TOPOLOGICAL SORTING

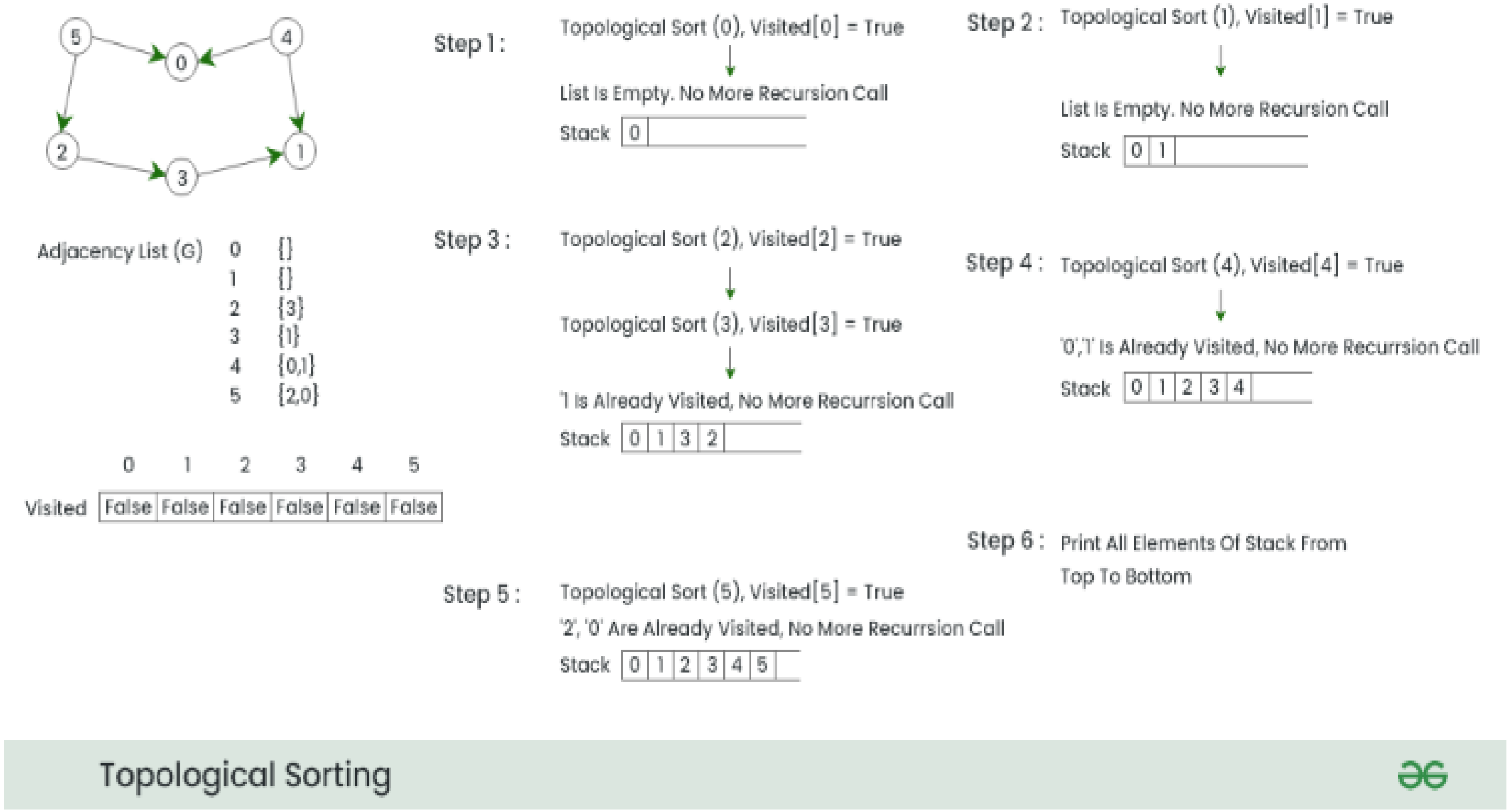
Algorithm for Topological Sorting using DFS:

Here’s a step-by-step algorithm for topological sorting using Depth First Search (DFS):

- Create a graph with **n** vertices and **m**-directed edges.
- Initialize a stack and a visited array of size **n**.
- For each unvisited vertex in the graph, do the following:
    - Call the DFS function with the vertex as the parameter.
    - In the DFS function, mark the vertex as visited and recursively call the DFS function for all unvisited neighbors of the vertex.
    - Once all the neighbors have been visited, push the vertex onto the stack.
- After all, vertices have been visited, pop elements from the stack and append them to the output list until the stack is empty.
- The resulting list is the topologically sorted order of the graph.

Illustration Topological Sorting Algorithm:

Below image is an illustration of the above approach:



Adjacency List (G)

0	{}
1	{}
2	{3}
3	{1}
4	{0,1}
5	{2,0}

	0	1	2	3	4	5
Visited	False	False	False	False	False	False

Step 1 : Topological Sort (0), Visited[0] = True

List Is Empty. No More Recursion Call

Stack | 0 |

Step 2 : Topological Sort (1), Visited[1] = True

List Is Empty. No More Recursion Call

Stack | 0 | 1 |

Step 3 : Topological Sort (2), Visited[2] = True

Topological Sort (3), Visited[3] = True

'1' Is Already Visited, No More Recurrsion Call

Stack | 0 | 1 | 3 | 2 |

Step 4 : Topological Sort (4), Visited[4] = True

'0','1' Is Already Visited, No More Recurrsion Call

Stack | 0 | 1 | 2 | 3 | 4 |

Step 5 : Topological Sort (5), Visited[5] = True

'2', '0' Are Already Visited, No More Recurrsion Call

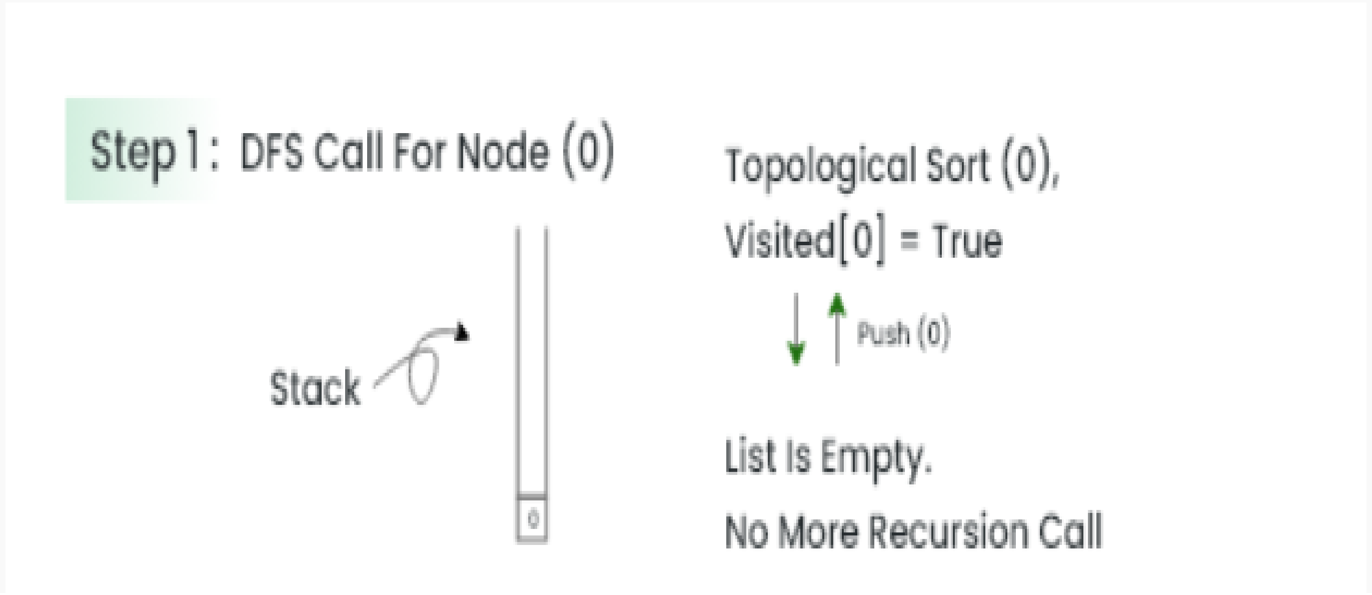
Stack | 0 | 1 | 2 | 3 | 4 | 5 |

Step 6 : Print All Elements Of Stack From Top To Bottom

Overall workflow of topological sorting

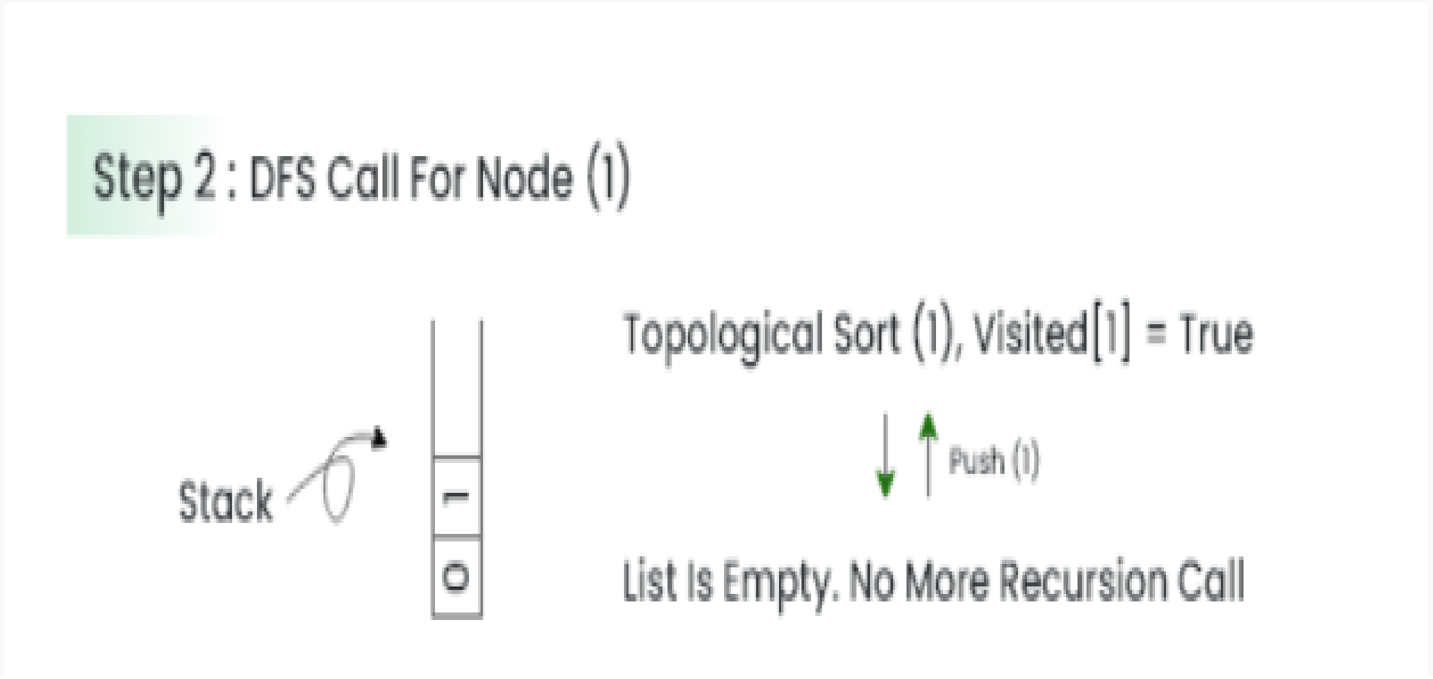
Step 1:

- *We start DFS from node 0 because it has zero incoming Nodes*
- *We push node 0 in the stack and move to next node having minimum number of adjacent nodes i.e. node 1.*



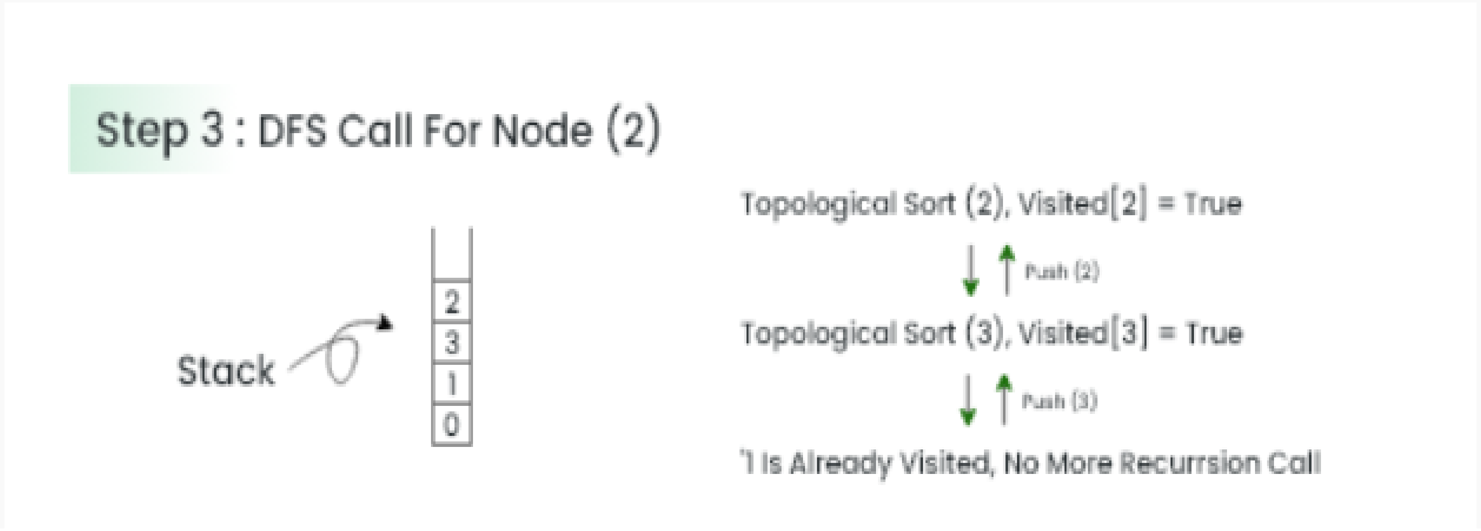
Step 2:

- *In this step , because there is no adjacent of this node so push the node 1 in the stack and move to next node.*


Step 2 : DFS Call For Node (1)
Stack
1
0
Topological Sort (1), Visited[1] = True
Push (1)
List Is Empty. No More Recursion Call

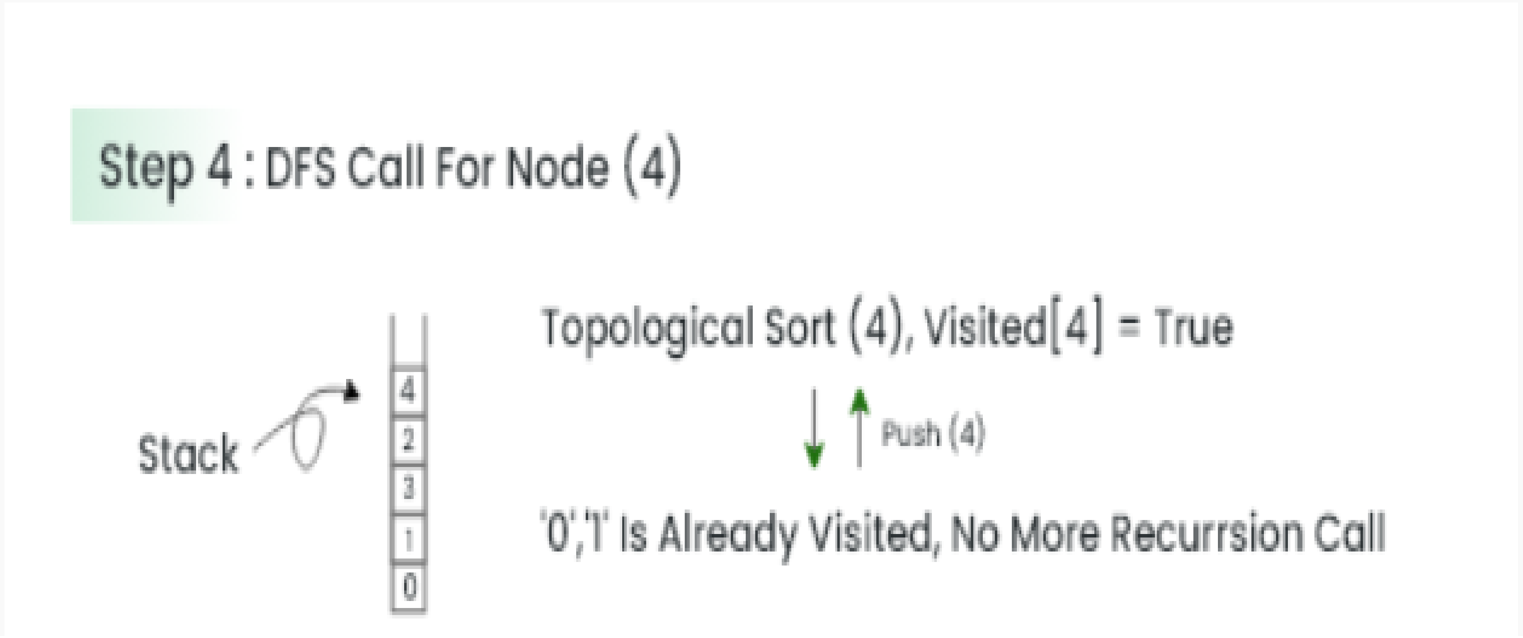
Step 3:

- *In this step , We choose node 2 because it has minimum number of adjacent nodes after 0 and 1 .*
- *We call DFS for node 2 and push all the nodes which comes in traversal from node 2 in reverse order.*
- *So push 3 then push 2 .*


Step 3 : DFS Call For Node (2)
Stack
2
3
1
0
Topological Sort (2), Visited[2] = True
Push (2)
Topological Sort (3), Visited[3] = True
Push (3)
'1 Is Already Visited, No More Recurrsion Call

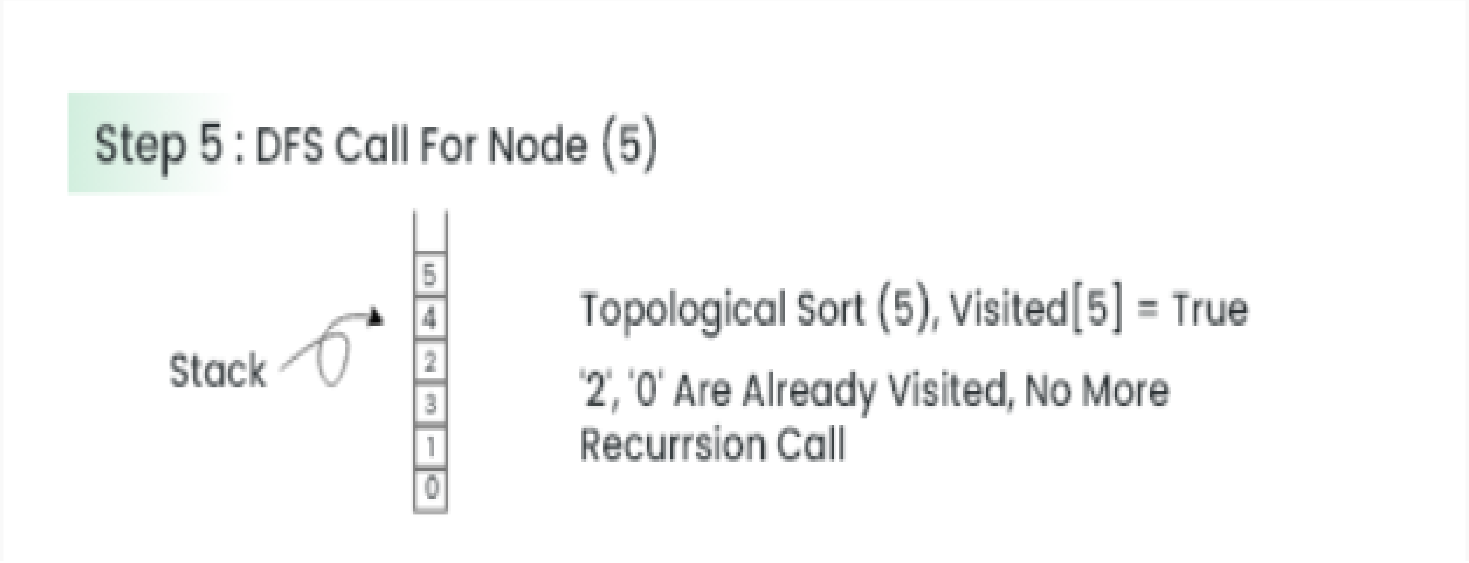
Step 4:

- *We now call DFS for node 4*
- *Because 0 and 1 already present in the stack so we just push node 4 in the stack and return.*


Step 4 : DFS Call For Node (4)
Stack
4
2
3
1
0
Topological Sort (4), Visited[4] = True
Push (4)
'0','1' Is Already Visited, No More Recurrsion Call

Step 5:

- *In this step because all the adjacent nodes of 5 is already in the stack we push node 5 in the stack and return.*


Step 5 : DFS Call For Node (5)
Stack
5
4
2
3
1
0
Topological Sort (5), Visited[5] = True
'2', '0' Are Already Visited, No More Recurrsion Call

Step 6: *This is the final step of the Topological sorting in which we pop all the element from the stack and print it in that order .*

SPANNING TREE

- *A spanning tree is a subset of Graph G, such that all the vertices are connected using minimum possible number of edges.*
- *A spanning tree does not have cycles and a graph may have more than one spanning tree.*

- **Properties of a Spanning Tree:**
- A Spanning tree does not exist for a disconnected graph.

- For a connected graph having **N** vertices then the number of edges in the spanning tree for that graph will be **N-1**.

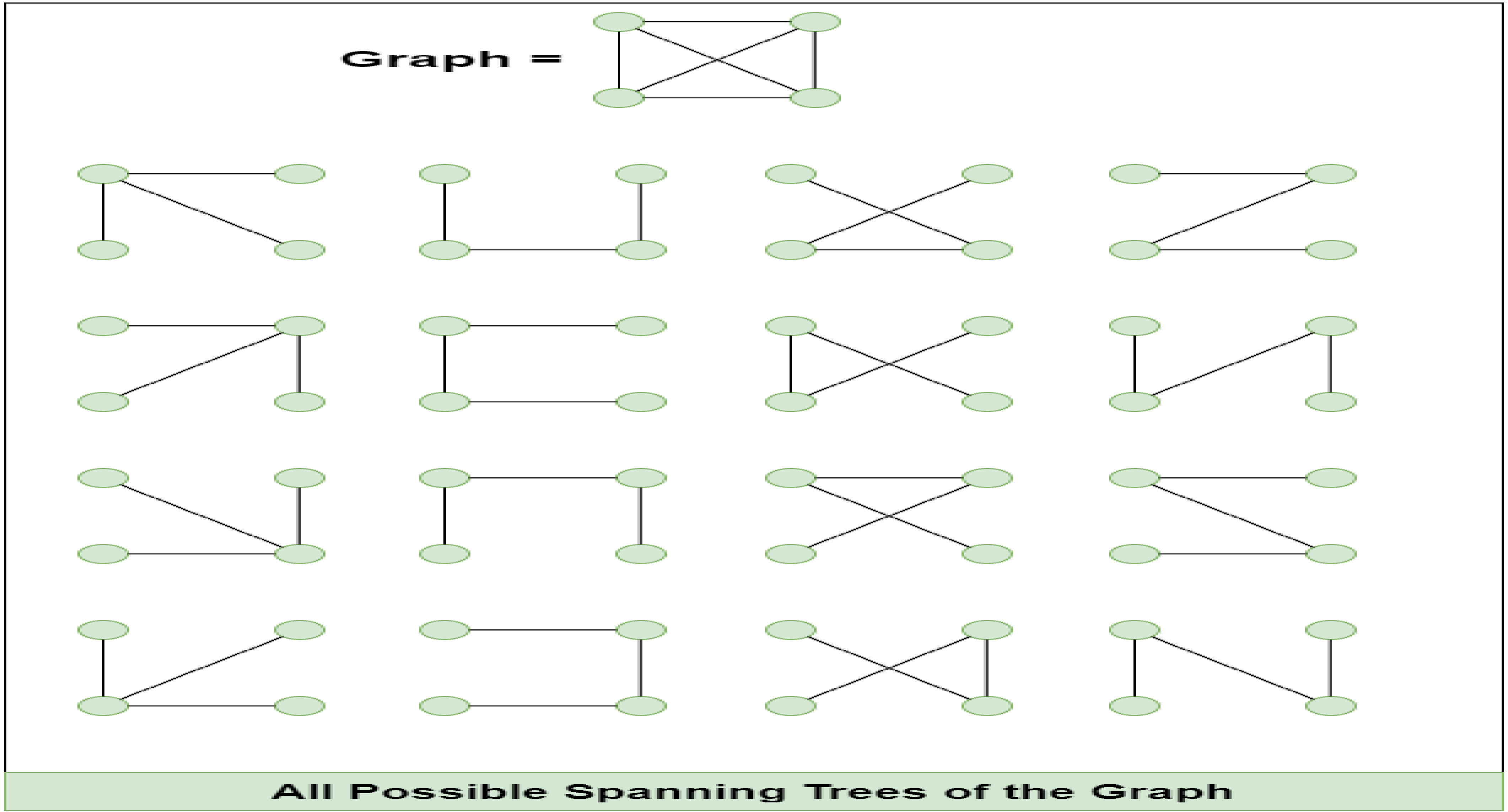
- A Spanning tree does not have any cycle.

- We can construct a spanning tree for a complete graph by removing **E-N+1** edges, where **E** is the number of Edges and **N** is the number of vertices.

- **Cayley's Formula:** It states that the number of spanning trees in a complete graph with N vertices is N^{N-2}

  - For example: N=4, then maximum number of spanning tree possible =4^{4-2} = 16 (shown in the above image).

A spanning tree does not have cycles and a graph may have more than one spanning tree.

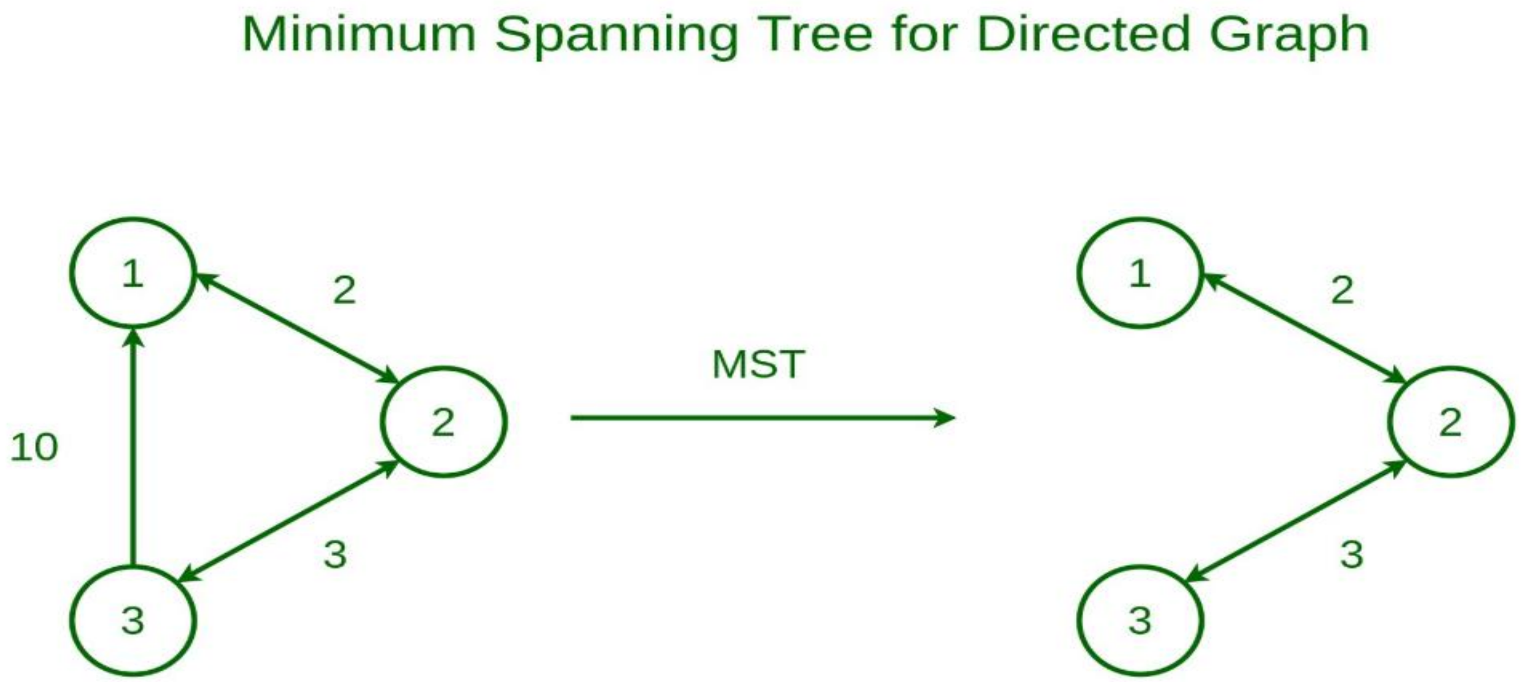


- **Real World Applications of A Spanning Tree:**
- Several path finding algorithms, such as Dijkstra's algorithm, Prim’s algorithm and A* search algorithm, internally build a spanning tree as an intermediate step.
- Building Telecommunication Network.
- Image Segmentation to break an image into distinguishable components.
- Computer Network Routing Protocol

What is Minimum Spanning Tree (MST)

- The weight of a spanning tree is determined by the sum of weight of all the edge involved in it.

A **minimum spanning tree (MST)** is defined as a spanning tree that has the minimum weight among all the possible spanning trees.



- Properties of Minimum Spanning Tree:
- A minimum spanning tree connects all the vertices in the graph, ensuring that there is a path between any pair of nodes.

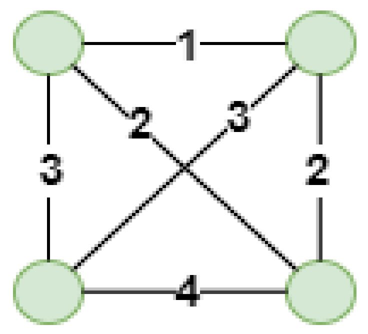
- An **MST** is **acyclic**, meaning it contains no cycles. This property ensures that it remains a tree and not a graph with loops.

- An **MST** with **V** vertices (where **V** is the number of vertices in the original graph) will have exactly **V - 1** edges, where **V** is the number of vertices.

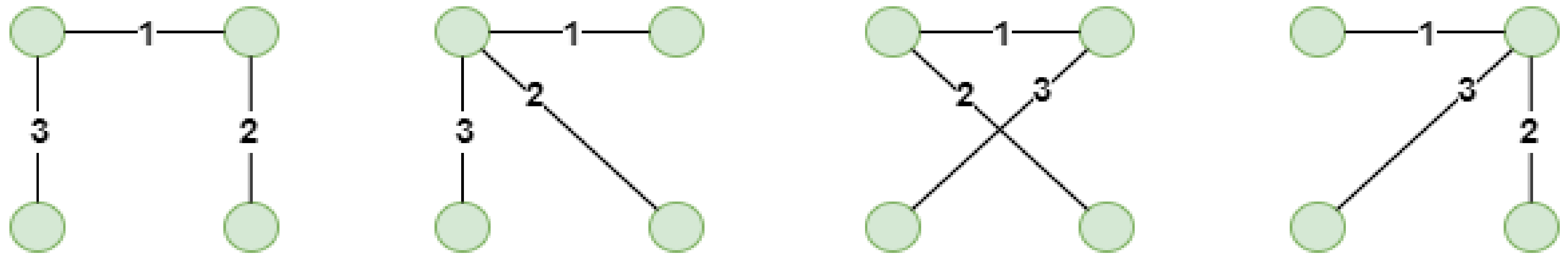
- An **MST** is optimal for minimizing the total edge weight, but it may not necessarily be unique.

- The cut property states that if you take any cut (a partition of the vertices into two sets) in the original graph and consider the minimum-weight edge that crosses the cut, that edge is part of the **MST**.

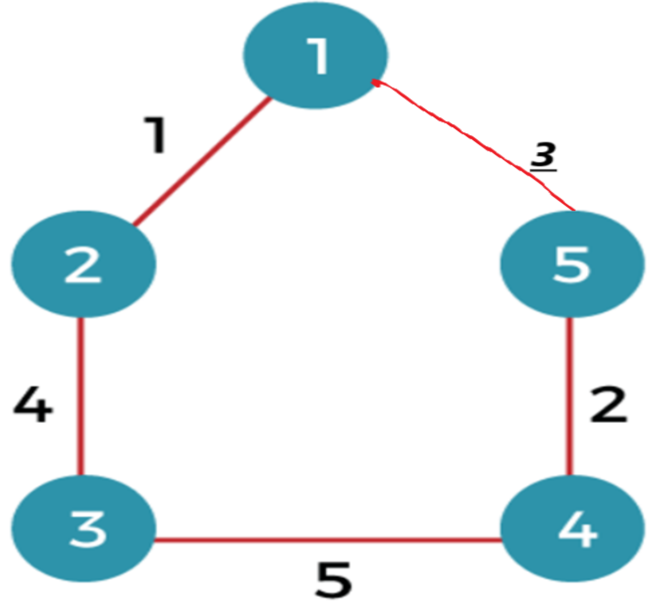
Graph(V,E) =

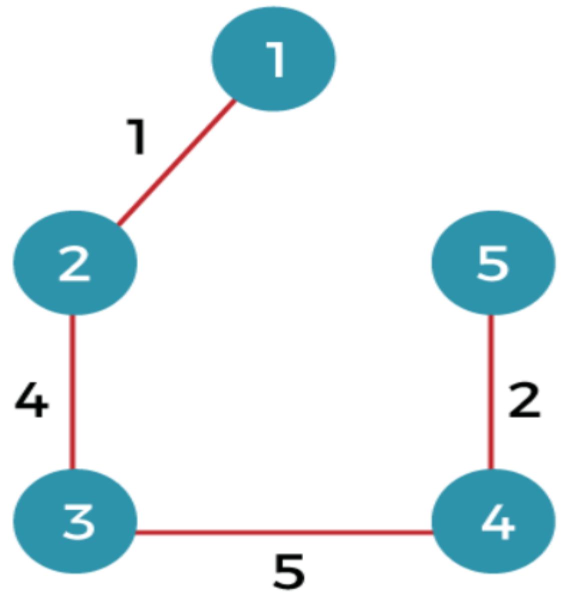


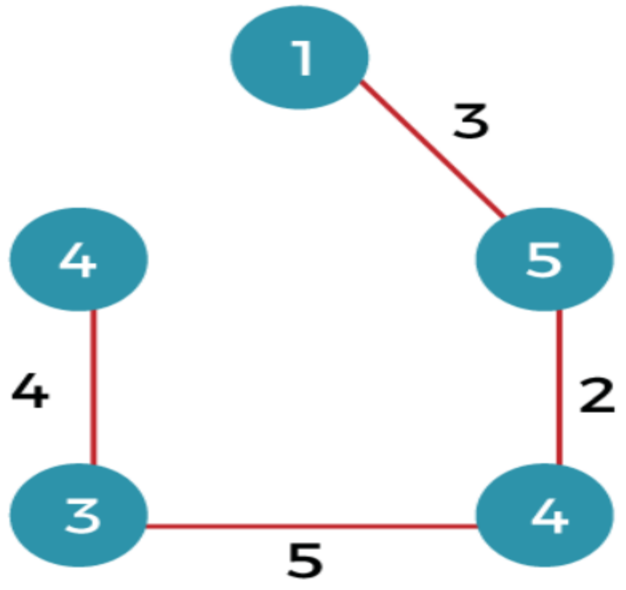
All Possible MST's of the above Graph

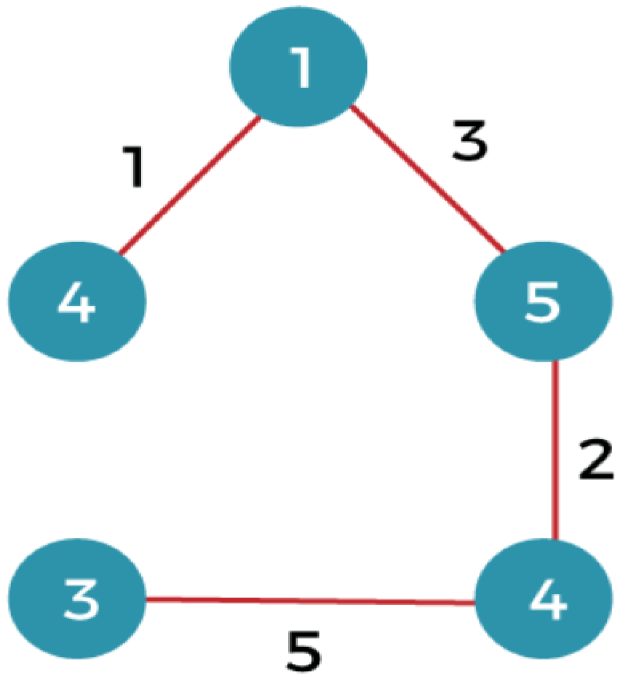


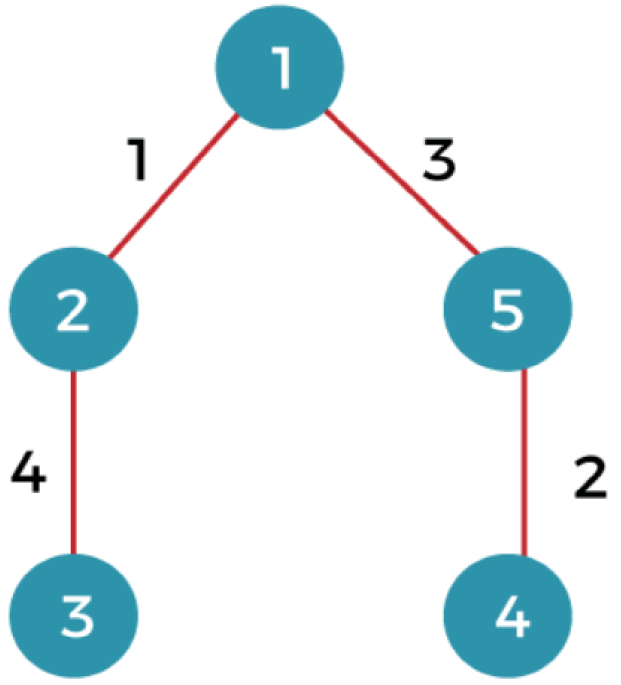
MST Cost =6 MST Cost =6 MST Cost =6 MST Cost =6


1
2
3
4
5
1
3
4
2
5


1
2
3
4
5
1
4
2
5


1
4
5
3
4
3
4
2
5


1
4
5
3
4
1
3
2
5


1
2
5
3
4
1
3
4
2

- **The following are the spanning trees that we can make from the above graph.**
- The first spanning tree is a tree in which we have removed the edge between the vertices 1 and 5 shown as below:
  The sum of the edges of the above tree is **(1 + 4 + 5 + 2): 12**
- The second spanning tree is a tree in which we have removed the edge between the vertices 1 and 2 shown as below:
  The sum of the edges of the above tree is **(3 + 2 + 5 + 4) : 14**
- The third spanning tree is a tree in which we have removed the edge between the vertices 2 and 3 shown as below:
  The sum of the edges of the above tree is **(1 + 3 + 2 + 5) : 11**
- The fourth spanning tree is a tree in which we have removed the edge between the vertices 3 and 4 shown as below:
  The sum of the edges of the above tree is **(1 + 3 + 2 + 4) : 10**. The edge cost 10 is minimum so it is a minimum spanning tree.

General properties of minimum spanning tree:

- If we remove any edge from the spanning tree, then it becomes disconnected. Therefore, we cannot remove any edge from the spanning tree.
- If we add an edge to the spanning tree then it creates a loop. Therefore, we cannot add any edge to the spanning tree.
- In a graph, each edge has a distinct weight, then there exists only a single and unique minimum spanning tree. If the edge weight is not distinct, then there can be more than one minimum spanning tree.
- A complete undirected graph can have an n^{n-2} number of spanning trees.
- Every connected and undirected graph contains atleast one spanning tree.
- The disconnected graph does not have any spanning tree.
- In a complete graph, we can remove maximum (e-n+1) edges to construct a spanning tree.

Application of A Minimum Spanning Tree:

- **Network design**: Spanning trees can be used in network design to find the minimum number of connections required to connect all nodes. Minimum spanning trees, in particular, can help minimize the cost of the connections by selecting the cheapest edges.

- **Image processing**: Spanning trees can be used in image processing to identify regions of similar intensity or color, which can be useful for segmentation and classification tasks.

- **Social network analysis**: Spanning trees and minimum spanning trees can be used in social network analysis to identify important connections and relationships among individuals or groups.