**Code:**

#BFS and DFS searching implementation

graph = {

  'A':['B','C'],

  'B':['D','E'],

  'C':['F','G'],

  'D':['H','K'],

  'E':['J'],

  'F':[],

  'G':[],

  'H':[],

  'I':[],

  'J':[],

  'K':[]

}

#function to print the value in table formated way

def print\_table\_row(step, node , fringe , visited):

    print(f"{step:<5} | {node:<10} | {str(fringe):<25} | {str(visited)}")

#DFS

def dfs(graph, start, goal):

    visited = []

    stack = [start]

    step = 0

    #print table header

    print("\n DFS fringe Tracking in Table")

    print("Step  |  Current   |  Fringe(stack)            |   Visited")

    print("------------------------------------------------------------")

    #DFS traversal

    while stack:

        node = stack.pop()

        step+=1

        if node not in visited:

            visited.append(node)

            #display the current seach space

            print\_table\_row(step, node, stack[::-1],visited)

        if node == goal:

            print(f"\n Goal found at {goal} with {step} steps\n")

            return

        #backtracking from left to right

        for child in reversed(graph[node]):

            if child not in visited:

                stack.append(child)

def bfs(graph, start, goal):

    visited = []

    queue = [start]

    step = 0

        #print table header

    print("\n BFS fringe Tracking in Table")

    print("Step  |  Current   |  Fringe(stack)            |   Visited")

    print("------------------------------------------------------------")

    #BFS Traversal

    while queue:

        node =  queue.pop(0)  #Take the front node

        step += 1

        if node not in visited:

            visited.append(node)

           #display current state

            print\_table\_row(step, node, queue, visited)

        if node == goal:

            print(f"\n Goal found at {goal} with {step} steps")

            break

        #enqueue all children not already visited or queue

        for child in graph[node]:

            if child not in visited and child not in queue:

                queue.append(child)

#Step the start point and goal node

start\_node = 'A'

goal\_node = 'K'

#Run dfs

dfs(graph, start\_node, goal\_node)

#Run bfs

bfs(graph, start\_node, goal\_node)

**Output:**

