1. **Hill Climbing:**

**Input:**

graph={'A':[('B',3),('C',4),('K',10)], 'B':[('D',1),('E',0)],

    'C':[('F',2),('G',3)], 'D':[], 'E':[], 'F':[], 'G':[], 'K':[] }

start=input("Enter the Start Node:")

goal=input("Enter Goal Node:")

def hill\_climb(start,goal,graph):

    open=[]

    close=[]

    open.append(start)

    while open:

        node=open.pop(0)

        if node==goal:

            print("Goal found")

            close.append(node)

            print("Path:",close)

            return True

        close.append(node)

        neighbour=graph[node]

        for i in neighbour:

            if i[0] not in open:

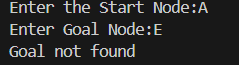
                open.append(i[0])

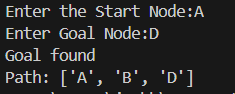
                break

    print("Goal not found")

hill\_climb(start,goal,graph)

**Output:**





1. **Steepest Hill Climbing:**

**Input:**

graph = {

    'A': [('B', 3), ('C', 4), ('K', 10)],

    'B': [('D', 1), ('E', 0)],

    'C': [('F', 2), ('G', 3)],

    'D': [],

    'E': [],

    'F': [],

    'G': [],

    'K': []

}

start = input("Enter the Start Node: ")

goal = input("Enter Goal Node: ")

def hill\_climb(start, goal, graph):

    open = []

    close = []

    open.append(start)

    while open:

        node = open.pop(0)

        if node == goal:

            print("Goal found")

            close.append(node)

            print("Path:",close)

            return True

        close.append(node)

        neighbours = graph[node]

        neighbours = sorted(neighbours, key=lambda x: x[1])  # sort by heuristic

        for i in neighbours:

            if i[0] not in open and i[0] not in close:

                open.append(i[0])

                break  # only take the best option

    print("Goal not found")

    return False

hill\_climb(start, goal, graph)

**Output:**

