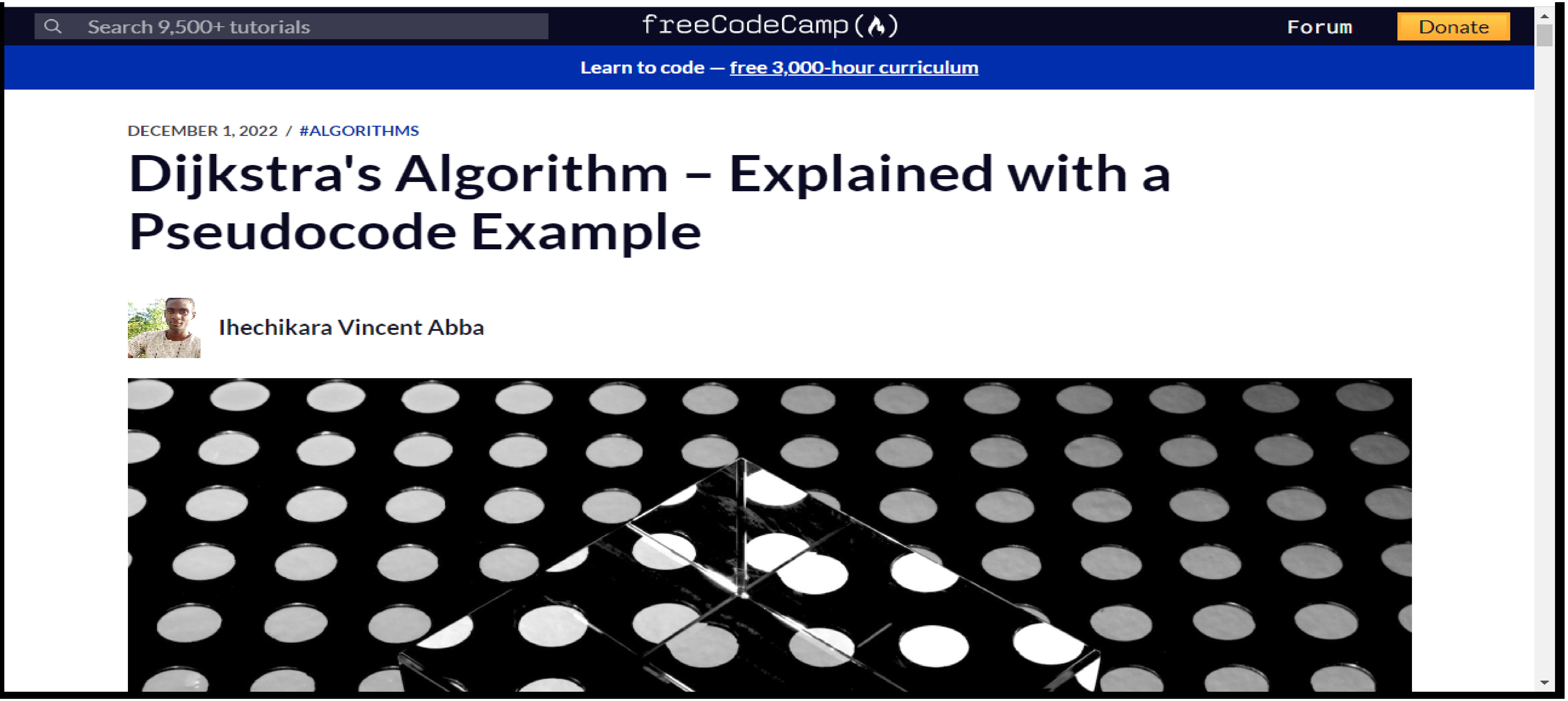
Path Finding in Graph Theory



Learn to code — free 3,000-hour curriculum

DECEMBER 1, 2022 / #ALGORITHMS

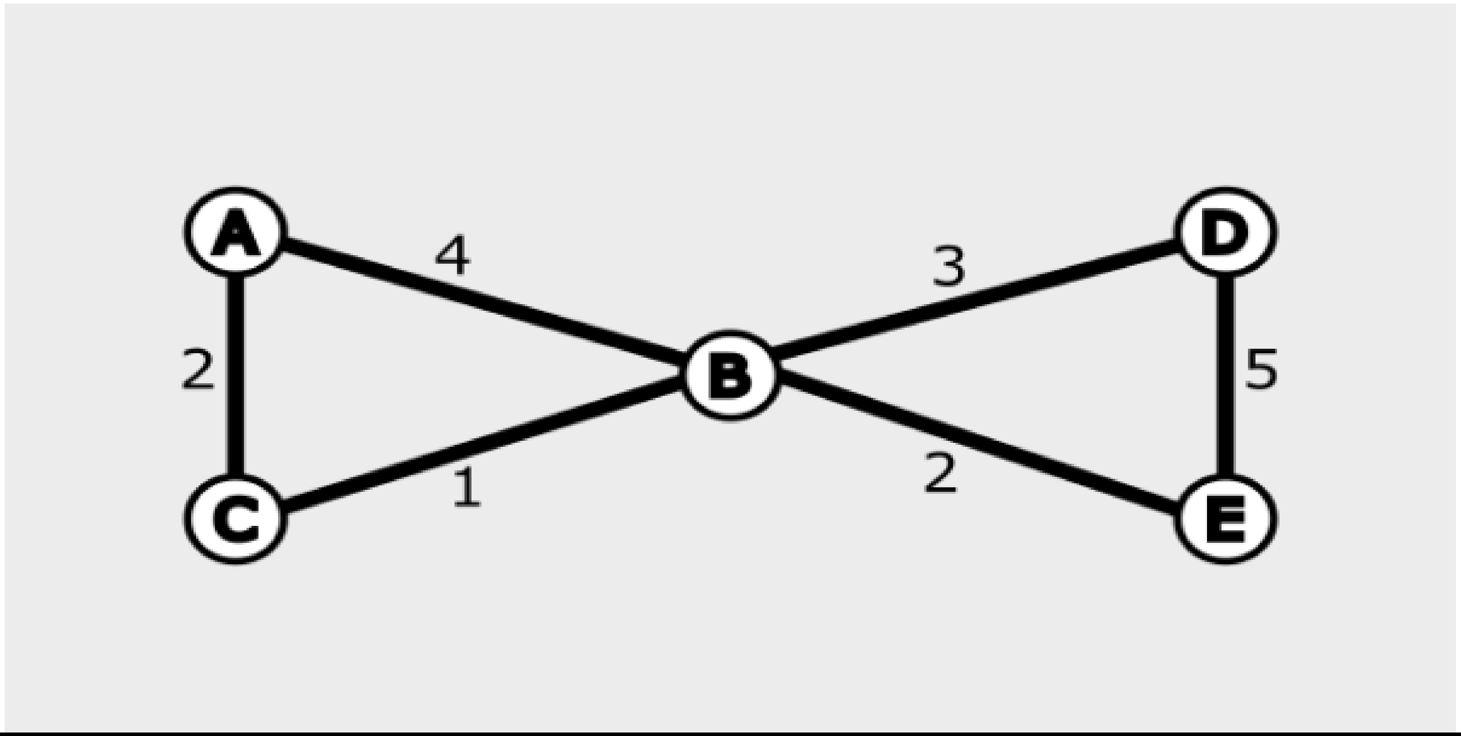
Dijkstra's Algorithm – Explained with a Pseudocode Example

Ihechikara Vincent Abba

Dijkstra's Algorithm Example

In this section, we'll take a look at a practical example that shows how Dijkstra's algorithm works.

Here's the graph we'll be working with:



We'll use the table below to put down the visited nodes and their distance from the fixed node:

NODE	SHORTEST DISTANCE FROM FIXED NODE
A	∞
B	∞
C	∞
D	∞
E	∞

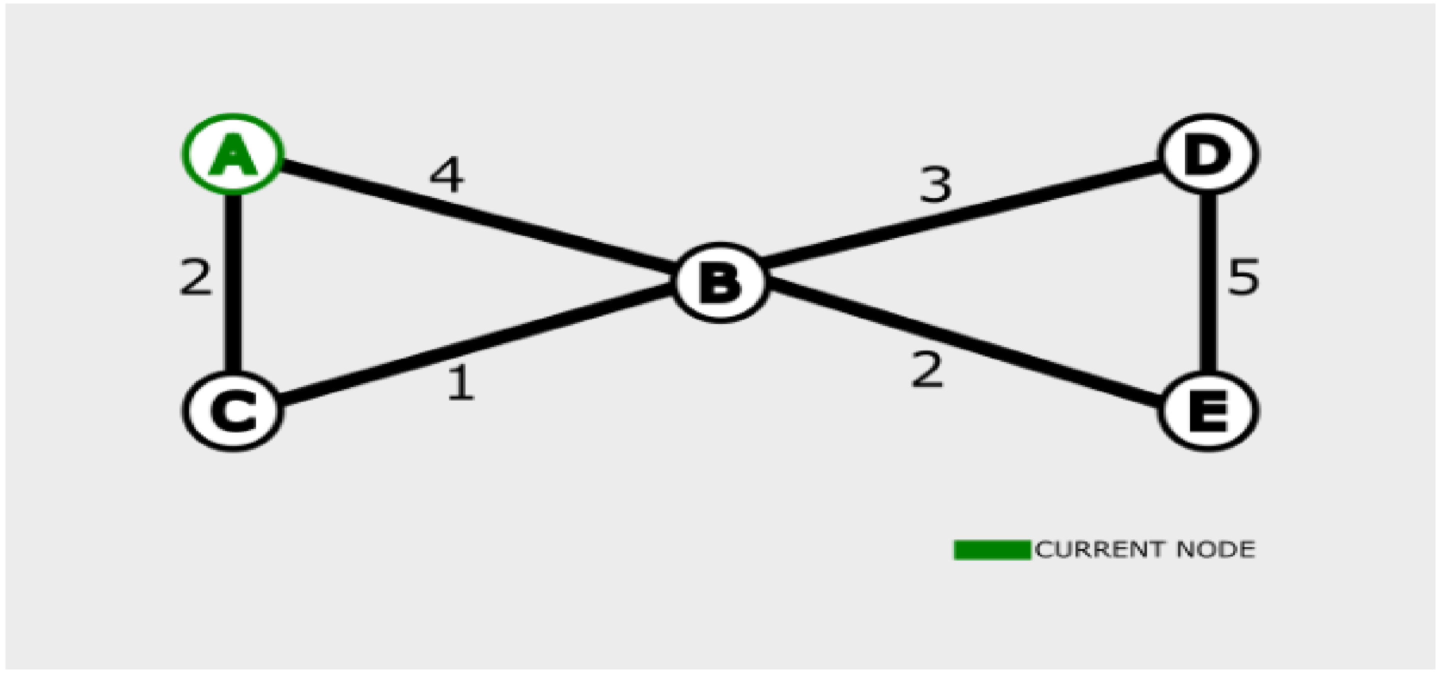
Visited nodes = []

Unvisited nodes = [A,B,C,D,E]

Iteration #1

Step #1 - Pick an unvisited node

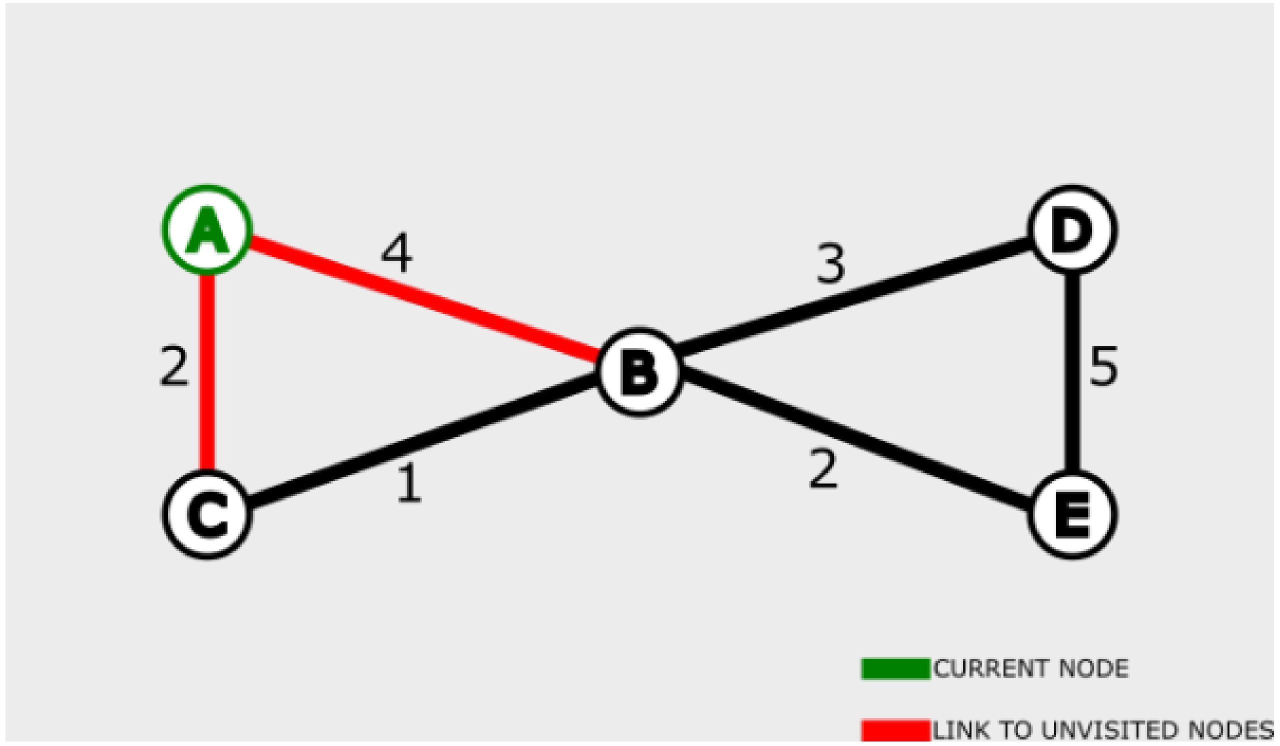
We'll choose A as the fixed node. So we'll find the shortest distance from A to every other node in the graph.



We're going to give **A** a distance of 0 because it is the initial node. So the table would look like this:

NODE	SHORTEST DISTANCE FROM FIXED NODE
A	0
B	∞
C	∞
D	∞
E	∞

Step #2 - Find the distance from current node



The next thing to do after choosing a node is to find the distance from it to the unvisited nodes around it.

ADVERTISEMENT

Step #3 - Update table with known distances

In the last step, we got 4 and 2 as the values of B and C respectively. So we'll update the table with those values:

NODE	SHORTEST DISTANCE FROM FIXED NODE
A	0
B	4
C	2
D	∞
E	∞

Step #4 - Update arrays

At this point, the first iteration is complete. We'll move node A to the visited nodes array:

Visited nodes = [A]
Unvisited nodes = [B,C,D,E]

Iteration #2

Step #1 - Pick an unvisited node

We have four unvisited nodes — [B,C,D,E]. So how do you know which node to pick for the next iteration?

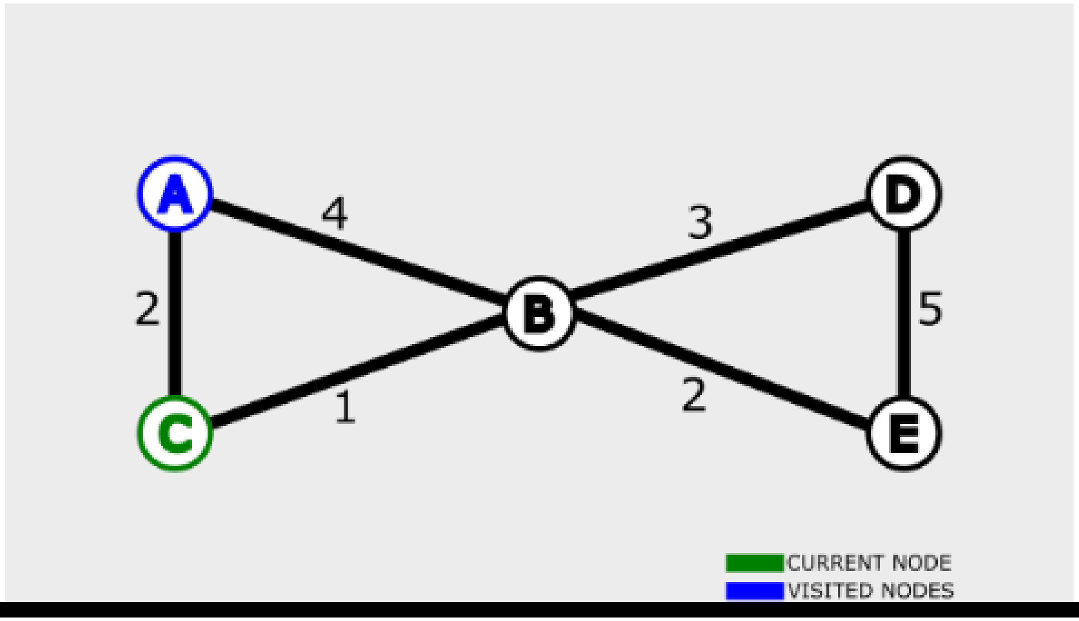
Well, we pick the node with the smallest known distance recorded in the table. Here's the table:

NODE	SHORTEST DISTANCE FROM FIXED NODE
A	0
B	4
C	2
D	∞
E	∞

So we're going with node **C**.

NODE	SHORTEST DISTANCE FROM FIXED NODE
A	0
B	4
C	2
D	∞
E	∞

So we're going with node C.

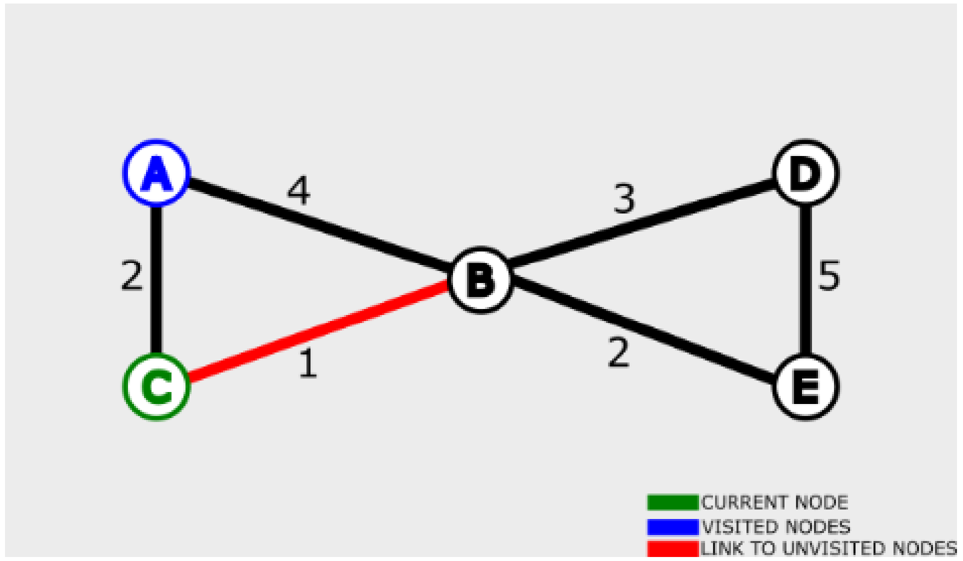


Step #2 - Find the distance from current node

To find the distance from the current node to the fixed node, we have to consider the nodes linked to the current node.

The nodes linked to the current node are **A** and **B**.

But **A** has been visited in the previous iteration so it will not be linked to the current node. That is:



Step #3 - Update table with known distances

In the last step, we got 4 and 2 as the values of B and C respectively. So we'll update the table with those values:

NODE	SHORTEST DISTANCE FROM FIXED NODE
A	0
B	4
C	2
D	∞
E	∞

Step #4 - Update arrays

At this point, the first iteration is complete. We'll move node A to the visited nodes array:

Visited nodes = [A]
Unvisited nodes = [B,C,D,E]

Iteration #2

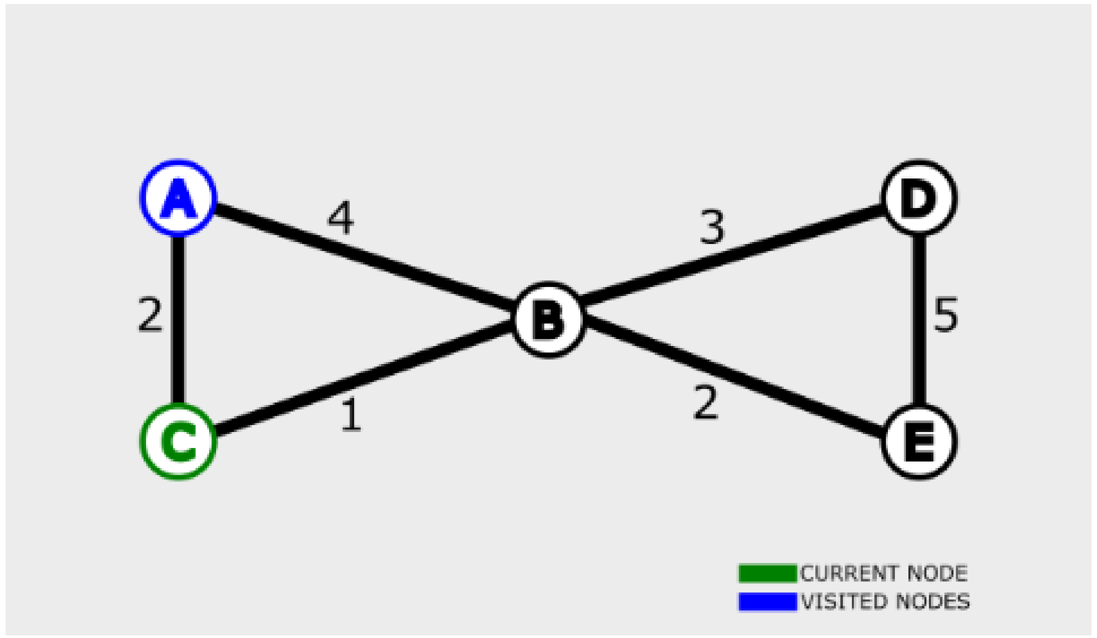
Step #1 - Pick an unvisited node

We have four unvisited nodes — [B,C,D,E]. So how do you know which node to pick for the next iteration?

Well, we pick the node with the smallest known distance recorded in the table. Here's the table:

NODE	SHORTEST DISTANCE FROM FIXED NODE
A	0
B	4
C	2
D	∞
E	∞

So we're going with node **C**.

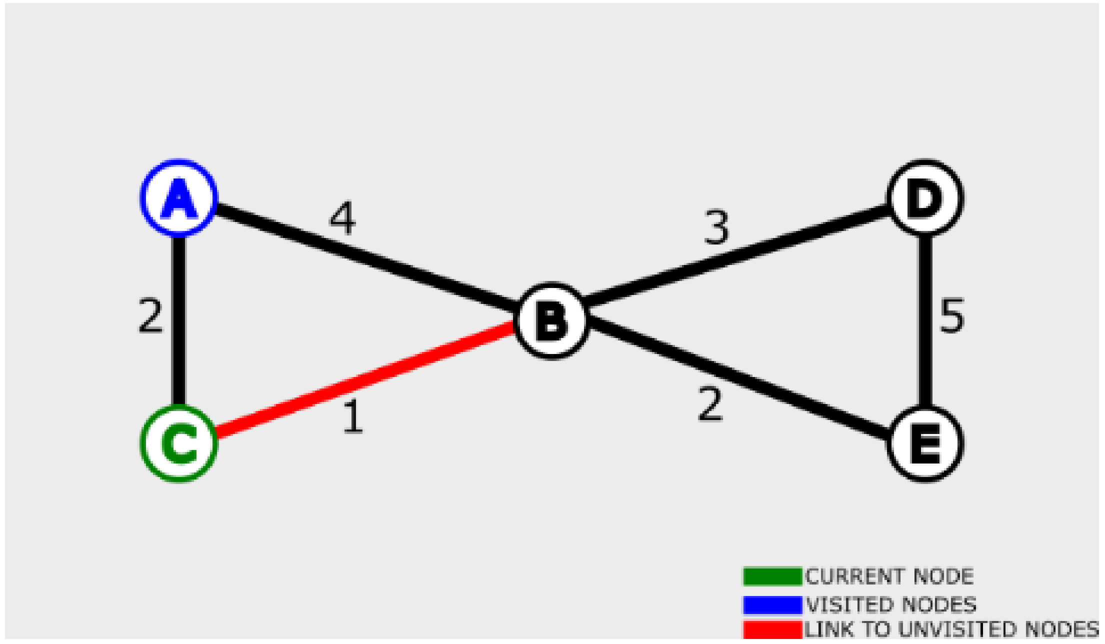


Step #2 - Find the distance from current node

To find the distance from the current node to the fixed node, we have to consider the nodes linked to the current node.

The nodes linked to the current node are **A** and **B**.

But **A** has been visited in the previous iteration so it will not be linked to the current node. That is:



From the diagram above,

- The green color denotes the current node.
- The blue color denotes the visited nodes. We cannot link to them or visit them again.
- The red color shows the link from the unvisited nodes to the current node.

Step #3 - Update table with known distances

In the last step, we got the value of **B** to be 3. In the first iteration, it was 4.

We're going to update the distance in the table to 3.

NODE	SHORTEST DISTANCE FROM FIXED NODE
A	0
B	3
C	2
D	∞
E	∞

So, **A** --> **B** = 4 (First iteration).

A --> **C** --> **B** = 3 (Second iteration).

The algorithm has helped us find the shortest path to **B** from **A**.

Step #4 - Update arrays

We're done with the last visited node. Let's add it to the visited nodes array:

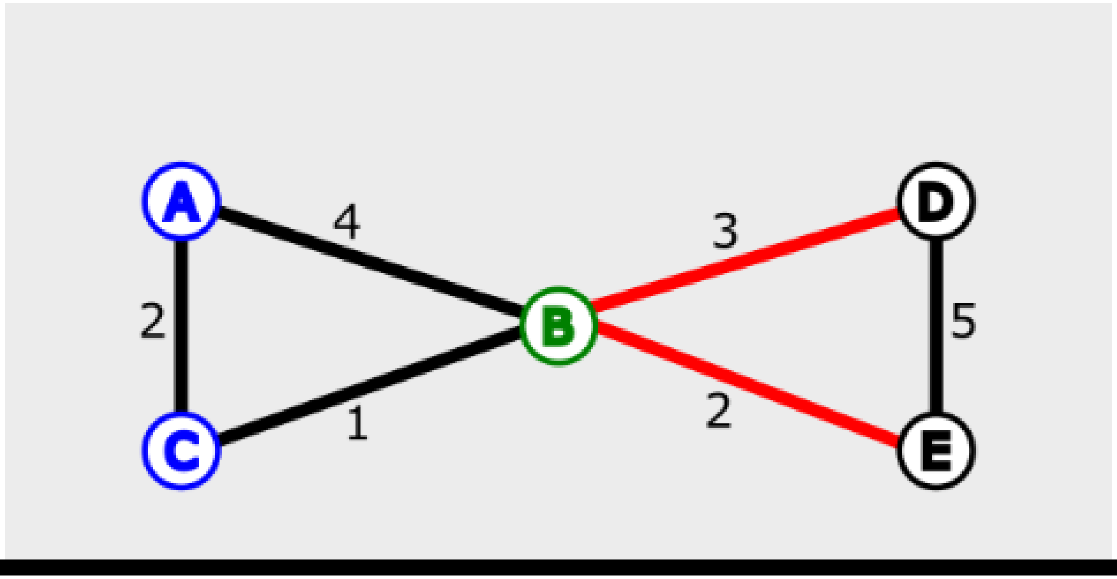
Visited nodes = [A,C]

Unvisited nodes = [B,D,E]

Iteration #3

Step #1 - Pick an unvisited node

We're down to three unvisited nodes — [B,D,E]. From the array, **B** has the shortest known distance.



Step #2 - Find the distance from current node

The nodes linked to the current node are **D** and **E**.

B (the current node) has a value of 3. Therefore,

For node **D**, 3 + 3 = 6.

For node **E**, 3 + 2 = 5.

Step #3 - Update table with known distances

NODE	SHORTEST DISTANCE FROM FIXED NODE
A	0
B	3
C	2
D	6
E	5

Step #4 - Update arrays

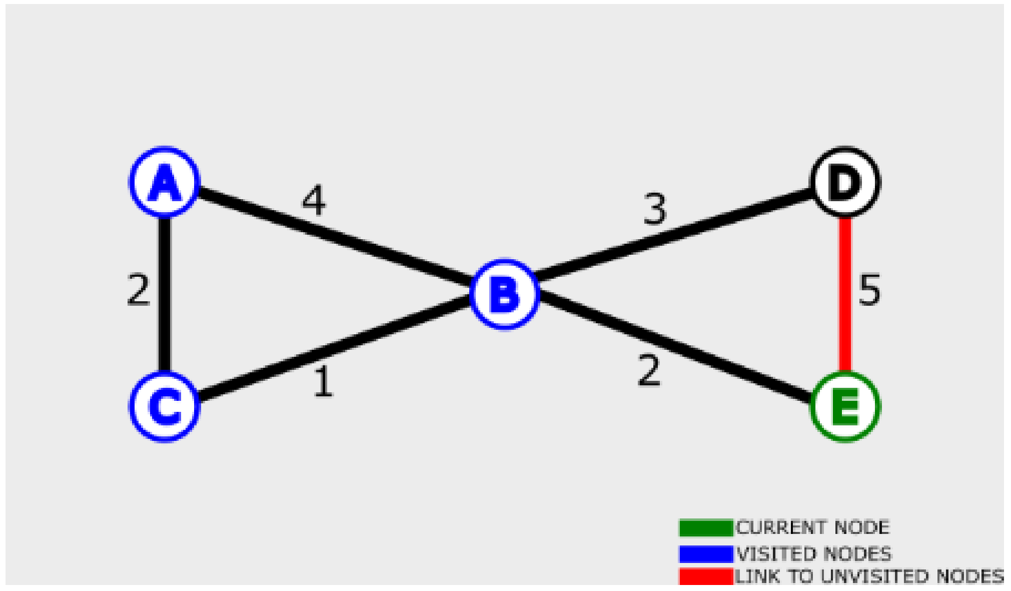
Visited nodes = [A,C,B]

Unvisited nodes = [D,E]

Iteration #4

Step #1 - Pick an unvisited node

Like other iterations, we'll go with the unvisited node with the shortest known distance. That is **E**.



Step #2 - Find the distance from current node

According to our table, **E** has a value of 5.

For **D** in the current iteration,

Step #3 - Update table with known distances

Our table remains the same:

NODE	SHORTEST DISTANCE FROM FIXED NODE
A	0
B	3
C	2
D	6
E	5

Step #4 - Update arrays

Visited nodes = [A,C,B,E]

Unvisited nodes = [D]

Iteration #5

Step #1 - Pick an unvisited node

We're currently left with one node in the unvisited array — **D.**

Step #2 - Find the distance from current node

The algorithm has gotten to the last iteration. This is because all nodes linked to the current node have been visited already so we can't link to them.

Step #3 - Update table with known distances

Our table remains the same:

NODE	SHORTEST DISTANCE FROM FIXED NODE
A	0
B	3
C	2
D	6
E	5

At this point, we have updated the table with the shortest distance from the fixed node to every other node in the graph.

Step #4 - Update arrays

Visited nodes = [A,C,B,E,D]

Unvisited nodes = []