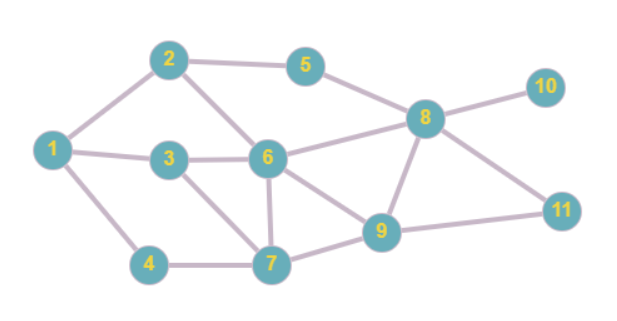
**Experiment NO: 3** **Date:**

**Aim:** To implement BFS algorithm

Theory: Breadth-First Search (BFS) is a fundamental graph traversal algorithm used to explore or traverse a graph or tree data structure. Unlike Depth-First Search (DFS), which explores as far as possible along a branch before backtracking, BFS explores the graph level by level.

The basic process of the Breadth-First Search is as follows:

1. **Starting Point:** The algorithm begins at a selected node as the starting point.
2. **Exploring Adjacent Nodes:** It visits all the neighboring nodes of the starting node at the current level before moving to the nodes at the next level.
3. **Queue:** BFS uses a queue data structure to keep track of the nodes to visit. Nodes are added to the queue as they are discovered and processed in the order they were added (FIFO order).
4. **Level-wise Exploration:** It explores the graph level by level. In each level, it visits all nodes adjacent to the current nodes before moving to the next level.
5. **Marking Visited Nodes:** Typically, nodes are marked as visited to prevent revisiting nodes and infinite loops in graphs with cycles.

****

**Code:**

from queue import Queue

def BFS(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="")

return

parent = {}

q = Queue()

q.put(start)

while not q.empty():

current = q.get()

if current in parent.values():

continue

print(f"{current} visited!")

if current == end:

print("End reached!")

break

for next in edges[current]:

if next not in parent.keys():

parent[next]=current

q.put(next)

if end in parent:

print\_sol(end, start)

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"],

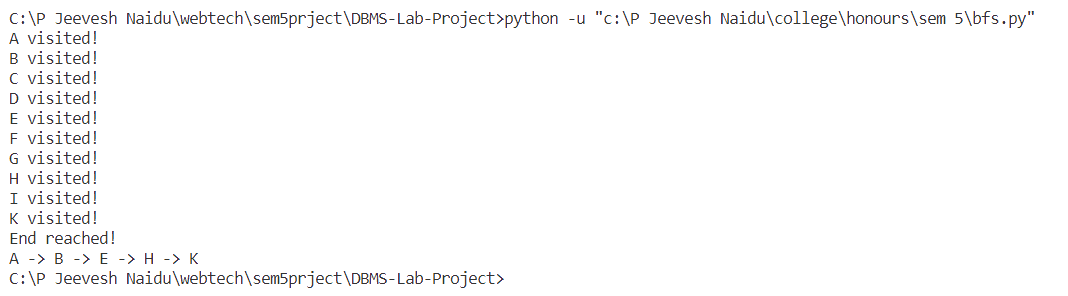
"J":["J"],

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

}

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

Output:



**Conclusion:**

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