Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

SAIRAM DIGITAL RESOURCES

YEAR II | SEM III

CS8391

DATA STRUCTURES (Common to CSE & IT)

UNIT NO 4

NON LINEAR DATA STRUCTURES

4.1 DEPTH FIRST TRAVERSAL

COMPUTER SCIENCE & ENGINEERING

DEPTH FIRST SEARCH

Graph G(V, E) directed or undirected

Adjacency list representation

Goal: Systematically explore every vertex and every edge

Idea: search deeper whenever possible – Using a LIFO queue (Stack; FIFO queue used in BFS)

- The DFS algorithm is a recursive algorithm that uses the idea of backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking.

DEPTH FIRST SEARCH

Depth First search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root (top) node of a tree and goes as far as it can down a given branch (path), then backtracks until it finds an unexplored path, and then explores it. The algorithm does this until the entire graph has been explored.

DEPTH FIRST SEARCH

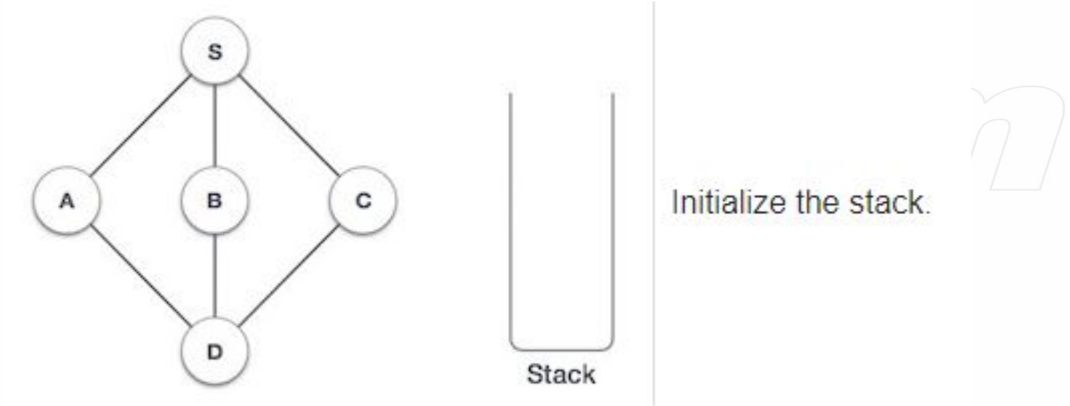
Rule 1 − Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack.

Rule 2 − If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the vertices from the stack, which do not have adjacent vertices.)

Rule 3 − Repeat Rule 1 and Rule 2 until the stack is empty.

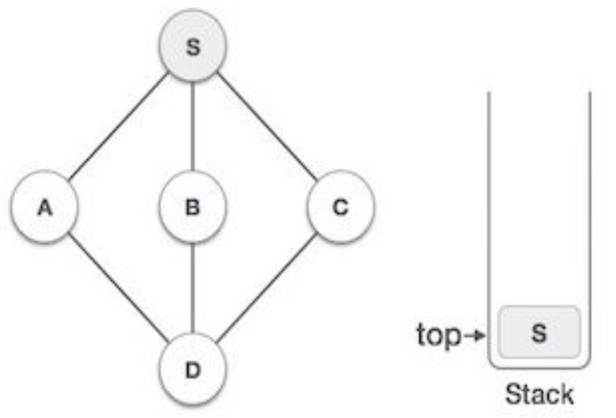
DEPTH FIRST SEARCH

Step :1



DEPTH FIRST SEARCH

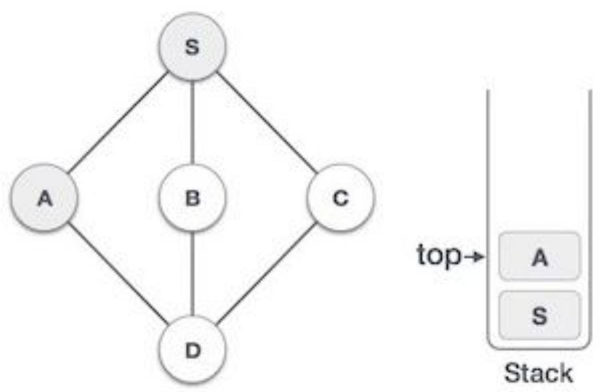
Step:2



Mark **S** as visited and put it onto the stack. Explore any unvisited adjacent node from **S**. We have three nodes and we can pick any of them. For this example, we shall take the node in an alphabetical order.

DEPTH FIRST SEARCH

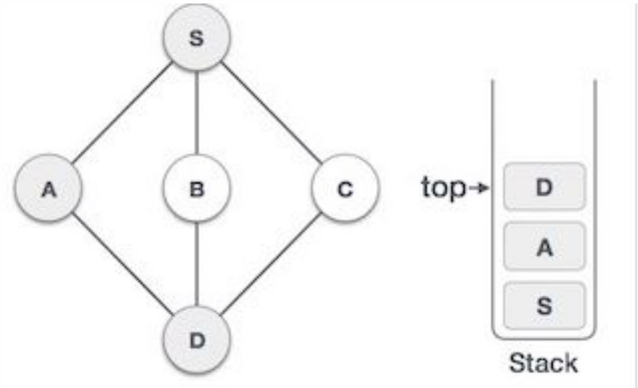
Step:3



Mark **A** as visited and put it onto the stack. Explore any unvisited adjacent node from A. Both **S** and **D** are adjacent to **A** but we are concerned for unvisited nodes only.

DEPTH FIRST SEARCH

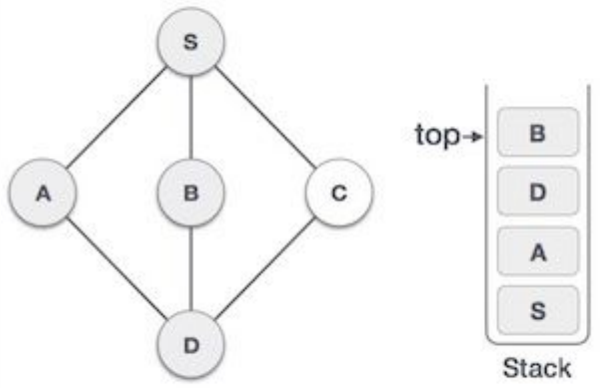
Step:4



Visit **D** and mark it as visited and put onto the stack. Here, we have **B** and **C** nodes, which are adjacent to **D** and both are unvisited. However, we shall again choose in an alphabetical order.

DEPTH FIRST SEARCH

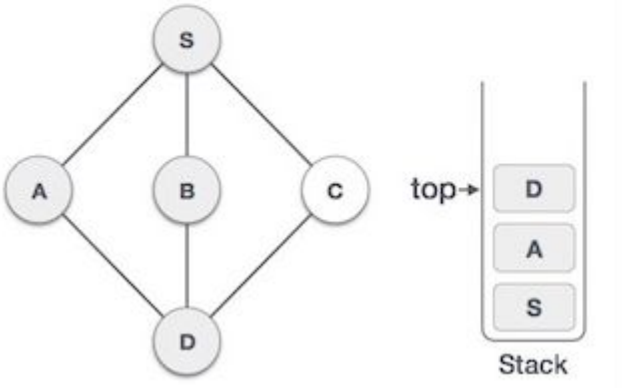
Step:5



We choose **B**, mark it as visited and put onto the stack. Here **B** does not have any unvisited adjacent node. So, we pop **B** from the stack.

DEPTH FIRST SEARCH

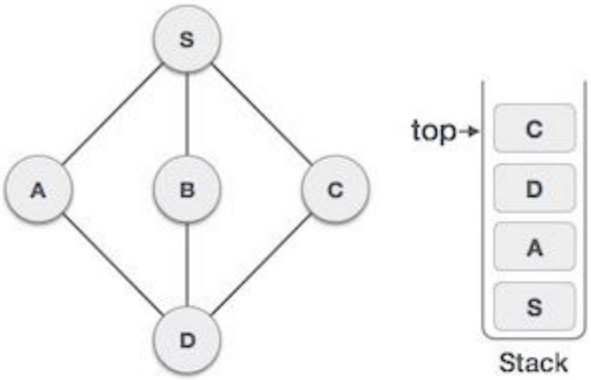
Step:6



We check the stack top for return to the previous node and check if it has any unvisited nodes. Here, we find **D** to be on the top of the stack.

DEPTH FIRST SEARCH

Step:7



Only unvisited adjacent node is from **D** is **C** now. So we visit **C**, mark it as visited and put it onto the stack.

Applications of DFS

1) For a weighted graph, DFS traversal of the graph produces the minimum spanning tree and all pair shortest path tree.

2) Detecting cycle in a graph

3) Path Finding

4) Topological Sorting

5) To test if a graph is bipartite

6) Finding Strongly Connected Components of a graph

7) Solving puzzles with only one solution, such as mazes.