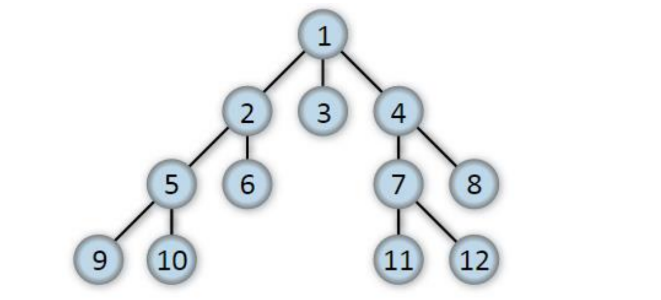
**Lab 3**

1. **Construct the given tree, write a breadth first search function def bfs(graph, start\_node, goal\_node) which should return goal node if the data is found in the tree else return None.**

****

**Objective**:The objective of this code is to implement the Breadth-First Search (BFS) algorithm to traverse a graph and find the shortest path from a specified start node to a goal node. It uses a queue to explore nodes level by level, ensuring each node is visited only once, and prints the path traversed.

**Code:**def bfs(graph, start\_node, goal\_node):

visited = set()

queue = [[start\_node]]

if start\_node == goal\_node:

return [start\_node]

while queue:

path = queue.pop(0)

node = path[-1]

if node not in visited:

visited.add(node)

for neighbour in graph[node]:

new\_path = list(path)

new\_path.append(neighbour)

queue.append(new\_path)

if neighbour == goal\_node:

return new\_path

return None

graph = {

1: [2, 3, 4],

2: [5, 6],

3: [],

4: [7, 8],

5: [9, 10],

6: [],

7: [11, 12],

8: [],

9: [],

10: [],

11: [],

12: []

}

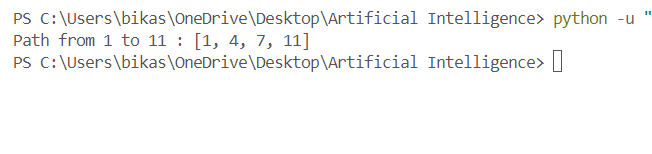
start\_node = 1

goal\_node = 11

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

print("Path from", start\_node, "to", goal\_node, ":", path)

**Output:**



**2.Construct the tree as in the previous question, write a depth first search**

**function def dfs(graph, start\_node, goal\_node) which should return goal**

**node if the data is found in the tree else return None.**

**Objective**:Implement a depth-first search function that finds and returns the path from the start node to the goal node in a given tree, represented as a graph.

**Code:**def depth\_first\_search(graph, start\_node, goal\_node):

visited = set()

stack = [[start\_node]]

if start\_node == goal\_node:

return [start\_node]

while stack:

path = stack.pop()

node = path[-1]

if node not in visited:

visited.add(node)

for neighbour in graph[node]:

new\_path = list(path)

new\_path.append(neighbour)

stack.append(new\_path)

if neighbour == goal\_node:

return new\_path

return None

# Define the graph and nodes outside the function

graph = {

1: [2, 3, 4],

2: [5, 6],

3: [],

4: [7, 8],

5: [9, 10],

6: [],

7: [11, 12],

8: [],

9: [],

10: [],

11: [],

12: []

}

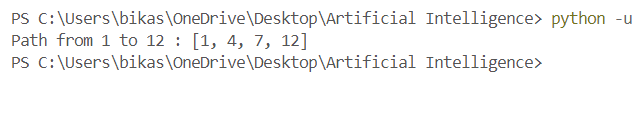
start\_node = 1

goal\_node = 12

path = depth\_first\_search(graph, start\_node, goal\_node)

print("Path from", start\_node, "to", goal\_node, ":", path)

**Output:**

****