690 Employee Importance

2018年5月3日 15:56

You are given a data structure of employee information, which includes the employee's **unique id**, his **importance value** and his **direct** subordinates' id.

For example, employee 1 is the leader of employee 2, and employee 2 is the leader of employee 3. They have importance value 15, 10 and 5, respectively. Then employee 1 has a data structure like [1, 15, [2]], and employee 2 has [2, 10, [3]], and employee 3 has [3, 5, []]. Note that although employee 3 is also a subordinate of employee 1, the relationship is **not direct**.

Now given the employee information of a company, and an employee id, you need to return the total importance value of this employee and all his subordinates.

Example 1:

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

Output: 11

Explanation:

Employee 1 has importance value 5, and he has two direct subordinates: employee 2 and employee 3. They both have importance value 3. So the total importance value of employee 1 is 5 + 3 + 3 = 11.

Note:

1. One employee has at most one **direct** leader and may have several subordinates.
2. The maximum number of employees won't exceed 2000.

来自 <https://leetcode.com/problems/employee-importance/description/>

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:

1. 一个员工最多有一个**直系**领导，但是可以有多个**直系**下属
2. 员工数量不超过2000。

Solution for Python3:

```python
"""
# Employee info
class Employee:
    def __init__(self, id, importance, subordinates):
        # It's the unique id of each node.
        # unique id of this employee
        self.id = id
        # the importance value of this employee
        self.importance = importance
        # the id of direct subordinates
        self.subordinates = subordinates
"""
class Solution1:
    def getImportance(self, employees, id):
        """
        :type employees: Employee
```

```
        :type id: int
        :rtype: int
        """
        emap = {e.id:e for e in employees}
        def dfs(eid):
            employee = emap[eid]
            return (employee.importance + sum(dfs(eid) for eid in
employee.subordinates))
        return dfs(id)

class Solution2:
    def getImportance(self, employees, id):
        """
        :type employees: Employee
        :type id: int
        :rtype: int
        """
        emap = {e.id:e for e in employees}
        deq = collections.deque([emap[id]])
        ans = 0
        while deq:
            employee = deq.popleft()
            ans += employee.importance
            deq.extend(emap[id] for id in employee.subordinates)
        return ans
```

Solution for C++:

```
/*
// Employee info
class Employee {
public:
    // It's the unique ID of each node.
    // unique id of this employee
    int id;
    // the importance value of this employee
    int importance;
    // the id of direct subordinates
    vector<int> subordinates;
};
*/
class Solution1 {
    unordered_map<int, Employee*> emap;
public:
    int getImportance(vector<Employee*> employees, int id) {
        for (Employee* e : employees)
            emap[e->id] = e;
        return dfs(id);
    }
    int dfs(int eid) {
        Employee* employee = emap[eid];
        int ans = employee->importance;
        for (int subId : employee->subordinates)
            ans += dfs(subId);
```

```
27          return ans;
28      }
29  };
30
31  class Solution2 {
32  public:
33      int getImportance(vector<Employee*> employees, int id) {
34          unordered_map<int, Employee*> emap;
35          for (Employee* e : employees)
36              emap[e->id] = e;
37          queue<Employee*> que;
38          que.push(emap[id]);
39          int ans = 0;
40          while (!que.empty()) {
41              Employee* employee = que.front();
42              que.pop();
43              ans += employee->importance;
44              for (int subordinate : employee->subordinates)
45                  que.push(emap[subordinate]);
46          }
47          return ans;
48      }
49  };
```