image.py

```
class Stack(object):
    def __init__(self):
        self.items = []
    def is_empty(self):
        return self.items == []
    def peek(self):
        return self.items[len(self.items) - 1]
    def size(self):
        return len(self.items)
    def push(self, item):
        self.items.append(item)
    def pop(self):
        return self.items.pop()
class Queue(object):
    def __init__(self):
        self.items = []
    def is_empty(self):
        return self.items == []
    def peek(self):
        return self.items[0]
    def size(self):
        return len(self.items)
    def add(self, item):
        self.items.append(item)
    def pop(self):
        return self.items.pop(0)
class imageNode(object):
    def __init__(self,data,next = None):
        self.data = data
        self.next = next
        self.marked = 0
def dfs(image, i, j):
    m = Stack()
    m.push(image[i])
    temp = image[i].next
    while (m.is_empty() != True):
        m.peek().marked = 1
        if (temp != None):
            if(temp.marked == 0):
                m.push(image[temp.data])
                m.peek().marked = 1
                if (m.peek().data == j):
                    return 1
                else:
                    if (m.peek().next == None):
                        m.pop()
                    else:
                        return dfs(image, m.peek().data, j)
```

```
            temp = temp.next
        else: m.pop()
    return 0
def bfs(image,i,j):
    m = Queue()
    m.add(image[i])
    m.peek().marked = 1
    temp = image[i].next
    while (temp != None):
        if (temp.marked == 0):
            m.add(image[temp.data])
            m.peek().marked = 1
            if (temp.data == j): return 1
            temp = temp.next
        else:
            temp = temp.next
    m.pop()
    while(m.is_empty() != True):
        if (m.peek().next == None):
            m.pop()
        else:
            return bfs(image, m.peek().data, j)
    return 0
```

test.py

```
from image import imageNode
import image
V = []
for i in range(4):
    a = imageNode(i + 1, None)
    V.append(a)
V.insert(0, imageNode(0, None))
temp2 = imageNode(3, None)
temp1 = imageNode(2, temp2)
V[1].next = temp1
temp1 = imageNode(4, None)
V[3].next = temp1
temp1 = imageNode(1, None)
V[4].next = temp1
print(image.dfs(V,1,4))
print(image.bfs(V,2,4))
```

image.py中定义了image类

dfs和bfs分别为深度优先搜索和宽度优先搜索

测试用的图为课本上的有向图G1