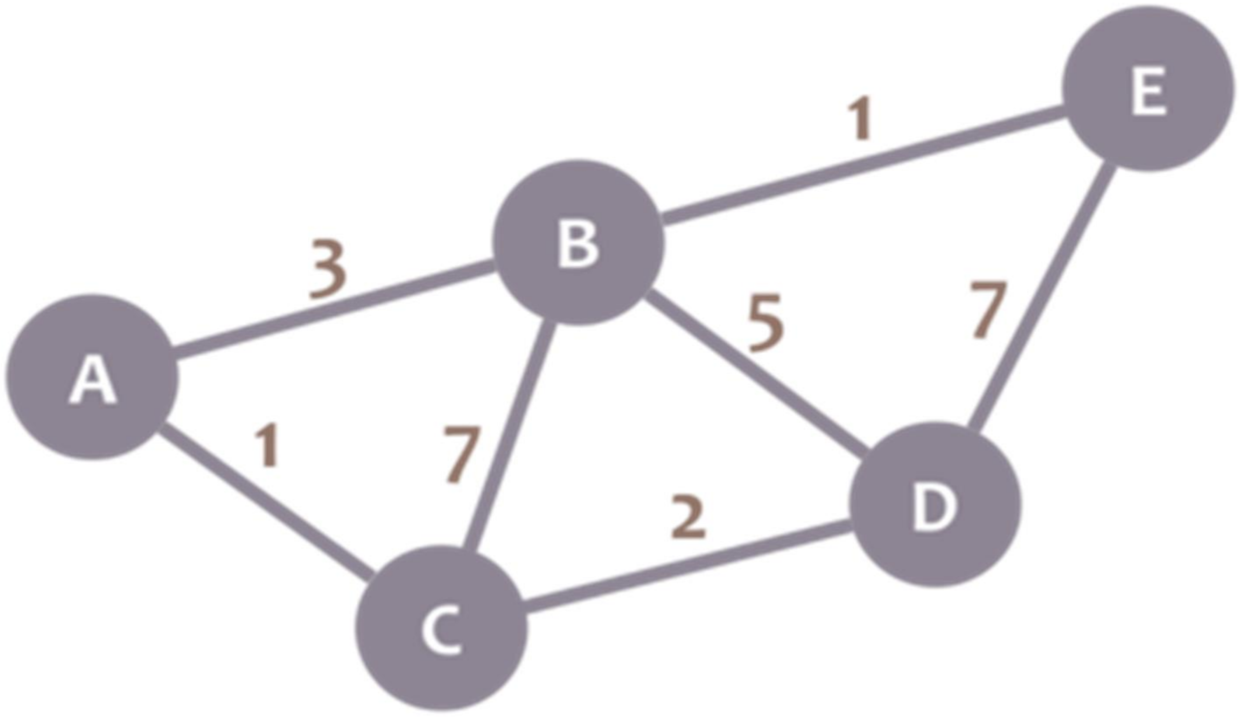
Link state routing algorithm - Dijkstra

In link state algorithm based routing protocol ensures every router has complete information about all other routers in the network and the traffic status of the network.

Dijkstra's Algorithm

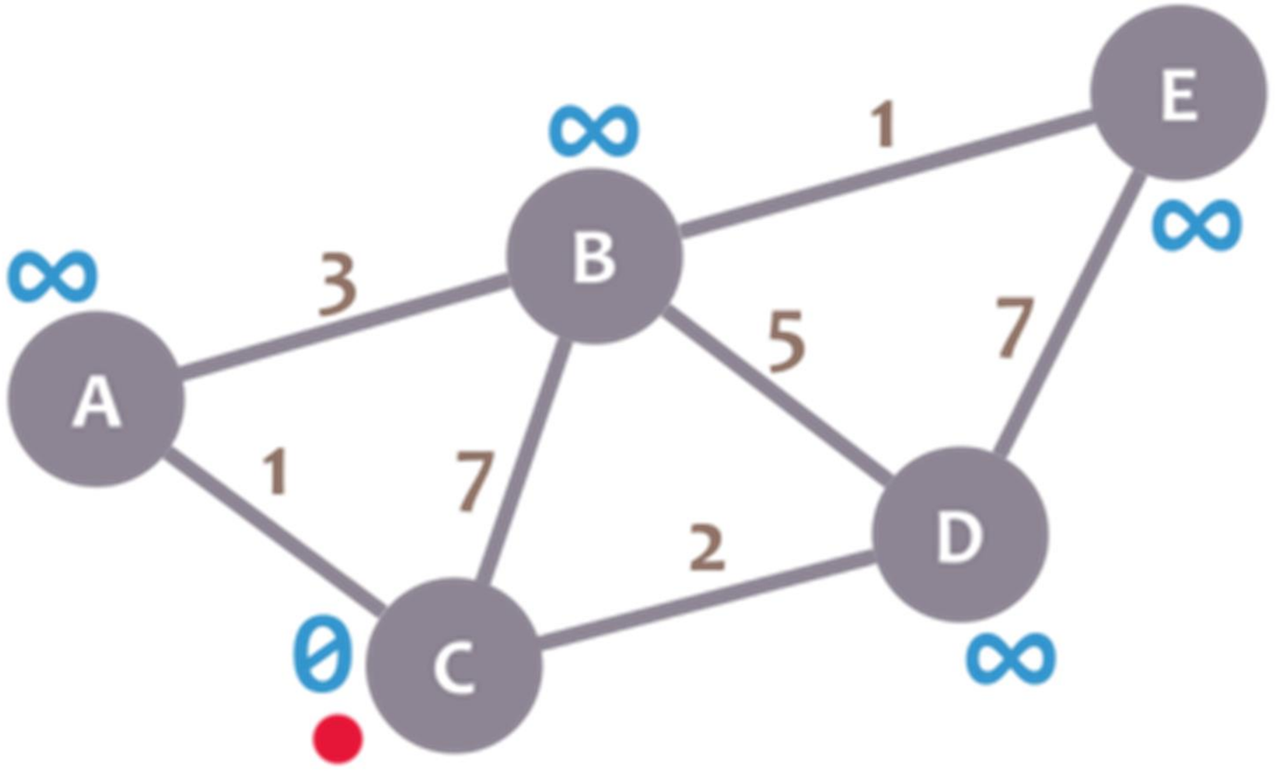
Dijkstra's Algorithm allows you to calculate the shortest path between SOURCE node and *every other node in the graph*.

Let's calculate the shortest path between node ‘C’ and the other nodes in our graph.

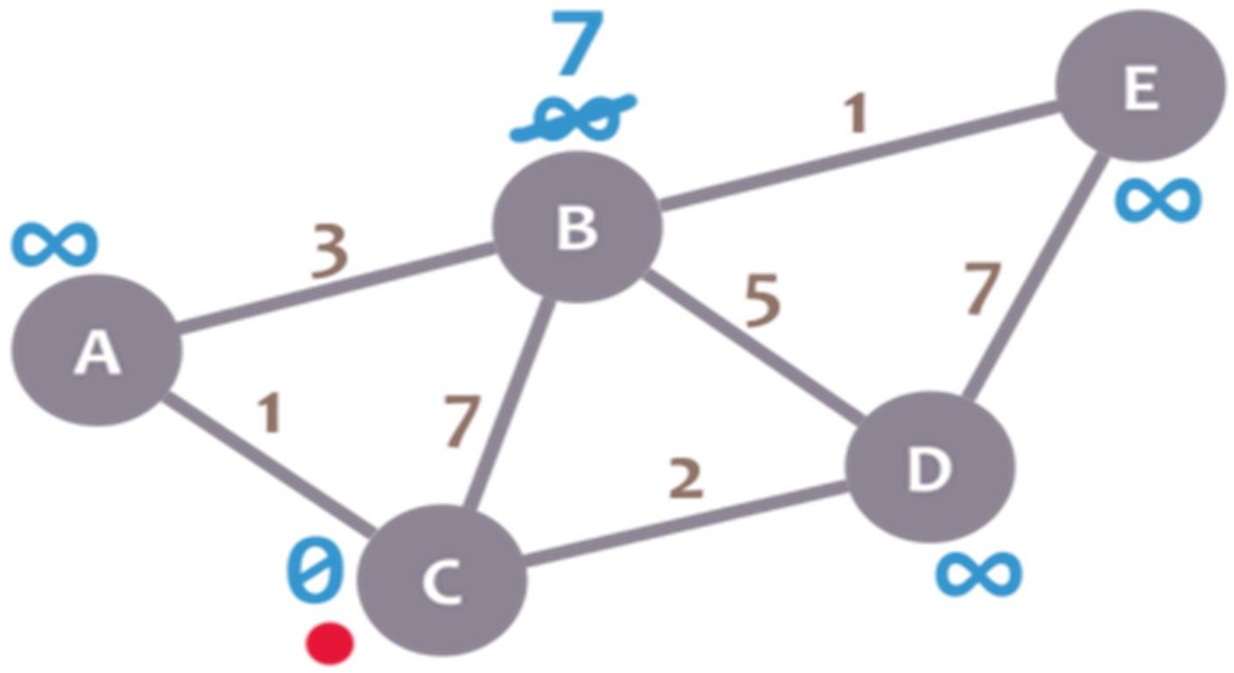


During the algorithm execution, mark every node with its *minimum distance* to node C. For node C, this distance is 0.

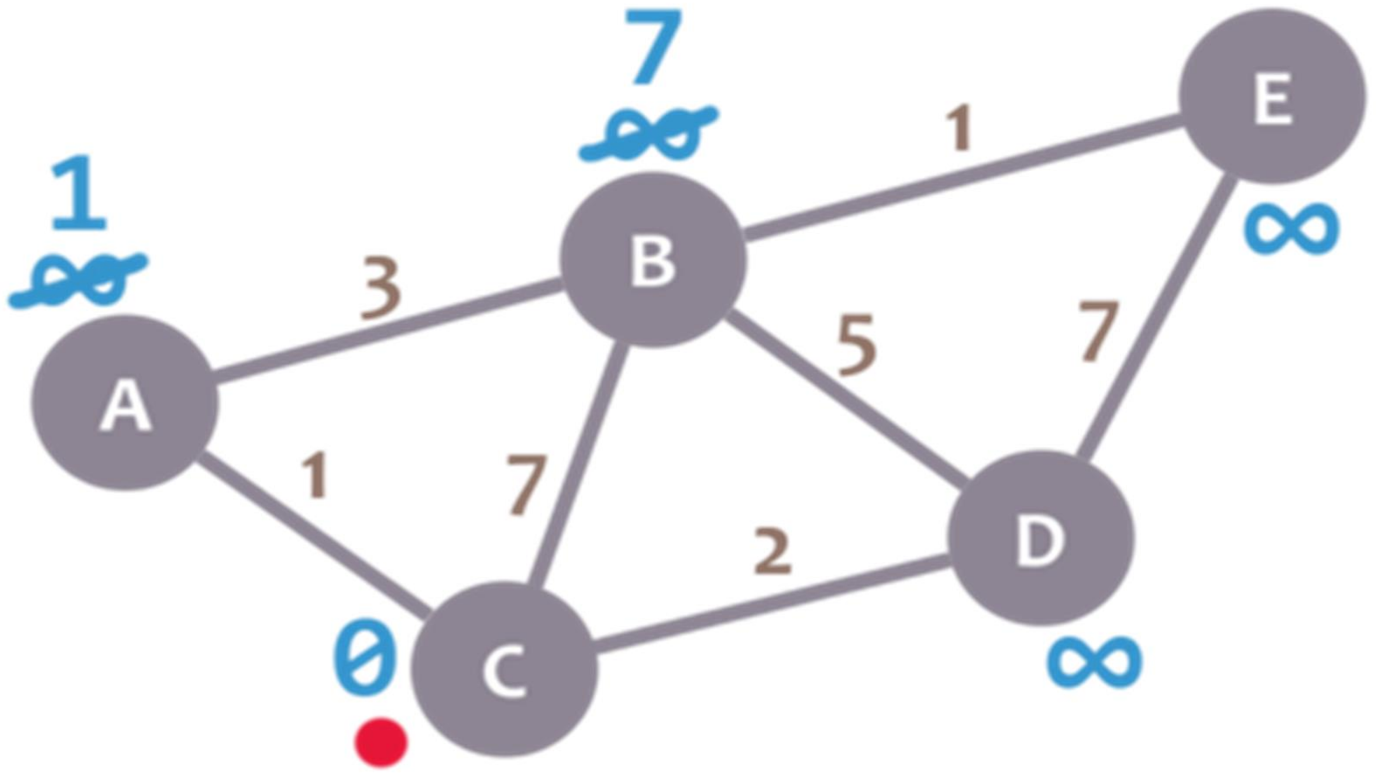
For the rest of nodes, as we still don't know that minimum distance, it starts being infinity (∞).



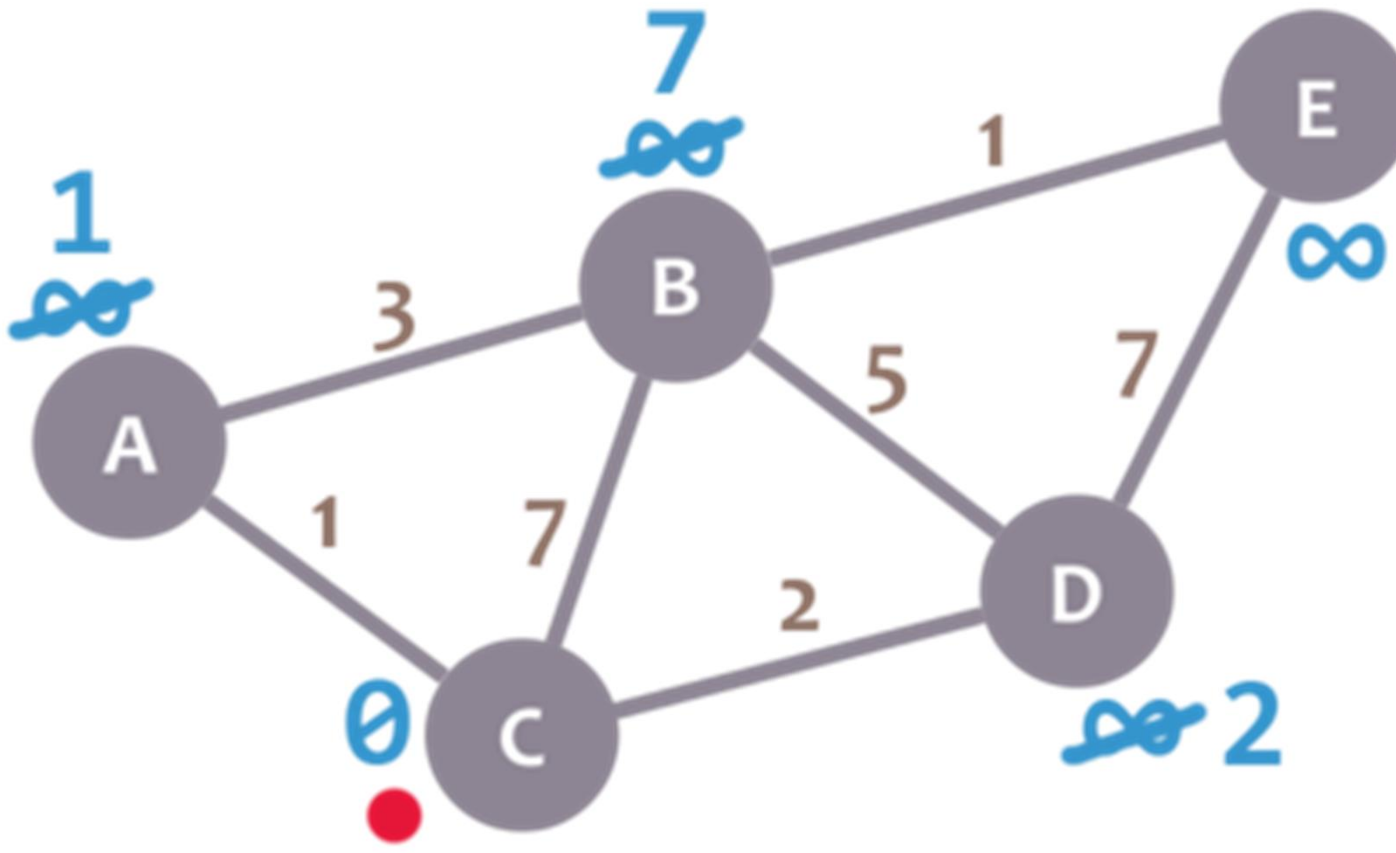
- We'll also have a *current node*. Initially, we set it to C. In the image, we mark the current node with a red dot.
- Now, we check the neighbours of our current node (A, B and D) in no specific order. Let's begin with B. We add the minimum distance of the current node (in this case, 0) with the weight of the edge that connects our current node with B (in this case, 7), and we obtain 0 + 7 = 7. We compare that value with the minimum distance of B (infinity).The lowest value is the one that remains as the minimum distance of B (in this case, 7 is less than infinity).



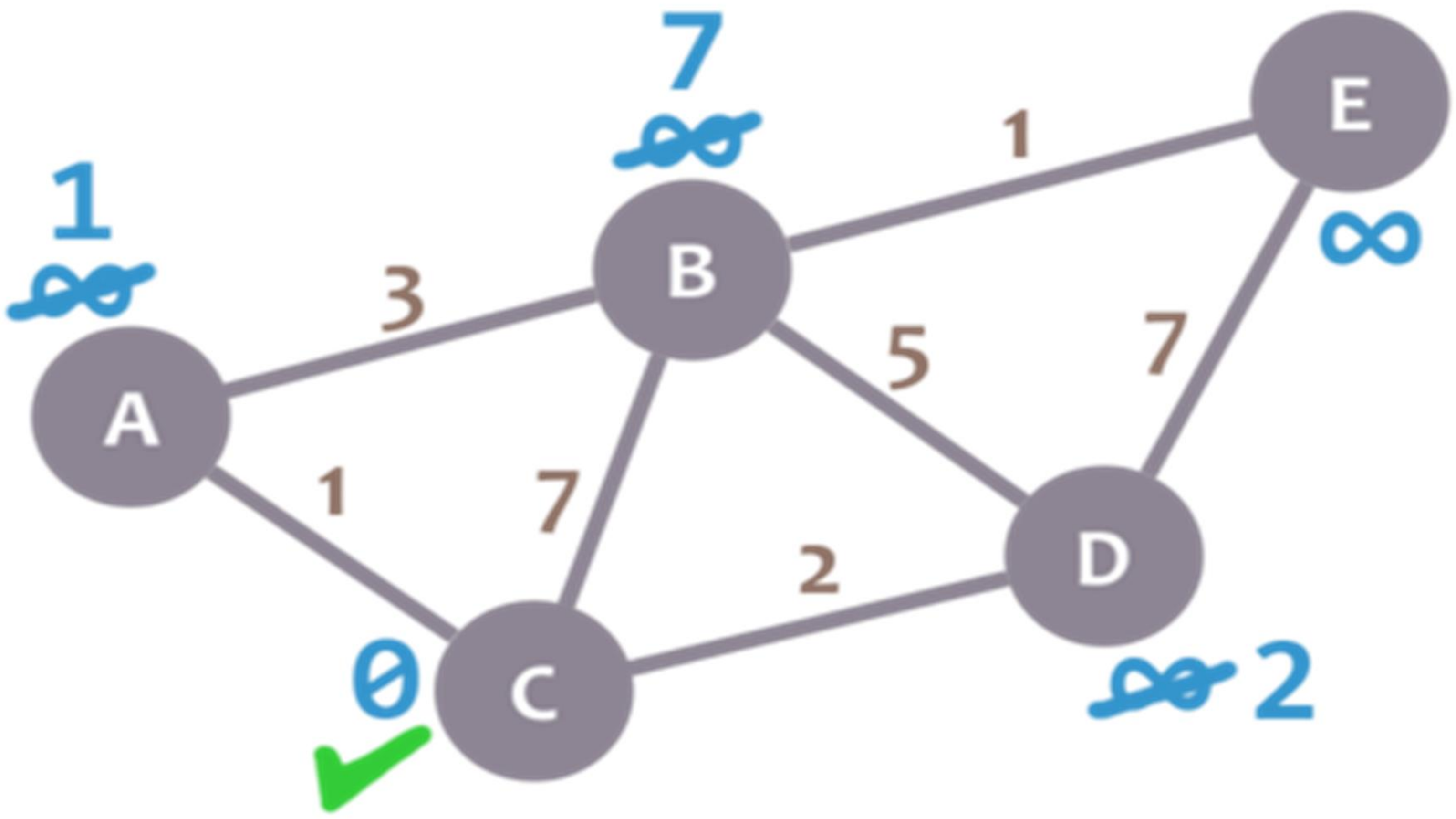
- Now, let's check neighbour A. We add 0 (the minimum distance of C, our current node) with 1 (the weight of the edge connecting our current node with A) to obtain 1.
- We compare that 1 with the minimum distance of A (infinity), and leave the smallest value.



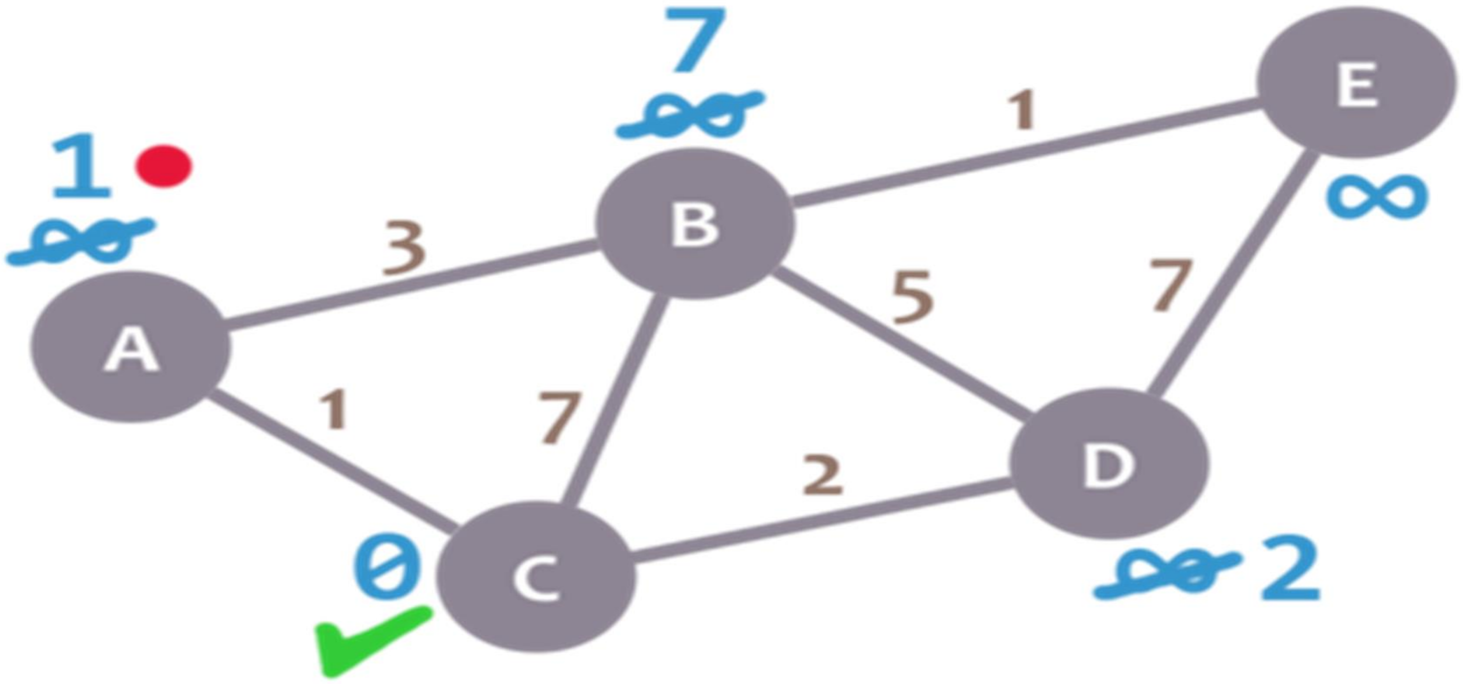
- Repeat the same procedure for D.



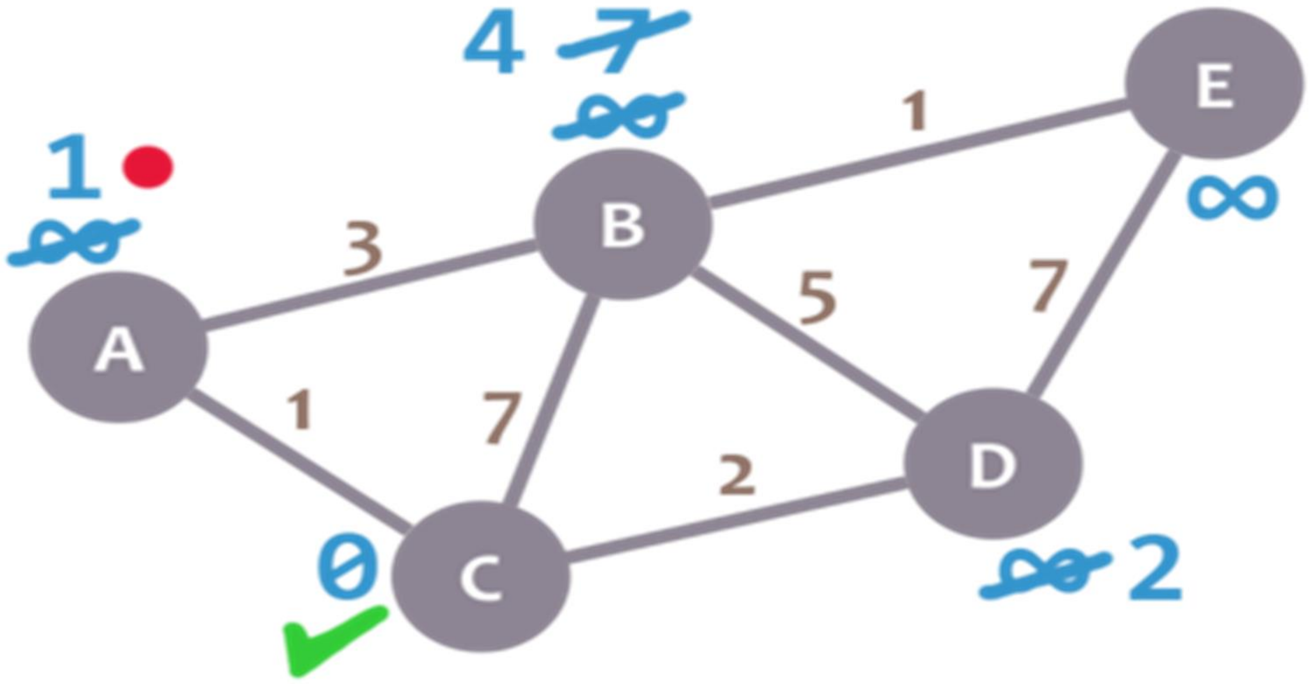
- We have checked all the neighbours of C. Hence, let us mark it as **visited**.
- Let's represent visited nodes with a **green check mark**.



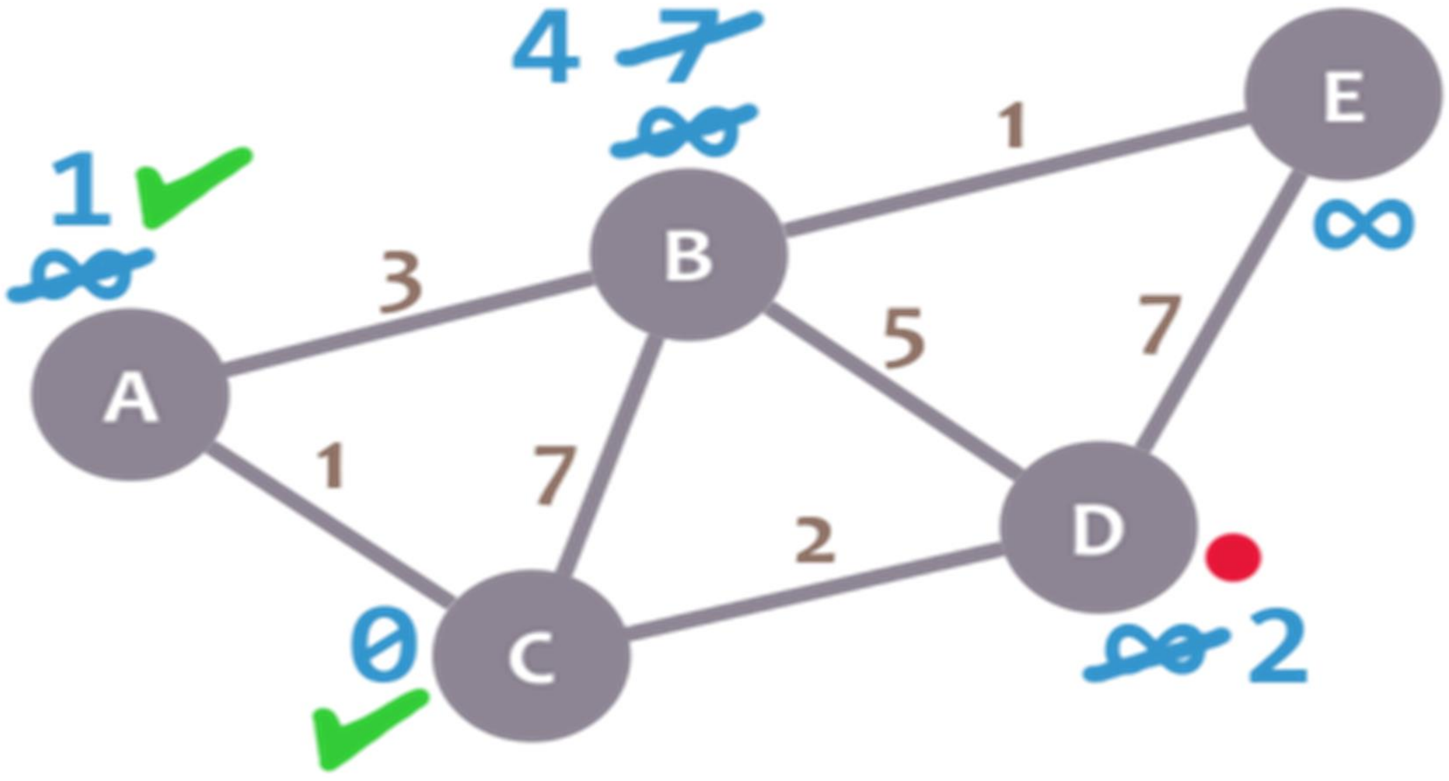
- We now need to pick a new **current node**. That node must be the unvisited node with the smallest minimum distance (so, the node with the smallest number and no check mark).
- That's A. Let's mark it with the **red dot**.



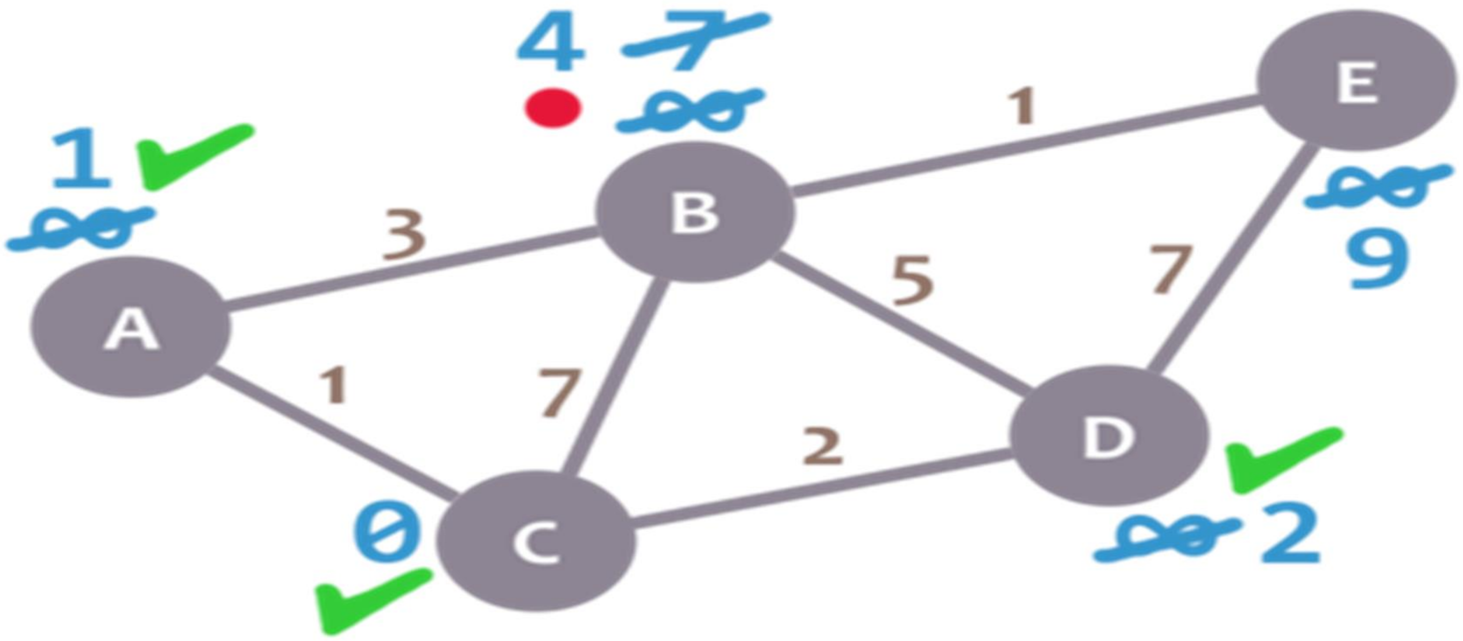
- And now we repeat the algorithm. We check the neighbours of our current node, **ignoring the visited nodes**. This means we only check B.
- For B, we add 1 (the minimum distance of A, our current node) with 3 (the weight of the edge connecting A and B) to obtain 4. We compare that 4 with the minimum distance of B (7) and leave the smallest value = 4.



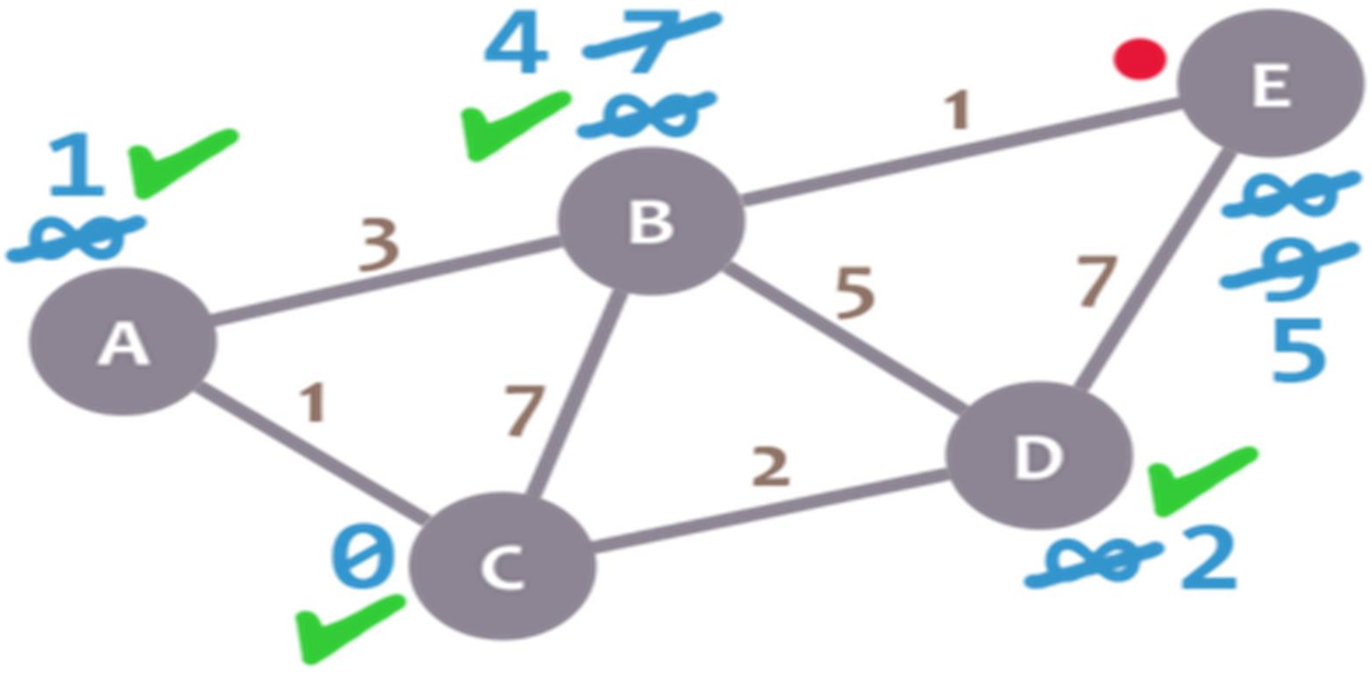
- Afterwards, let us mark A as visited and pick a new current node: D, which is the unvisited node with the smallest current distance.



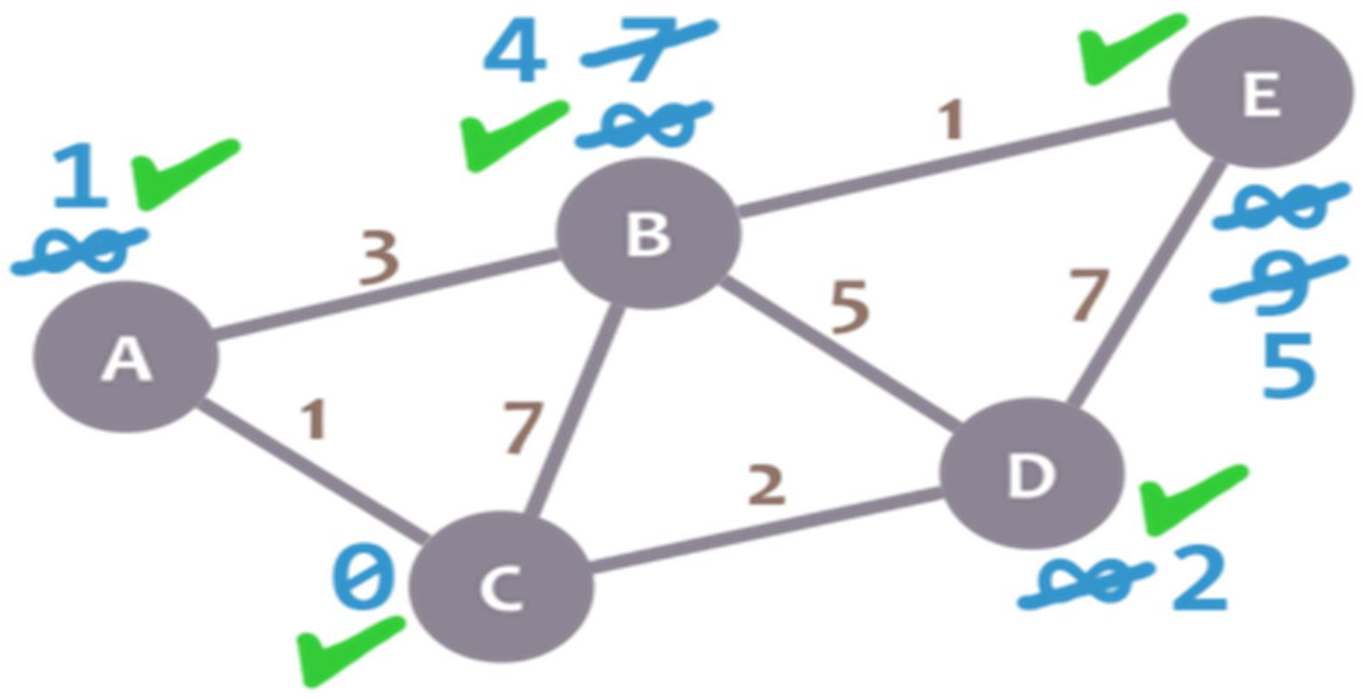
- We repeat the algorithm again. This time, we check B and E.
- For B, we obtain 2 + 5 = 7. We compare that value with B's minimum distance (4) and leave the smallest value (4). For E, we obtain 2 + 7 = 9, compare it with the minimum distance of E (infinity) and leave the smallest one (9).
- Let us mark D as visited and set our current node to B.



- Now, we need to check only E.
- 4 + 1 = 5, which is less than E's minimum distance (9), so we leave the 5.
- Then, let us mark B as visited and set E as the current node.

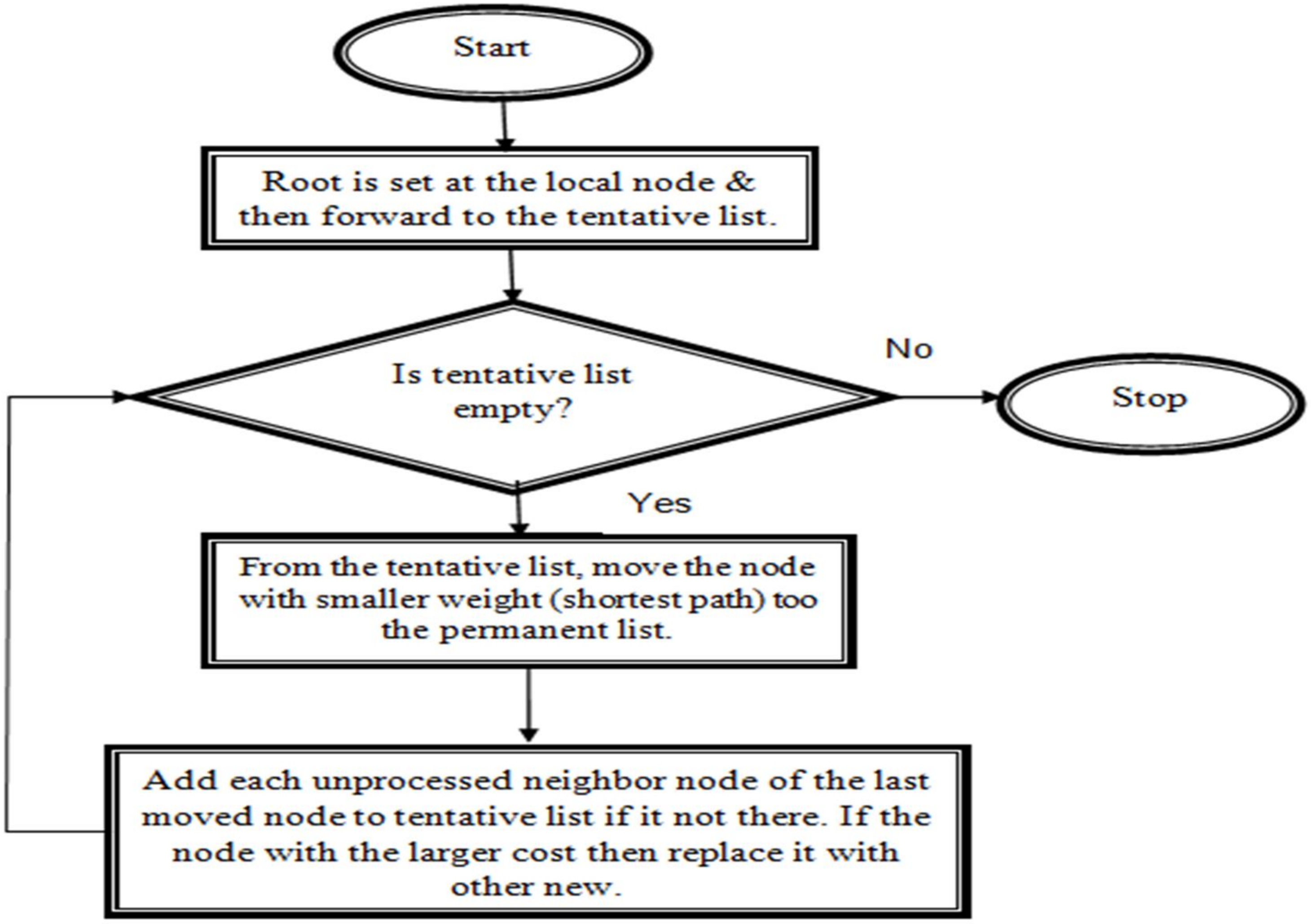


- Finally, E doesn't have any non-visited neighbours, so we don't need to check anything.
- Hence, Let us mark it as visited.



- As there are no more unvisited nodes, the algorithm has found the minimum distance of each node with source node C.

Dijkstra's Algorithm - Flowchart


Start
Root is set at the local node & then forward to the tentative list.
Is tentative list empty?
No
Stop
Yes
From the tentative list, move the node with smaller weight (shortest path) too the permanent list.
Add each unprocessed neighbor node of the last moved node to tentative list if it not there. If the node with the larger cost then replace it with other new.