**UNIT 5**

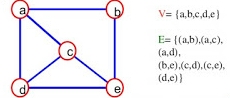
**# INTROUCTION TO GRAPHS**

We can easily understand the data structure graph on the basis of the following points:-

1:- Graph is a [non-primitive, non-linear data structure .](https://ehindistudy.com/2015/11/25/what-is-data-structure-in-hindi-classification-in-hindi/)

2:- A graph is a set of vertex (node). One vertex is connected to another vertex and the connection between two vertex is called edge. Edge acts as a communication link between two nodes.

3:- Graph is a set of (V,E) where V is the set of vertex and E is the set of edges.

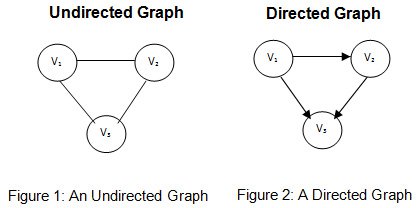
[](https://ehindistudy.com/wp-content/uploads/2015/11/wpid-graphs-in-data-structure-2-638.jpg)

* **Types of data structure graph**

There are following types of graphs in data structure:-

**1: Directed graph**

The graph in which the edges have any direction is called a directed graph. And this type of edges are called directed edges. Directed edges are also called acres. In a graph, the edges are represented by a line and if there is an arrow mark in each line then it is called a directed graph. Directed graph is also called diagram.

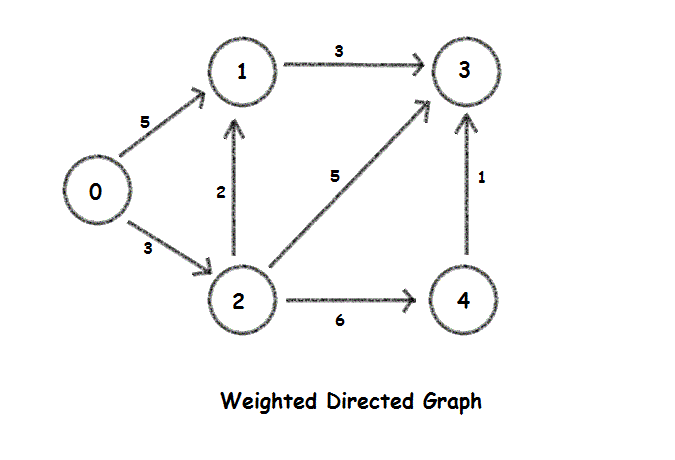
[](https://ehindistudy.com/wp-content/uploads/2015/11/wpid-differencebetween_directed_undirected_graphs1.jpg)

**2: Undirected graph**

The graph in which the edges do not have direction, that is, there is no arrow mark in it. Is called undirected graph.

**3: Weighted graph and non-weighted graph**

Sometimes graphs have edges that carry weight. These weights are real numbers. Both directed and undirected graphs can be weighted graphs. Those graphs which do not carry weight are called non-weighted graphs.

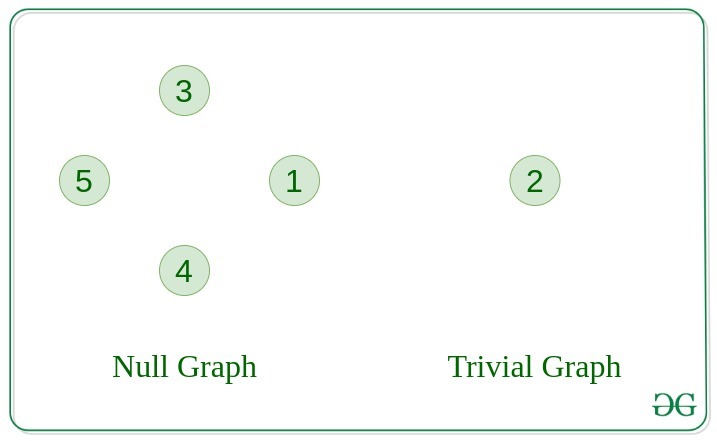
[](https://ehindistudy.com/wp-content/uploads/2015/11/wpid-weighted_graphs.gif)

**4. Null Graph**

A graph is known as a null graph if there are no edges in the graph.

**5. Trivial Graph**

Graph having only a single vertex, it is also the smallest graph possible.

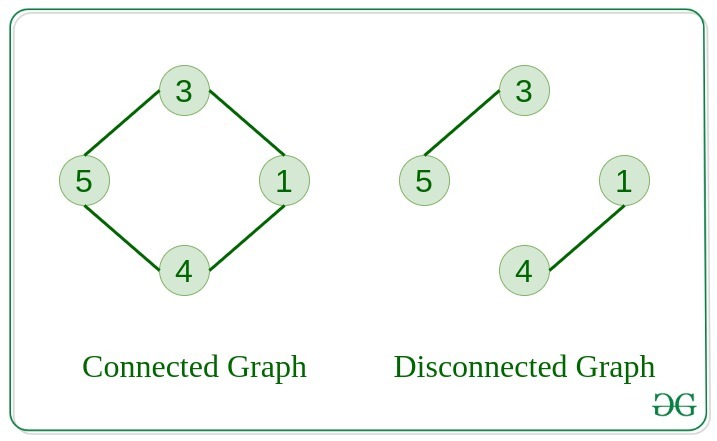
[](https://media.geeksforgeeks.org/wp-content/uploads/20200630113942/null_graph_trivial.jpg)

**6. Connected Graph**

The graph in which from one node we can visit any other node in the graph is known as a connected graph.

**7. Disconnected Graph**

The graph in which at least one node is not reachable from a node is known as a disconnected graph.

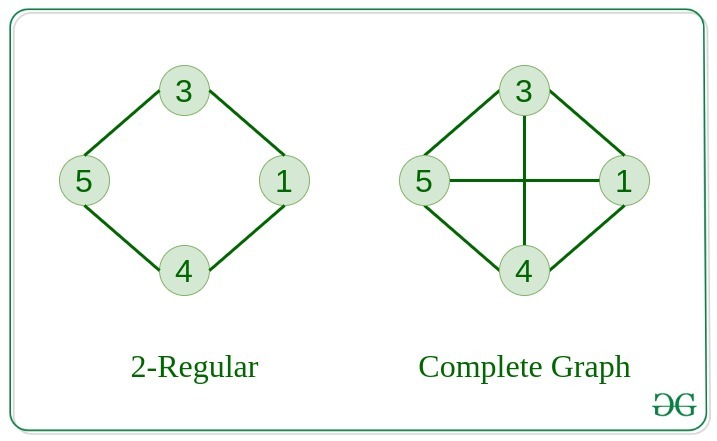
[](https://media.geeksforgeeks.org/wp-content/uploads/20200630121400/connected1.jpg)

**8. Regular Graph**

The graph in which the degree of every vertex is equal to K is called K regular graph.

**9. Complete Graph**

The graph in which from each node there is an edge to each other node.



.

* **Basic Operations on Graphs**

1. Insertion of Nodes
2. Deletion of Nodes
3. Searching on Graphs.
4. Traversal of Graphs

**# BASIC TERMINOLOGY OF GRAPH**

1. **Path -** We can define a path by a sequence of nodes. The work done to reach from the initial node to the terminal node is called a path.

V U

1. **Closed Path -** A path is called closed path when the initial node is the terminal node.

The path will be closed path if VO =VN

1. **Cycle -** A cycle defines the path that does not have repeated edges.

**# REPRESNTATION OF GRAPH**

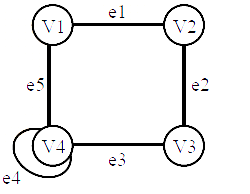
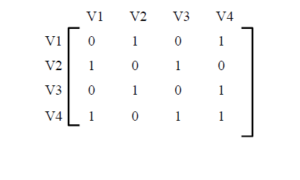
Representation of graph is a way of representing a graph. There are following ways to represent the graph in memory.

1. Adjacency Matrix
2. Indecency Matrix
3. Adjacency List Representation
4. Multi-list Representation

**Adjacency Matrix:-**

In this type of representation, the relationship of one vertex with another is presented through a matrix.

The adjacency matrix representation of the graph shown in the following figure is shown below the figure.



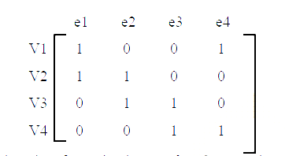
Here we have written 1 below those vertices which are adjacent to the vertex written in front. 0 is written below the remainder.

Like V1, V2 and V4 are adjacent, we have written 1 below V2 and V4 and 0 for V1 and V3.

**Indecency matrix**

In this type of submission, the relation of vertex to different edges is presented through a matrix.

The indecency matrix representation of the graph shown in the following figure is shown below.

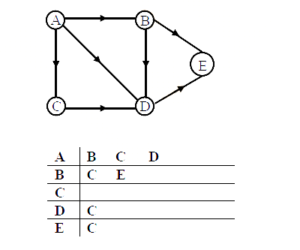


Here we have written 1 in front of those vertices which are indecent to the edge given above.

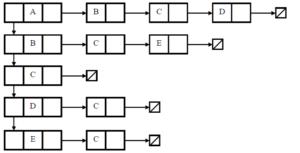
Like V1,e1 and e5 are incidents, 1 is written below e1 and 35 and 0 is written below e2,e3 and e4.

**Adjacency list representation**

We will present an adjacency list for the graph shown in the following figure. For this, first of all we will create a table, In which the adjacent nodes of each node are written in front of it.



Now we will convert this table into a list as follows.

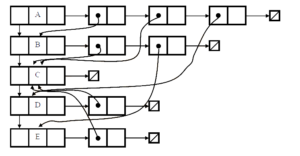


Here two pointers have been used for the node whose adjacent nodes are mentioned.

The first pointer is indicating the next node and the second pointer is indicating the adjacent node.

**Multi-list representation**

In this type of representation, we present the graph through a list, but in the adjacent nodes, not their information but their pointers are kept.



**# GRAPH TRAVERSAL**

Graph traversal means visiting each node of the graph. Here we will talk about two types of traversal. Which are as follows:-  
1:- BFS (breadth first search)

2:- DFS (depth first search)

**1:- BFS (breadth first search)**

BFS is an algorithm to traverse and search [graph data structures .](https://ehindistudy.com/2015/11/16/what-is-graph-in-hindi/)

It is used to find the shortest path in a graph and solve puzzle games.

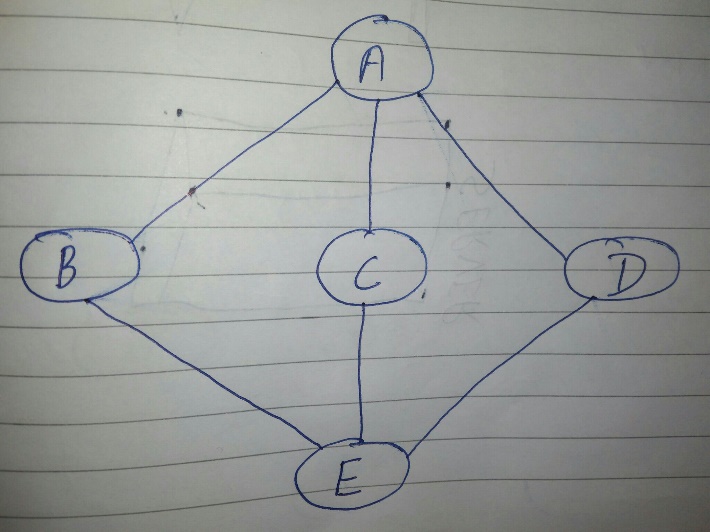
[In data structures, queues](https://ehindistudy.com/2015/11/17/what-is-queues-in-hindi/) are used to implement BFS .

In BFS, nodes are visited breadth wise.

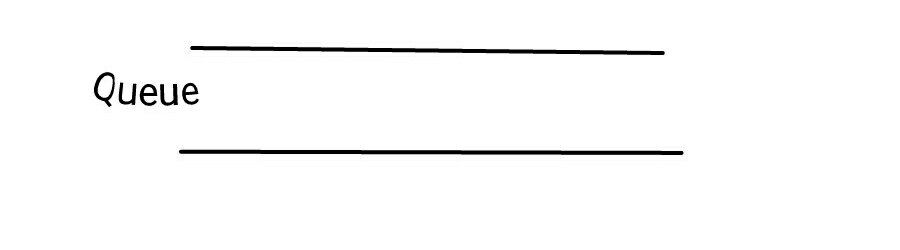
In BFS, first any one node is visited and then its adjacent nodes are visited. After this, all the adjacent nodes of these adjacent nodes are also visited. And this process continues until all the nodes are visited.

**BFS algorithm:-**

Let us understand its algorithm through the following example. Suppose we have the following graph which we have to traverse.

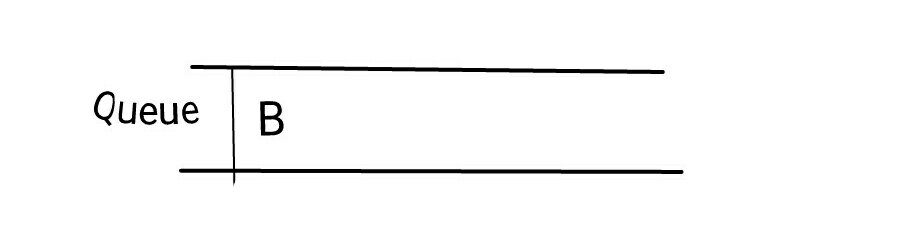


**Step 1:-** The queue is initialized.

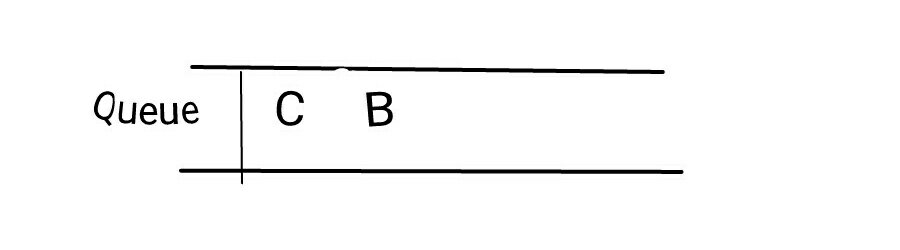


**Step 2: –** First of all we visit node A (initial node) and mark it as visited.

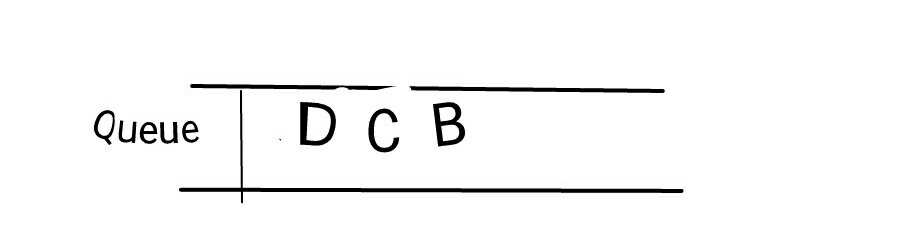
**Step 3: –** After this we look at the adjacent nodes of A. Its adjacent nodes are B, C and D. In this picture we first visit B and put it in the queue.



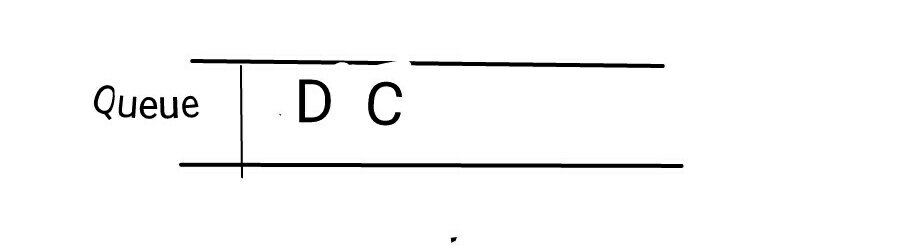
**Step 4: –** After this we visit node C adjacent to node A and keep it in the queue.



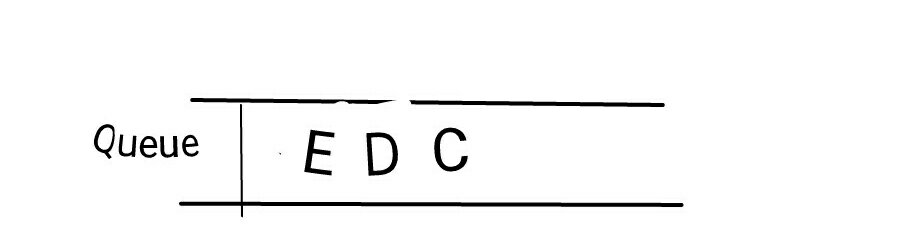
**Step 5:-** After this the last adjacent node of A is visited D and placed in the queue.



**Step 6**:- After this, there are no adjacent nodes of A left, so we remove B from the queue and search its adjacent ones.



**Step 7:-** The adjacent node of node B is E, so we visit E and keep it in the queue.



Now we have no nodes left to visit but we need to remove all the nodes from the queue. And when the queue becomes empty the program will end.

**2:- DFS (depth first search)**

DFS is also an algorithm to traverse and search graph data structures like [BFS .](https://ehindistudy.com/2017/11/23/bfs-hindi/)

In BFS, nodes are visited depth wise.

[Stack](https://ehindistudy.com/2015/11/12/what-is-stack-in-hindi/) is used to implement DFS in data structure .

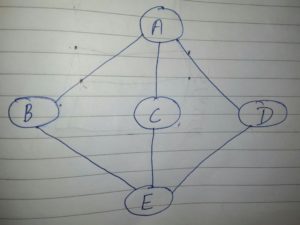
DFS is a recursive algorithm that works on the principle of backtracking.

DFS is used to analyze networks, map routes and solve other computer science problems.

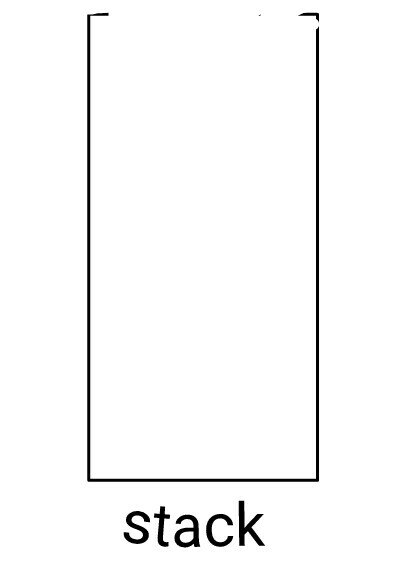
**DFS algorithm**

In DFS we first visit the starting node through the stack. Then by putting all its adjacent nodes in the stack, visiting the node located at the top of the stack, putting all its adjacent nodes in the stack and repeating the process until the stack is empty.

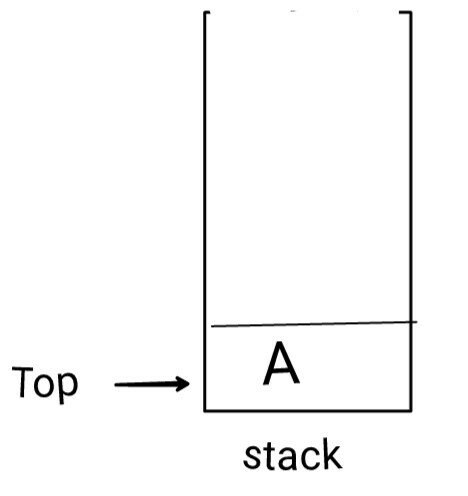
Let us understand its algorithm through the following example.



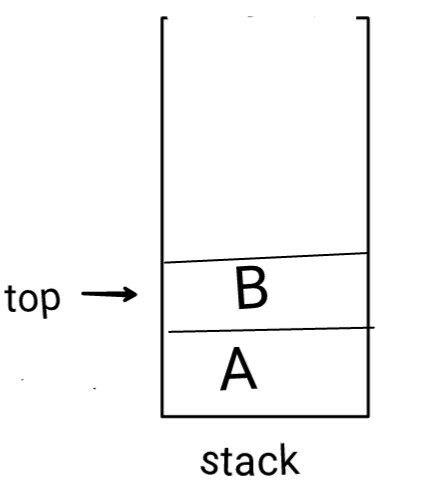
**Step 1: –** The stack is initialized.



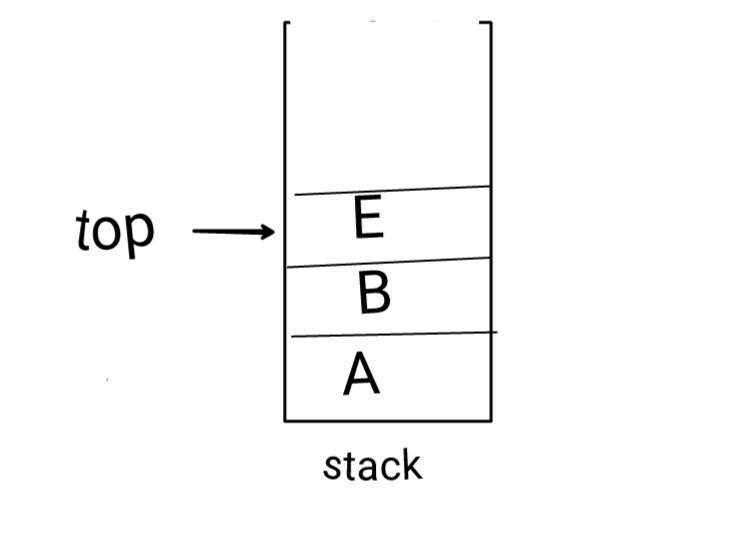
**Step 2: –** Mark node A as visited and put it in the stack. Node A has three adjacent nodes B, C, and D. We can choose any of these nodes. We will choose in alphabetical order.



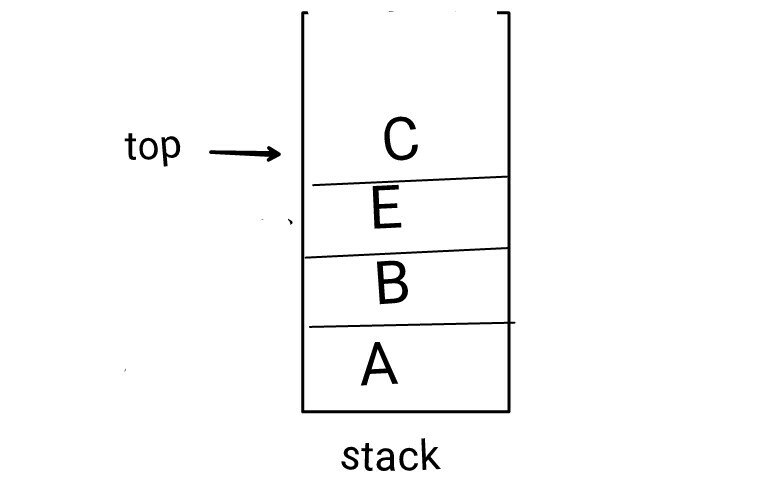
**Step 3:-** Mark node B as visited and put it in the stack. Now we will select any unvisited adjacent nodes of B. The adjacent nodes of B are A and E. Since A has already been visited, we will select E.



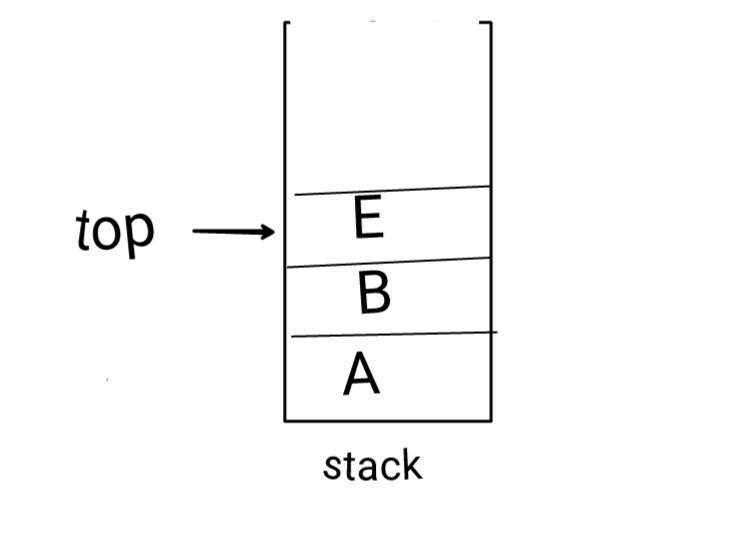
**Step 4:-** Will visit E and mark it as visited and put it in the stack. Here node E has two adjacent nodes C and D, both are unvisited so we will select C. (In alphabetical order.)



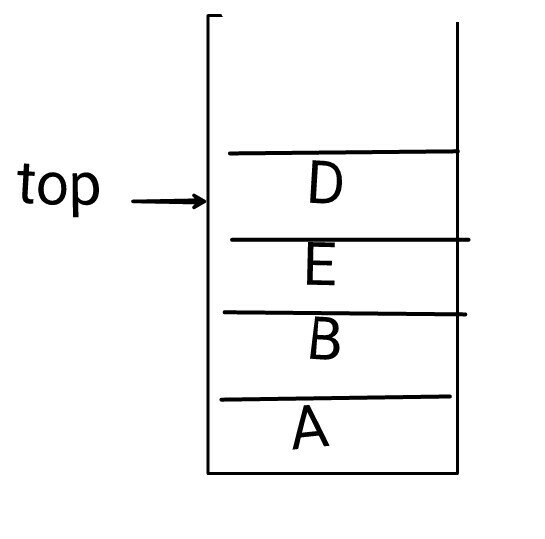
Step 5:- We will visit c and mark it as visited and put it in the stack. Here B has no unvisited adjacent node so it will be removed from the stack.



**Step 6:-** Now E is back on the top of the stack, now we will see whether it has any unvisited adjacent node or not. Its adjacent unvisited node is D.



**Step 7:-** Now we will visit node D and mark it as visited and put it in the stack.



Now there are no nodes left to visit. Now we will remove all the nodes from the stack and when the stack is empty the program will terminate.

## ****# SPANNING TREE****

Spanning tree is created by a [graph](https://ehindistudy.com/2015/11/16/what-is-graph-in-hindi/) . That is, spanning tree is a subset of a graph.

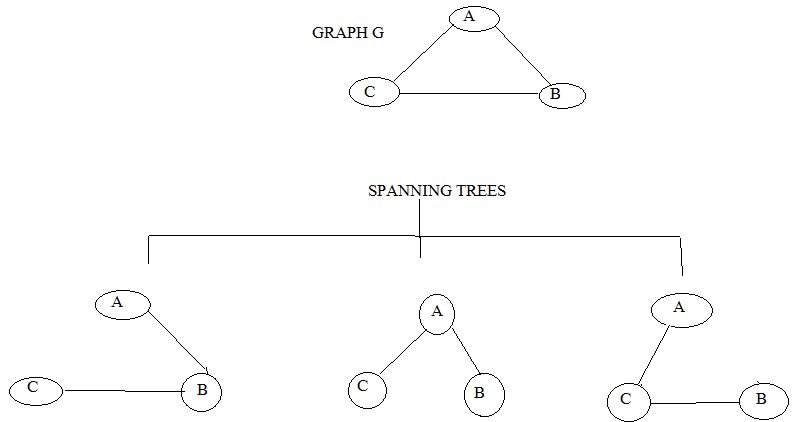
A spanning tree will have all vertices and some edges. Meaning, the number of vertices required in a graph to create a spanning tree. Will use all those vertices but will use only some of all the edges of the graph.

The graph should not be in the form of a cycle. A spanning tree never produces a cycle. And it should not be disconnected.

So we can say that every connected and undirected graph has at least one spanning tree.

* **Properties of spanning tree**

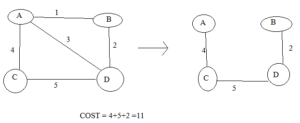
1:- A connected graph can have more than one spanning tree.  
2:- There is no cycle in spanning tree.  
3:- If we remove one edge from the spanning tree then the graph will become disconnected.  
4:- If we add an edge to the spanning tree then it will form a loop or circuit.



Spanning tree can be created from a graph in many ways. Here in the picture above, three types of spanning trees have been created from a graph. In which the graph has all the vertices and some edges. Keep in mind that no spanning tree should create a cycle.

* **Cost of spanning tree :-**

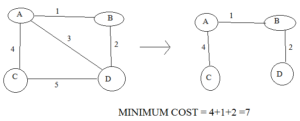
The cost of a spanning tree can be calculated by adding the weights of its vertices.

As :-  


**# MINIMUM SPANNING TREE**

Minimum spanning tree is only a spanning tree. The spanning tree which has the lowest cost is called minimum spanning tree.

**For example: -** Here a spanning tree has been made from a graph whose vertices are joined together resulting in the lowest cost.



There are two types of algorithms to find the minimum spanning tree –

1:- Prim’s algorithm  
2:- Kruskal’s algorithm

* **Prim’s algorithm**

Prim's algorithm is a greedy algorithm by which we find a minimum spanning tree. Meaning, we will find a subset of the edges of a tree which includes all the edges. Prim's algorithm shares similarity with shortest first path.

This algorithm treats a single tree as nodes and adds new nodes to it to create a spanning tree from the given graph.

To create minimum spanning tree in Prim's algorithm, we will first start from a vertex, it can be any arbitrary vertex, we can start creating spanning tree from anywhere. After choosing a vertex, we will look at all the vertexes connected to that vertex to see which vertex has the lowest cost. We will choose the vertex whose cost is lowest and combine it with the previous vertex. This process will continue until all the vertices are connected and at the same time it will also be kept in mind that not a single cycle is created.

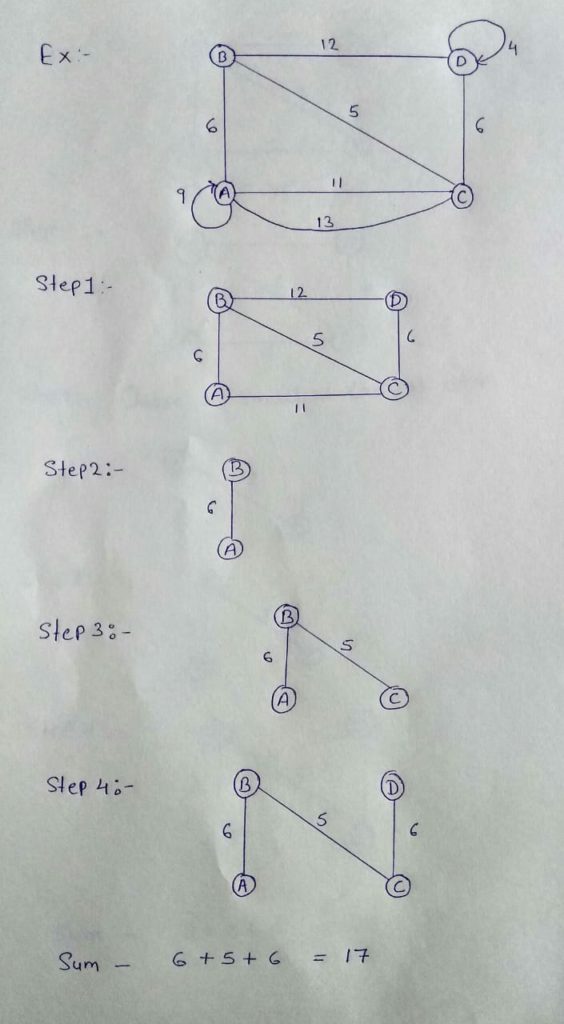
**Algorithm :-**

1:- Remove all the loops and all the parallel edges.  
(will separate all loops and parallel edges)

2:- Choose any arbitrary node as root node.  
(We will choose one node as the root node, this node can be any, we can start creating spanning tree from anywhere.)

3:- Check outgoing edges and select the one with less cost.  
(We will check the outgoing edges and select one edge which has the lowest cost.)

**Example 1**



**EXAMPLE 2,3,4 HINDI WALE NOTES ME HAI**

* **Kruskal’s algorithm**

Kruskal's algorithm finds the minimum spanning tree using a greedy approach. It follows all the properties of minimum spanning tree. In this algorithm the graph is treated like a forest and all the nodes are treated like an individual tree.

To find the minimum spanning tree with the help of Kruskal algorithm, first of all we will choose an edge with the lowest cost and then again we will choose the edge with the lowest cost. This process will continue until all the vertices are connected and simultaneously Not even a single cycle should be formed. In this, only the edge with lowest cost will be chosen, even if one vertex is not connected to another vertex.

**Algorithm :-**

1:- Remove all the loops and all parallel edges.

(will separate all loops and parallel edges)

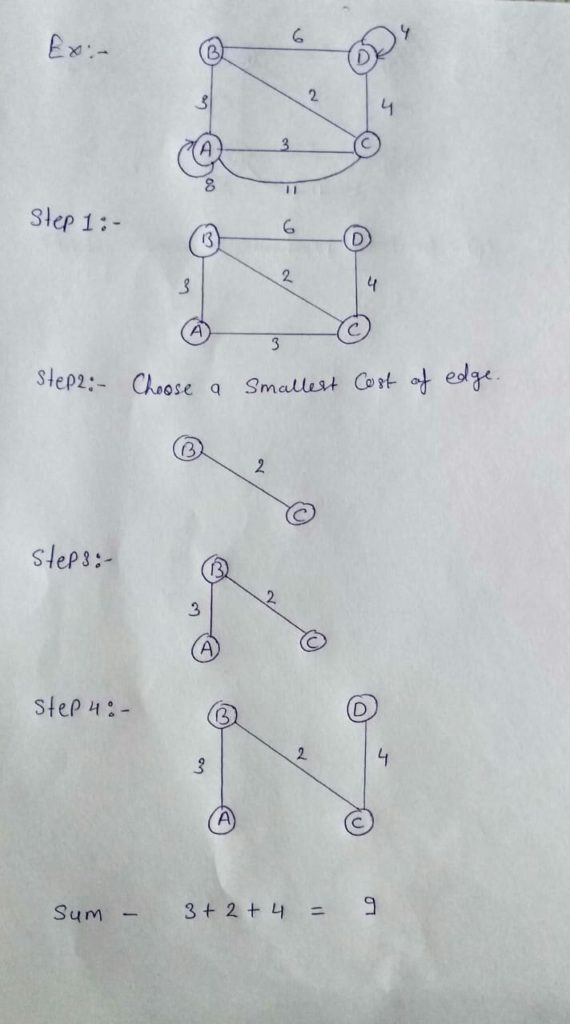
2:- Arrange all edges in their increasing order of weight.

(All the edges will be arranged in their increasing order of weight.)

3:- Add all edges which has the least weightage.

(All edges that have the lowest importance or cost will be added.)

**Example:-**



**# SHORTEST PATH PROBLEM USING DIJKSTRA METHOD**

Dijkstra's algorithm solves the problem of single-source shortest paths on a directed weighted graph.  
G = (V,E) where all the edges are non-negative.

This algorithm is also similar to prim's algorithm. Like prim's, in this also we generate SPT (shortest path [tree](https://ehindistudy.com/2015/10/03/tree-in-hindi/) ) with the given source . This algorithm was proposed by Dutch scientist Edsger Dijkstra in 1959. This is a greedy algorithm.

Below you are given the algorithm in which we have used a function Extract-min(), which extracts the node with the smallest key.

**Algorithm: Dijkstra’s-Algorithm (G, w, s) for each vertex v Є G.V**

v.d := ∞

v.∏ := NIL

s.d := 0

S := Ф

Q := G.V

while Q ≠ Ф

u := Extract-Min (Q)

S := S U {u}

for each vertex v Є G.adj[u]

if v.d > u.d + w(u, v)

v.d := u.d + w(u, v)

v.∏ := u

[The complexity](https://ehindistudy.com/2019/12/06/time-complexity-and-space-complexity-in-hindi/) of this algorithm completely depends on the implementation of the Extract –min() function. If the extract-min function is implemented using linear search then the complexity of this algorithm will be – O(V 2  + E) . If we use min heap then its complexity will further reduce.

* **Disadvantage of Dijkstra’s algorithm**

Its disadvantages are as follows:-

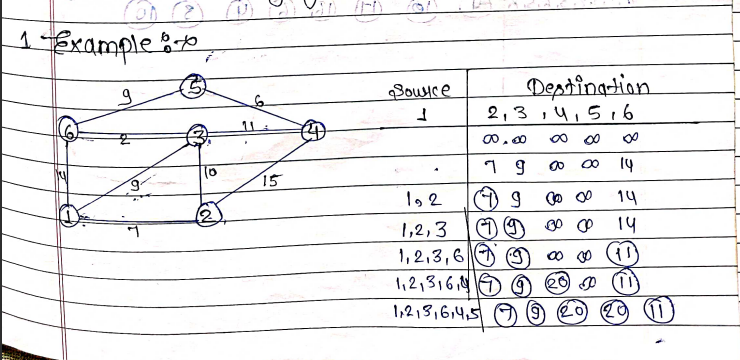
1. It does a blind search, so a lot of time is wasted in processing.
2. It cannot handle negative edges.
3. This leads to an acyclic graph and most of the times it does not achieve the correct shortest path.
4. We need to keep track of the vertices that have been visited once.

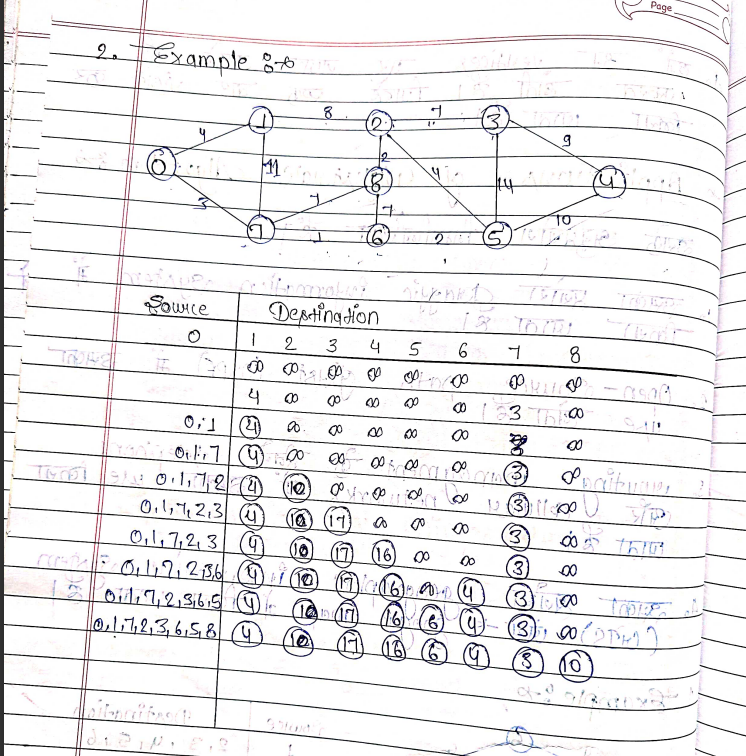
* **Applications of Dijkstra’s algorithm**

Its applications are as follows:-

1. It is used in traffic information system.
2. It is used in [open-source path first (OSPF) .](https://ehindistudy.com/2017/11/01/routing-protocol-hindi/)
3. It is used in telephone and cellular networks for routing management.
4. It is used in geographical information system (GIS) like:- Google Map.

**Example:-1**





**Example:- 2**