

In [2]:

```

1 class ArrayStack:
2     def __init__(self,size):
3         self.size=size
4         self.data=[0 for i in range(size)]
5         self.top=0
6     def isEmpty(self):
7         if self.top==0:
8             return True
9         else:
10            return False
11    def Push(self,value):
12        if self.top==self.size:
13            print("Stack Overflow !")
14        else:
15            self.data[self.top]=value
16            self.top+=1
17    def Pop(self):
18        if self.isEmpty():
19            print("Stack Underflow !")
20        else:
21            self.top-=1
22            x=self.data[self.top]
23            self.data[self.top]=0
24            return x
25    class Node:
26        def __init__(self,value):
27            self.value=value
28            self.next=None
29    class ListQueue:
30        def __init__(self,size):
31            self.size=size
32            self.head=None
33            self.tail=None
34        def enqueue(self,value):
35            if self.Count()==self.size:
36                pass
37            else:
38                newnode=Node(value)
39                x=self.tail
40                if self.head == None:
41                    self.head=newnode
42                    self.tail=newnode
43                else:
44                    self.tail.next=newnode
45                    self.tail=newnode
46        def dequeue(self):
47            if self.head==None:
48                pass
49            else:
50                x=self.head
51                a=x.value
52                x=x.next
53                self.head=x
54                return a
55        def isEmpty(self):
56            if self.Count==0:
57                return True
58            else:
59                return False
60        def Count(self):
61            x=self.head
62            count=0
63            while x:
64                count+=1
65                x=x.next
66            return count
67    class Graph:
68        def __init__(self,vertex):
69            self.vertex=vertex
70            self.adj=[[0 for i in range(vertex)]for j in range(vertex)]
71            self.visited=[]
72        def AddEdge(self,src,dest):
73            if src==dest:
74                print("Source and destination are same.")
75            else:
76                self.adj[src][dest]=1
77                self.adj[dest][src]=1
78        def PrintMatrix(self):
79            for i in range(self.vertex):
80                for j in range(self.vertex):
81                    print(self.adj[i][j],end=" ")
82                print("\r")
83        def GetNeighbours(self,vertex):
84            lst=[]
85            for i in range(self.vertex):

```

```

86         if self.adj[vertex][i] == 1:
87             lst.append(i)
88             #print("Position :",i,"to",vertex,"and",vertex,"to",i)
89         return lst
90     def DFS(self,source):
91         s=ArrayStack(self.vertex)
92         s.Push(source)
93         self.visited.append(source)
94         while not s.isEmpty():
95             x=s.Pop()
96             print("Visited {}".format(x))
97             neighbours=self.GetNeighbours(x)
98             for i in neighbours:
99                 if i not in self.visited:
100                     s.Push(i)
101                     self.visited.append(i)
102     def BFS(self,source):
103         self.visited=[]
104         s=ListQueue(self.vertex)
105         s.enqueue(source)
106         self.visited.append(source)
107         while not s.isEmpty():
108             x=s.dequeue()
109             if x!=None:
110                 print("Visited {}".format(x))
111                 neighbours=self.GetNeighbours(x)
112                 for i in neighbours:
113                     if i not in self.visited:
114                         s.enqueue(i)
115                         self.visited.append(i)
116             else:
117                 break
118 a=Graph(4)
119 a.AddEdge(1,2)
120 a.AddEdge(1,0)
121 a.AddEdge(0,2)
122 a.AddEdge(2,3)
123 a.PrintMatrix()
124 vertex=2
125 for i in a.GetNeighbours(vertex):
126     print("Position :",i,"to",vertex,"and",vertex,"to",i)
127 print("-----DFS-----")
128 a.DFS(2)
129 print("-----BFS-----")
130 a.BFS(2)
0 1 1 0
1 0 1 0
1 1 0 1
0 0 1 0
Position : 0 to 2 and 2 to 0
Position : 1 to 2 and 2 to 1
Position : 3 to 2 and 2 to 3
-----DFS-----
Visited 2
Visited 3
Visited 1
Visited 0
-----BFS-----
Visited 2
Visited 0
Visited 1
Visited 3

```