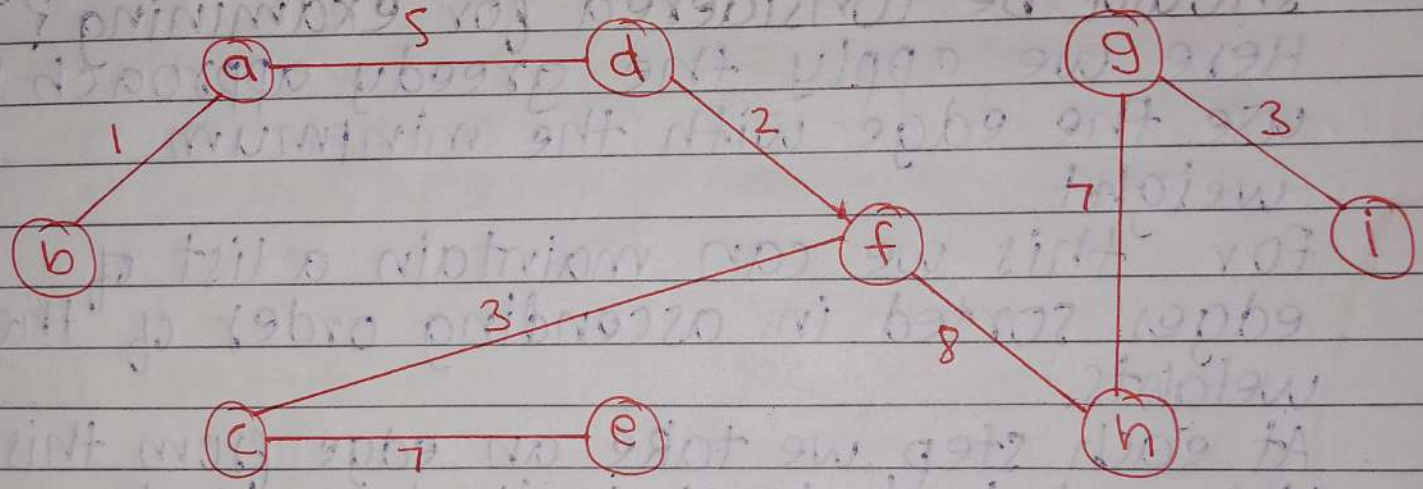
Now, i pe jane ke 2 route hai, one from g and one from h, minimum wala is g → i. So, we choose this path.



Now, we have seen that we cover all the vertices and there is no cycle created in our spanning tree. Also, our vertices are 9 and our edges are (9-1), i.e., 8.

It means, we satisfied all the properties of Spanning Tree.

Final Spanning Tree from Prim's Algorithm are :-



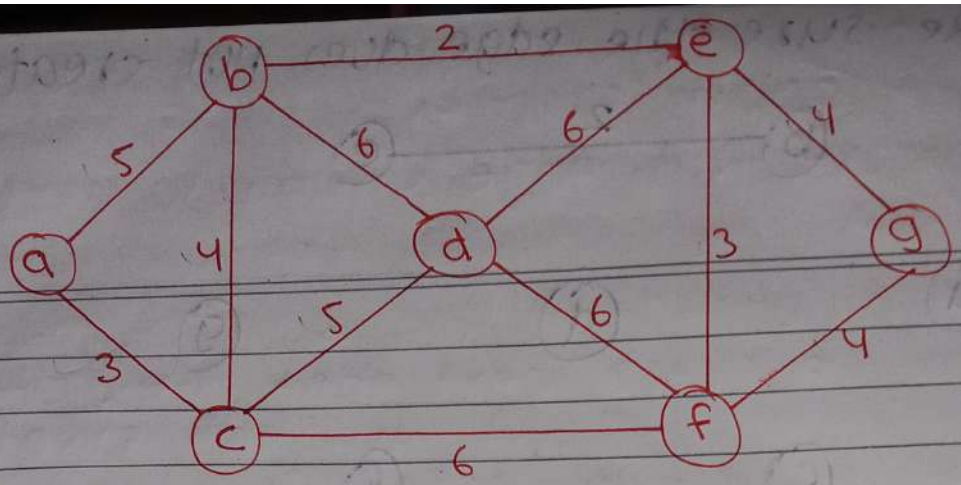
* Total cost of this spanning tree is :-

(1 + 5 + 2 + 3 + 7 + 8 + 7 + 3) = 36.

* Kruskal's Algorithm :- At the start of this algorithm, each tree is a single vertex tree and no edges are there. In, each step of the algorithm an edge is examined and it is included in the spanning tree if its inclusion does not form a cycle. If this edge forms a cycle than it is rejected.

* Now the Question arises as to which edge should be considered for examining? Here we apply the greedy approach & use the edge with the minimum weight.
For this we can maintain a list of edges sorted in ascending order of their weights.
At each step we take an edge from this list and include it if it does not form a cycle. If the edge forms a cycle, we reject it. Say, we have n vertices, & our algorithm complete when n-1 edges have been included.
If the graph contain less than n-1 edges then it means that graph is not connected & no spanning tree is possible.

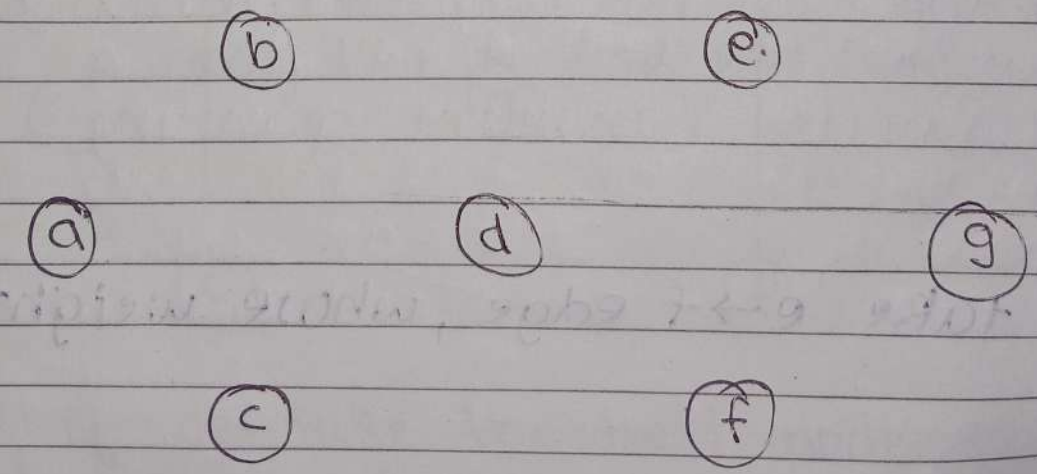
* Initially, in Kruskal's Algorithm, the trees are not connected, but at the end the resultant tree is connected MST.



First we maintain the list of edges in ascending order:-

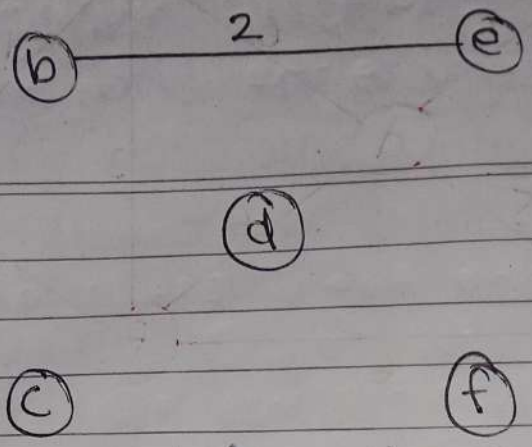
b → e = 2	a → b = 5
a → c = 3	c → d = 5
e → f = 3	b → d = 6
b → c = 4	e → d = 6
e → g = 4	d → f = 6
g → f = 4	c → f = 6

Now, we take an edge from the list & include it, if it does not form a cycle. If the edge form a cycle we ignore it.



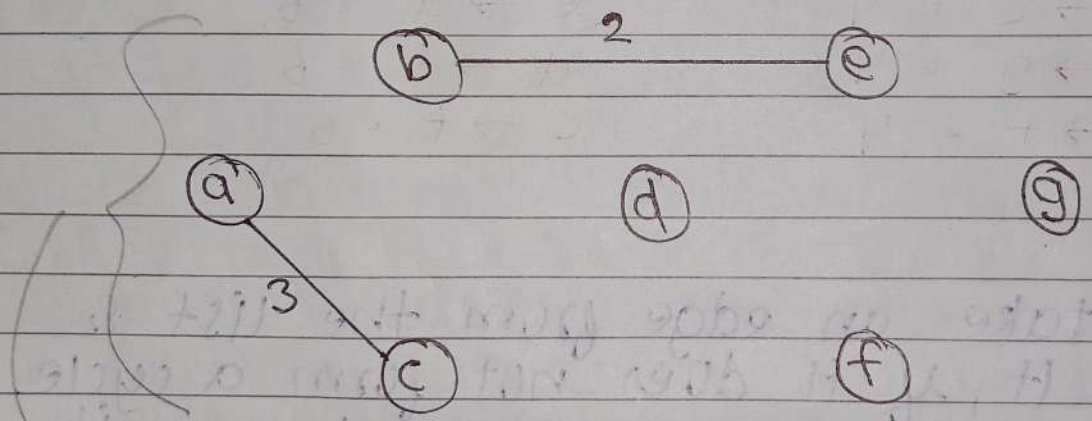
Initially, we have an empty tree with 7 veritex. Now, we take a edge from list and include it in our tree.

And make sure the edge does not create a cycle.



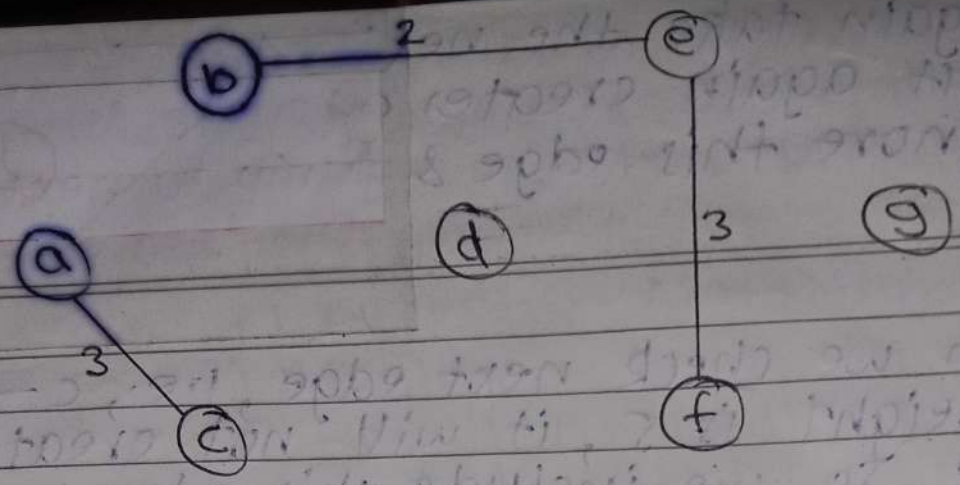
First, we create the minimum weighted edge, i.e., b → e, whose weight is 2.

After that we create a → c edge, whose weight is 3.

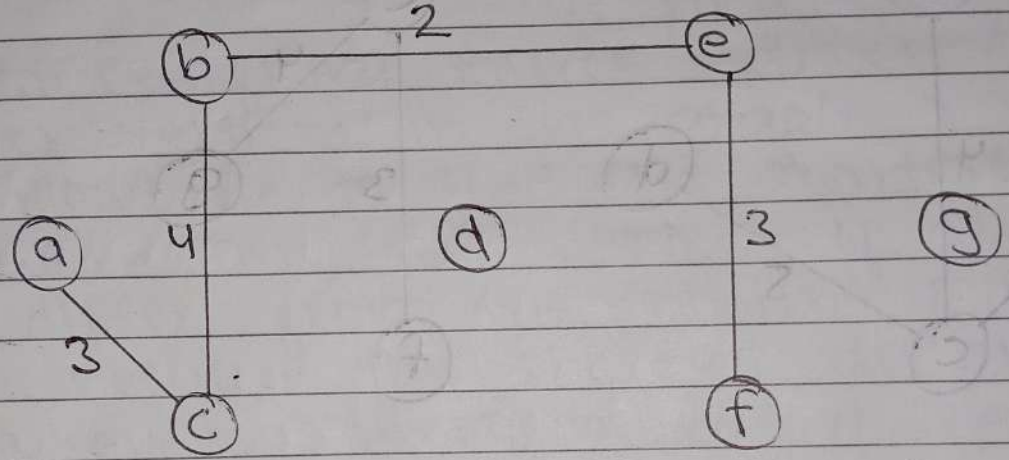


→ the trees are not connected initially but at the end it will form a connected minimum spanning tree.

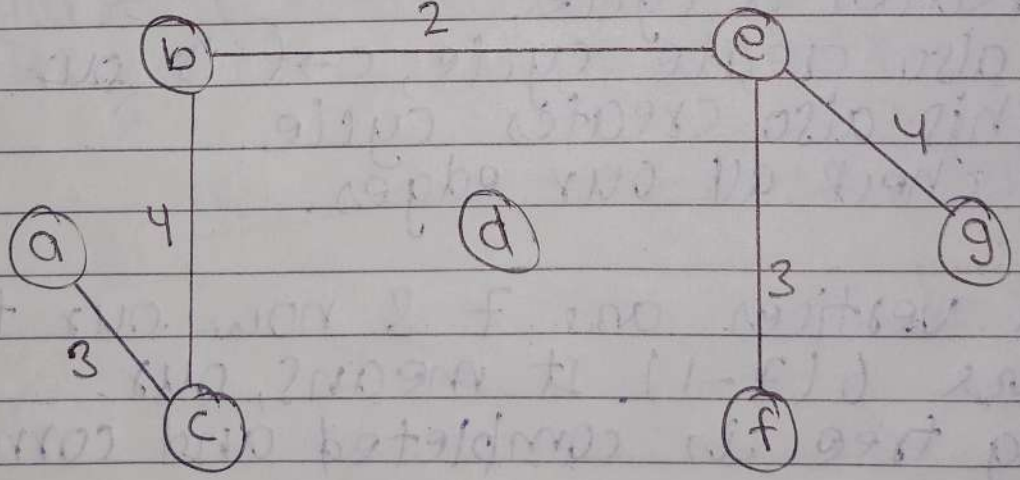
Now, we take e → f edge, whose weight is 3.



Now, we take the edge b → c, whose weight is 4.



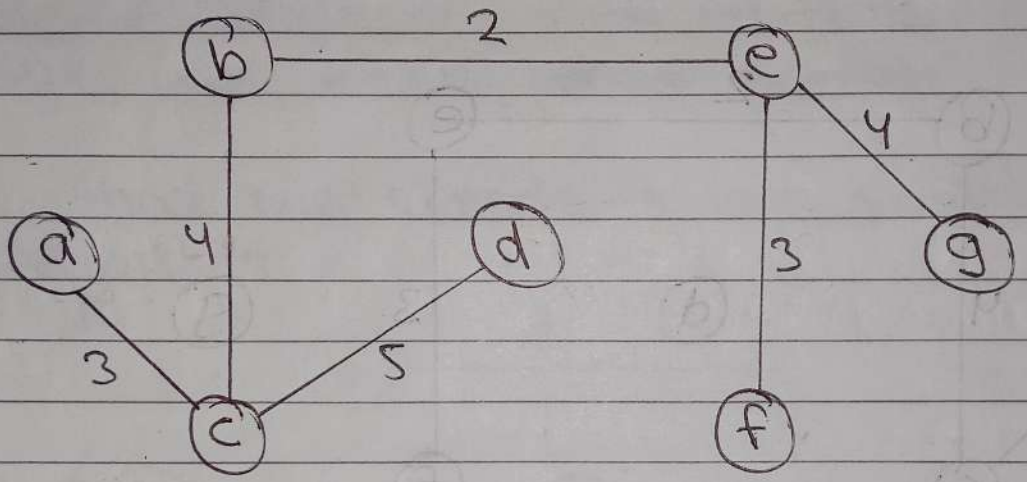
Now, we take e → g, whose weight is also 4.



Now, if we take the next node from the list, i.e., g → f then it creates a cycle in our tree. So, we ignore this edge, and move two the next edge.

If we again take the next edge, i.e., a→b, it again creates a cycle. So, we again ignore this edge & jump to next edge.

Now, when we check next edge, i.e., c→d whose weight is 5, it will not create the cycle. So, we include this edge in our tree.



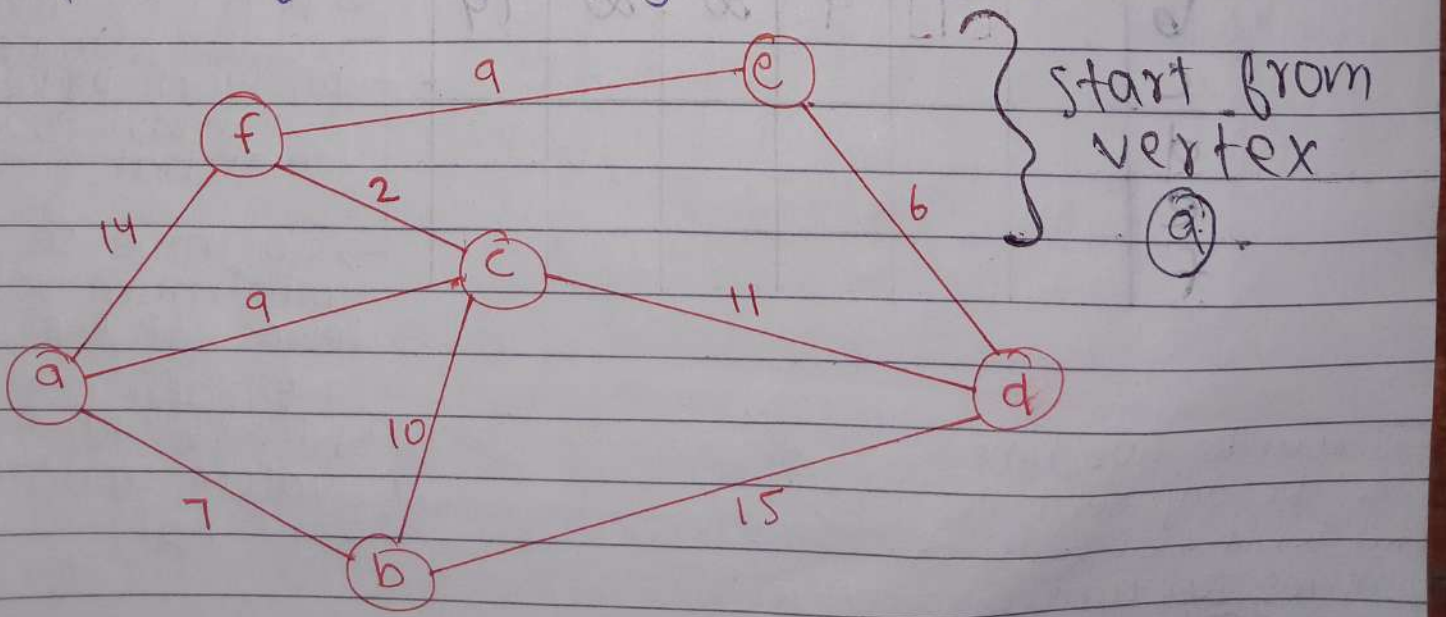
Next edge, i.e., b→d again create cycle. So, we jump to next edge, i.e., e→d. This edge again creates a cycle.
d→f also create cycle, c→f is our last edge & this also creates cycle.
Now, we check all our edges.

Our total vertices are 7 & now our total edges are 6 (7-1). It means, our spanning tree is completed and correct.

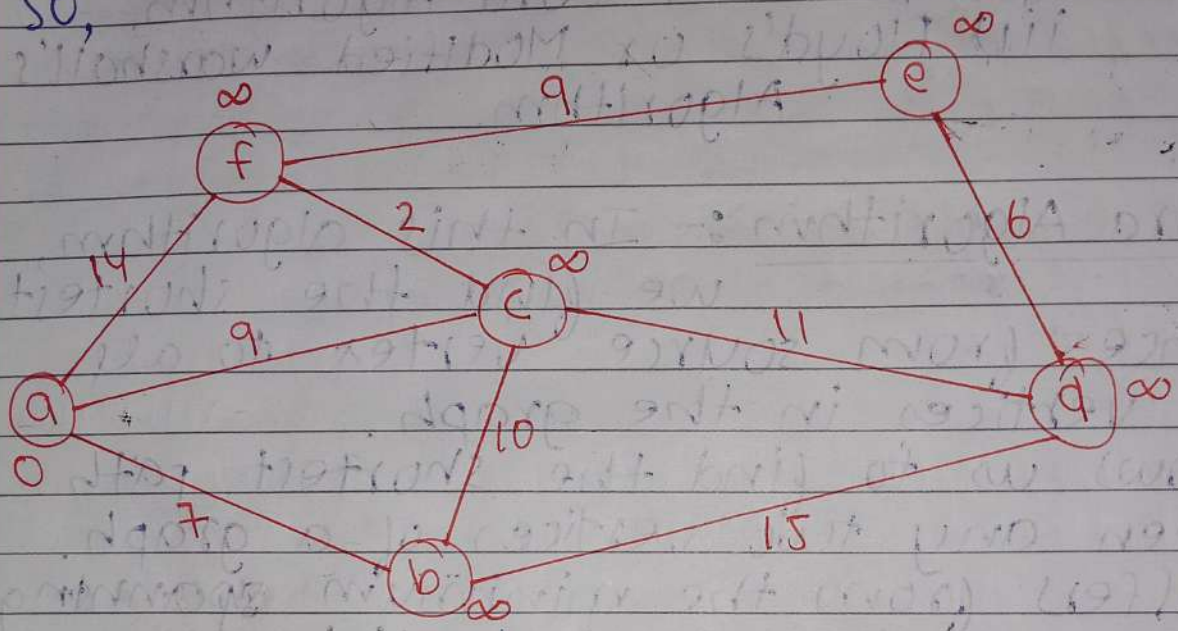
* Shortest Path Problem Algorithms :-
There are basically 3 algorithms for finding out the shortest path in a weighted graph :-

i) Dijkstra Algorithm
ii) Bellman Ford Algorithm.
iii) Floyd's or Modified warshall's Algorithm.

* Dijkstra Algorithm :- In this algorithm, we find the shortest distance from source vertex to all other vertices in the graph.
It allows us to find the shortest path between any two vertices of a graph.
It differs from the minimum spanning tree because the shortest distance between two vertices might not include all the vertices of the graph.
It's a single source shortest path algorithm, means we find the shortest path from a single source.



Our starting vertex is (a), so, at this time we write 0 as the value of (a) and ∞ as the value of all other nodes because we don't know the value of all other nodes.
so,



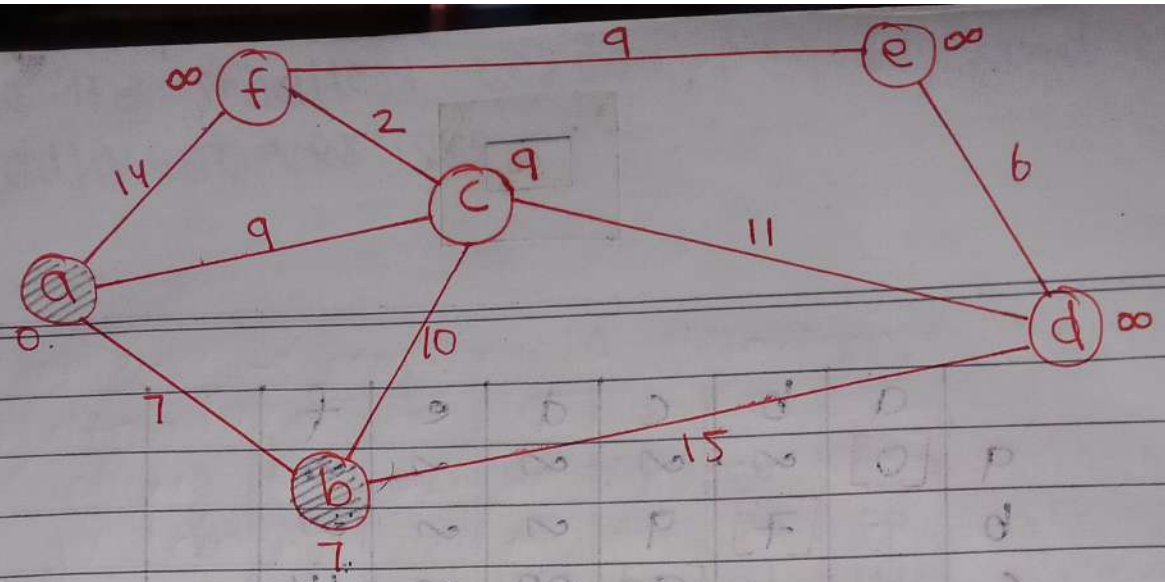
Now we create table for this graph with current values,

	a	b	c	d	e	f
a	[0]	∞	∞	∞	∞	∞
b		[7]	9	∞	∞	14

Initially hme a se start krna h to uspe 0 likhega baaki sb ∞ hoga or uske baad "a" wale ko block krke minimum value leni hai.

→ ab inme se minimum value lena hai, i.e., 7 (graph me bhi "b" ki value update krni hai)

is row ke liye hme "a" ke adjacent vertices ko lena hai, baki ki values ko upar se as it is copy paste krna hai.



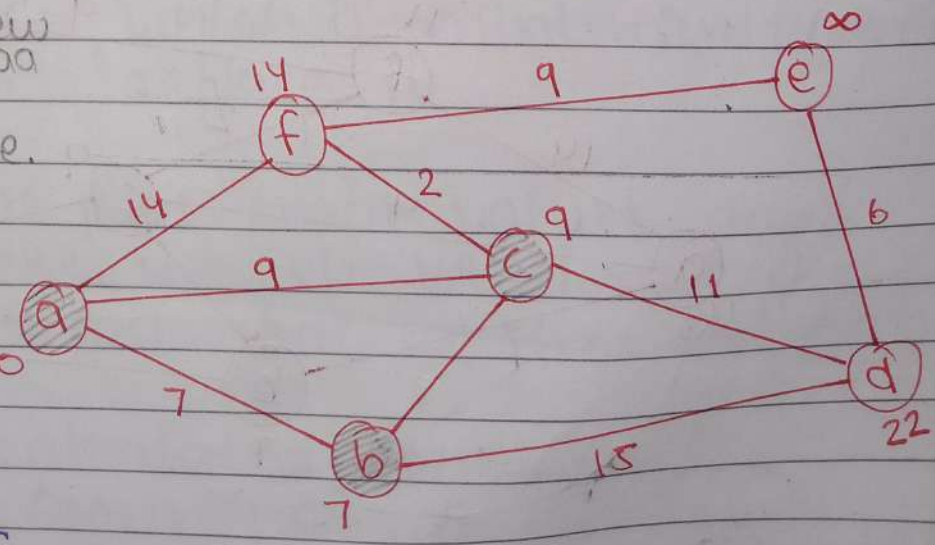
Now, in our table, minimum value is b, i.e. 7, now we take this b as our vertex, block b's column & select b's adjacent vertices.

paths ↓	a	b	c	d	e	f
a	[0]	∞	∞	∞	∞	∞
b		[7]	9	∞	∞	14
c			[9]	22	∞	14

→ ab isme se minimum lenge 'c' wala column block krdege or c ke adjacent ka uthayega baaki copy paste hojayega

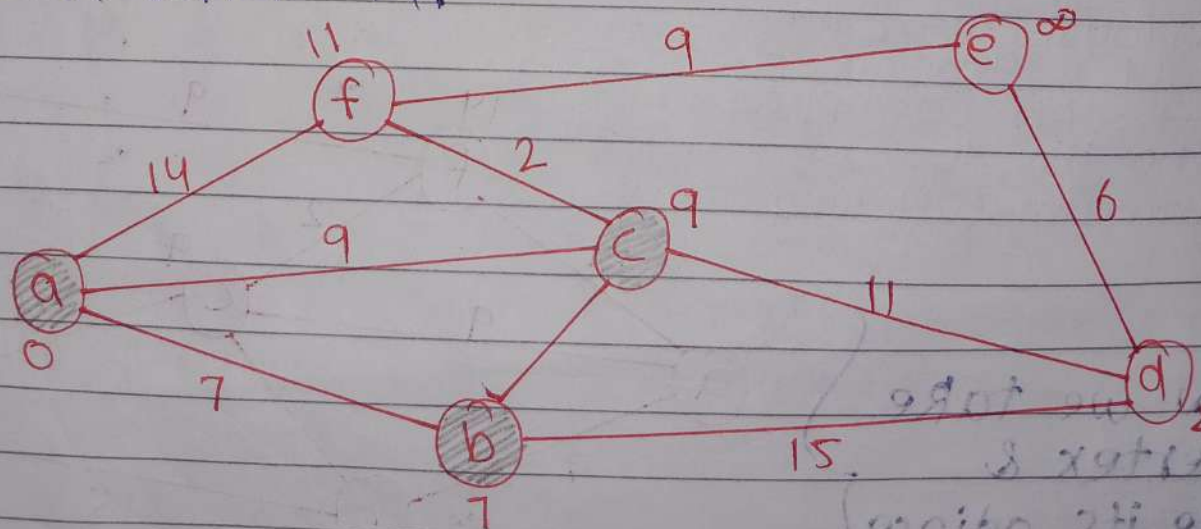
yaha hmne b→c ki value 9 is liya hai because previously a→c ka cost 9 tha or new b→c ka cost 17 aa rha tha, to hum minimum cost lenge.

now we take "c" vertex & take its adjacent vertices, i.e; f and d.



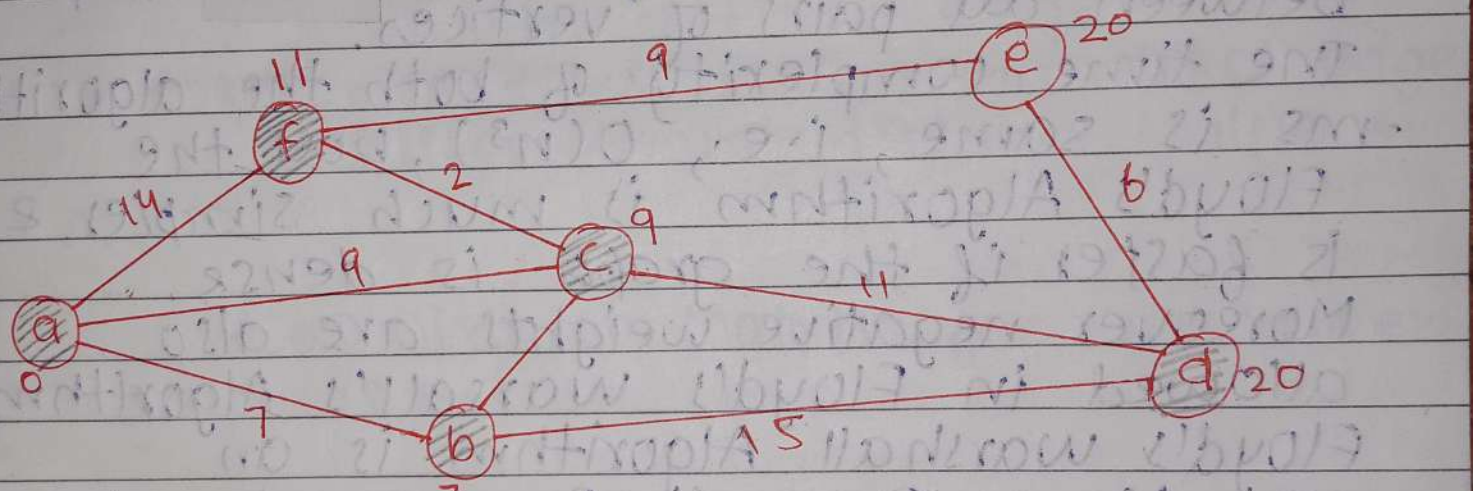
	a	b	c	d	e	f
a	[0]	∞	∞	∞	∞	∞
b		[7]	9	∞	∞	∞
c			[9]	22	∞	14
f				20	∞	[11]

Now, when we consider the adjacent of "c" vertex then we have only 2 adjacent of "c", i.e., f and d.
Now, when we calculate the path from c → d, its 9 + 11 = 20, and our previous value of d is 22. So, we replace 20 with 22, & update the value of d.
Same happens with c → f, current cost is 14 but new cost from c → f is 9 + 2 = 11. So update 11 with 14.



take the smallest vertex, i.e., f, and ch... it's adjacent vertices.

	a	b	c	d	e	f
a	[0]	∞	∞	∞	∞	∞
b		[7]	9	∞	∞	∞
c			[9]	22	∞	14
f				20	∞	[11]
d				[20]	20	
e					[20]	

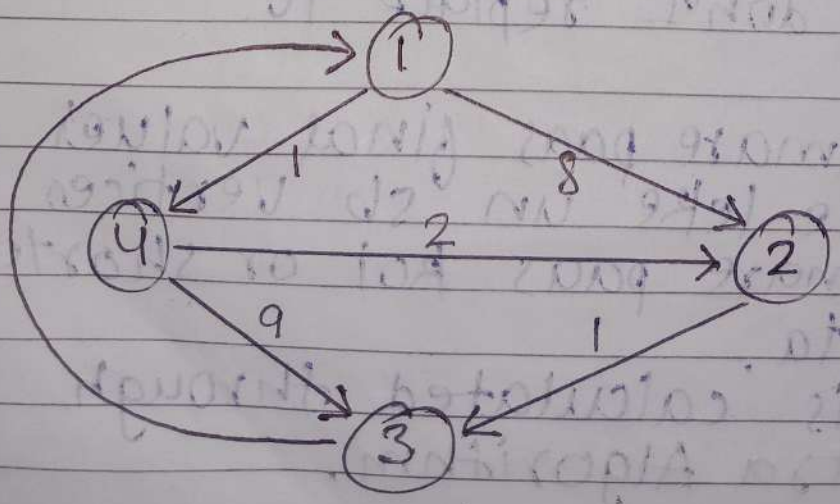


If we take "d's" adjacent vertices than is "e", & the cost from d→e is 26 but already e's cost is 20, which is minimum from 26. So, we don't replace it.

Now, ab jo hmare paas final value's aayi "a" vertex se leke un sb vertices tk jane ka hmare paas koi or shortest path nahi ho skta.
And, this is calculated through Dijktra Algorithm.

* Floyd's warshall's Algorithm :- This is an all path/pair shortest path problem, i.e., it finds the shortest path between all pairs of vertices in the graph.
We may apply Dijkstra Algorithm n times, once for each vertex as the source vertex and find the shortest path between all pairs of vertices.
The time complexity of both the algorithms is same, i.e., $O(n^3)$, but the Floyd's Algorithm is much simpler & is faster if the graph is dense.
Moreover negative weights are also allowed in Floyd's warsall's Algorithm
Floyd's warshall Algorithm is an example of Dynamic Programming Approach.

Example :-



Step 1: Remove all the self loops and parallel edges from the graph.

Step 2: create the initial distance matrix. It represents the distance between every pair of vertices in the form of given weights.

* For diagonal elements (representing self loops), distance value = 0.
* For vertices having a direct edge between them, distance value = weight of that edge.
* For vertices having no direct edge between them, distance value = ∞.

Initial Distance Matrix for the given graph is :-

D_0 =

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4	∞	0	∞
4	∞	2	9	0

Step 3 : Now, we create D_1 Matrix and the Base of D_1 Matrix is D_0 Matrix. In D_1 Matrix, we copy the First Row and First column from the D_0 Matrix, and then find the remaining "∞" values.

D_1 Matrix,

	1	2	3	4
1	0	8	∞	1
2	∞	0		
3	4		0	
4	∞			0

1st row & 1st column copied D_1 = D_0 matrix.

means, if we want to calculate [2,3], then we have to go in like [2,1] then [1,3], because its D_1 Matrix.

(calculate the remaining values, but we have to include (1) in the path, because of D_1 Matrix.

Same goes with [2,4], we will go like [2,1] then [1,4].

The formula is,

Base is D_0 matrix, so, we compare new value with old value (right hand side).

$$D_0[2,3] = D_0[2,1] + D_0[1,3]$$

$$1 = \infty + \infty$$

$$1 = \infty$$

Here, 1 is not equal to infinity its less than infinity.

So, we write 1 as it, it in the new D_1 matrix, at the place of $D_1[2,3]$

$D_1 =$

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	
3	4		0	
4	∞			0

} Like we calculate 1 for $D_1[2,3]$. same, we calculate all other values.

$D_0[2,4] = D_0[2,1] + [1,4]$

$\infty = \infty + 1$

$\infty = \infty$ } new value is infinity so we write ∞, at the place of $D_1[2,4]$

$D_1 =$

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4		0	
4	∞			0

$D_0[3,2] = D_0[3,1] + D_0[1,2]$

$\infty = 4 + 8$

$\infty = 12$ } $12 < \infty$, so. we replace 12 at $D_1[3,2]$

$D_1 =$

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4	12	0	
4	∞			0

$D_0[3,4] = D_0[3,1] + D_0[1,4]$

$\infty = 4 + 1$

$\infty = 5$ } $5 < \infty$, so we replace 5 at $D_1[3,4]$.

$D_1 =$

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4	12	0	5
4	∞			0

$D_0[4,2] = D_0[4,1] + D_0[1,2]$

$2 = \infty + 8$

$2 = \infty$ { $2 < \infty$, so we replace 2 at [4,2]

$D_1 =$

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4	12	0	5
4	∞	2		0

$D_0[4,3] = D_0[4,1] + D_0[1,3]$

$9 = \infty + \infty$

$9 = \infty$ { $9 < \infty$, replace 9 with value of $D_1[4,3]$

$D_1 =$

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4	12	0	5
4	∞	2	9	0

now this is our updated matrix we have calculated.

now, we create D_2 matrix with base D_1. (Forgot D_0 matrix)

Step 4: Now, we create D_2 Matrix and the Base for D_2 Matrix is the Final D_1 Matrix. In D_2 Matrix, we copy the second row and second column from D_1 Matrix & then find the remaining values as we calculated for D_1 matrix. The same procedure follow in D_2 Matrix.

D_2 Matrix,

$D_2 = $

	1	2	3	4
1	0	8	9	1
2	∞	0	1	∞
3	4	12	0	5
4	∞	2	3	0

2nd row & 2nd column copied from D_1 Matrix

calculate the remaining values, but we have to include (2) in the path, because of D_2 Matrix.

$D_1[1,3] = D_1[1,2] + D_1[2,3]$

$\infty = 8 + 1$

$\infty = 9$ } $9 < \infty$, replace 9 with the value at $D_2[1,3]$

$D_1[1,4] = D_1[1,2] + D_1[2,4]$

$1 = 8 + \infty$

$1 = \infty$ } $1 < \infty$, replace at $D_1[1,4]$

$D_1[3,1] = D_1[3,2] + D_1[2,1]$

$4 = 12 + \infty$

$4 = \infty$ } $4 < \infty$, replace at $D_2[3,1]$

$D_1[3,4] = D_1[3,2] + D_1[2,4]$

$5 = 12 + \infty$

$5 = \infty$ } $5 < \infty$, replace at $D_2[3,4]$.

$D_1[4,1] = D_1[4,2] + D_2[2,1]$

$\infty = 2 + \infty$

$\infty = \infty$ } keep ∞, at $D_2[4,1]$

$D_1[4,3] = D_1[4,2] + D_1[2,3]$

$9 = 2 + 1$

$9 = 3$ } $3 < 9$, replace 3 at $D_2[4,3]$

Final D_2 Matrix is,

$D_2 =$

	1	2	3	4
1	0	8	9	1
2	∞	0	1	∞
3	4	12	0	5
4	∞	2	3	0

now this is our updated matrix that we have calculated,

now, we create D_3 matrix with base D_2.

Step 5: Now, we create D_3 Matrix and the Base for D_3 Matrix is the Final D_2 Matrix. In D_3 Matrix, we copy the third row and third column from D_2 matrix & then find the remaining values as we calculated for D_2 matrix. The same procedure follow in D_3 matrix.

D_3 Matrix,

$D_3 =$

	1	2	3	4
1	0	8	9	1
2	5	0	1	6
3	4	12	0	5
4	7	2	3	0

3rd row & 3rd column copied from D_2 matrix

calculate the remaining values but we have to include (3) in the path, because it's D_3 matrix.

$D_2[1,2] = D_2[1,3] + D_2[3,2]$

$8 = 9 + 12$

$8 = 21$ } $8 < 21$ replace 8 with the value at $D_2[1,2]$

$D_2[1,4] = D_2[1,3] + D_2[3,4]$

$1 = 9 + 5$

$1 = 14$ } $1 < 14$ replace 1 with the value at $D_2[1,4]$

$D_2[2,1] = D_2[2,3] + D_2[3,1]$

$\infty = 1 + 4$

$\infty = 5$ } $5 < \infty$; replace 5 with the value at $D_2[2,1]$

$D_2[2,4] = D_2[2,3] + D_2[3,4]$

$\infty = 1 + 5$

$\infty = 6$ } $6 < \infty$, replace 6 with the value at $D_2[2,4]$

$D_2[4,1] = D_2[4,3] + D_2[3,1]$

$\infty = 3 + 4$

$\infty = 7$ } $7 < \infty$, replace 7 with the value at $D_2[4,1]$

$D_2[4,2] = D_2[4,3] + D_2[3,2]$

$2 = 3 + 12$

$2 = 15$ } $2 < 15$, replace 2 with the value at $D_2[4,2]$

Final D_3 Matrix is,

$$D_3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 8 & 9 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 12 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix} \end{array}$$

} this is our update matrix that we have calculated.

now we create D_4 matrix with base D_3

Step 6: Now, we create our last matrix, i.e., D_4 matrix and the base for D_4 matrix is our final D_3 matrix. In D_4 matrix, we copy the fourth row & fourth column from D_3 matrix & then find the remaining values as we calculated for D_3 matrix. The same procedure follow in D_4 matrix.

D_4 Matrix,

4th row & 4th column copied from D_3 matrix

$D_4 =$

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

calculate the remaining values but we have to include (4) in the path, because it's D_4 matrix.

$D_3[1,2] = D_3[1,4] + D_3[4,2]$

$8 = 2 + 2$

$8 = 3\}\ 3<8$, replace 3 with the value at $D_3[1,2]$

$D_3[1,3] = D_3[1,4] + D_3[4,3]$

$9 = 1 + 3$

$9 = 4\}\ 4<9$, replace 4 with the value at $D_3[1,3]$

$D_3[2,1] = D_3[2,4] + D_3[4,1]$

$5 = 6 + 7$

$5 = 13$ } $5 < 13$ so replace 5 with the value at $D_3[2,1]$

$D_3[2,3] = D_3[2,4] + D_3[4,3]$

$1 = 6 + 3$

$1 = 9$ } replace 1 with value at $D_3[2,3]$

$D_3[3,1] = D_3[3,4] + D_3[4,1]$

$4 = 5 + 7$

$4 = 12$ } $4 < 12$, replace 4 with value at $D_3[3,1]$

$D_3[3,2] = D_3[3,4] + D_3[4,2]$

$12 = 5 + 2$

$12 = 7$ } $7 < 12$, replace 7 with value at $D_3[3,2]$

Final D_4 Matrix is,

$D_4 =$

	1	2	3	4
1	0	3	4	1
2	5	0	1	6
3	4	7	0	5
4	7	10	3	0

The final matrix which has the minimum path is our last D_4 matrix, i.e.,

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

} there is no shortest path other than this, from the one vertex to another.