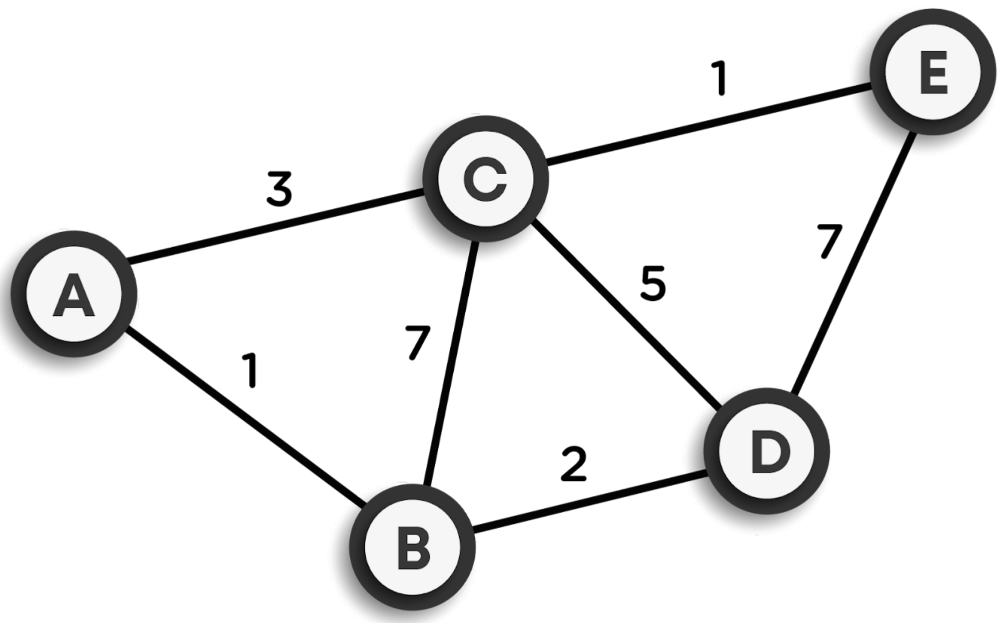
Dijkstra’s Algorithm: Minimum cost path

This algorithm is used to find the shortest distance between any two vertices in a weighted non-cyclic graph.

Here, we will be using a slight modification of the algorithm according to which we will be figuring out the minimum distance of all the vertices from the particular source vertex.

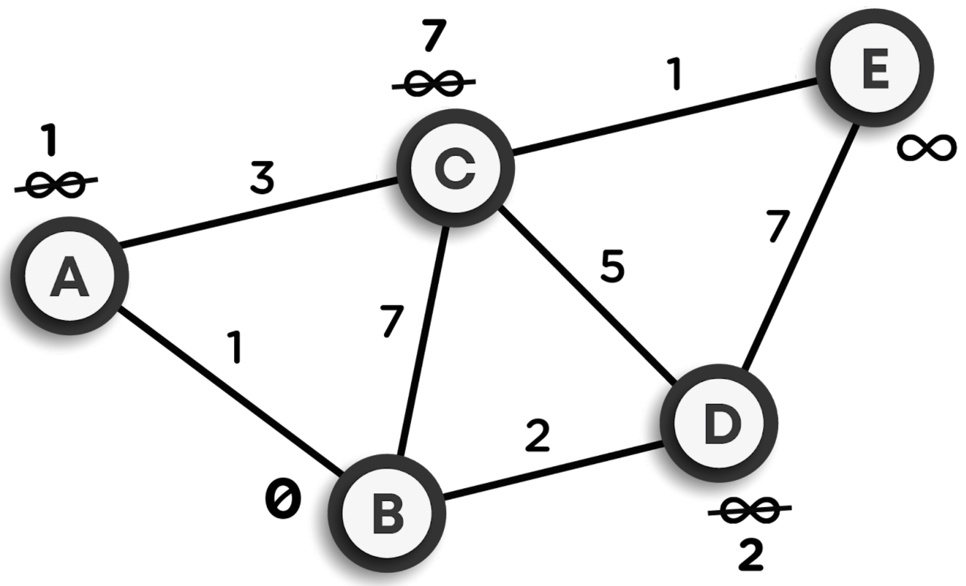
Let’s consider the algorithm with an example:

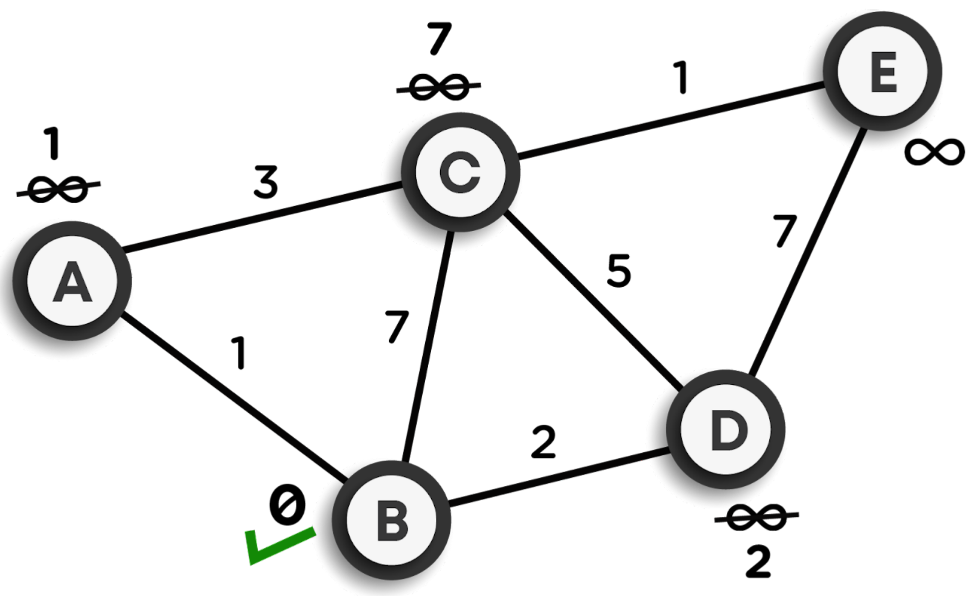
- We want to calculate the shortest path between the source vertex C and all other vertices in the following graph.



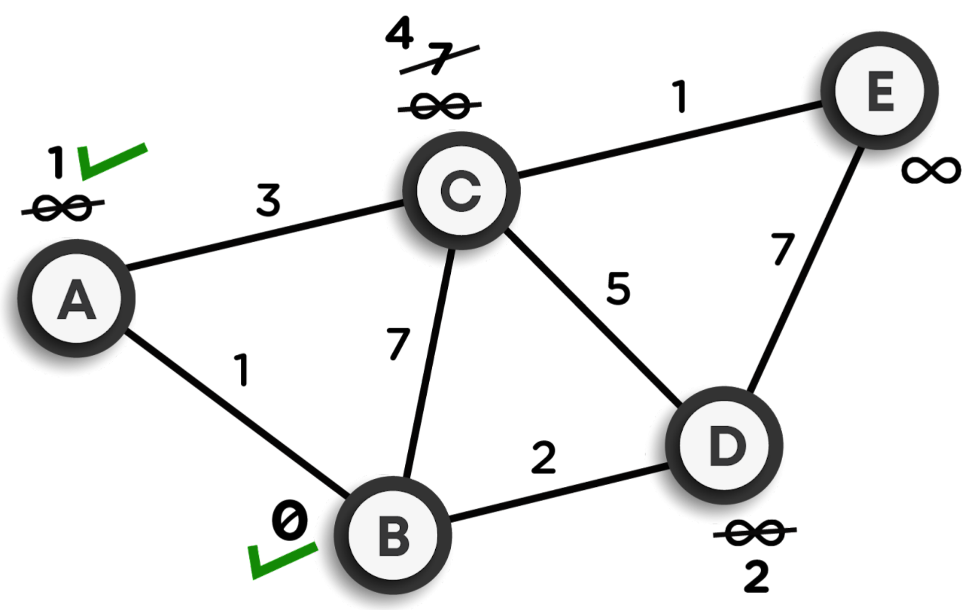
- While executing the algorithm, we will mark every node with its **minimum distance** to the selected node, which is C in our case. Obviously, for node C itself, this distance will be 0, and for the rest of the nodes, we will assume that the distance is infinity, which also denotes that these vertices have not been visited till now.

- Now, we will check for the neighbors of the current node, which are A, B, and D. Now, we will add the minimum cost of the current node to the weight of the edge connecting the current node and the particular neighbor node. For example, for node B, its weight will become minimum(infinity, 0+7) = 7. This same process is repeated for other neighbor nodes.

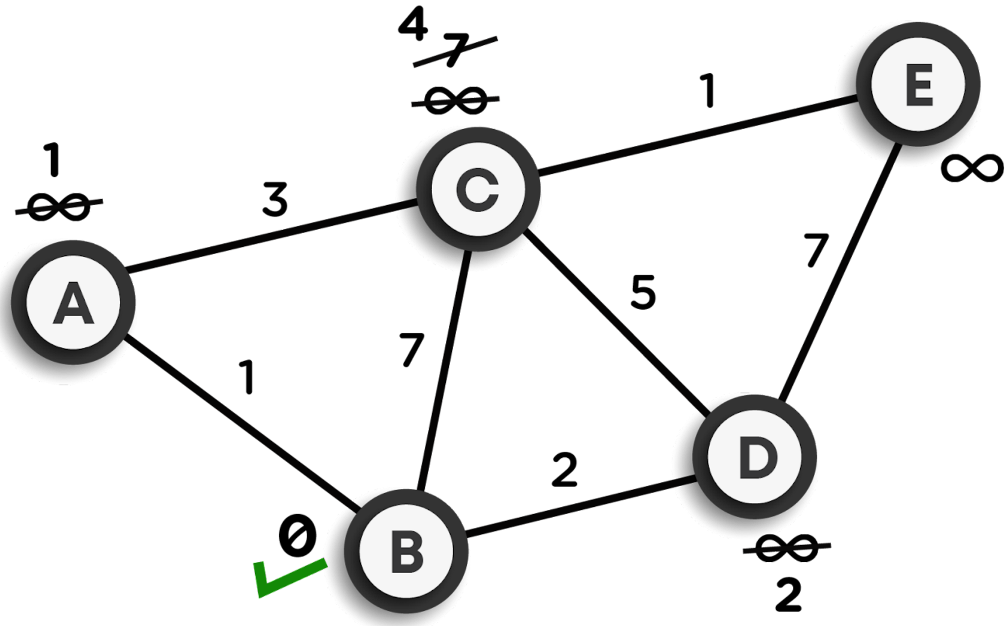




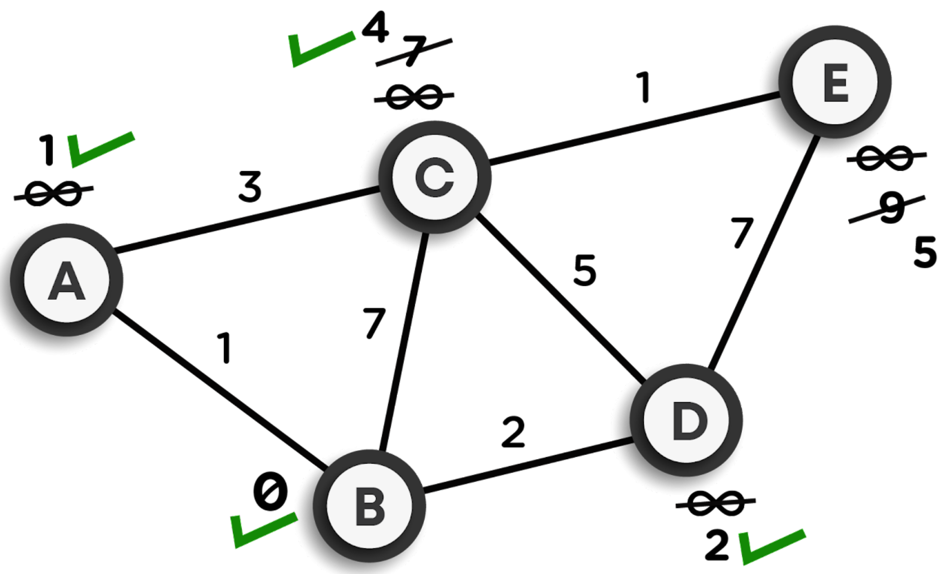
- Now, as we have updated the distance of all the neighbor nodes of the current node, we will mark the current node as visited.



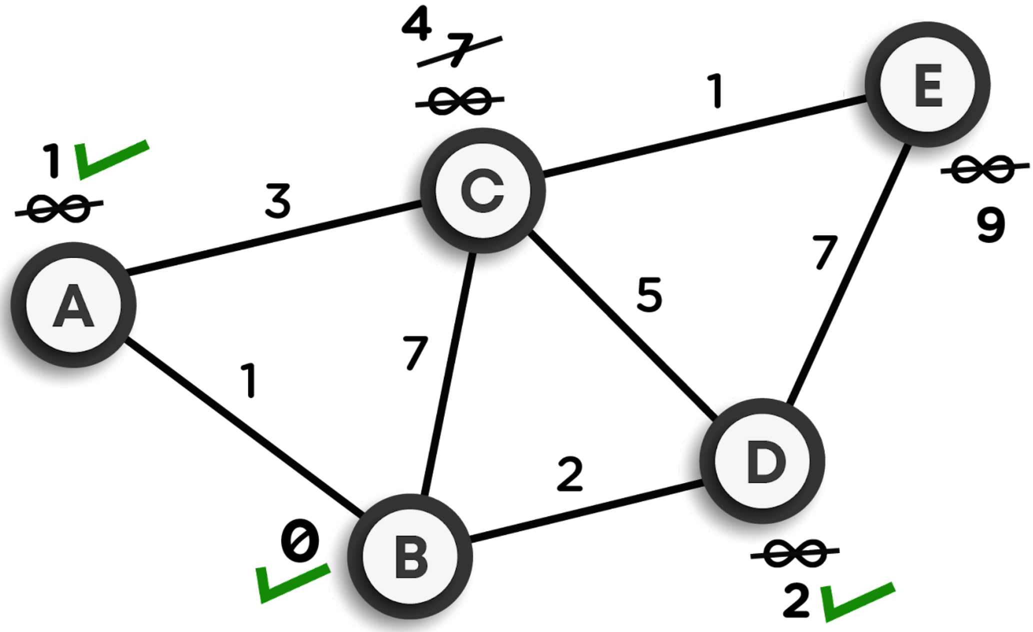
- After this, we will be selecting the minimum weighted node among the remaining vertices. In this case, it is node A. Take this node as the current node.
- Now, we will repeat the above steps for the rest of the vertices. The pictorial representation of the same is shown below:

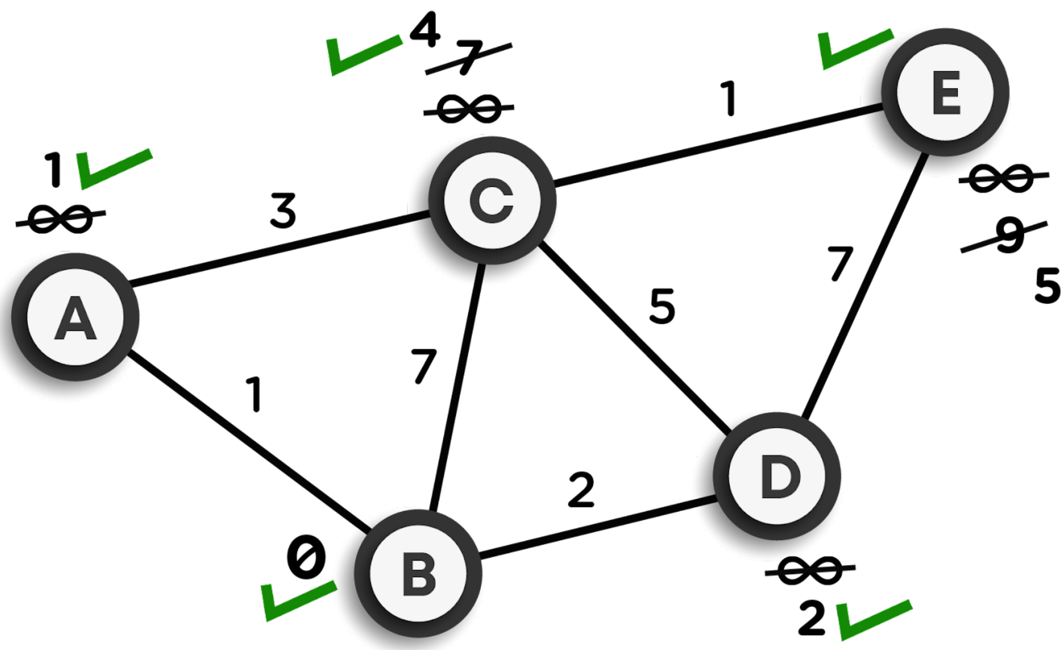


4
7
∞
1
∞
3
1
E
∞
C
7
5
A
1
7
D
2
0
B
∞
2

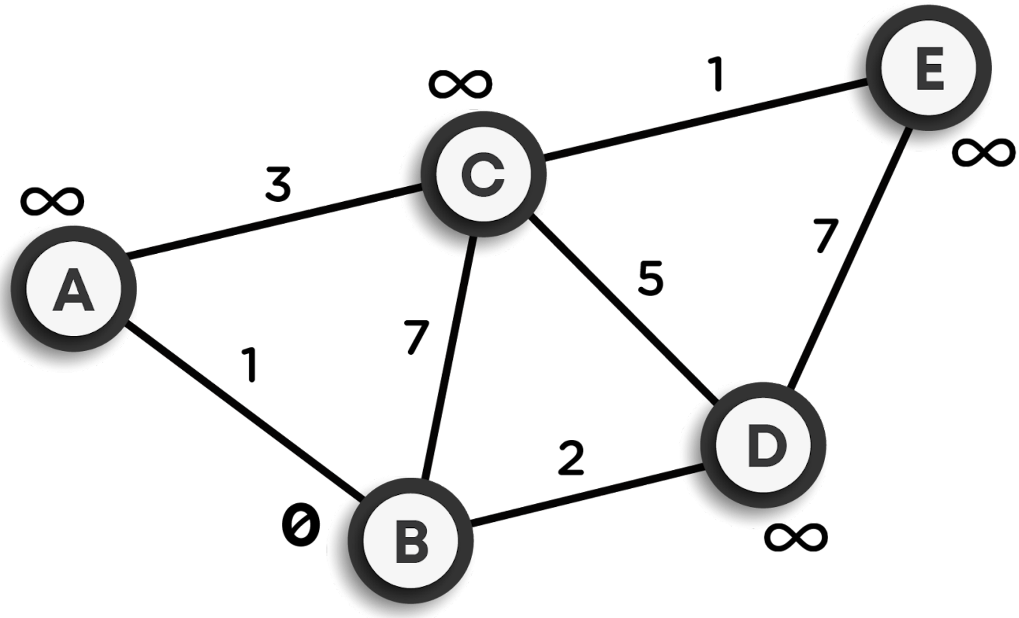


4
7
∞
1
E
1
∞
3
C
∞
9
5
7
A
5
1
7
D
2
0
B
∞
2





- Finally, we will get the graph as follows:
- The distances finally marked at each node are minimum from node C.



Time Complexity of Dijkstra's algorithm:

The time complexity is also the same as that of Prim's algorithm, i.e., **O(n2)**. This can be reduced by using the same approaches as discussed in Prim's algorithm's content.