Breadth First Search

In [2]:
```
#+1
from collections import defaultdict

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):
        visited = [False] * (max(self.graph) + 1)
        queue = []
        queue.append(s)
        visited[s] = True
        while queue:
            # Dequeue a vertex from queue and print it
            s = queue.pop(0)
            print (s, end = " ")
            for i in self.graph[s]:
                if visited[i] == False:
                    queue.append(i)
                    visited[i] = True

# Create a graph tree
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 0")
g.BFS(2)
```

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

Depth First Search

In [3]:
```
from collections import defaultdict

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)

    def DFSUtil(self, v, visited):
        visited.add(v)
        print(v, end=' ')
        for neighbour in self.graph[v]:
            if neighbour not in visited:
                self.DFSUtil(neighbour, visited)

    # The function to do DFS traversal.
    def DFS(self, v):
        # Create a set to store visited vertices
        visited = set()
        # Call the recursive helper function to print DFS traversal
        self.DFSUtil(v, visited)

#Create a graph
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 DFS starting from vertex 0")
g.DFS(1)
```

```
Following is DFS starting from vertex 0
1 2 0 3
```

In [ ]: