Dijkstra’s Algorithm

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.

For a given source node in the graph, the algorithm finds the shortest path between that node and every other. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities.

Dijkstra's Algorithm:

Step 1: Remove all the loops (Any edge that starts and ends at the same vertex is loop)

Step 2: Remove all parallel edges between two vertices except the one with least weight.

Step 3: create the weight matrix table

(i) Set 0 to the source vertex and infinite to the remaining vertices.

For all vertices, repeat (ii) and (iii)

(ii) Mark the smallest unmarked value and mark that vertex.

(iii) Find those vertices which are directly connected with marked vertex and update all.

Update value formula:

New Destination value = minimum (Old Destination value, Marked value+ Edge Weight)

Step 4: Find the shortest path from source to destination using **backtracking.**

Put destination vertex in 'shortest path'

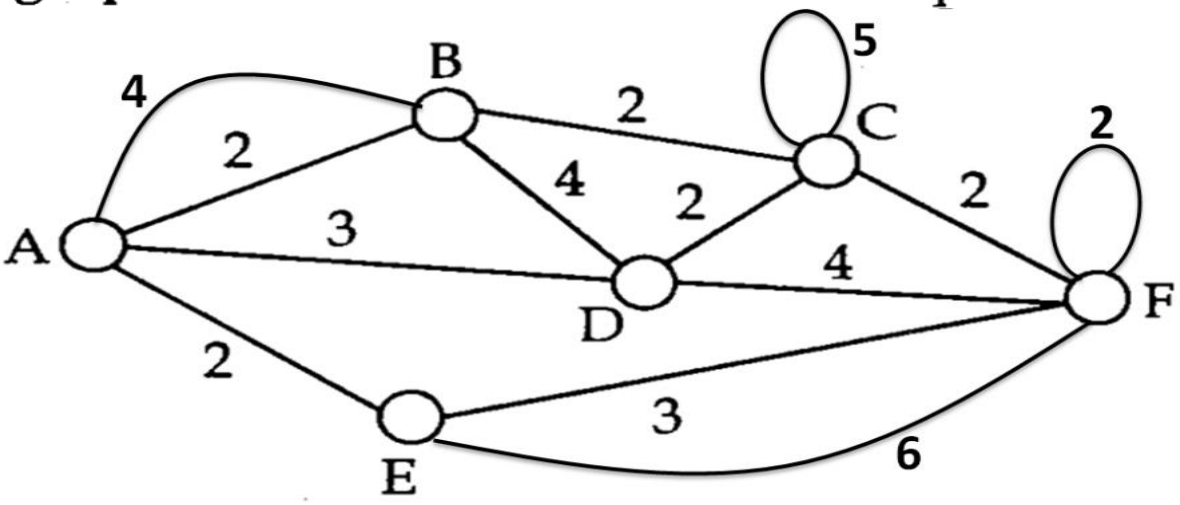
(i) Set pointer to the last marked value and put that vertex to the 'shortest path' list.

(ii) Move the pointer up until marked value is change.

(iii) If the last marked value has been changed then move the pointer to the marked value in that row and put that vertex to the 'shortest path' list.

Repeat (ii) to (iii) until the pointer will move to source vertex.

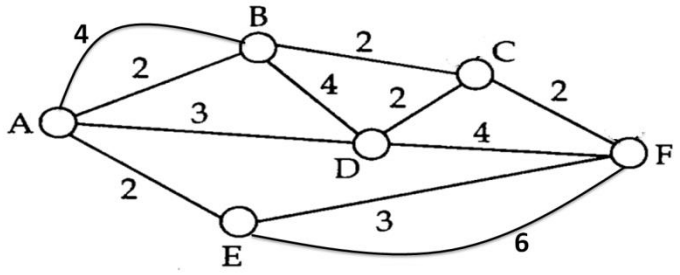
Using Dijkstra's algorithm, find the shortest path between A and F for the following graph. Show the intermediate steps also.



Solution:-

Step 1: Remove all the loops. (Edges that start and end at the same vertex is called loop)

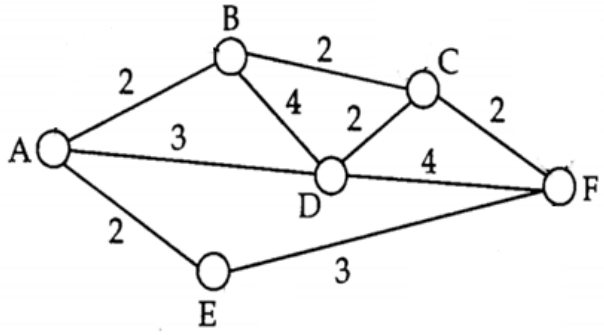
In the above graph there are two loops. One is in the vertex C and another one is in the Vertex F(starting and ending vertex are same), thus we remove this two edge.



Step 2: Remove all parallel edges between two vertices except the one with least weight.

In the above graph there are two edges from A vertex to B vertex, and there cost/weight are 4 and 2.This two edges are called parallel edges. We remove that edge which one has greater value (4>2) i.e. 4. So we remove that edge of cost 4.

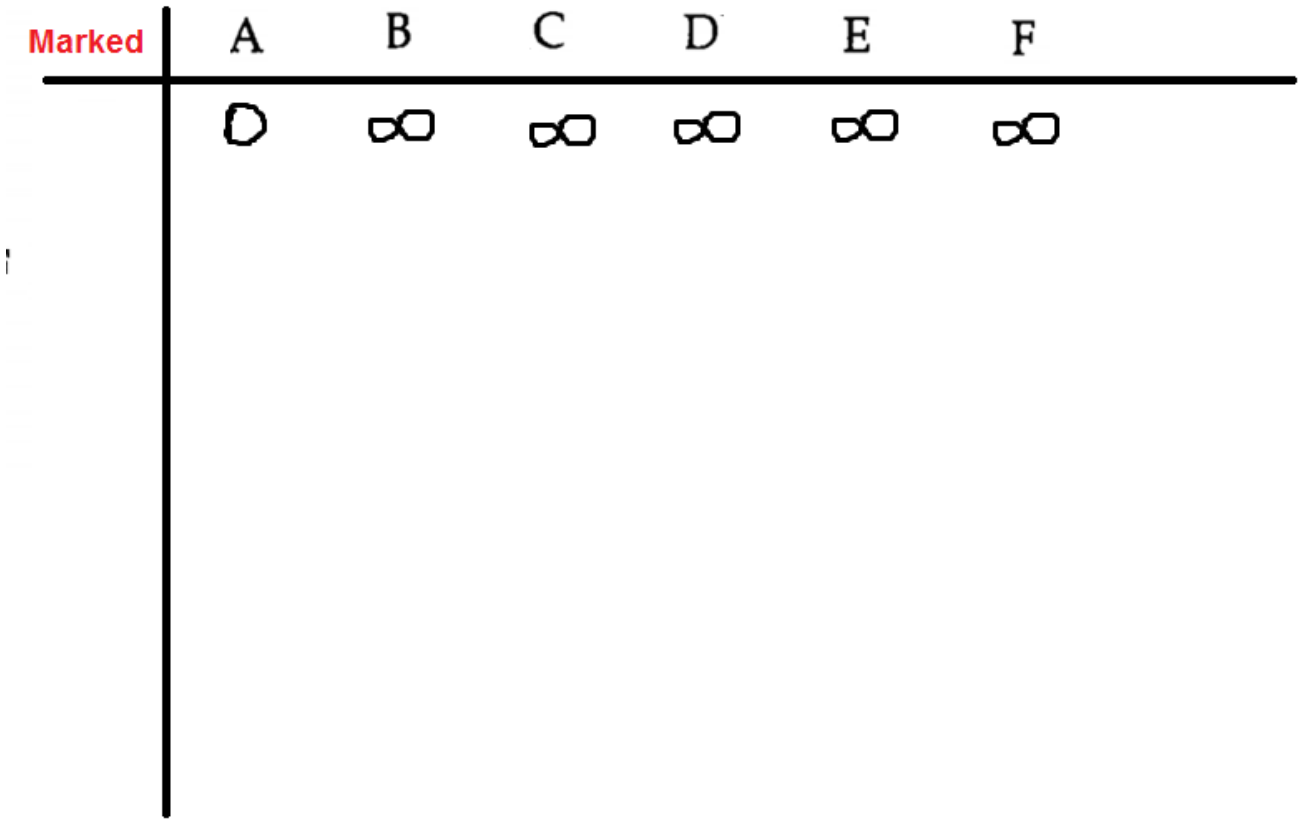
Similarly, there is another parallel edge from E to F, and as per the rules we remove that edge which one has greater value (6>3) i.e. 6. So we remove that edge of cost 6.



Step 3: create the weight matrix table .

(i) Set 0 to the source vertex and infinite to the remaining vertices.

Weight Matrix Table:-

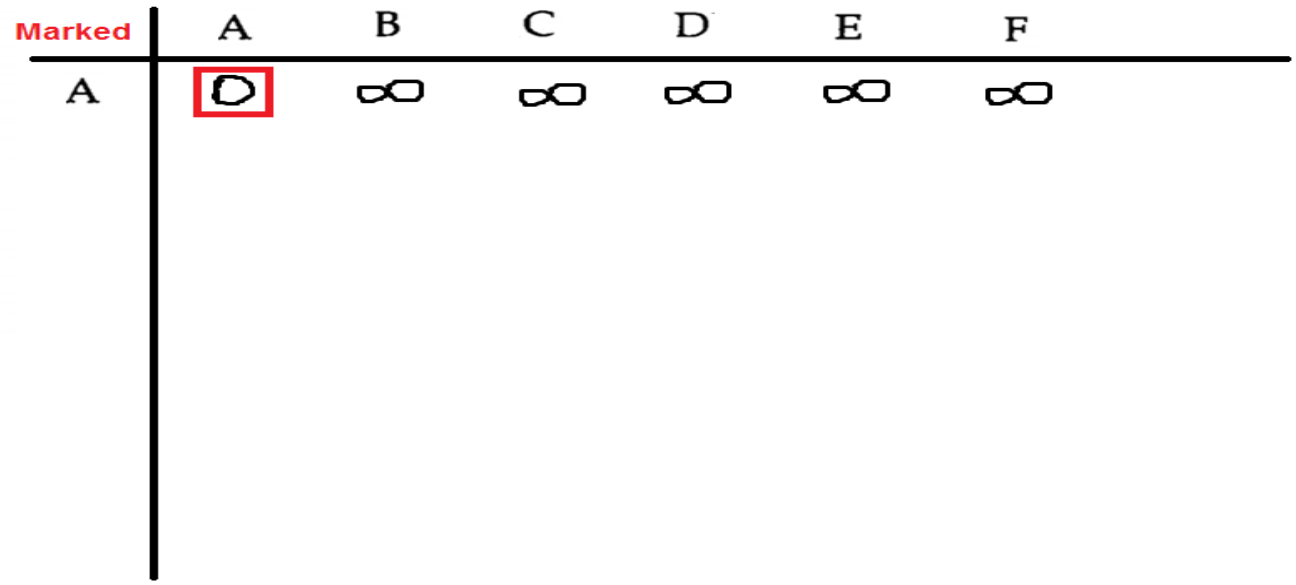


For all vertices, repeat (ii) and (iii)

(ii) Mark the smallest unmarked value and mark that vertex.

(iii) Find those vertices which are directly connected with marked vertex and update all.

So, smallest value in the first row is 0. So we marked 0 and put A as Marked.



Now, Find those vertices which are directly connected with A vertex. i.e. B, D and E.

Now, update B, D and E.

Update value formula:

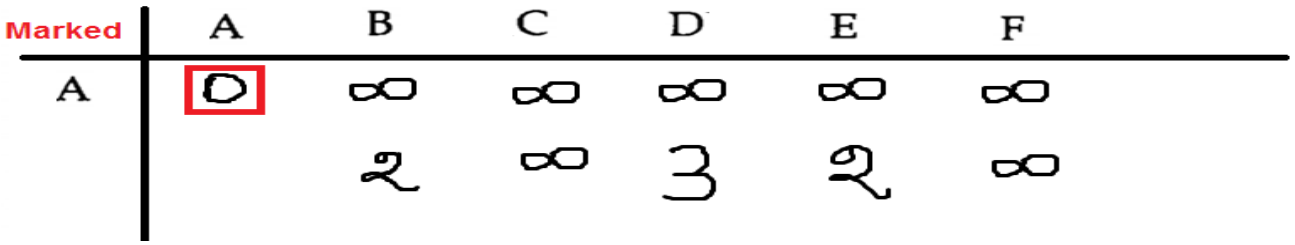
New Destination value = minimum (Old Destination value, Marked value+ Edge Weight)

New Value of B = minimum (∞, 0+2)= minimum(∞, 2)= 2

New Value of D = minimum (∞, 0+3)= minimum(∞, 3)= 3

New Value of E = minimum (∞, 0+2)= minimum(∞, 2)= 2

Value of all other vertices remain same. Now the table is:



Now repeat:

Smallest value in the second row is 2. If there are multiple smallest value we can marked any one of them. We marked first 2 and put B as Marked.

Marked	A	B	C	D	E	F
A	0	∞	∞	∞	∞	∞
B		2	∞	3	2	∞

Now, Find those vertices which are directly connected with B vertex. i.e. A, D and C. But A is already Marked, so we will not chose A.

Now, update D and C.

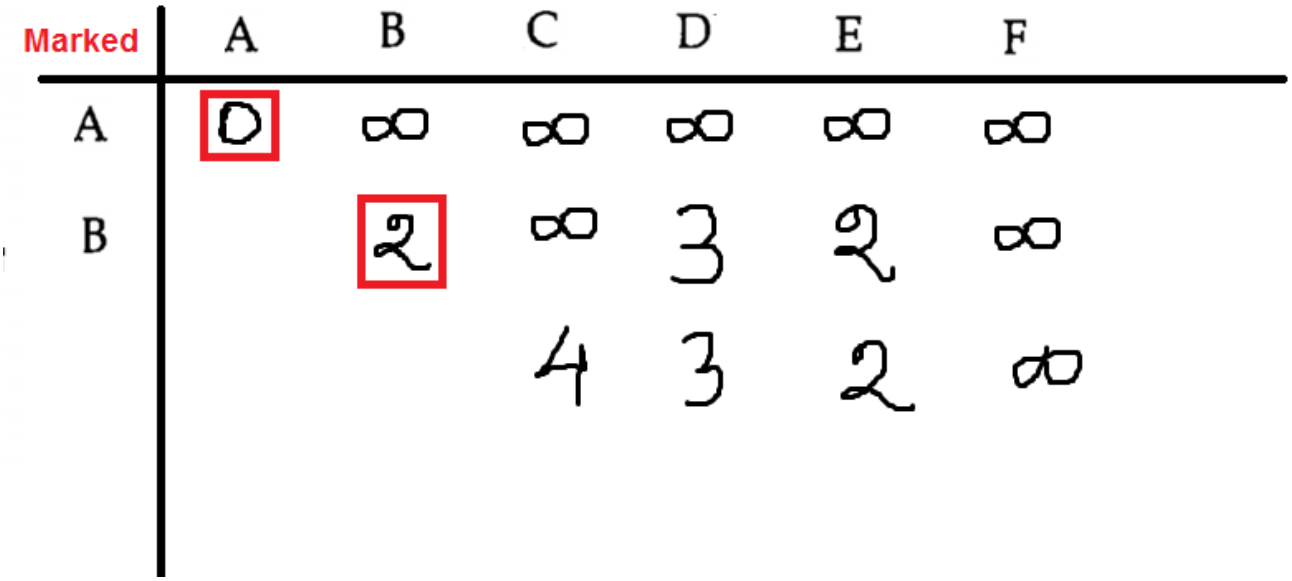
Update value formula:

New Destination value = minimum (Old Destination value, Marked value+ Edge Weight)

New Value of D = minimum (**3**, 2+4)= minimum(**3**, 6)= 3

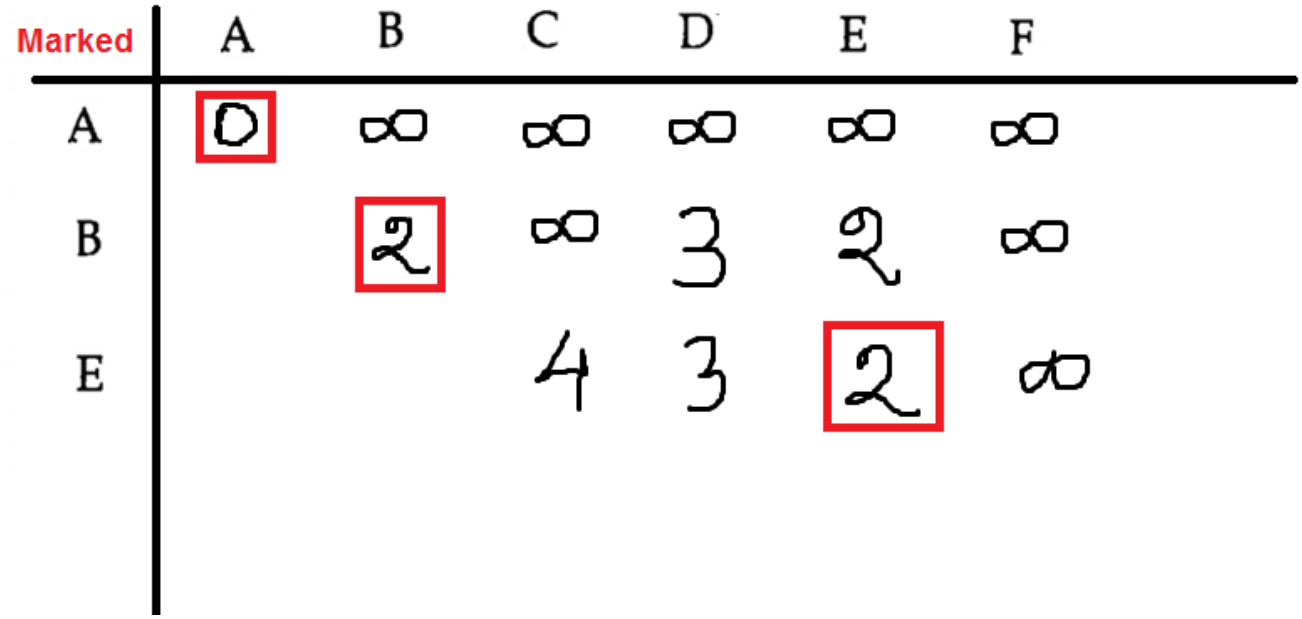
New Value of C = minimum (∞, 2+2)= minimum(∞, 4)= 4

Value of all other vertices remain same. Now the table is:



Now repeat:

Smallest value in the 3rd row is 2. So,We marked first 2 and put E as Marked.



Now, Find those vertices which are directly connected with E vertex. i.e. A, and F. But A is already Marked, so we will not chose A.

Now, update F.

Update value formula:

New Destination value = minimum (Old Destination value, Marked value+ Edge Weight)

New Value of F = minimum (∞, 2+3)= minimum(∞, 5)= 5

Value of all other vertices remain same. Now the table is:

Marked	A	B	C	D	E	F
A	0	∞	∞	∞	∞	∞
B		2	∞	3	2	∞
E			4	3	2	∞
			4	3		5

Now repeat:

Smallest value in the 4th row is 3. So,We marked first 3 and put D as Marked.

Marked	A	B	C	D	E	F
A	0	∞	∞	∞	∞	∞
B		2	∞	3	2	∞
E			4	3	2	∞
D			4	3		5

Now, Find those vertices which are directly connected with D vertex. i.e. A, B, C, and F. But A and B are already Marked, so we will not chose A and B.

Now, update C and F.

Update value formula:

New Destination value = minimum (Old Destination value, Marked value+ Edge Weight)

New Value of C = minimum (**4**, 3+2)= minimum(**4**, 5)= 4

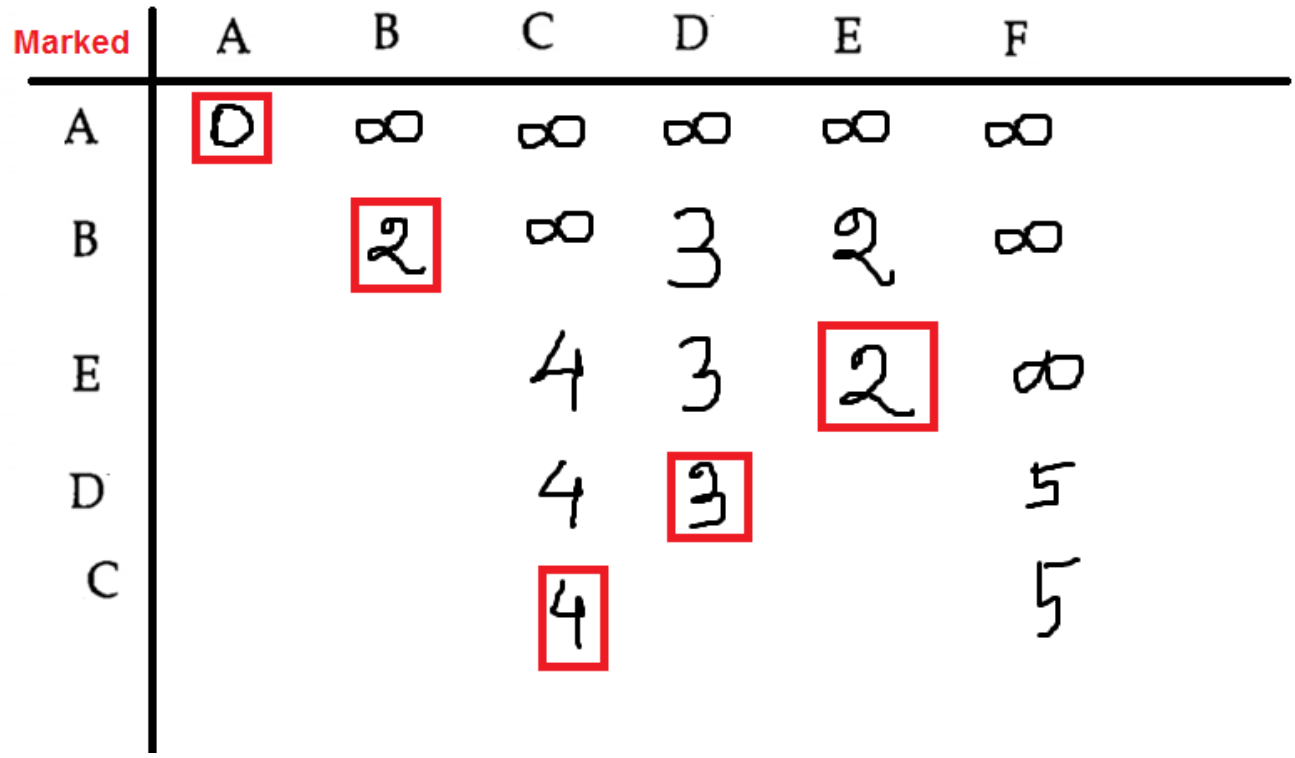
New Value of F = minimum (**5**, 4+3)= minimum(**5**, 7)= 5

Value of all other vertices remain same. Now the table is:

Marked	A	B	C	D	E	F
A	0	∞	∞	∞	∞	∞
B		2	∞	3	2	∞
E			4	3	2	∞
D			4	3		5
			4			5

Now repeat:

Smallest value in the 5th row is 4. So,We marked first 4 and put C as Marked.



Now, Find those vertices which are directly connected with C vertex. i.e. B, D, and F. But B and D are already Marked, so we will not chose B and D.

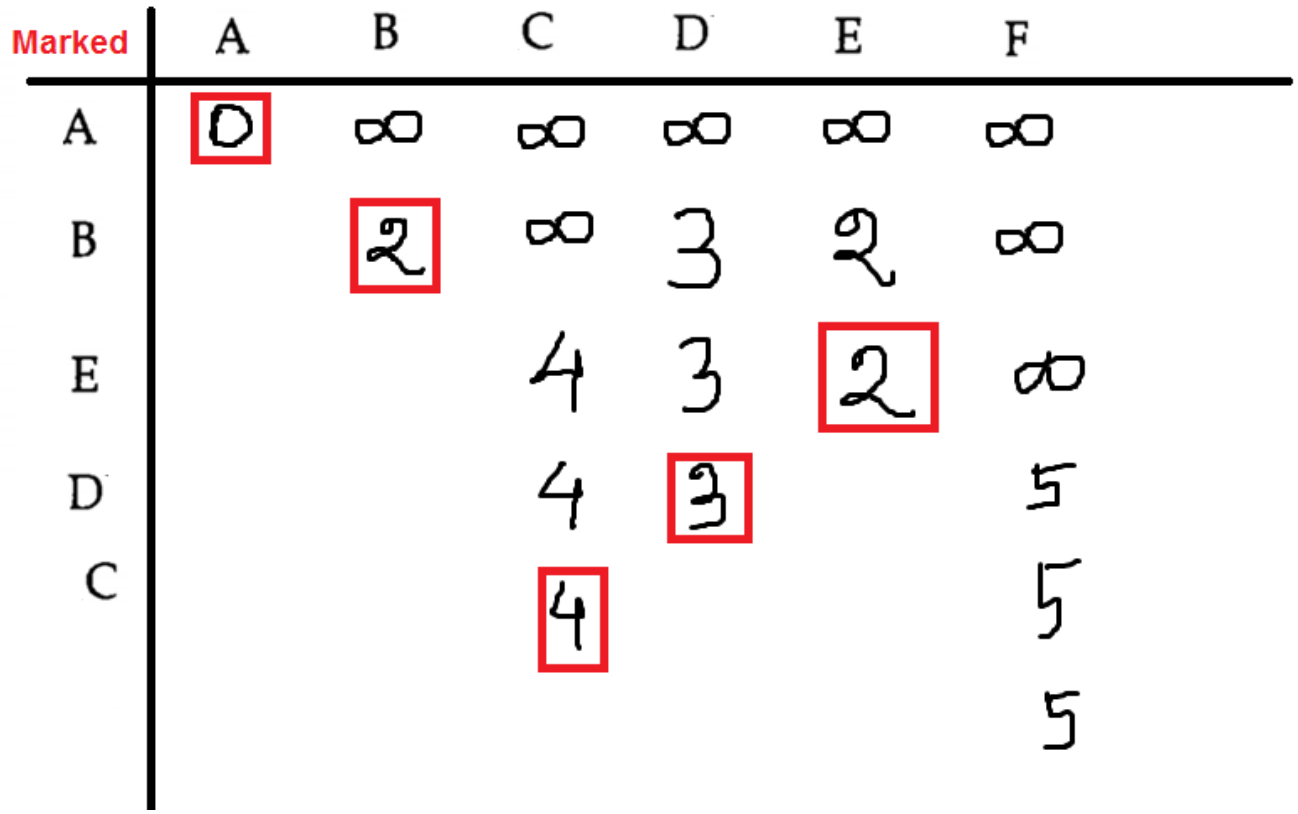
Now, update F.

Update value formula:

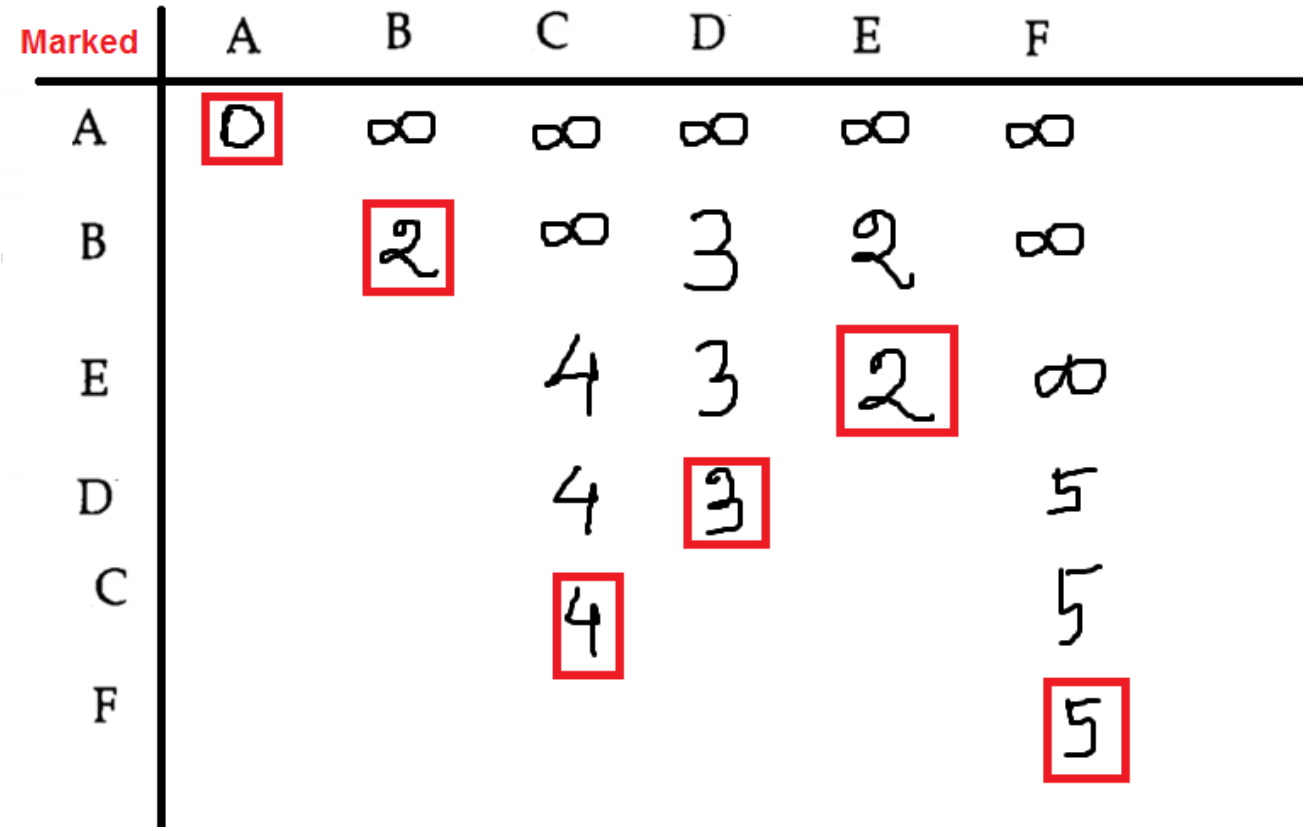
New Destination value = minimum (Old Destination value, Marked value+ Edge Weight)

New Value of F = minimum (**5**, 4+2)= minimum(**5**, 6)= 5

Value of all other vertices remain same. Now the table is:



As we see there is only one unmarked vertex remaining i.e. F. So marked 5 and put F as Marked.



So we cover all vertex. Step 3 is over.

Now,

Step 4: Find the shortest path from source to destination using **backtracking.**

Put destination vertex in ‘shortest path’

(i) Set pointer to the last marked value and put that vertex to the ‘shortest path’ list.

(ii) Move the pointer up until marked value is change.

(iii) If the last marked value has been changed then move the pointer to the marked value in that row and put that vertex to the ‘shortest path’ list.

Repeat (ii) to (iii) until the pointer will move to source vertex.

The last marked value is 5 so put F to the ‘shortest path’ list.

shortest path : F

now, Move the pointer up, the value is same. So again move the pointer up again, the value is same. So again move the pointer, the value has changed(infinity). So now, move the pointer to the marked value in that row which is 2 and put the vertex E to the ‘shortest path’ list.

shortest path : F -> E

now again, Move the pointer up, the value is same. So again move the pointer up again, the value has changed(infinity). So now, move the pointer to the marked value in that row which is 0 and put the vertex A to the ‘shortest path’ list.

shortest path : F -> E -> A

So minimum cost from A to F is: 2 + 3 = 5

Thank you so much friends.

Don’t forget to subscribe our channel

Channel URL: https://www.youtube.com/c/examaasaanhai