

***A Project Report***

***On***

# **Dijktras-Map**

*conducted as part of the course CSE CS3130 Submitted by*

**Mehul kumar Sinha 209301105**

**Pradyumna Singh 209301309**

## ***GROUP-1***

*in partial fulfilment for the award of the degree  
of*

**BACHELOR OF TECHNOLOGY**

**In**

**Computer Science & Engineering**



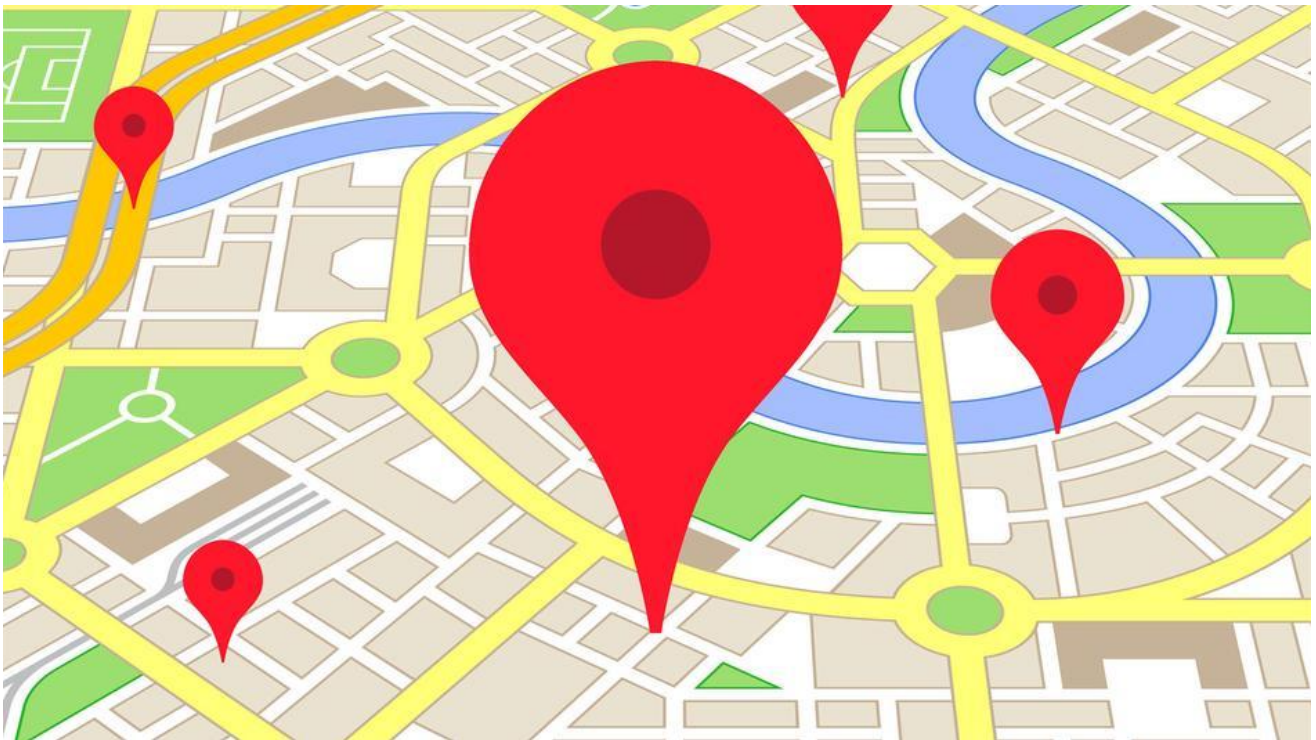
**MANIPAL UNIVERSITY  
JAIPUR**

**Department of Computer Science & Engineering,  
School of Computing and IT,  
Manipal University Jaipur,  
*November 2022***

## **PROBLEM STATEMENT**

Practically, often confuse travellers lack of information include the shortest route via city from the origin to the destination. Traveler usually have less option to choose what route (city) to take from one location to another location. Normally, travellers have fix time to travel, for example, they have only one week to travel due to this traveler need to find the shortest route in order for them shorten the travelling cost and save cost

This project is inspired by Google Map. It uses the Dijkstra algorithm to find the shortest path from a dummy map. You can get the time needed to reach a destination with different modes of transport.



## **AIM AND OBJECTIVE**

**Generally, the objective to develop Route Suggestion System (RSS) is to help the traveler around the world to find the shortest route trip from one location to location.**




- i. To design a Route Suggestion System (RSS) that can give information about the route (city) to choose for travelling from the origin location to the destination**
- ii. To implement PHP Server Page technique in the Route Suggestion System (RSS)**
- iii. To test the functionality of the system, so travellers will have an option to choose the shortest route (city) to take from one location to another location**

**Route Suggestion is a web based system. The scope of this system to help the travellers around the world to find the best route. In this Route Suggestion system, travellers, and the other one are the admin is the user of this system. Travellers need to make registration or login before travellers can choose the route (city). Travellers will enter the destination they want to go then the system will give an option the best route (city) to choose. Travellers have the option to choose from one location to another location via a different route (city). They will choose based on that. Admin's scope, the admin will check a monthly report, manage user and arrange a database.**

## MOTIVATION

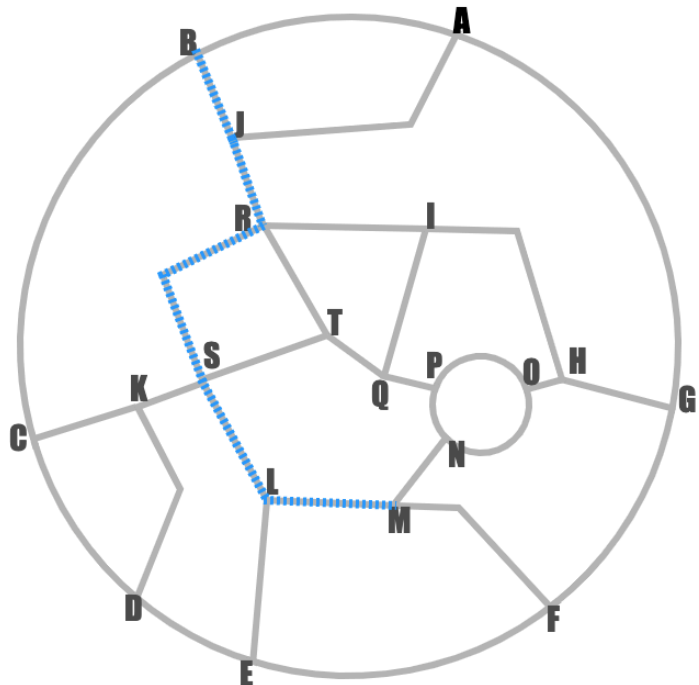
- Expansion of knowledge in the domain of DSA and DAA.
- Improve one's capability for an overall development in competitive and algorithmic skills of an individual.
- Using the attained knowledge as a medium that enables a person to be self-reliant in the domain of Development of algorithms.
- Equipping ourselves in these ways that's helps a persons in ways more than one.

START



Distance: 8 km  
Time: ~5 min

BJ	~1 min	1 Km
JR	~1 min	1 Km
RS	~2 min	3 Km
SL	~1 min	2 Km
	~1 min	



## Study and Analysis of Shortest Path Algorithms

Definition Shortest path problem is a fundamental technique in computer networking for route discovery. Utilising the shortest path algorithms overall costs of setting the network is reduced. Shortest path trees are made when shortest paths of remaining nodes are calculated from on a single root node. The algorithms determining the shortest path that is summarised in the following subsections have been classified under an efficiency analysis.

Dijkstra's algorithm calculates the shortest path between a given initial node to a single destination node and all other nodes of the graph when all edges have positive weight values. The algorithm selects any node  $l$  as a starting node. Then the algorithm builds a tree  $T$  that ultimately spans all vertices reachable from  $S_1$ . Vertices are inserted into a tree  $T$  according to the distances, i.e., first  $S_1$ , then the vertex closest to  $S_1$  and so on. Initially the distance between each node is set to infinity.

At the beginning of the algorithm the source node  $i$  is set as the current node. From this node the tentative distances of all unvisited neighbour nodes are calculated that are connected by edge. The current node is then marked as the visited node and the next current node is selected which has the minimum tentative distance from the previous current node.

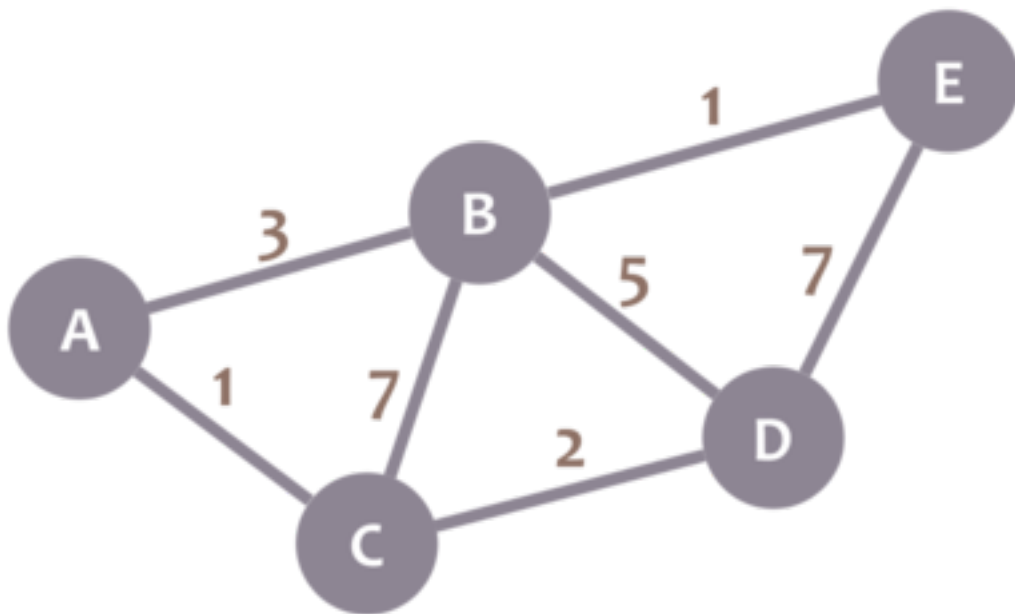
The algorithm is terminated when all nodes have been visited and the set of unvisited nodes is empty. This time can be reduced by using different variants of Dijkstra's algorithm.

## Algorithm

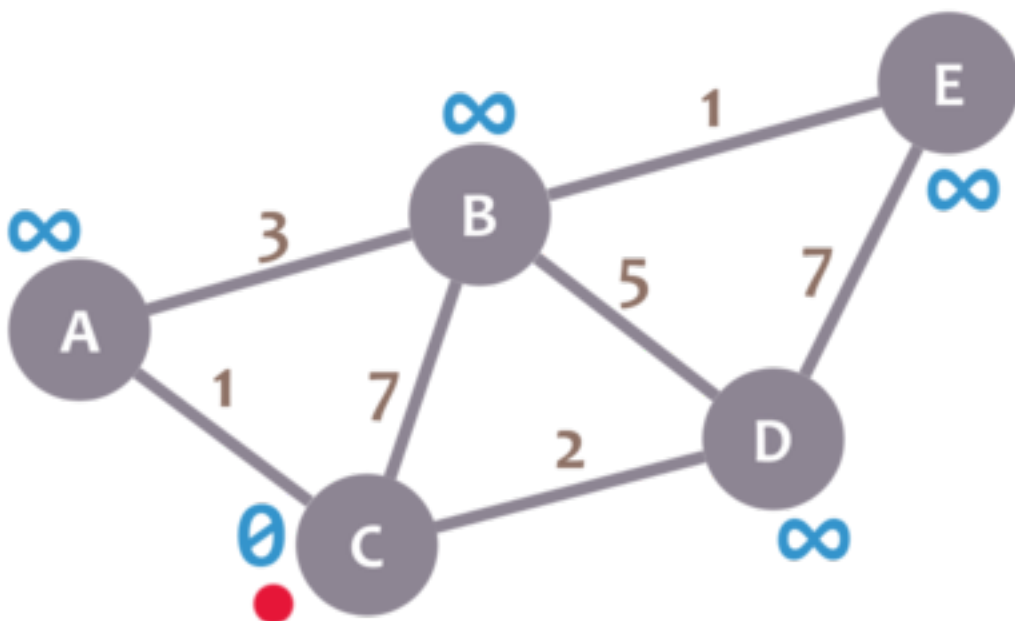
```
function dijkstra(G, S)
  for each vertex V in G
    distance[V] <- infinite
    previous[V] <- NULL
  If V != S, add V to Priority Queue Q
  distance[S] <- 0

  while Q IS NOT EMPTY
    U <- Extract MIN from Q
    for each unvisited neighbour V of U
      tempDistance <- distance[U] + edge_weight(U, V)
      if tempDistance < distance[V]
        distance[V] <- tempDistance
        previous[V] <- U
  return distance[], previous[]
```

## Working example Illustration

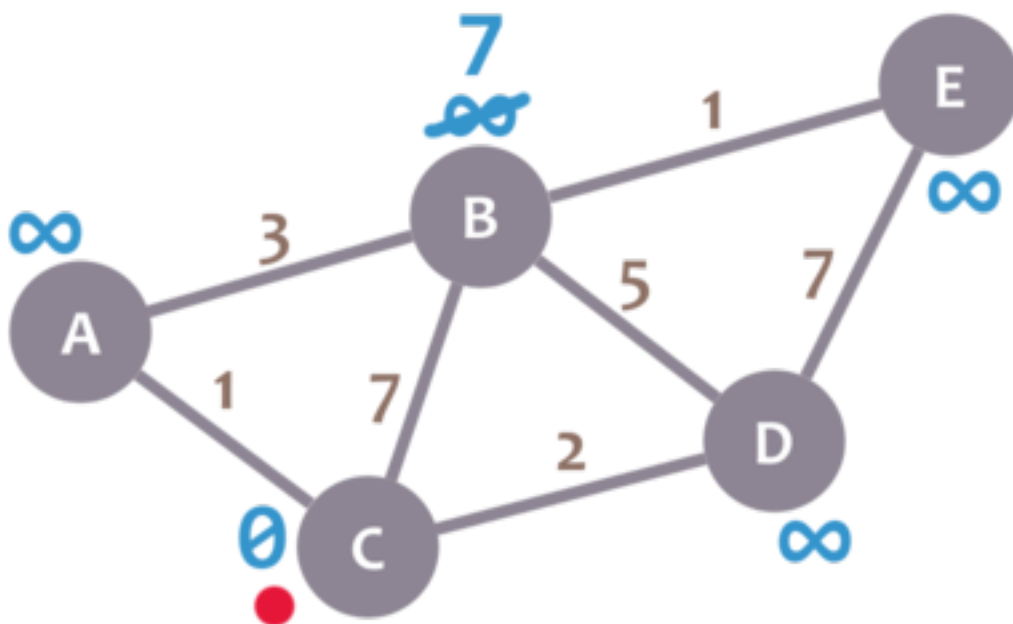


During the algorithm execution, we'll mark every node with its *minimum distance* to node C (our selected node). For node C, this distance is 0. For the rest of nodes, as we still don't know that minimum distance, it starts being infinity ( $\infty$ ):



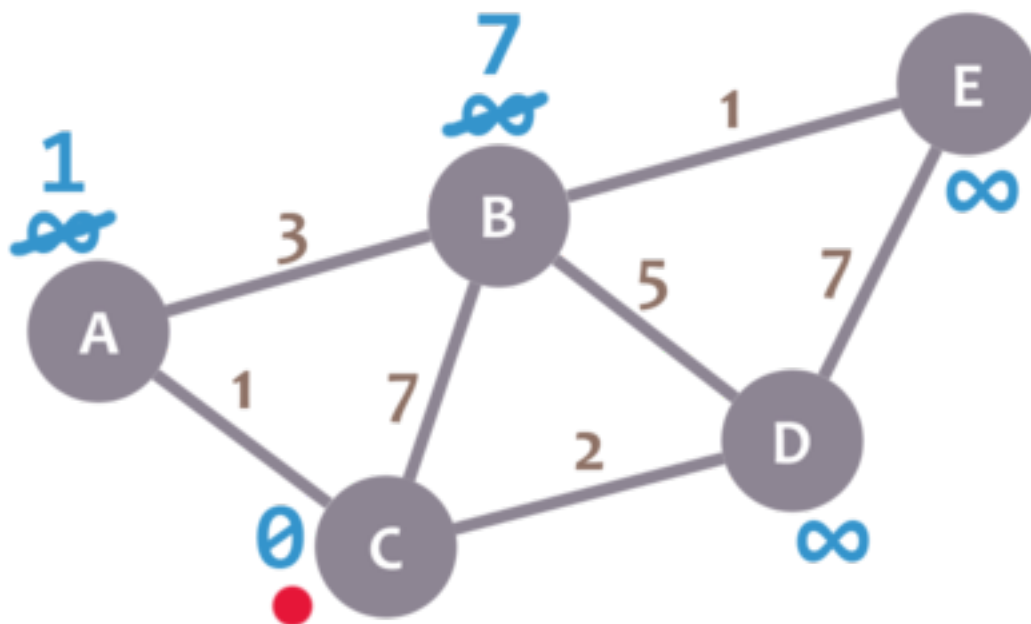
We'll also have a *current node*. Initially, we set it to C (our selected node). In the image, we mark the current node with a red dot.

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):

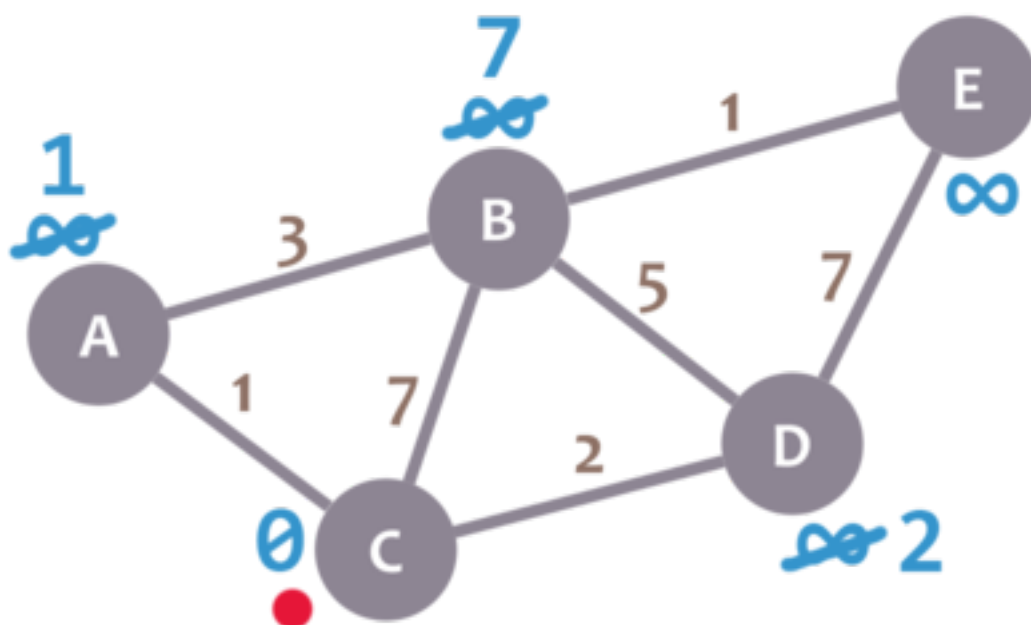


So far, so good. Now, let's check neighbour A. We add 0 (the minimum distance of C, our current node) with 1 (the weight of the edge connecting our current node with A) to obtain 1. We compare that 1 with the minimum distance of A (infinity), and leave the smallest value:

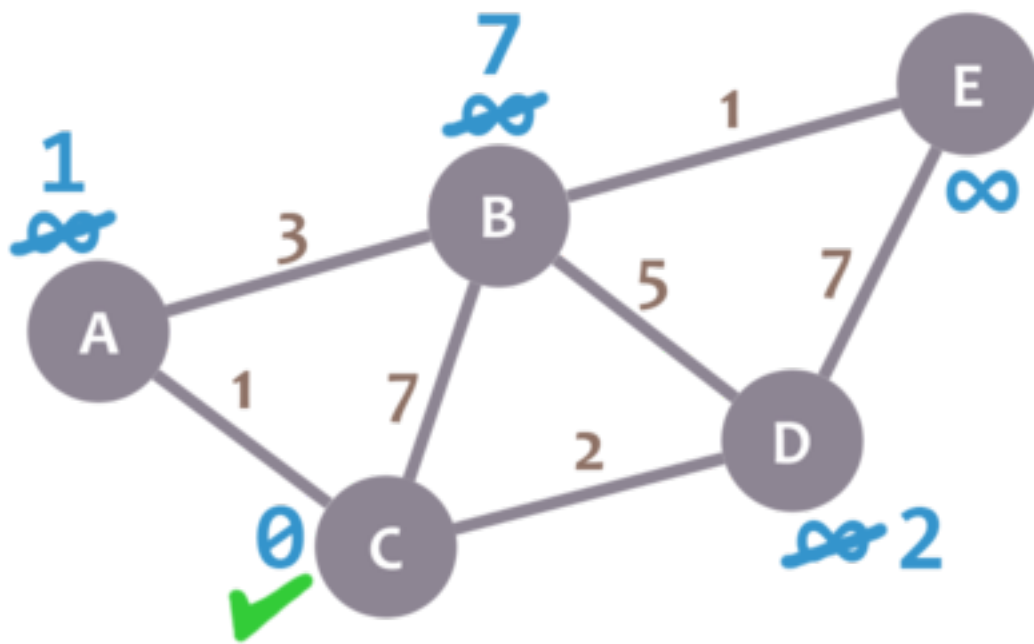




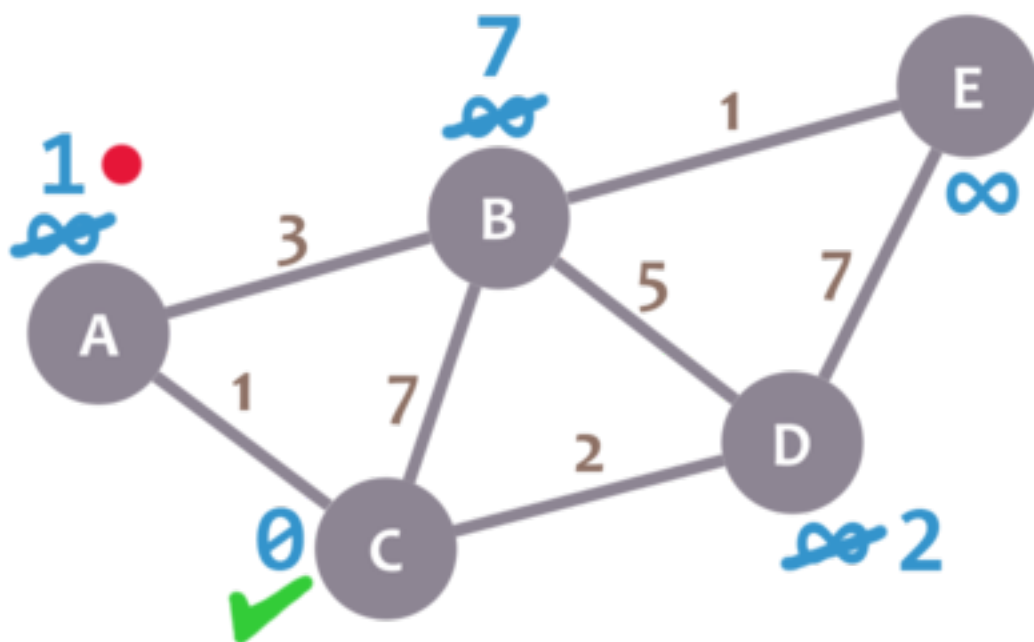
OK. Repeat the same procedure for D:



Great. We have checked all the neighbours of C. Because of that, we mark it as *visited*. Let's represent visited nodes with a green check mark:

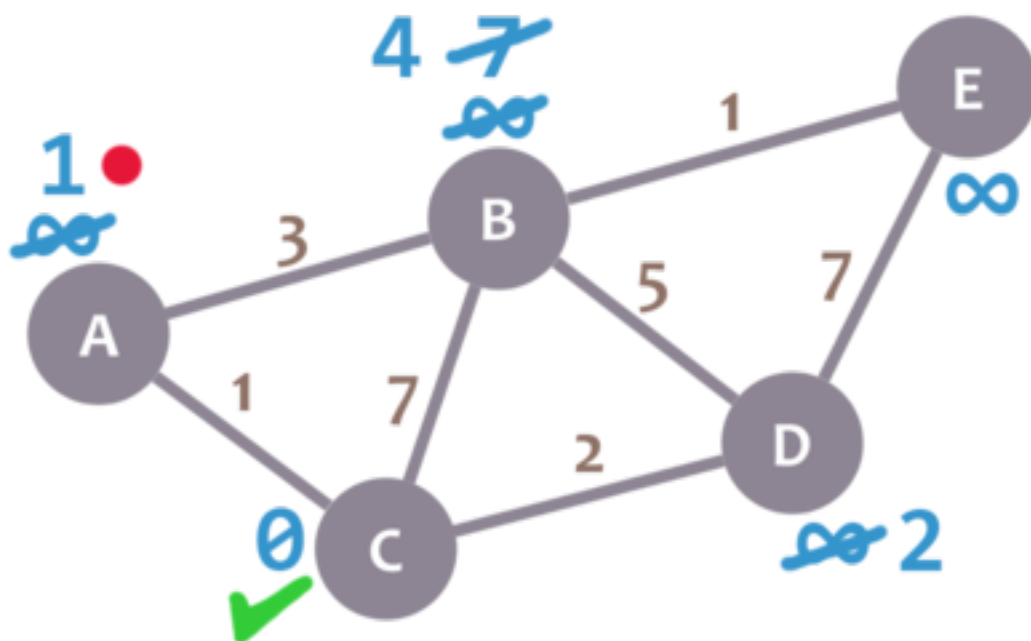


We now need to pick a new *current node*. That node must be the unvisited node with the smallest minimum distance (so, the node with the smallest number and no check mark). That's A. Let's mark it with the red dot:

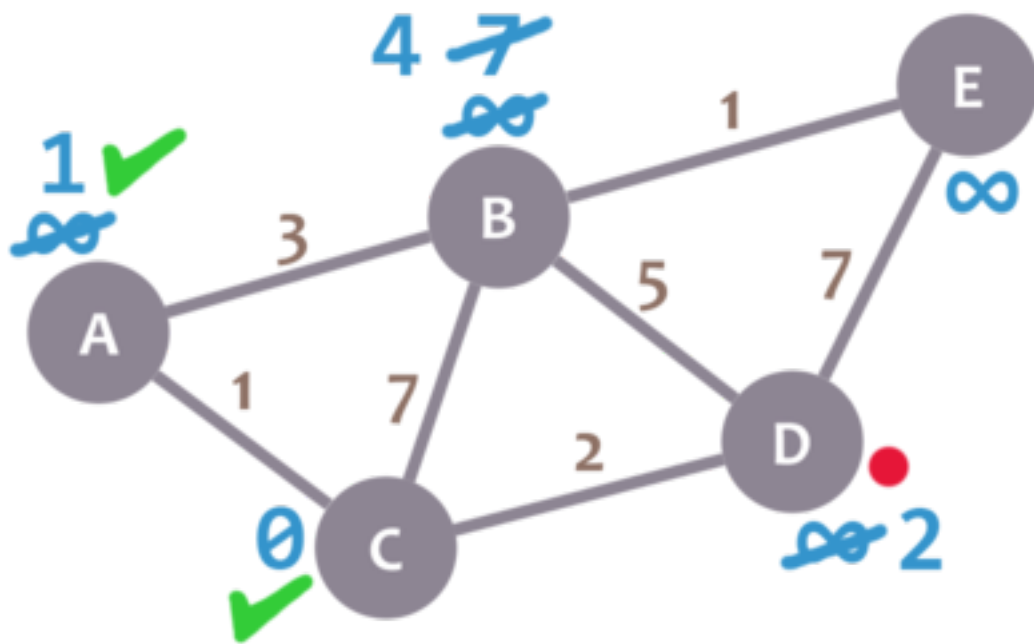


And now we repeat the algorithm. We check the neighbours of our current node, ignoring the visited nodes. This means we only check B.

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.



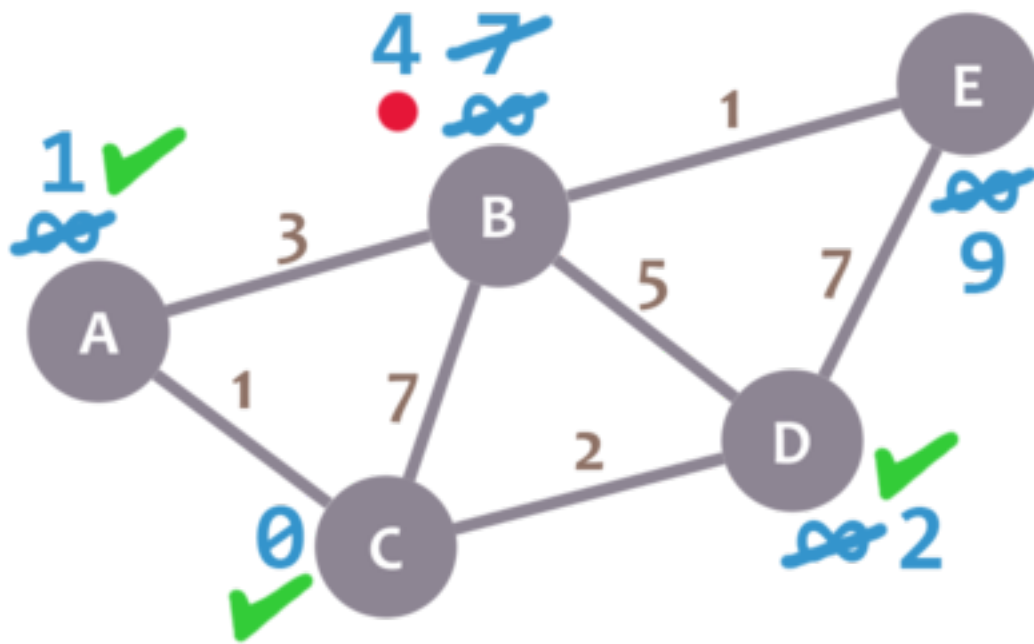
Afterwards, we mark A as visited and pick a new current node: D, which is the non-visited node with the smallest current distance.



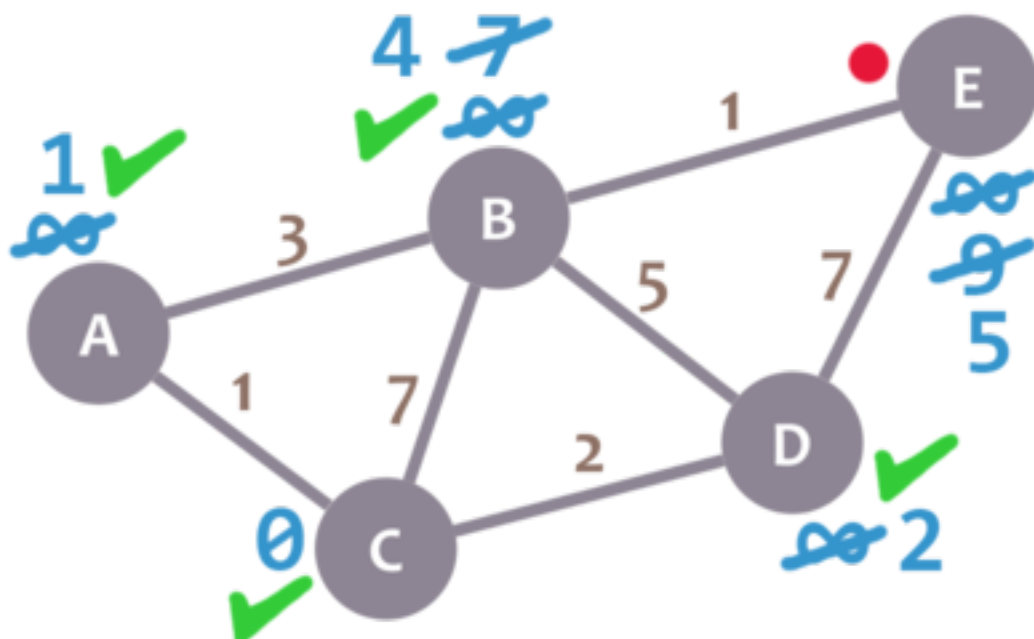
We repeat the algorithm again. This time, we check B and E.

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).

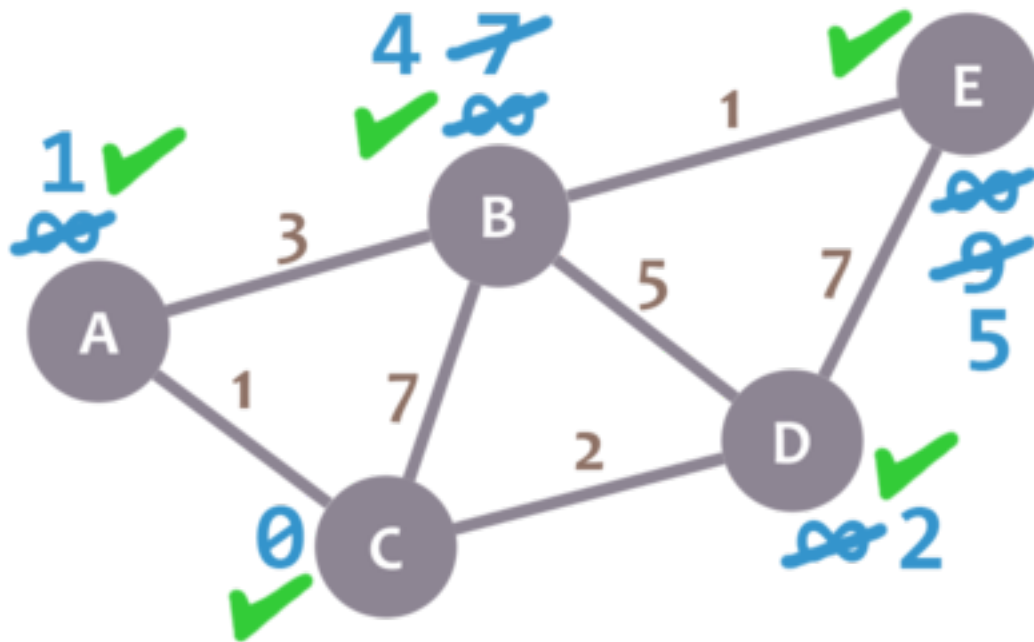
We mark D as visited and set our current node to B.



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



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



As there are not unvisited nodes, we're done! The minimum distance of each node now actually represents the minimum distance from that node to node C (the node we picked as our initial node)!