# 哈希映射： 1743. 从相邻元素对还原数组

存在一个由 `n` 个不同元素组成的整数数组 `nums` ，但你已经记不清具体内容。好在你还记得 `nums` 中的每一对相邻元素。

给你一个二维整数数组 `adjacentPairs` ，大小为 `n - 1` ，其中每个 `adjacentPairs[i] = [$u_i$, $v_i$]` 表示元素 $u_i$ 和 $v_i$ 在 `nums` 中相邻。

题目数据保证所有由元素 `nums[i]` 和 `nums[i+1]` 组成的相邻元素对都存在于 `adjacentPairs` 中，存在形式可能是 `[nums[i], nums[i+1]]` ，也可能是 `[nums[i+1], nums[i]]` 。这些相邻元素对可以 **按任意顺序** 出现。

返回 **原始数组** `nums` 。如果存在多种解答，返回 **其中任意一个** 即可。

**示例 1：**

```
输入：adjacentPairs = [[2,1],[3,4],[3,2]]
输出：[1,2,3,4]
解释：数组的所有相邻元素对都在 adjacentPairs 中。
特别要注意的是，adjacentPairs[i] 只表示两个元素相邻，并不保证其 左-右 顺序。
```

## 我的代码：哈希映射（超时）

```cpp
class Solution {
public:
    vector<int> restoreArray(vector<vector<int>>& adjacentPairs) {
        //建立哈希映射
        unordered_map<int,vector<pair<int,bool>>>mymap;
        for(auto temp:adjacentPairs){
            mymap[temp[0]].push_back(make_pair(temp[1],true));
            mymap[temp[1]].push_back(make_pair(temp[0],true));
        }
        //遍历以求结果
```

```
        vector<int>res;
        res.push_back(adjacentPairs[0][0]);
        res.push_back(adjacentPairs[0][1]);
        for(auto &tp:mymap[adjacentPairs[0][0]]){
            if(tp.first==adjacentPairs[0][1]){
                tp.second=false;
                break;
            }
        }
        for(auto &tp:mymap[adjacentPairs[0][1]]){
            if(tp.first==adjacentPairs[0][0]){
                tp.second=false;
                break;
            }
        }
        bool flag=true;
        while(flag){
            //向后
            flag=false;
            int num=res.back();
            for(auto &tp:mymap[num]){
                if(tp.second){
                    flag=true;
                    for(auto &t:mymap[tp.first]){
                        if(t.first==res.back()){
                            t.second=false;
                            break;
                        }
                    }
                    res.push_back(tp.first);
                    tp.second=false;
                    break;
                }
            }
        }
        flag=true;
        while(flag){
            //向前
            flag=false;
            int num=res[0];
            for(auto &tp:mymap[num]){
                if(tp.second){
                    flag=true;
                    for(auto&t:mymap[tp.first]){
                        if(t.first==res[0]){
                            t.second=false;
                            break;
                        }
                    }
```

```
                    //此处insert会耗费太多
                    res.insert(res.begin(),tp.first);
                    tp.second=false;
                    break;
                }
            }
        }
        return res;
    }
};
```

# 哈希映射：优化

```
class Solution {
public:
    vector<int> restoreArray(vector<vector<int>>& adjacentPairs) {
        unordered_map<int, vector<int>> mp;
        for (auto& adjacentPair : adjacentPairs) {
            mp[adjacentPair[0]].push_back(adjacentPair[1]);
            mp[adjacentPair[1]].push_back(adjacentPair[0]);
        }//此处建立哈希映射

        //取得结果
        int n = adjacentPairs.size() + 1;
        //找头结点
        vector<int> ret(n);
        for (auto& [e, adj] : mp) {
            if (adj.size() == 1) {
                ret[0] = e;
                break;
            }
        }

        //根据数字的唯一性向下寻找
        ret[1] = mp[ret[0]][0];
        for (int i = 2; i < n; i++) {
            auto& adj = mp[ret[i - 1]];
            ret[i] = ret[i - 2] == adj[0] ? adj[1] : adj[0];
        }
        return ret;
    }
};
```

作者：LeetCode-Solution
链接：https://leetcode-cn.com/problems/restore-the-array-from-adjacent-pairs/solution/cong-xiang-lin-yuan-su-dui-huan-yuan-shu-v55t/
来源：力扣（LeetCode）
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。