CS208 Lab6 Practice

12312110 李轩然

DDL: Apr.27

Max path in tree use divide-and-conquer

Description

Given a binary tree where each node has an integer value, find a path (a sequence of nodes connected parent-to-child) with the maximum sum of node values. The path does not need to pass through the root, and each node can be visited at most once.

Input:

First line has 1 integer, is the number of node n;
Second line has n integers, are the values of each node;
The next $n-1$ lines, are the edges.

The root is node 1.

Analysis

Define sum as the sum of node values. For the root u of any subtree, there are two cases: 1. the longest path is in u's left / right subtree; 2. the longest path contains u. In case 1, we just use the same method to search the maximum path in left / right subtree; in case 2, we do the same thing, and add them and the value of root. If it's larger than sum, update sum. For case 1, the connected root must be the father of u, and in this subtree, the maximum value of the branch is: value of u add the maximum value of left / right subtree. Until we backtrack to the root 1, and sum is the final answer. For every nodes will be iterated several time, the time complexity will be $O(n)$.

C++ Code

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <climits>
using namespace std;

int sum = INT_MIN;

struct TreeNode {
    int val;
    TreeNode* left;
```

```
    TreeNode* right;
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};

int maxpath(TreeNode* root) {
    if (root == nullptr) return 0;
    int l = max(0, maxpath(root->left));
    int r = max(0, maxpath(root->right));
    sum = max(sum, root->val + l + r);
    return root->val + max(l, r);
}

int main()
{
    int n;
    cin >> n;
    vector<int> a(n + 1);
    for (int i = 1; i <= n; i++)
        cin >> a[i];

    vector<TreeNode*> node(n + 1);
    for (int i = 1; i <= n; i++)
        node[i] = new TreeNode(a[i]);

    for (int i = 1; i < n; i++)
    {
        int u, v;
        cin >> u >> v;
        if (node[u]->left == nullptr) node[u]->left = node[v]; else node[u]->right =
node[v];
    }

    maxpath(node[1]);
    cout << sum << endl;
    return 0;
}
```