Leetcode(python) 深度优先搜索 简单题

690员工的重要性

题目：

给定一个保存员工信息的数据结构，它包含了员工唯一的id，重要度 和 直系下属的id。

比如，员工1是员工2的领导，员工2是员工3的领导。他们相应的重要度为15, 10, 5。那么员工1的数据结构是[1, 15, [2]]，员工2的数据结构是[2, 10, [3]]，员工3的数据结构是[3, 5, []]。注意虽然员工3也是员工1的一个下属，但是由于并不是直系下属，因此没有体现在员工1的数据结构中。

现在输入一个公司的所有员工信息，以及单个员工id，返回这个员工和他所有下属的重要度之和。

示例 1:

输入: [[1, 5, [2, 3]], [2, 3, []], [3, 3, []]], 1

输出: 11

解释:

员工1自身的重要度是5，他有两个直系下属2和3，而且2和3的重要度均为3。因此员工1的总重要度是 5 + 3 + 3 = 11。

注意:

一个员工最多有一个直系领导，但是可以有多个直系下属

员工数量不超过2000。

思路：

代码：

1. """
2. # Employee info
3. class Employee(object):
4. def \_\_init\_\_(self, id, importance, subordinates):
5. # It's the unique id of each node.
6. # unique id of this employee
7. self.id = id
8. # the importance value of this employee
9. self.importance = importance
10. # the id of direct subordinates
11. self.subordinates = subordinates
12. """
13. **class** Solution(object):
14. **def** getImportance(self, employees, id):
15. """
16. :type employees: Employee
17. :type id: int
18. :rtype: int
19. """
20. emap={employee.id:employee **for** employee **in** employees}
21. res=emap[id].importance
22. **for** sub **in** emap[id].subordinates:
23. res+= self.getImportance(employees,sub)
24. **return** res

733图像渲染

题目：

有一幅以二维整数数组表示的图画，每一个整数表示该图画的像素值大小，数值在 0 到 65535 之间。

给你一个坐标 (sr, sc) 表示图像渲染开始的像素值（行 ，列）和一个新的颜色值 newColor，让你重新上色这幅图像。

为了完成上色工作，从初始坐标开始，记录初始坐标的上下左右四个方向上像素值与初始坐标相同的相连像素点，接着再记录这四个方向上符合条件的像素点与他们对应四个方向上像素值与初始坐标相同的相连像素点，……，重复该过程。将所有有记录的像素点的颜色值改为新的颜色值。

最后返回经过上色渲染后的图像。

示例 1:

输入:

image = [[1,1,1],[1,1,0],[1,0,1]]

sr = 1, sc = 1, newColor = 2

输出: [[2,2,2],[2,2,0],[2,0,1]]

解析:

在图像的正中间，(坐标(sr,sc)=(1,1)),

在路径上所有符合条件的像素点的颜色都被更改成2。

注意，右下角的像素没有更改为2，

因为它不是在上下左右四个方向上与初始点相连的像素点。

注意:

image 和 image[0] 的长度在范围 [1, 50] 内。

给出的初始点将满足 0 <= sr < image.length 和 0 <= sc < image[0].length。

image[i][j] 和 newColor 表示的颜色值在范围 [0, 65535]内。

思路：

代码：

1. **class** Solution(object):
2. **def** floodFill(self, image, sr, sc, newColor):
3. """
4. :type image: List[List[int]]
5. :type sr: int
6. :type sc: int
7. :type newColor: int
8. :rtype: List[List[int]]
9. """
10. curcolor=image[sr][sc]
11. **if** curcolor==newColor:
12. **return** image
13. l=len(image)
14. r=len(image[0])
16. **def** dfs(left,right):
17. **if** image[left][right]==curcolor:
18. image[left][right]=newColor
19. **if** left>=1: dfs(left-1,right)
20. **if** left<l-1:dfs(left+1,right)
21. **if** right>=1:dfs(left,right-1)
22. **if** right<r-1:dfs(left,right+1)
23. dfs(sr,sc)
24. **return** image