**Experiment NO: 4** **Date:**

**Aim:** To implement DFS algorithm

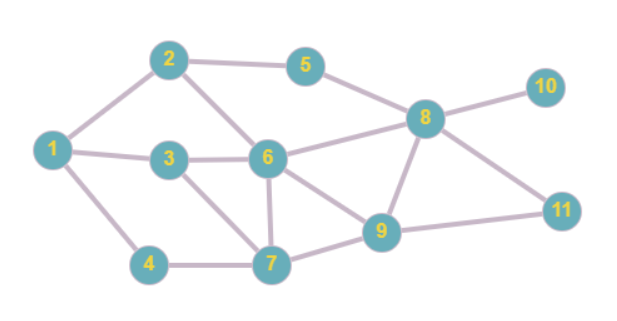
Theory: Depth-First Search (DFS) is an algorithm used for traversing or searching tree or graph data structures. The basic idea of DFS is to explore as far as possible along each branch before backtracking.

Here's a high-level overview of how DFS works:

1. Start at a Node: Choose a starting node in the graph or tree.
2. Explore as Far as Possible: Move to an adjacent, unvisited node and explore as far as possible along that branch before backtracking.
3. Backtrack: If you reach a dead end (a node with no unvisited neighbors), backtrack to the previous node and continue the process.
4. Mark Visited Nodes: To avoid visiting the same node multiple times, mark each node as visited.

DFS can be implemented using recursion or an explicit stack data structure. It's often used in various applications, such as finding connected components in graphs, solving mazes, and topological sorting.

DFS has applications in various fields, including computer science, data analysis, and artificial intelligence.

****

**Code:**

def DFS(vertices, edges, start, end):

assert start in vertices, "Start must be a vertex"

assert end in vertices, "End must be a vertex"

def print\_sol(current, start):

if current == start:

print(f"{current} ", end="")

return

print\_sol(parent[current], start)

print(f"-> {current} ",end="")

def dfs\_helper(current):

visited.add(current)

print(f"{current} visited!")

if current == end:

print("End reached!")

return True

for next\_vertex in edges[current]:

if next\_vertex not in visited:

parent[next\_vertex] = current

if dfs\_helper(next\_vertex):

return True

return False

parent = {}

visited = set()

dfs\_helper(start)

if end in parent:

print\_sol(end, start)

# Example usage

Gvertices = {"A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K" }

Gedges = {

"A" : ["B", "C", "D"],

"B" : ["A", "E", "F", "C"],

"C" : ["A", "B", "F", "G"],

"D": ["A", "G"],

"E": ["B", "H"],

"F": ["B", "C", "G", "H", "I"],

"G": ["C", "F", "D", "I"],

"H": ["E", "F", "I", "K", "J"],

"I": ["F", "G", "H", "K"],

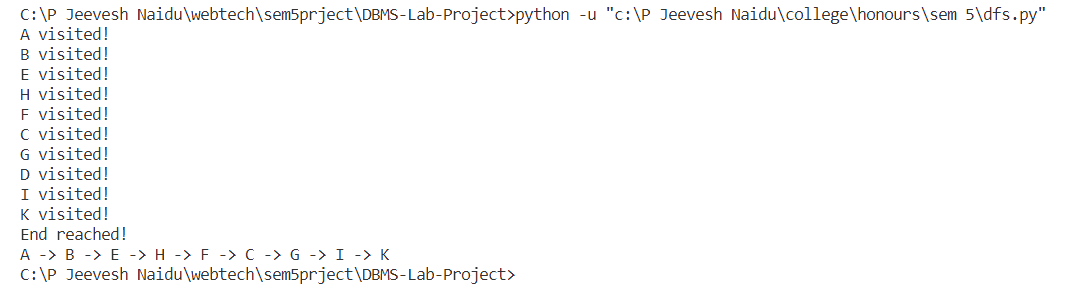
"J":["H"],

"K":["H", "I"]

}

DFS(Gvertices, Gedges, "A", "K")

Output:



**Conclusion:**

Studied the Depth-First Search technique and implemented successfully for Graphs in Python.