实验五 图的深度和广度遍历

姓名：王天一

学号：320200931301

2022年6月12日

1 实验要求

要求图用邻接表方式存储，编写能够通过键盘输入一个无向图并存储，实现图的深度优先遍历及广度优先遍历的程序。

2 测试用例

共有5个节点，14条边

```
[['1', '3', '2', '5', '4'],
 ['2', '3', '1', '5', '3'],
 ['3', '2', '2', '4'],
 ['4', '3', '1', '5', '3'],
 ['5', '3', '4', '1', '2']]
```

输出结果：

```
1 2 5 4 3
1 2 5 4 3
```

由于巧合，我们发现此图的深度和广度遍历是一样的。

3 代码实现

3.1 图的构建

对于图的构建，我直接建立了一个类。

```
class graph():
    def __init__(self,node,edge):
        self.node = node
        self.edge = edge
        self.linktable = []
        for _ in range (node):
            line = input("Please enter the link table: ").split(",")
            self.linktable.append(line)
        print(self.linktable)
        if self.checkLinktable():
            print("The link table is correct!")
        else:
            print("The link table is not correct! Please check!")
```

其初始化需要给两个参数：节点个数和边的个数。

在初始化的过程中，我们需要输入邻接表。

3.2 对于输入的检测

为了防止输入错误，我构建了一个检测函数，用来检测输入是否合法。

```
def checkLinktable(self):
        if len(self.linktable)!=self.node:
            print("The number of nodes is not correct!")
            return False
        count = 0
        for i in self.linktable:
            count+=int(i[1])
        if count!=self.edge:
            print("The number of edges is not correct!")
            return False
        return True

```

在这个函数中，我会检测节点的个数和边的数量。如果有一个输入和邻接表不匹配，则会输出提示信息。

3.3 深度遍历

```
def dfs(self,visited = None,i=None):
        if visited == None:
            visited=[False]*len(self.linktable)
        visited[i-1]=True
        for j in range(2,len(self.linktable[i-1])):
            if visited[int(self.linktable[i-1][j])-1]==False:
                visited[int(self.linktable[i-1][j])-1]=True
                self.dfs(visited,int(self.linktable[i-1][j]))
```

对于图的深度遍历，我使用了递归实现。

3.4 广度遍历

```
    def bfs(self,i):
        myqueue=[]
        myqueue.append(i)
        visited=[False]*len(self.linktable)
        visited[i-1]=True
        while len(myqueue)>0:
            i=myqueue.pop(0)
            print(i,end=" ")
            for j in range(2,len(self.linktable[i-1])):
                if visited[int(self.linktable[i-1][j])-1]==False:
                    myqueue.append(int(self.linktable[i-1][j]))
                    #print(int(linktable[i-1][j]))
                    visited[int(self.linktable[i-1][j])-1]=True
```

对于图的广度遍历，我主要使用了模拟队列的方式实现。