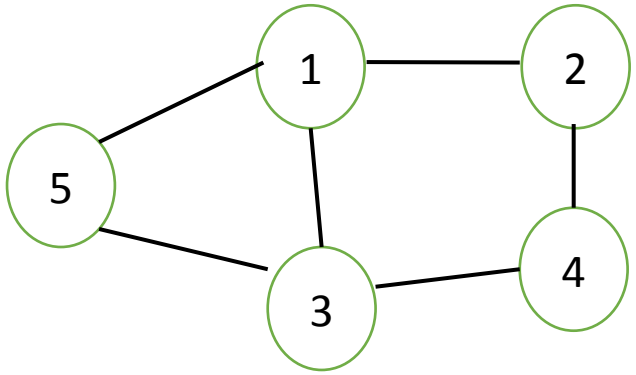
DEPTH and BREADTH FIRST SEARCH

DFS

- traverses a graph in a depth wise

Working

- Visiting a vertex
- Exploration of vertex – visiting its adjacent vertex
  - After exploring a node, a new node is selected from its adjacent nodes
  - adjacent node visited and explored
  - this Rules repeated till all nodes are visited and explored completely
- Stack is used to implement DFS



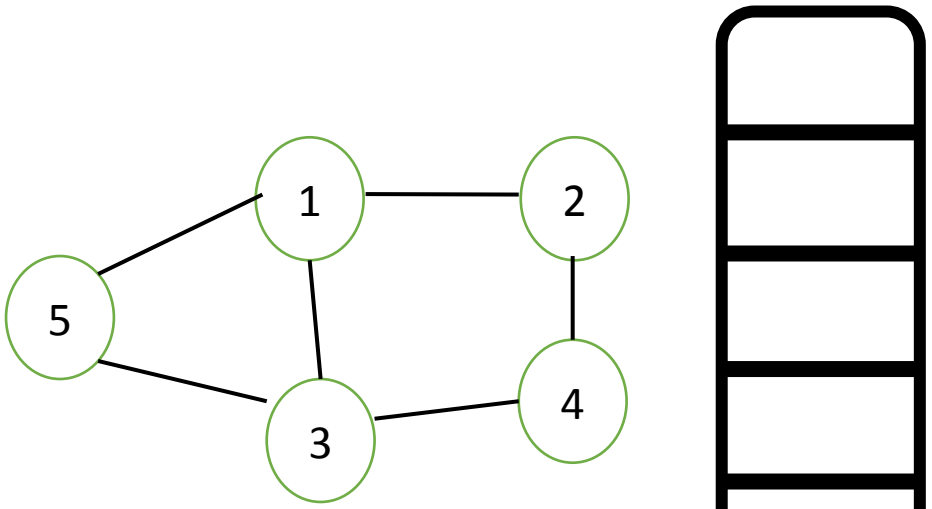
NODES	ADJACENT NODES
1	2,3,5
2	1,4
3	1,4,5
4	2,3
5	1,3

RULES OF DFS

Rule1 − Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack.

Rule2− If no adjacent vertex is found, pop up a vertex from the stack.

Rule3− Repeat Rule1 and 2 until the stack is empty.



COMPLEXITY OF DEPTH FIRST SEARCH

The time complexity of the DFS algorithm is represented in the form of O(V + E), where V is the number of nodes and E is the number of edges.

The space complexity of the algorithm is O(V).

APPLICATION

1. For finding the path
2. To test if the graph is bipartite
3. For finding the strongly connected components of a graph
4. For detecting cycles in a graph

BFS

- traverses a graph in a breadth wise

Working

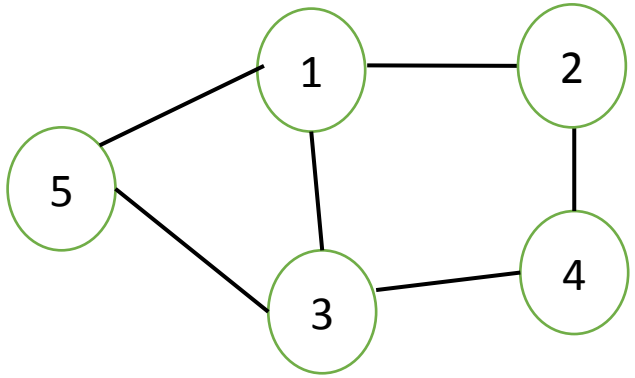
- Visiting a vertex
- Exploration of vertex – visiting its adjacent vertex
  - After exploring a node, a new node is selected from its adjacent nodes
  - adjacent node visited and explored
  - then other adjacent nodes of same vertex is visited and explored completely
  - this Rules repeated till all nodes are visited and explored
- Queue is used to implement DFS
- also called as Level order traversal

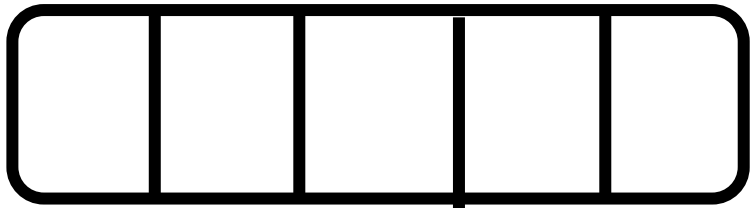
RULES OF BFS

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

Rule 2 − If no adjacent vertex is found, remove the first vertex from the queue.

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





COMPLEXITY OF BREADTH FIRST SEARCH

- The time complexity of the BFS algorithm is represented in the form of O(V + E), where V is the number of nodes and E is the number of edges.
- The space complexity of the algorithm is O(V).

APPLICATIONS

1. To build index by search index
2. For GPS navigation
3. Path finding algorithms
4. In Ford-Fulkerson algorithm to find maximum flow in a network
5. Cycle detection in an undirected graph
6. In minimum spanning tree