**实验一**

实验题目：DFS（深度优先搜索算法）

题目讲解：深度优先搜索算法是一种用于遍历或搜索树或图的算法。沿着树的深度遍历树的节点，尽可能深的搜索树的分支。当节点v的所在边都已被探寻过或在搜索时节点不满足条件，搜索将回到发现节点v的那条边的起始节点。整个过程反复进行直到所有节点都被访问为止。

深度搜索：时间复杂度大，空间复杂度小。

**时间复杂度（time complexity）：**

**空间复杂度（space complexity）：**

算法过程

   # 深度优先算法的实现  
    def DFS(self, root):  
        # 首先判断根节点是否为空节点  
        if root != None:  
            search\_queue = deque()  
            search\_queue.append(root)  
   
            visited = []  
        else:  
            print(&apos;root is None&apos;)  
            return -1  
   
        while search\_queue:  
            person = search\_queue.popleft()  
            self.order.append(person)  
   
            if (not person in visited) and (person in self.neighbor.keys()):  
                tmp = self.neighbor[person]  
                tmp.reverse()  
   
                for index in tmp:  
                    search\_queue.appendleft(index)  
   
                visited.append(person)  
   
    def clear(self):  
        self.order = []  
   
    def node\_print(self):  
        for index in self.order:  
            print(index, end=&apos;  &apos;)  
   
   
if \_\_name\_\_ == &apos;\_\_main\_\_&apos;:  
    # 创建一个二叉树图  
    g = Graph()  
    g.add\_node((&apos;A&apos;, [&apos;B&apos;, &apos;C&apos;]))  
    g.add\_node((&apos;B&apos;, [&apos;D&apos;, &apos;E&apos;]))  
    g.add\_node((&apos;C&apos;, [&apos;F&apos;]))  
  
    # 进行深度优先搜索  
    print(&apos;\n\n深度优先搜索:&apos;)  
    print(&apos;  &apos;, end=&apos;  &apos;)  
    g.DFS(&apos;A&apos;)  
    g.node\_print()  
    print()