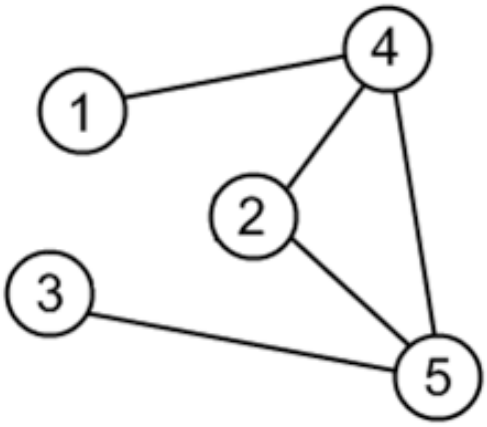
Task 1(A). Breadth First Search for Graph Traversal

Print the Breadth First Search (BFS) Traversal on following graph. The starting node of the graph is 1 in the give graph.



Input Format:

Index of nodes and edges of problem graph.

Output Format:

Sequence of visited nodes of problem graph

Sample Code:

```
from collections import defaultdict

# This class represents a directed graph
# using adjacency list representation
class Graph:

    # Constructor
    def __init__(self):

        # default dictionary to store graph
        self.graph = defaultdict(list)

    # function to add an edge to graph
    def addEdge(self,u,v):
        self.graph[u].append(v)

    # Function to print a BFS of graph
    def BFS(self, s):

        # Mark all the vertices as not visited
        visited = [False] * (max(self.graph) + 1)

        # Create a queue for BFS
        queue = []

        # Mark the source node as
        # visited and enqueue it
        queue.append(s)
        visited[s] = True
```

```
            while queue:

                    # Dequeue a vertex from
                    # queue and print it
                    s = queue.pop(0)
                    print (s, end = " ")

                    # Get all adjacent vertices of the
                    # dequeued vertex s. If a adjacent
                    # has not been visited, then mark it
                    # visited and enqueue it
                    for i in self.graph[s]:
                            if visited[i] == False:
                                    queue.append(i)
                                    visited[i] = True

# Driver code

# Create a graph given in
# the above diagram
g = Graph()
g.addEdge(0, 1)
g.addEdge(0, 2)
g.addEdge(1, 2)
g.addEdge(2, 0)
g.addEdge(2, 3)
g.addEdge(3, 3)

print ("Following is Breadth First Traversal"
```

```
" (starting from vertex 1)")

g.BFS(1)
```