834. Sum of Distances in Tree

[我的提交](https://leetcode-cn.com/contest/weekly-contest-84/problems/sum-of-distances-in-tree/submissions/)[返回竞赛](https://leetcode-cn.com/contest/weekly-contest-84/)

* 用户通过次数1
* 用户尝试次数8
* 通过次数1
* 提交次数18
* 题目难度**Hard**

An undirected, connected tree with N nodes labelled 0...N-1 and N-1 edges are given.

The ith edge connects nodes edges[i][0] and edges[i][1] together.

Return a list ans, where ans[i] is the sum of the distances between node i and all other nodes.

**Example 1:**

**Input:** N = 6, edges = [[0,1],[0,2],[2,3],[2,4],[2,5]]

**Output:** [8,12,6,10,10,10]

**Explanation:**

Here is a diagram of the given tree:

0

/ \

1 2

/|\

3 4 5

We can see that dist(0,1) + dist(0,2) + dist(0,3) + dist(0,4) + dist(0,5)

equals 1 + 1 + 2 + 2 + 2 = 8. Hence, answer[0] = 8, and so on.

Note: 1 <= N <= 10000

class Solution {

public int[] sumOfDistancesInTree(int n, int[][] edges) {

int[] from = new int[n-1];

int[] to = new int[n-1];

for(int i = 0;i < n-1;i++){

from[i] = edges[i][0];

to[i] = edges[i][1];

}

int[][] g = packU(n, from, to);

int[][] pars = parents3(g, 0);

int[] par = pars[0], ord = pars[1], dep = pars[2];

int[] dp = new int[n];

int[] des = new int[n];

Arrays.fill(des, 1);

for(int i = n-1;i >= 1;i--){

des[par[ord[i]]] += des[ord[i]];

}

for(int i = n-1;i >= 0;i--){

int cur = ord[i];

for(int e : g[cur]){

if(par[cur] == e)continue;

dp[cur] += dp[e] + des[e];

}

}

for(int i = 1;i < n;i++){

int cur = ord[i];

int p = par[cur];

dp[cur] += dp[p] - dp[cur] - des[cur] + (n - des[cur]);

}

return dp;

}

public int[][] parents3(int[][] g, int root) {

int n = g.length;

int[] par = new int[n];

Arrays.fill(par, -1);

int[] depth = new int[n];

depth[0] = 0;

int[] q = new int[n];

q[0] = root;

for (int p = 0, r = 1; p < r; p++) {

int cur = q[p];

for (int nex : g[cur]) {

if (par[cur] != nex) {

q[r++] = nex;

par[nex] = cur;

depth[nex] = depth[cur] + 1;

}

}

}

return new int[][] { par, q, depth };

}

int[][] packU(int n, int[] from, int[] to) {

int[][] g = new int[n][];

int[] p = new int[n];

for (int f : from)

p[f]++;

for (int t : to)

p[t]++;

for (int i = 0; i < n; i++)

g[i] = new int[p[i]];

for (int i = 0; i < from.length; i++) {

g[from[i]][--p[from[i]]] = to[i];

g[to[i]][--p[to[i]]] = from[i];

}

return g;

}

}