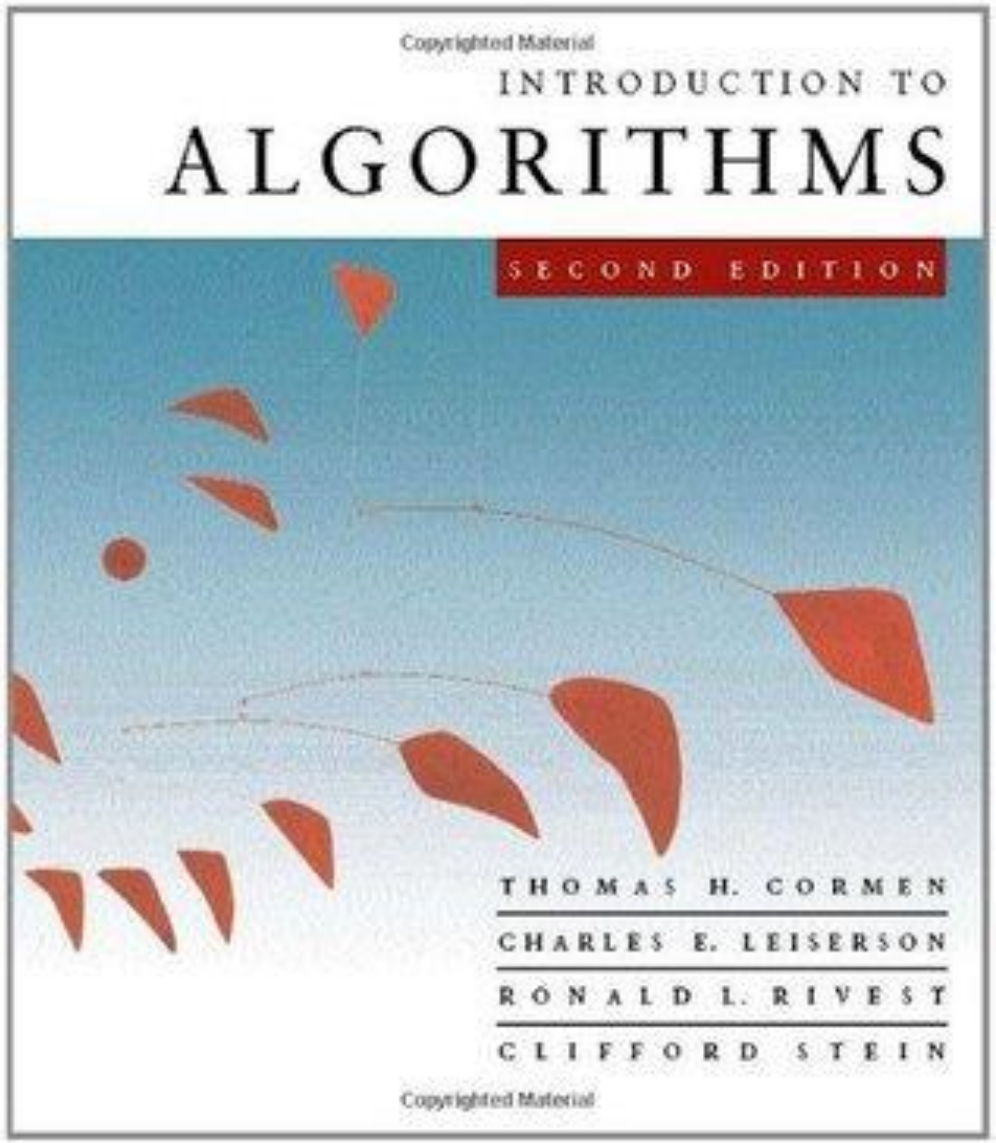
Fall 2024

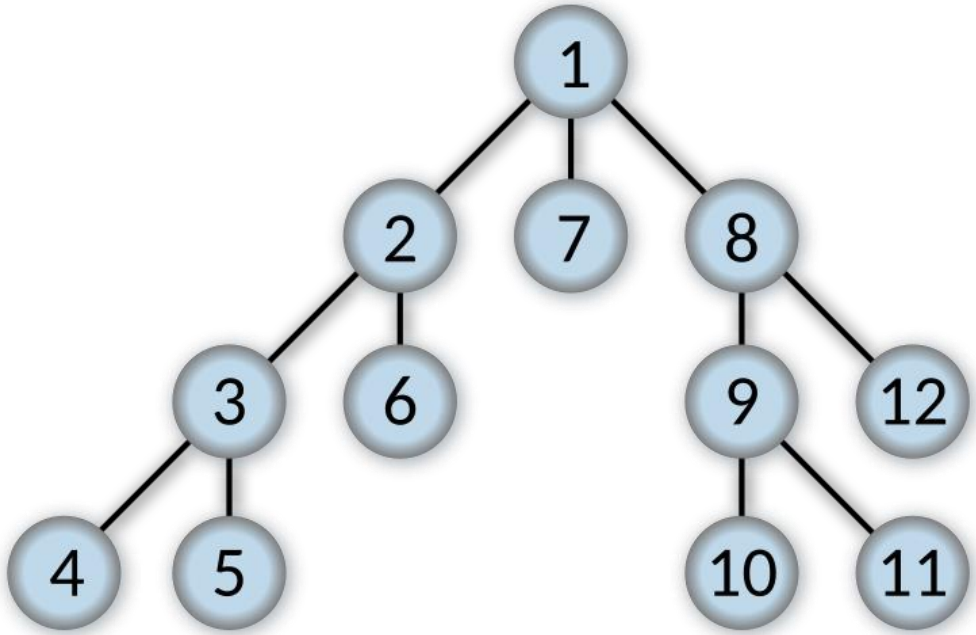
Prof. Alaa Sheta

Algorithms



Depth First Search Algorithm

- **Depth First Search (DFS)** is a graph traversal algorithm used to explore vertices and edges of a graph by starting at a root (or any arbitrary node), **moving as far as possible along each branch before backtracking**.
- **It explores one branch of the graph as deeply as possible before moving to another.**

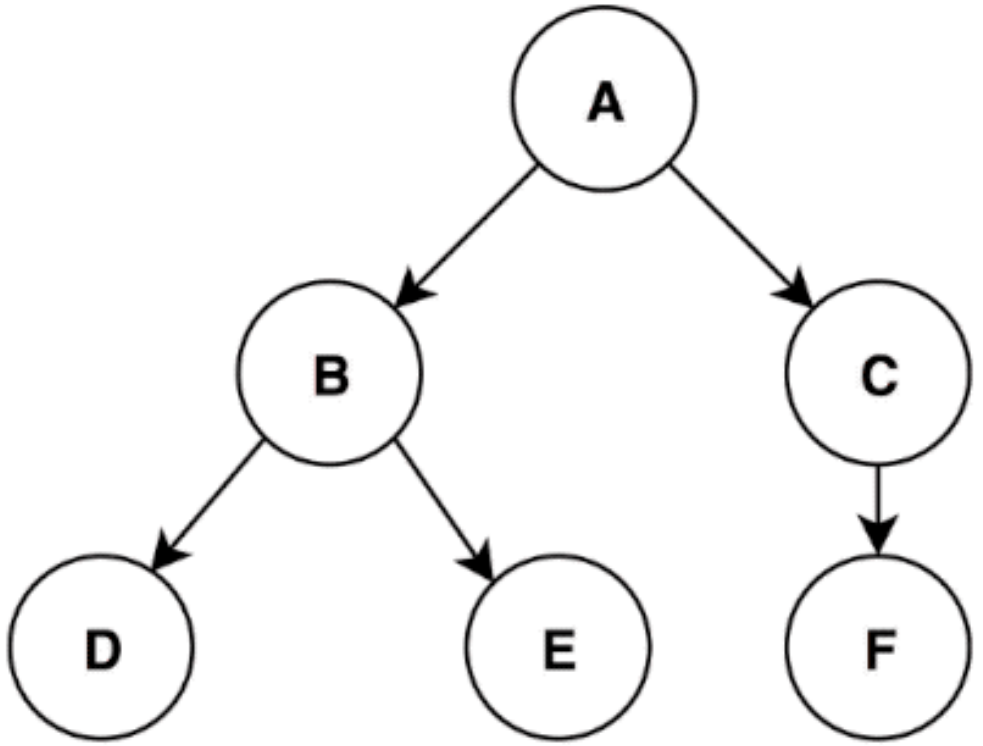


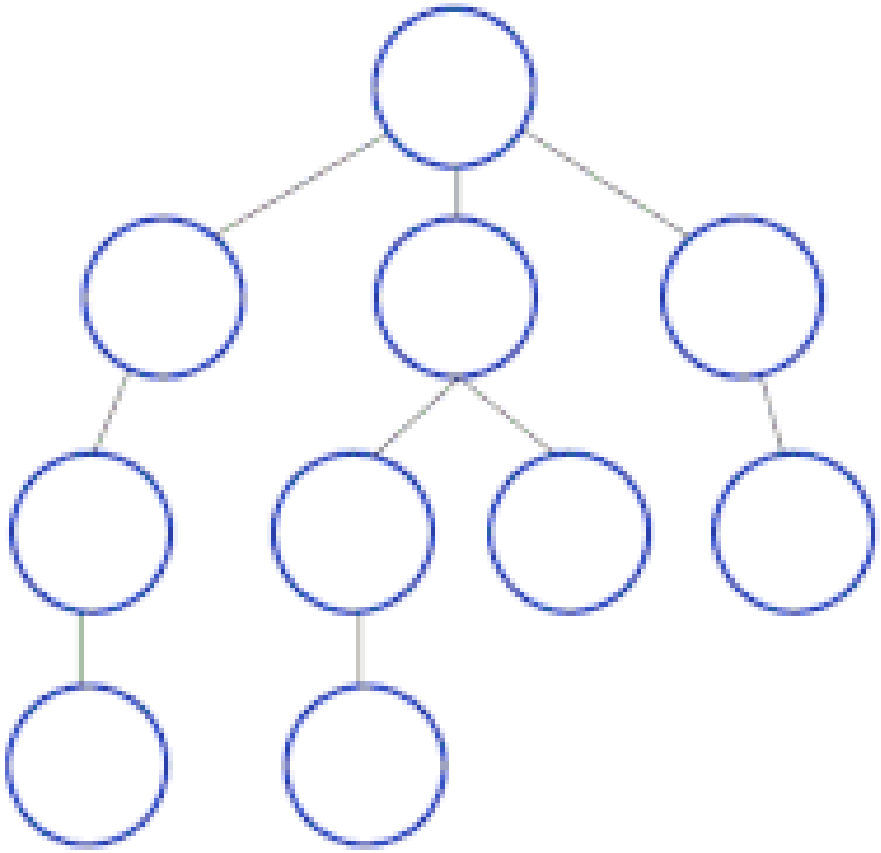
Characteristics of DFS

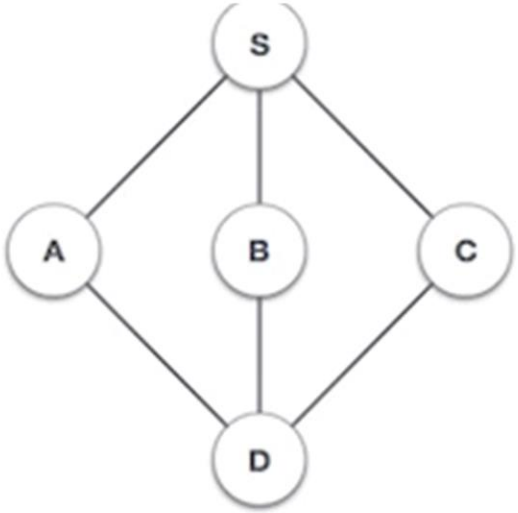
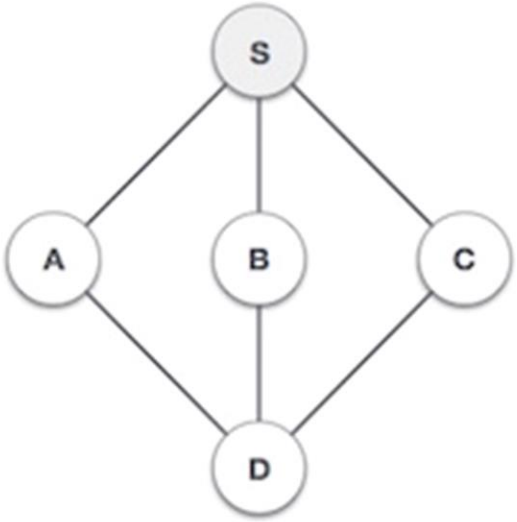
- It is called a **depth-first** search because it prioritizes going deep in the graph/tree before exploring other branches.
- It uses a **stack** (often implemented using recursion) to keep track of the vertices to visit.
- DFS can be applied to both **directed** and **undirected** graphs.

DFS Algorithm

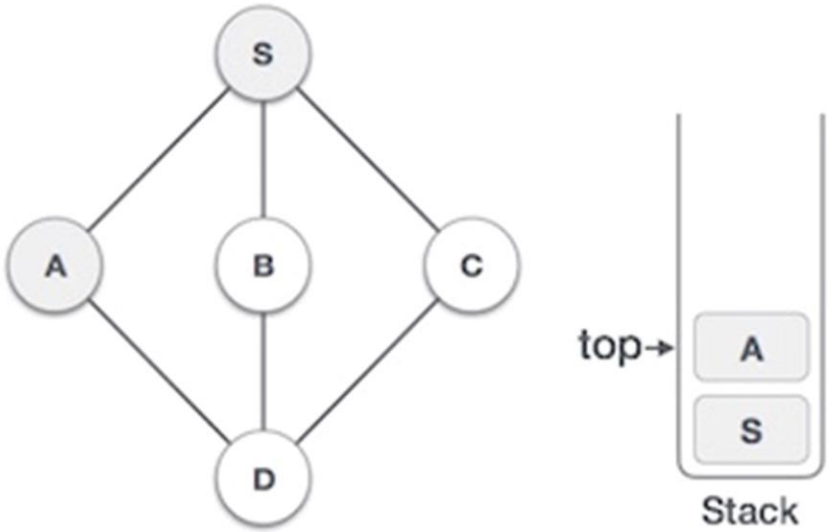
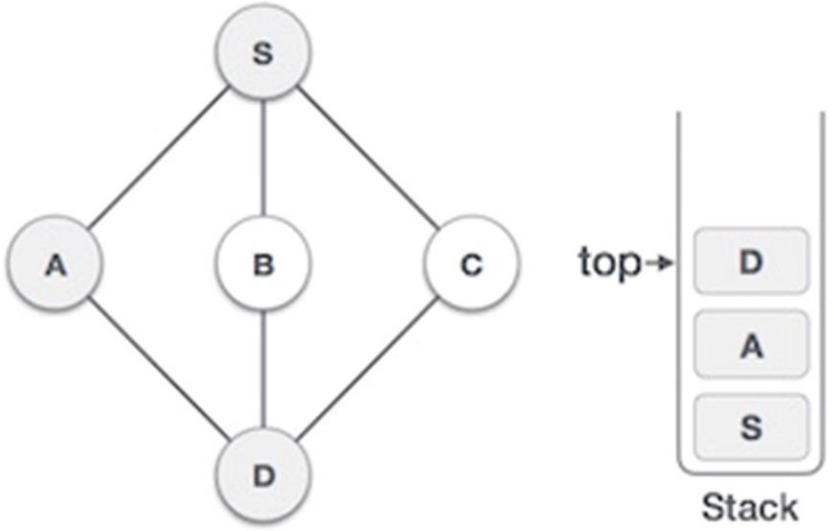
1. **Start** at an initial vertex.
2. **Visit** the vertex and mark it as visited.
3. For each **unvisited neighbor** of the current vertex, recursively visit that neighbor.
4. If all neighbors are visited, **backtrack** to the previous vertex and explore other unvisited vertices.
5. Repeat this process until all vertices have been visited.



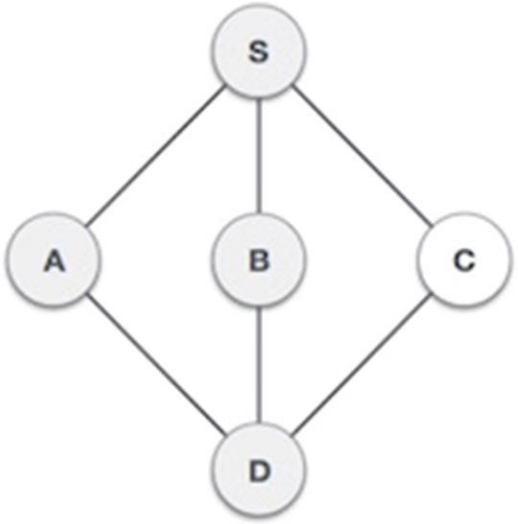
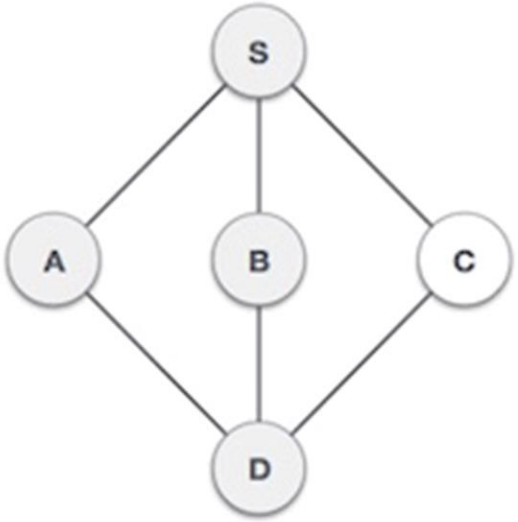


1			Initialize the stack.
2			Mark **S** as visited and put it onto the stack. Explore any unvisited adjacent node from **S**. We have three nodes and we can pick any of them. For this example, we shall take the node in an alphabetical order.

Output = S

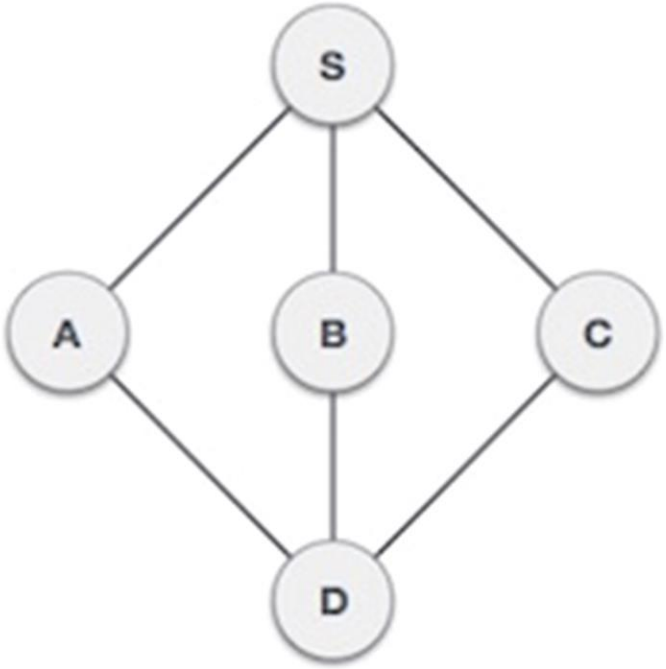
3		Mark **A** as visited and put it onto the stack. Explore any unvisited adjacent node from A. Both **S** and **D** are adjacent to **A** but we are concerned for unvisited nodes only.
4		Visit **D** and mark it as visited and put onto the stack. Here, we have **B** and **C** nodes, which are adjacent to **D** and both are unvisited. However, we shall again choose in an alphabetical order.

Output = S, A, D

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.
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.

Output = S, A, D, B

7

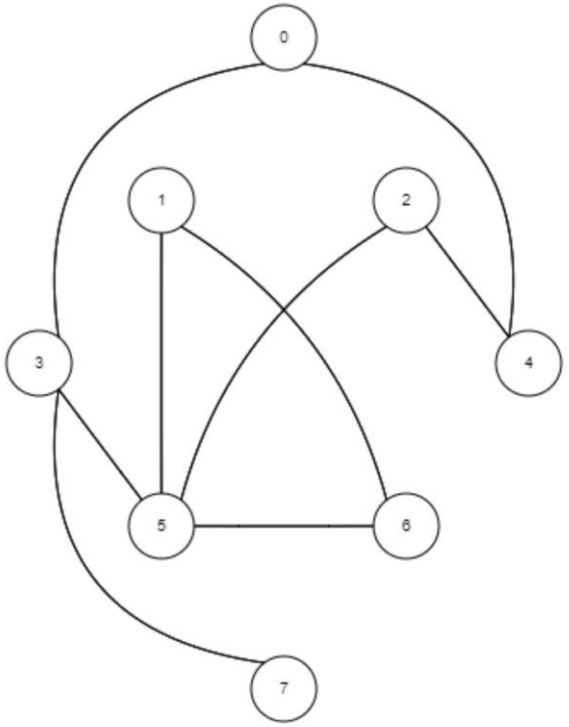


Only unvisited adjacent node is from **D** is **C** now. So we visit **C**, mark it as visited and put it onto the stack.

Output = S, A, D, B, C

https://www.youtube.com/watch?v=iaBEKo5sM7w

Start from node 0



DFS(0)

DFS(3)

DFS(5)

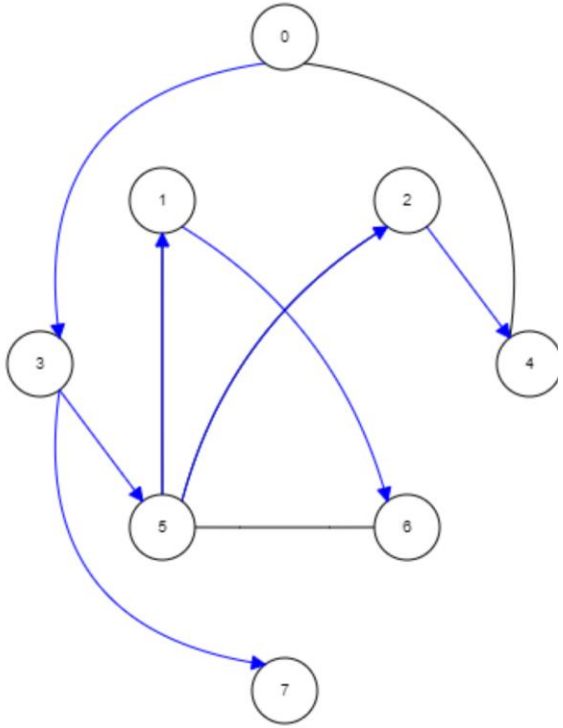
DFS(1)

DFS(6)

DFS(2)

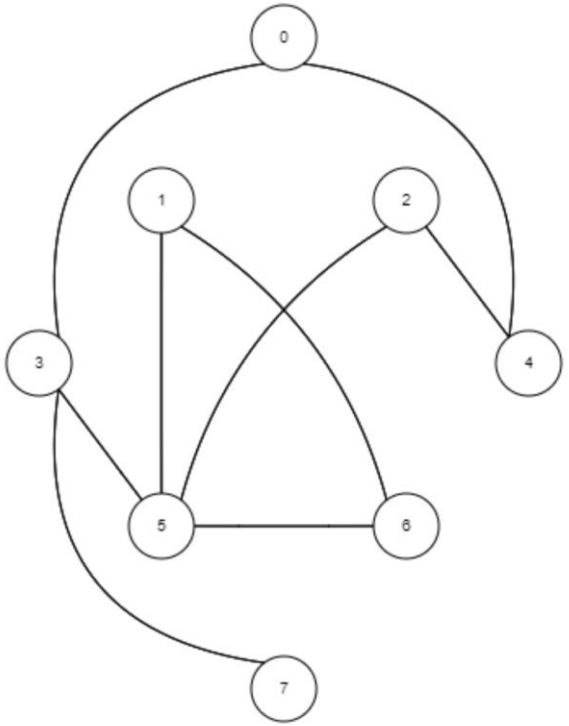
DFS(4)

DFS(7)



https://www.cs.usfca.edu/~galles/visualization/DFS.html

Start from node 2



DFS(2)

DFS(4)

DFS(0)

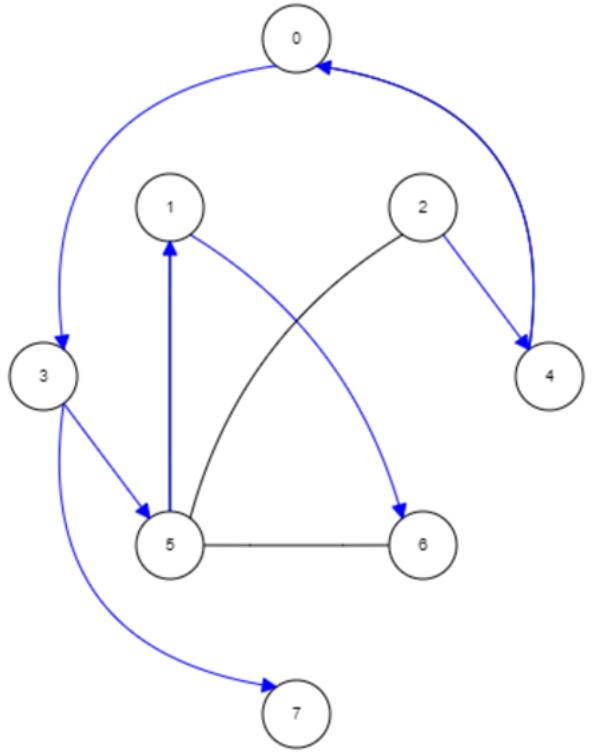
DFS(3)

DFS(5)

DFS(1)

DFS(6)

DFS(7)



https://www.cs.usfca.edu/~galles/visualization/DFS.html

DFS Characteristics:

- **Memory efficient**: Since DFS only needs to keep track of the current path and its adjacent nodes, it uses less memory than BFS in many cases.
- **Suitable for exploring deep graphs**: DFS is well-suited when solutions are likely to be found deep in the graph.

Time Complexity of DFS

For a graph with V vertices and E edges:

- **Time Complexity**: $O(V + E)$, because each vertex and each edge is processed once.
- **Space Complexity**: $O(V)$ for storing the recursion stack or the stack used in the iterative implementation.

Breadth First Search Algorithm

- **Breadth First Search (BFS)** is a graph traversal algorithm used to explore nodes and edges of a graph level by level. Unlike **Depth First Search (DFS)**, which explores as deep as possible before backtracking, BFS explores all the neighbors of a node before moving on to their neighbors.

Key Characteristics of BFS

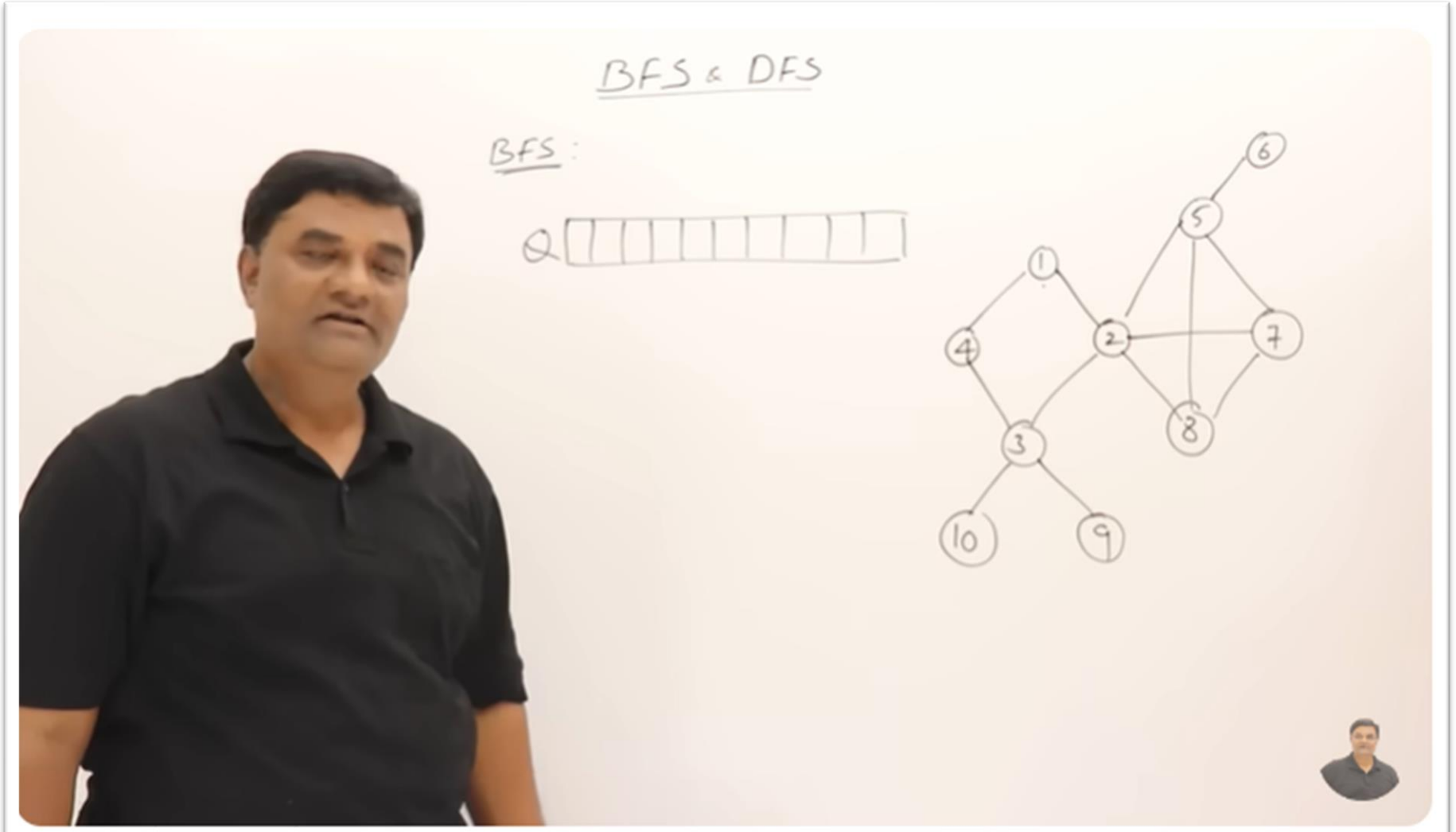
- BFS uses a **queue** data structure to keep track of the vertices to visit.
- It explores all nodes at the present depth (or "level") before moving on to nodes at the next level.
- BFS is commonly used to find the **shortest path** in unweighted graphs and is well-suited for **level-order traversal** in trees.

https://www.youtube.com/watch?v=QRq6p9s8NVg

1. A
2. B, S
3. S
4. C
5. C, G
6. G, D, E, F
7. D, E, F, H
8. E, F, H
9. F, H
10. H
11. H

BFS Characteristics:

- **Guaranteed to find the shortest path**: BFS is guaranteed to find the shortest path in unweighted graphs, which makes it useful in routing algorithms.
- **Requires more memory**: BFS typically requires more memory than DFS because it stores all nodes at the current level before moving on to the next level.



5.1 Graph Traversals - BFS & DFS -Breadth First Search and Depth First Search

https://www.youtube.com/watch?v=pcKY4hjDrxk&t=12s

Thank You!