深度优先遍历

深度优先搜索

        深度优先搜索属于图算法的一种，是一个针对图和树的遍历算法，英文缩写为DFS即Depth First Search。深度优先搜索是图论中的经典算法，利用深度优先搜索算法可以产生目标图的相应拓扑排序表，利用拓扑排序表可以方便的解决很多相关的图论问题，如最大路径问题等等。一般用堆数据结构来辅助实现DFS算法。其过程简要来说是对每一个可能的分支路径深入到不能再深入为止，而且每个节点只能访问一次。

实验代码：

**class** Graph(object):  
 **def** \_\_init\_\_(self, maps):  
 self.maps = maps  
 self.nodenum = self.get\_nodenum()  
 self.edgenum = self.get\_edgenum()  
  
 **def** get\_nodenum(self):  
 **return** len(self.maps)  
  
 **def** get\_edgenum(self):  
 count = 0  
 **for** i **in** range(self.nodenum):  
 **for** j **in** range(i):  
 **if** self.maps[i][j] > 0:  
 count += 1  
 **return** count  
  
 **def** insert\_node(self):  
 **for** i **in** range(len(self.maps)):  
 self.maps[i].append(-1)  
 self.maps.append([-1]\*(self.nodenum) + [0])  
 self.nodenum += 1  
  
 **def** insert\_edge(self, x, y, weight):  
 **if** x < 0 **or** x >= self.nodenum **or** y < 0 **or** y > self.nodenum **or** weight <= 0 **or** x == y:  
 **return  
 else**:  
 self.maps[x][y] = self.maps[y][x] = weight  
 self.edgenum += 1  
 **def** depth\_first\_search(self):  
 res = []  
 visited = [**False**]\*self.nodenum  
 **def** dfs(i):  
 res.append(i)  
 visited[i] = **True  
 for** j **in** range(self.nodenum):  
 **if** self.maps[i][j] > 0 **and** visited[j] == **False**:  
 dfs(j)  
 **if** self.nodenum > 0:  
 dfs(0)  
 **for** i **in** range(self.nodenum):  
 **if** visited[i] == **False**:  
 dfs(i)  
 **return** res  
maps = [[0,3,9,-1], [3,0,-1,5], [9,-1,0,-1], [-1,5,-1,0]]  
graph = Graph(maps)  
print(**'邻接矩阵为\n%s'**%graph.maps)  
print(**'节点数据为%d，边数为%d\n'**%(graph.nodenum, graph.edgenum))  
graph.insert\_node()  
print(**'-------插入一个节点--------'**)  
print(**'邻接矩阵为%s'**%graph.maps)  
print(**'节点数据为%d，边数为%d\n'**%(graph.nodenum, graph.edgenum))  
graph.insert\_edge(0, 4, 7)  
print(**'-------插入一个边--------'**)  
print(**'邻接矩阵为%s'**%graph.maps)  
print(**'节点数据为%d，边数为%d\n'**%(graph.nodenum, graph.edgenum))  
print(**'-------深度优先遍历--------'**)  
print(graph.depth\_first\_search())

