In [2]:
```python
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)
        self.visited = []

    # 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):
        # Create a queue for BFS
        queue = []

        # Add the source node in visited and enqueue it
        queue.append(s)
        self.visited.append(s)

        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 an adjacent has not been visited, then add it in visited and en
            for i in self.graph[s]:
                if i not in self.visited:
                    queue.append(i)
                    self.visited.append(i)

# 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 2)")
g.BFS(2)
```

```
Following is Breadth First Traversal (starting from vertex 2)
2 0 3 1
```

In [ ]:
```python

```