* **BFS**

**Code:-**

from collections import deque

def bfs(graph, start, destination):

queue = deque([(start, [start])])

visited = set()

while queue:

current, path = queue.popleft()

if current == destination:

print("Path found:", ' -> '.join(path))

return

visited.add(current)

for neighbor in graph[current]:

if neighbor not in visited:

queue.append((neighbor, path + [neighbor]))

visited.add(neighbor)

india\_city\_graph = {

'Delhi': ['Mumbai', 'Jaipur', 'Lucknow'],

'Mumbai': ['Delhi', 'Chennai', 'Bangalore'],

'Chennai': ['Mumbai', 'Bangalore'],

'Bangalore': ['Mumbai', 'Chennai', 'Hyderabad'],

'Hyderabad': ['Bangalore', 'Chennai'],

'Jaipur': ['Delhi', 'Lucknow'],

'Lucknow': ['Delhi', 'Jaipur']

}

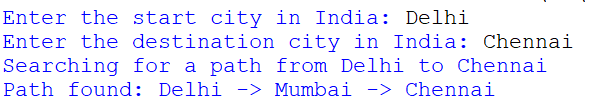
start\_city = input("Enter the start city in India: ")

destination\_city = input("Enter the destination city in India: ")

print(f"Searching for a path from {start\_city} to {destination\_city}")

bfs(india\_city\_graph, start\_city, destination\_city)

**Output:-**



* **DFS**

**Code:-**

def dfs(graph, current, destination, visited=None, path=None):

if visited is None:

visited = set()

if path is None:

path = []

path = path + [current]

visited.add(current)

if current == destination:

print("Path found:", ' -> '.join(path))

return

for neighbor in graph[current]:

if neighbor not in visited:

dfs(graph, neighbor, destination, visited, path)

india\_city\_graph = {

'Delhi': ['Mumbai', 'Jaipur', 'Lucknow'],

'Mumbai': ['Delhi', 'Chennai', 'Bangalore'],

'Chennai': ['Mumbai', 'Bangalore'],

'Bangalore': ['Mumbai', 'Chennai', 'Hyderabad'],

'Hyderabad': ['Bangalore', 'Chennai'],

'Jaipur': ['Delhi', 'Lucknow'],

'Lucknow': ['Delhi', 'Jaipur']

}

start\_city = input("Enter the start city in India: ")

destination\_city = input("Enter the destination city in India: ")

print(f"Searching for a path from {start\_city} to {destination\_city}")

dfs(india\_city\_graph, start\_city, destination\_city)

**Output:-**

