Unit 9: Graphs

9.1 Graph Terminology
9.2 Directed and undirected graph
9.3 Graph Traversal: BFS and DFS
9.4 Minimum Spanning Trees: Kruskal Algorithm
9.5 Shortest Path Algorithms: Dijkstra's Algorithm

Practical Works

9.1 Write a program to implement graph traversal algorithms: BFS and DFS
9.2 Write program to implement Kruskal algorithm and Dijkstra's algorithm

- **Definition of Graph**

Graph is a non linear data structure, which contains a set of points known as *nodes* (or *vertices*) and set of links known as *edges* (or *arcs*) which connects the vertices.

Thus a graph G is a collection of two sets V and E where V is the set of vertices V_0, V_1….V_{n-1} and E is the set of edges e_1, e2…….e_n. This can be represented as G= (V,E) where

V(G) = (V_0, V_1….V_{n-1}) or set of vertices

E(G) = (e_1,e_2…….e_n) or set of edges

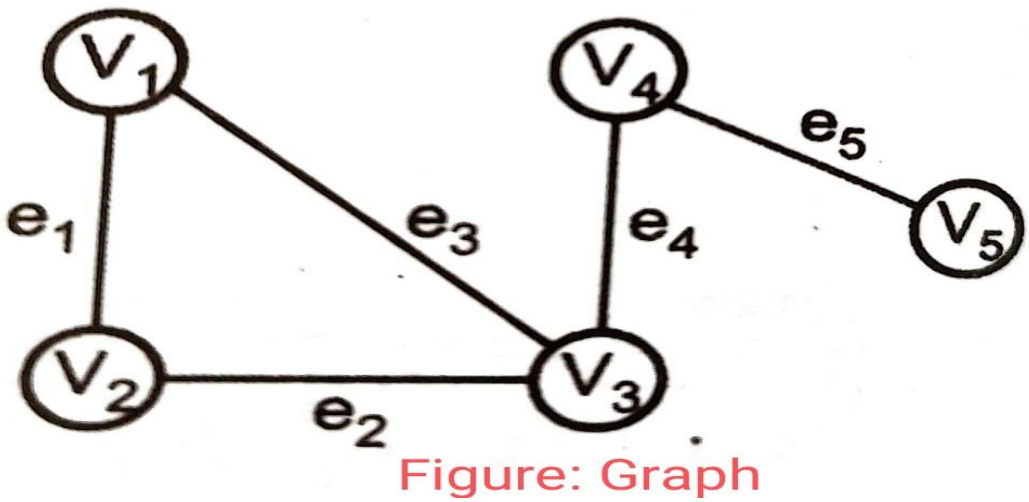


Figure: Graph

Figure 9.1

In the above figure, (V_1, V_2, V_3, V_4 and $V_{5)}$ are set of vertices 'V' and (e_1,e_2, $e_{3,}$ e_4 and e_5) are set of edges 'E' which represent a graph 'G'.

9.1 **Graph Terminology**

- An edge is (together with vertices) one of the two basic units out of which graphs are constructed. Each edge has two vertices to which it is attached, called its endpoints.
- Two vertices are called adjacent if they are endpoints of the same edge.
- Outgoing edges of a vertex are directed edges that the vertex is the origin.
- Incoming edges of a vertex are directed edges that the vertex is the destination.
- The degree of a vertex in a graph is the total number of edges incident to it.

- In a directed graph, the out-degree of a vertex is the total number of outgoing edges, and the in-degree is the total number of incoming edges.
- A vertex with in-degree zero is called a source vertex, while a vertex with out-degree zero is called a sink vertex.
- An isolated vertex is a vertex with degree zero, which is not an endpoint of an edge.
- Path is a sequence of alternating vertices and edges such that the edge connects each successive vertex.
- Cycle is a path that starts and ends at the same vertex.
- Simple path is a path with distinct vertices.
- A graph is strongly connected if it contains a directed path from u to v and a directed path from v to u for every pair of vertices u, v.
- A directed graph is called weakly connected if replacing all of its directed edges with undirected edges produces a connected (undirected) graph. The vertices in a weakly connected graph have either out-degree or in-degree of at least 1.
- Connected component is the maximal connected subgraph of an unconnected graph.
- A bridge is an edge whose removal would disconnect the graph.
- Forest is a graph without cycles.
- Tree is a connected graph with no cycles. If we remove all the cycles from DAG (Directed Acyclic Graph), it becomes a tree, and if we remove any edge in a tree, it becomes a forest.
- Spanning tree of an undirected graph is a subgraph that is a tree that includes all the vertices of the graph.

9.2 Directed and Undirected Graph

1. Directed Graph or Digraph:

If the pair of vertices are ordered, then Graph G is called directed graph.

Directed graph of digraph is a graph which has ordered pair of vertices (V_1, V_2) where V_1 is the tail and V_2 is the head of the edge. If the graph is directed, then the line segments or arcs have arrow heads indicating the direction.

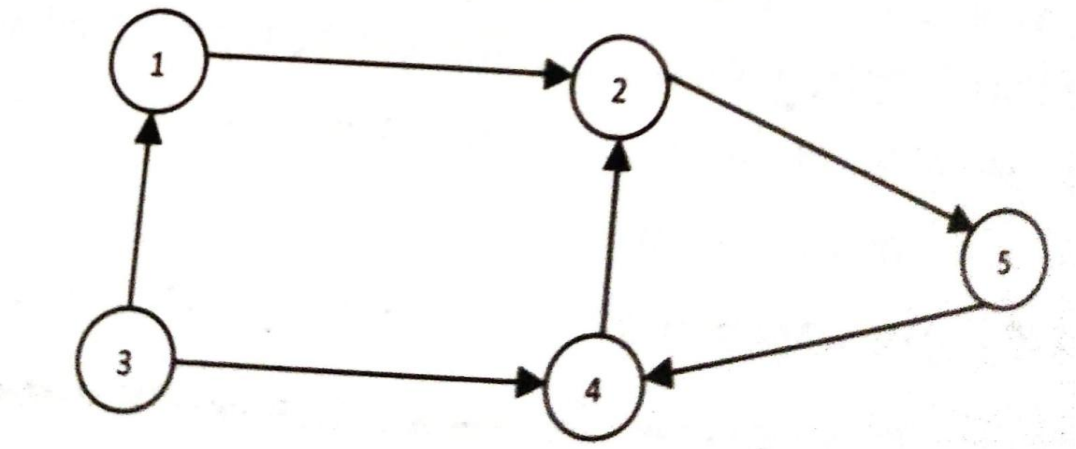


Figure 9.2 Digraph

In the above figure of digraph,

V(G) = {1,2,3,4,5}

E(G) = {(1,2), (2,5), (5,4), (4,2), (3,4), (3,1)}

This graph has 5 nodes (vertices) and 6 edges (arcs).

2. Undirected Graph:

If the pair of vertices is unordered, then Graph G is called an undirected graph. If there is an edge between V_1 and V_2, then it can be represented as (V_1, V_2) or (V_2, V_1) also.

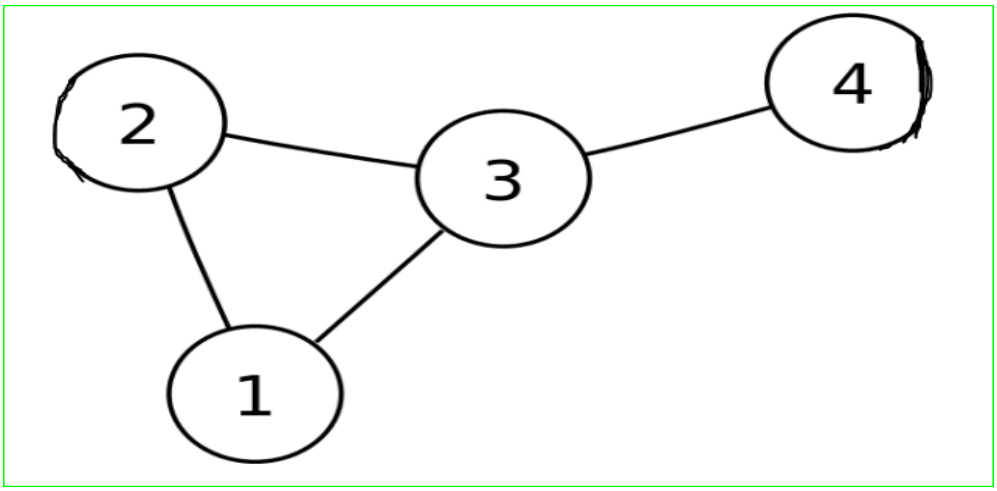


Figure 9.3 Undirected Graph

In the above figure of undirected graph,

V (G) = {1, 2, 3, 4}

E(G) = {(1,2), (2,1), (2,3), (3,2), (1,3), (3,1), (3,4), (4,3)}

This graph has 4 nodes (vertices) and 8 edges (arcs).

9.3 Graph Traversal: BFS and DFS

Traversing is the process of visiting each node in a graph in some systematic approach/order.

Graph traversal is technique used for searching a vertex in a graph. The graph traversal is also used to decide the order of vertices to be visited in the search process. A graph traversal finds the edges to be used in the search process without creating loops that means using graph traversal, we visit all vertices of graph without getting into looping path.

There are two graph traversal techniques. They are described below:

1. Breadth First Search (BFS):

BFS traversal of a graph, produces a spanning tree as final result. Spanning tree is a graph without any loops. The aim of BFS algorithm is to traverse the graph as close as possible to the root node. Queue data structure is used in the implementation of the Breadth First Search (BFS) traversal of a graph.

A standard BFS implementation puts each vertex of the graph into one of two categories:

a. Visited

b. Not Visited

The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

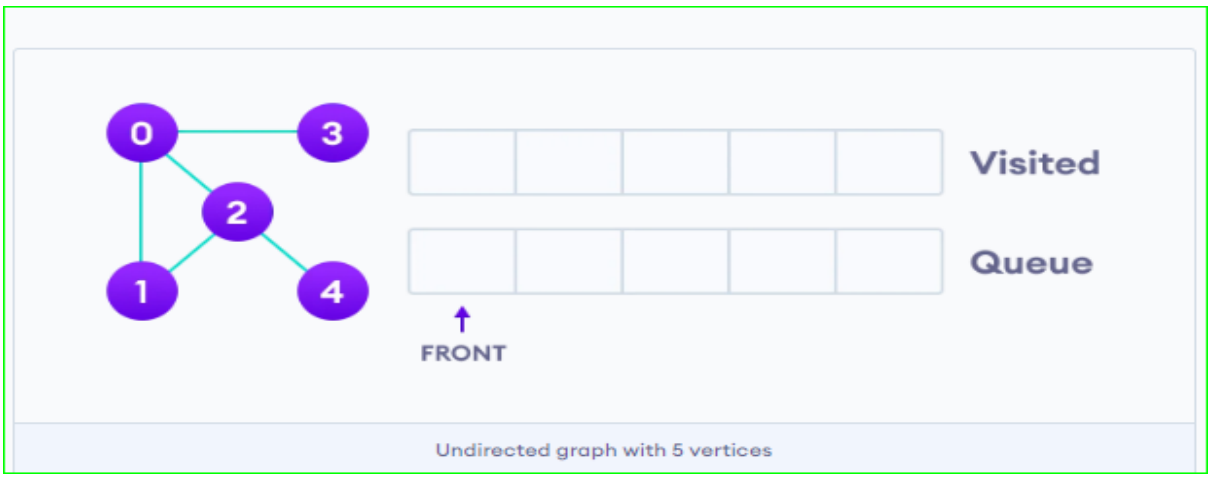
The algorithm works as follows:

1. Start by putting any one of the graph's vertices at the back of a queue.
2. Take the front item of the queue and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.
4. Keep repeating steps 2 and 3 until the queue is empty.

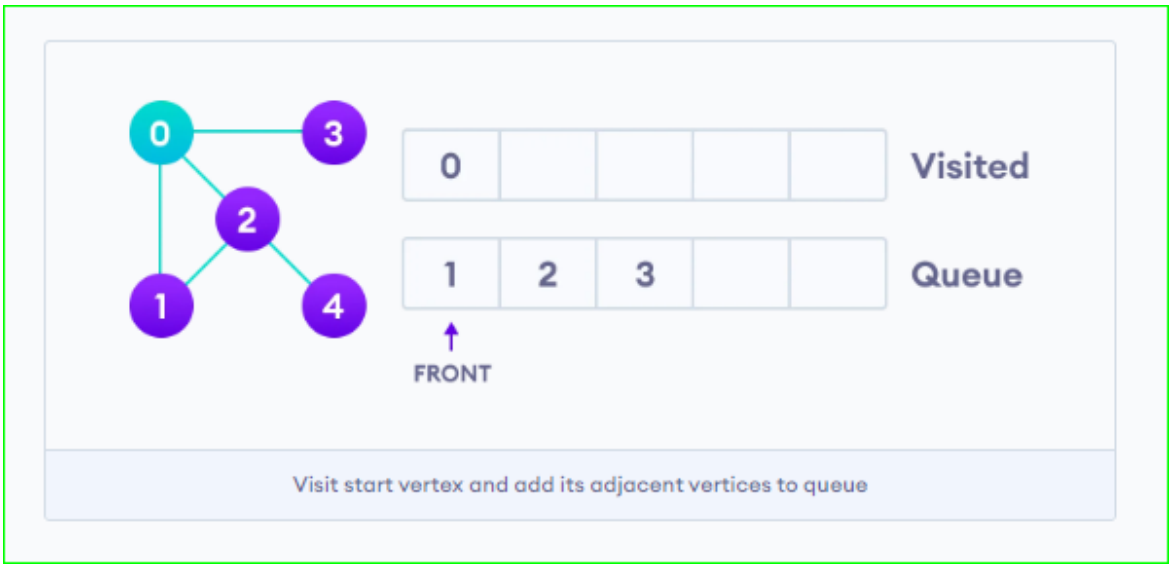
The graph might have two different disconnected parts so to make sure that we cover every vertex, we can also run the BFS algorithm on every node.

Example of BFS:

We use an undirected graph with 5 vertices.



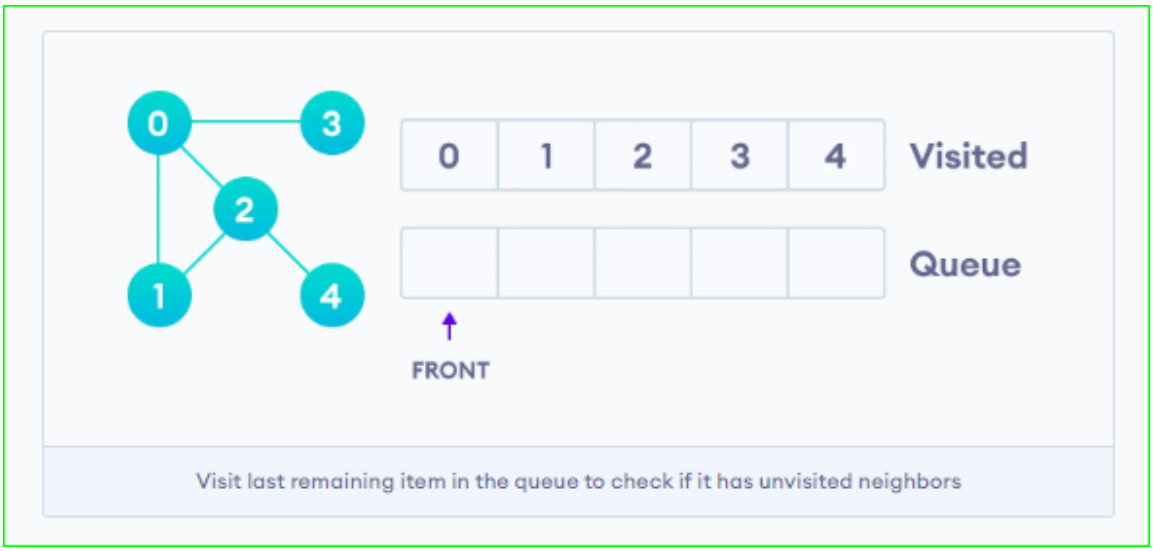
We start from vertex 0, the BFS algorithm starts by putting it in the **visited list** and putting all its adjacent vertices in the queue.



Next, we visit the element at the front of queue i.e. 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.

Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the back of the queue and visit 3, which is at the front of the queue.

Only 4 remains in the queue since the only adjacent node of 3 i.e. 0 is already visited. We visit it.



Since the queue is empty, we have completed the Breadth First Search/Traversal of the graph.

2. Depth First Search (BFS):

DFS traversal of a graph produces a spanning tree as a final result. Spanning tree is a graph without any loops. The aim of DFS algorithm is to traverse the graph in such a way that it tries to go far from the root node. Stack data structure is used in the implementation of the Depth First Search (DFS) traversal of a graph.

A standard DFS implementation puts each vertex of the graph into one of two categories:

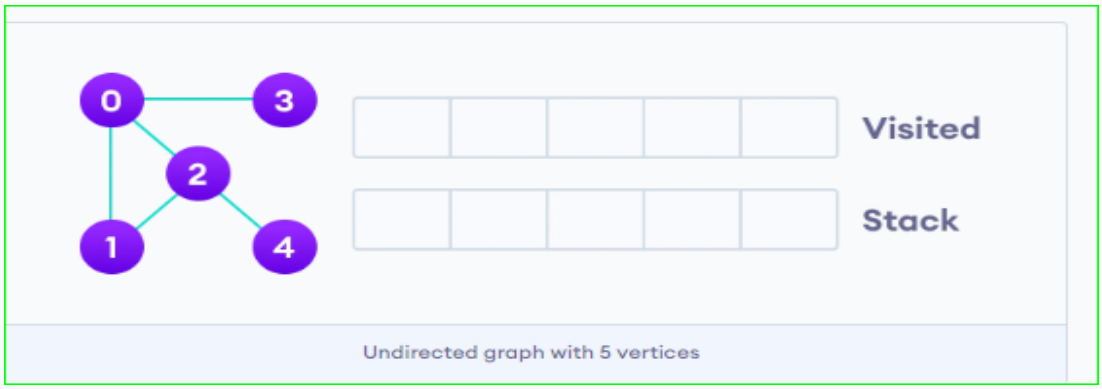
1. Visited
2. Not Visited

The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.
The DFS algorithm works as follows:

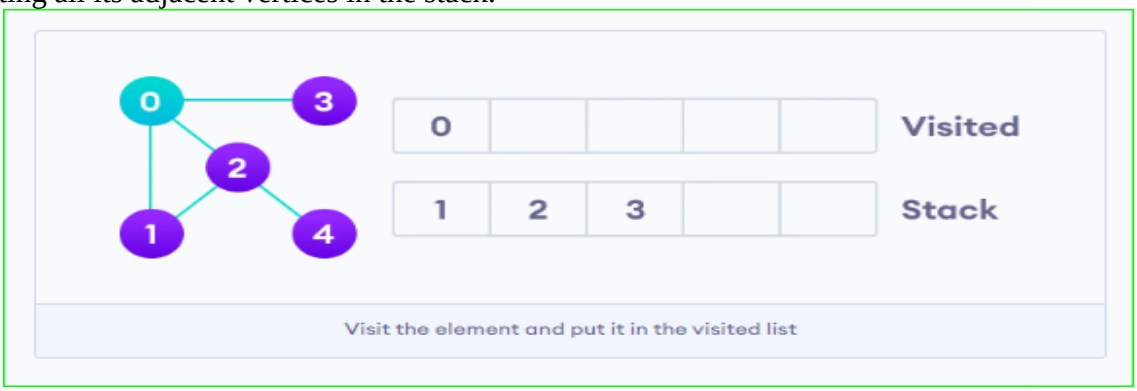
1. Start by putting any one of the graph's vertices on top of a stack.
2. Take the top item of the stack and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack.
4. Keep repeating steps 2 and 3 until the stack is empty.

Example of DFS:

We use an undirected graph with 5 vertices.



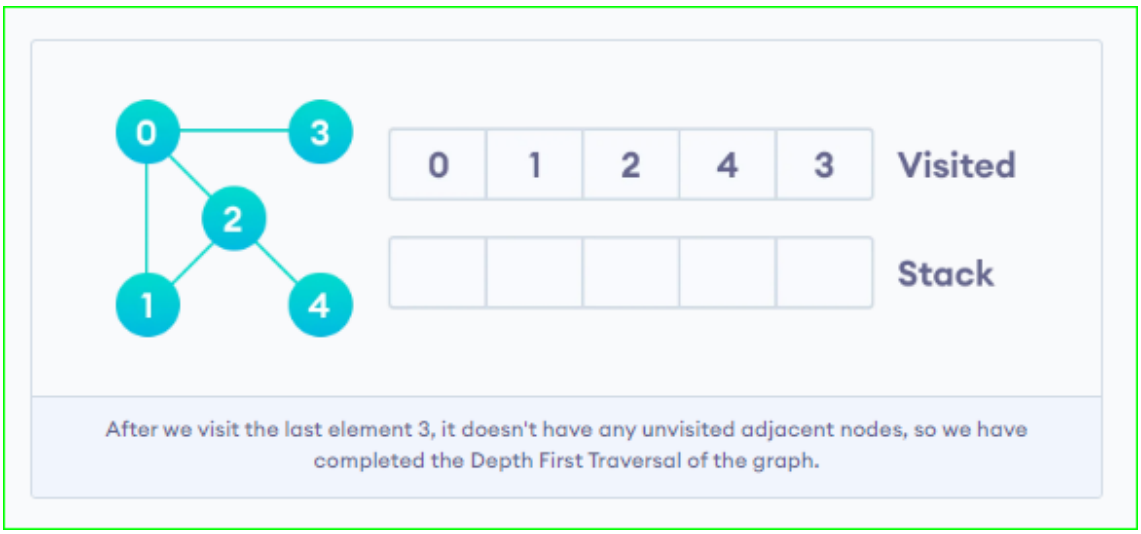
We start from vertex 0, the DFS algorithm starts by putting it in the visited list and putting all its adjacent vertices in the stack.



Next, we visit the element at the top of the stack i.e. 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.

Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

. After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth First Search/Traversal of the graph.



9.4 Minimum Spanning Trees: Kruskal Algorithm

A spanning tree of a graph is just a sub-graph that contains all the vertices and is a tree. A spanning tree of a connected graph G contains all the vertices and has the edges which connect all the vertices. So, the number of edges will be one less than number of nodes.

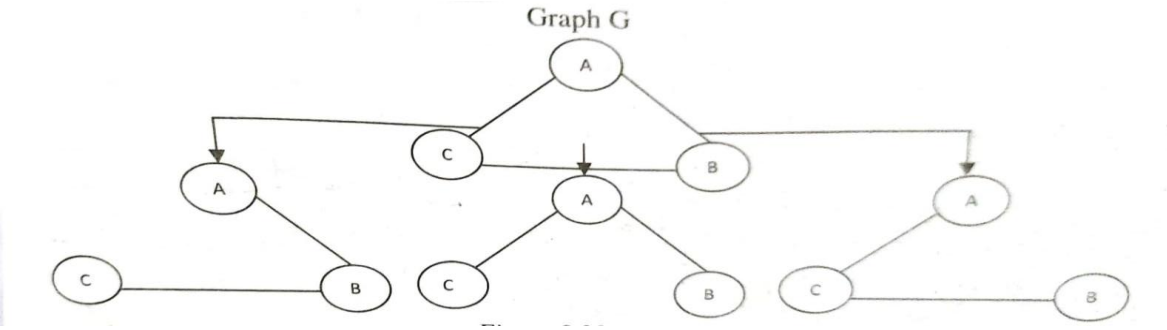


Figure 9.6

A minimum spanning tree is a special kind of tree that minimizes the lengths (or weights) of the edges of the tree.

There are number of techniques for creating a minimum spanning tree for a weighted graph. One of them is

a. Kruskal Algorithm

Kruskal's algorithm is a greedy algorithm that finds a minimum spanning free for a connected weighted graph.

It finds the subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized.

This algorithm is directly based on the MST (Minimum Spanning Tree) Property.

This algorithm creates a forest of trees. Initially, this forest consists of n single node trees and no edges.

Algorithm:

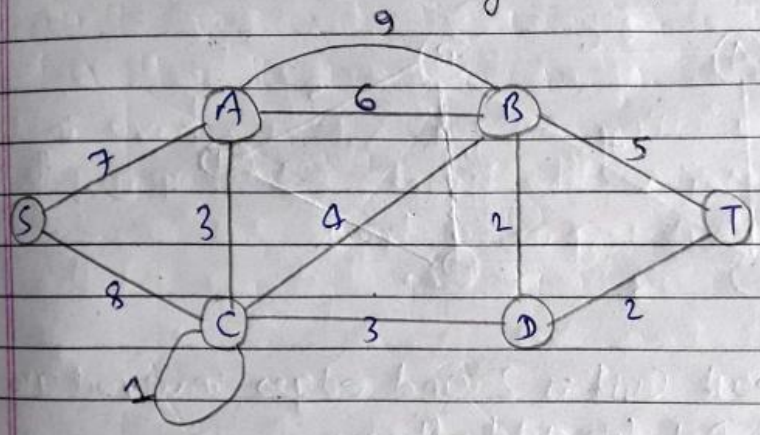
Step 1: Sort all the edges in increasing order of their weight.

Step 2: Take the edge with lowest weight and add it to the spanning tree. If adding the edge created a cycle, then reject the edge.

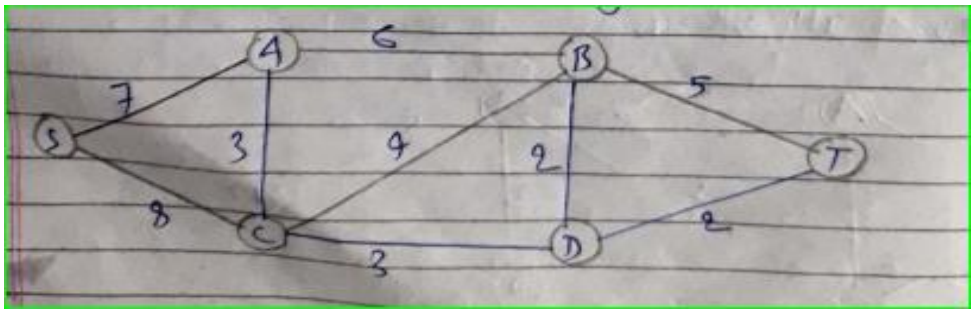
Step 3: Keep adding edges until we reach all vertices.

Example 1

To understand Kruskal's algorithm, let us consider the following example:



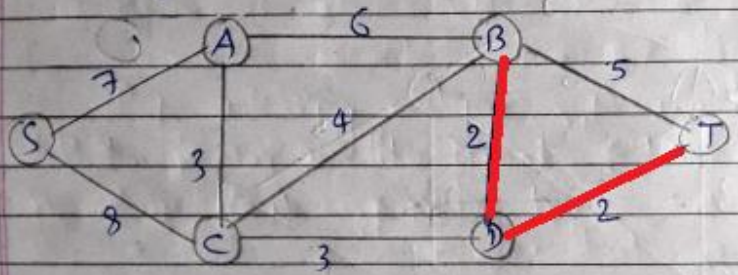
Step 1 Remove all loops and parallel edges. ~~Keep one which has the~~ in case of parallel edges keep one which has the least cost associated and remove all others.



Step 2 - Arrange all edges in their increasing order of weight.

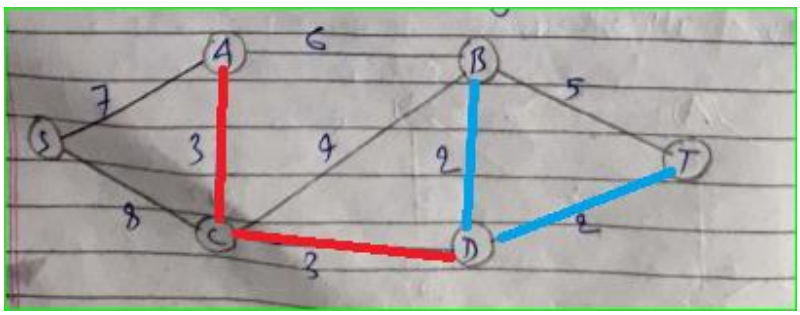
B,D	D,T	A,C	C,D	C,B	B,T	A,B	S,A	S,C
2	2	3	3	4	5	6	7	8

Step 3 - Add the edge which has the least weightage.

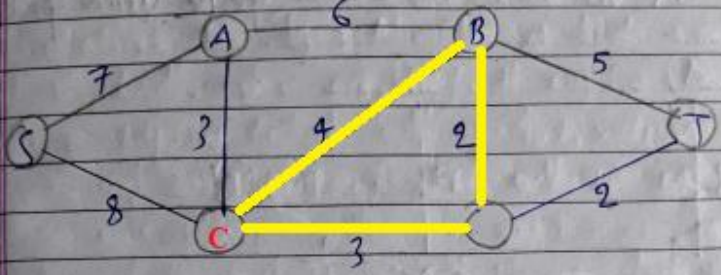


The least cost is 2 and edges involved are B,D and D,T. We add them.

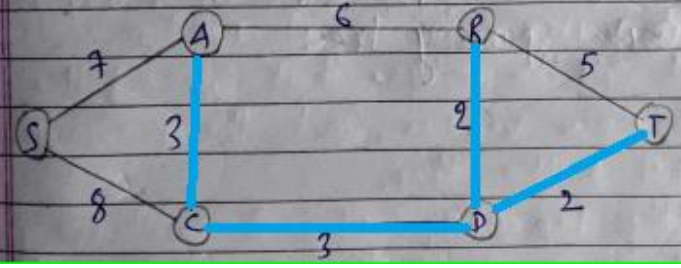
Next cost is 3, and associated edges are A,C and D. We add them again -



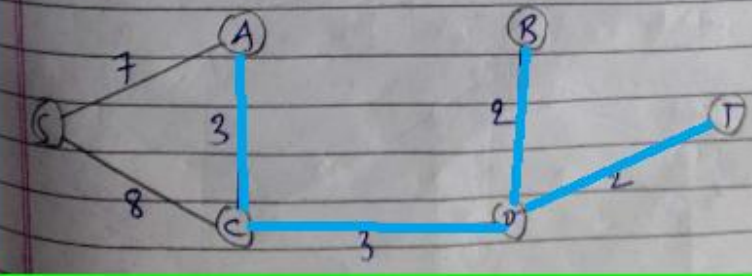
Next cost is 4, and we observe that adding will create a circuit in the graph.

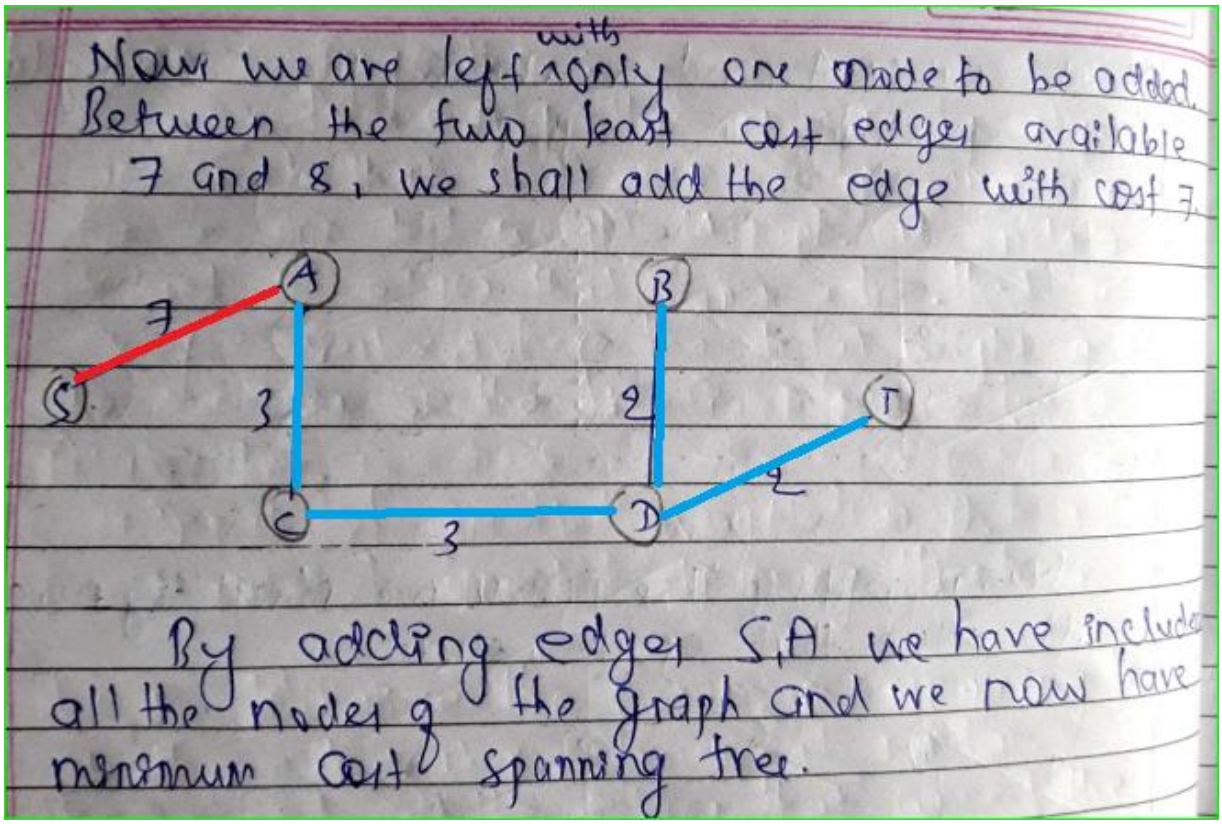


We ignore it. In this process, we shall ignore/avoid all edges that create a circuit.



We observe that edges ~~cost~~ with cost 5 and 6 also create circuits. We ignore them and move on.





9.5 Shortest Path Algorithms: Dijkstra's Algorithm

A path from source vertex s to t is shortest path from s to t if there is no path from s to t with lower weights. Shortest may be least number of edges, least total weight etc. Shortest simple path is a path that does not visit any vertex twice.

In a graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized.

- **Dijkstra's Algorithm -**

Dijkstra's Algorithm can be used to determine the shortest path from one node in graph to every other node within the same graph data structure, provided that the nodes are reachable from the starting node.

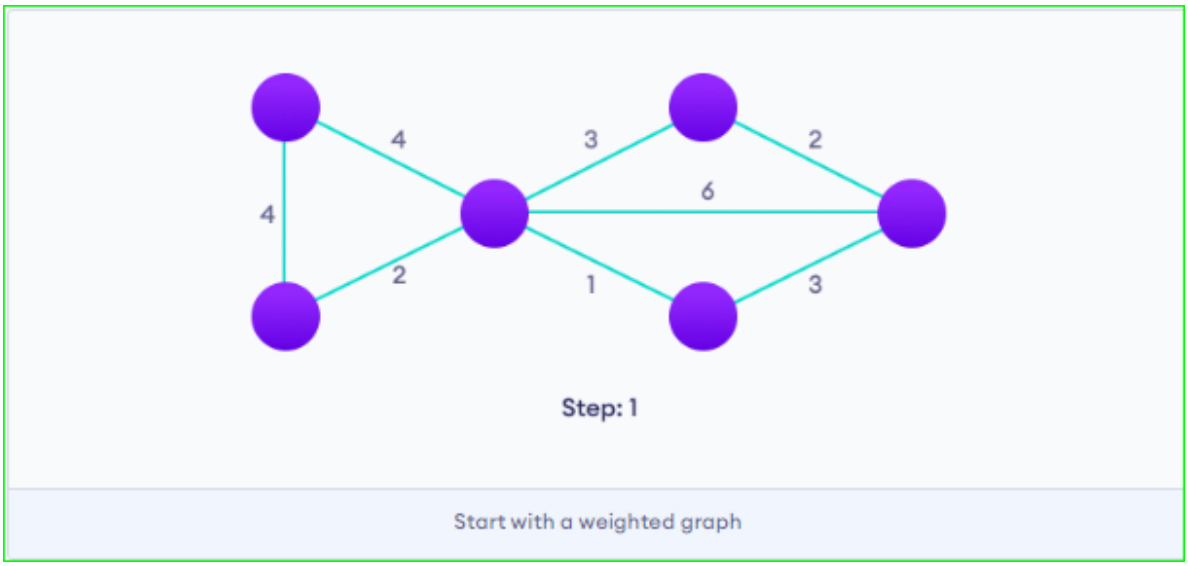
It was conceived by computer science scientist Edsger Dijkstra. in 1956 and published three years later.

- Algorithm

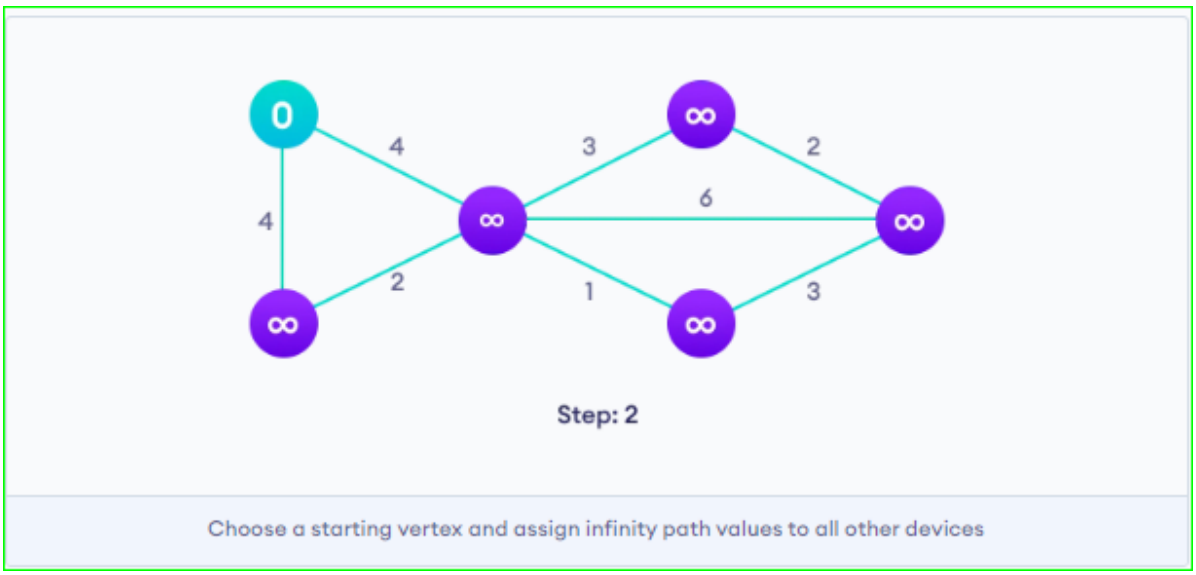
1. From the starting node, visit the vertex with the smallest known distance/cost. .
2. Once we have moved to the smallest cost vertex, check each of its neighboring nodes.
3. Calculate the distance/ cost for the neighboring nodes by summing the cost of the edges leading from the start vertex.
4. Finally, if the distance/ cost to a node is less than a known distance, update the shortest distance for that vertex.

- Example of Dijkstra's Algorithm:

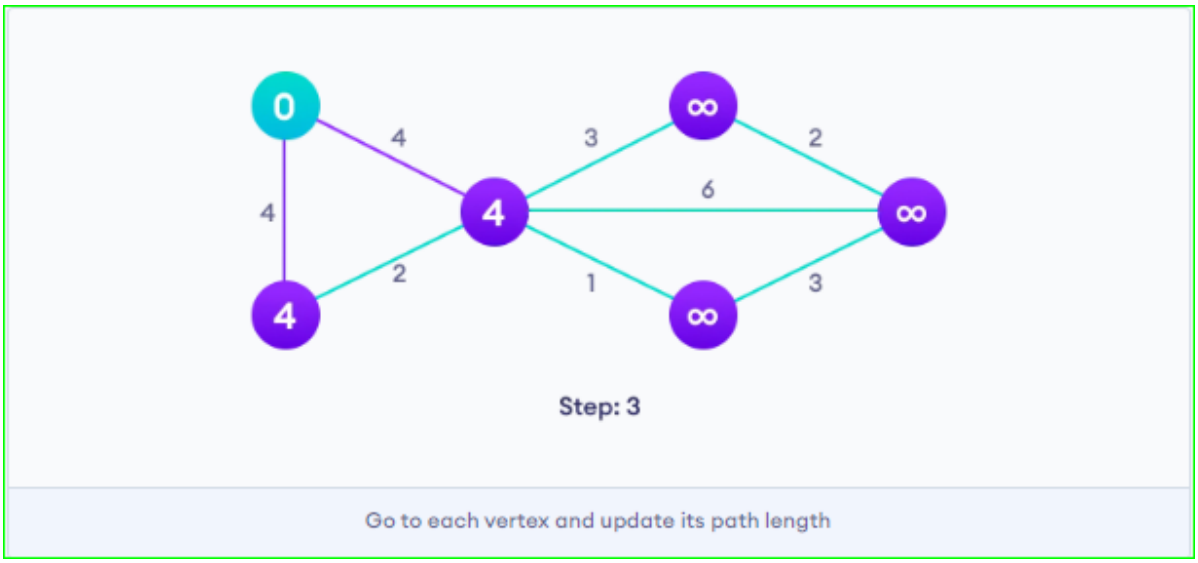
Step 1 : Start with a weighted graph.



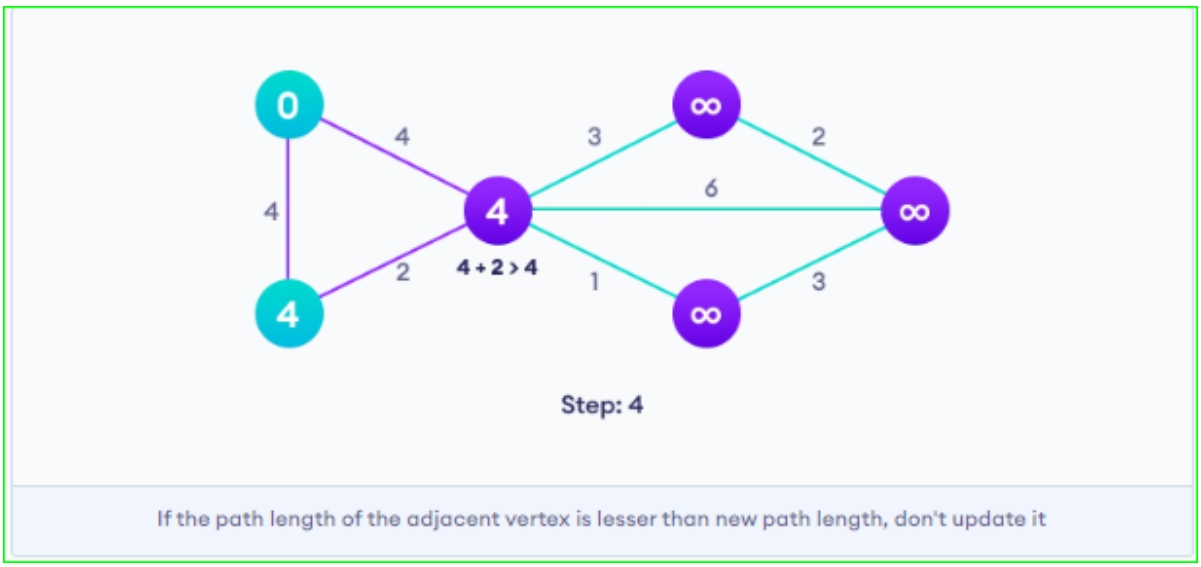
Step 2: Choose a starting vertex and assign infinity path values to all other devices.



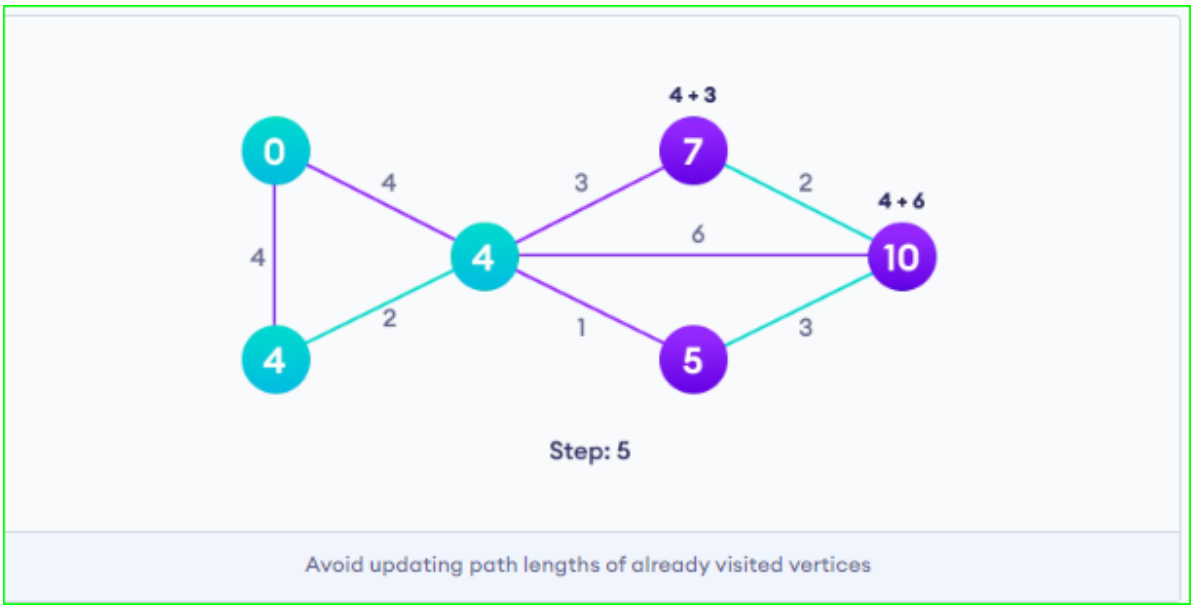
Step 3: Go to each vertex and update its path length.



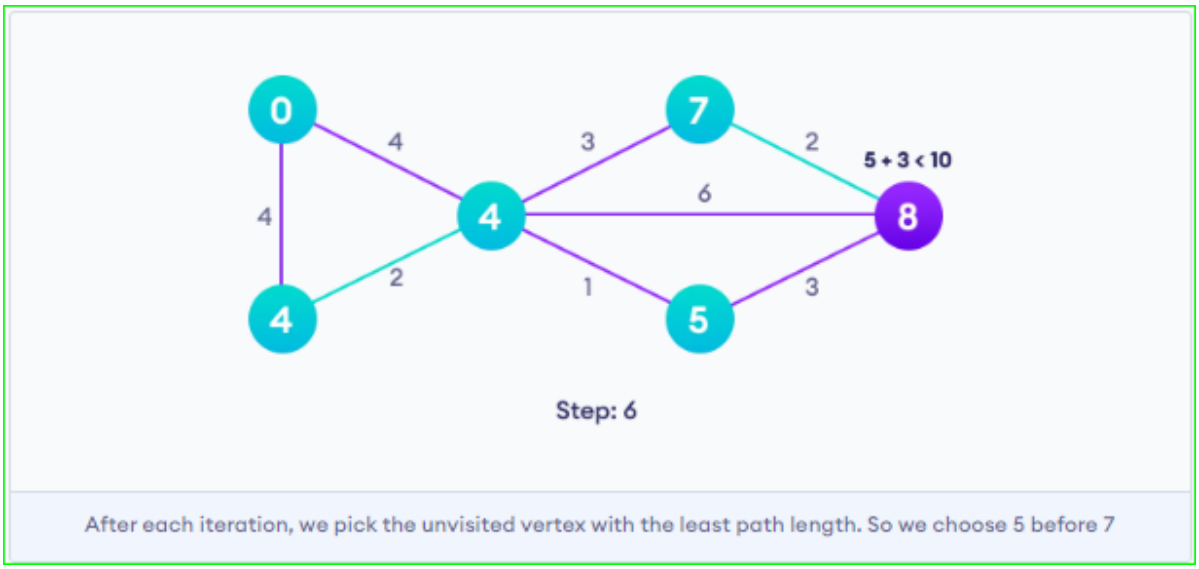
Step 4: If the path length of the adjacent vertex is lesser than new path length, don’t update it.



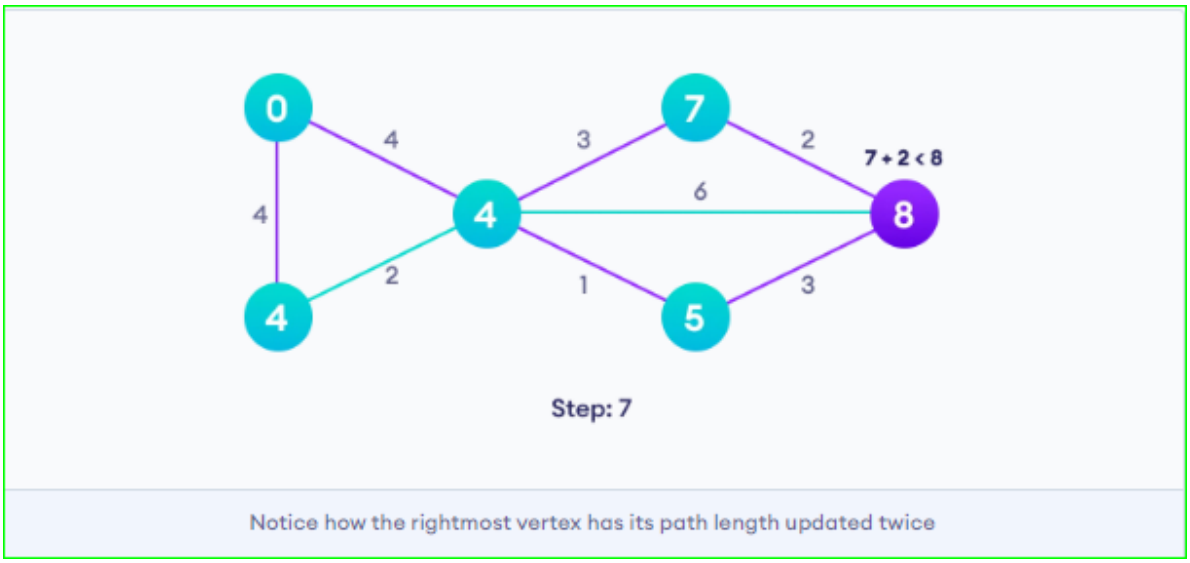
Step 5 : Avoid updating path lengths of already visited vertices.



Step 6: After each iteration, we pick the unvisited vertex with the least path length. So, we choose 5 before 7.



Step 7: Notice how the rightmost vertex has its path length updated twice.



Step 8: Repeat until all the vertices have been visited.