Wildflower

Calvin Fung

July 17, 2025

1 Problem

Yousef has a rooted tree* consisting of exactly n vertices, which is rooted at vertex 1. You would like to give Yousef an array a of length n, where each a_i $(1 \le i \le n)$ **can either be** 1 **or** 2.

Let s_u denote the sum of a_v where vertex v is in the subtree† of vertex u. Yousef considers the tree *special* if all the values in s are **pairwise distinct** (i.e., all subtree sums are unique).

Your task is to help Yousef count the number of different arrays a that result in the tree being *special*. Two arrays b and c are different if there exists an index i such that $b_i \neq c_i$.

2 Solution

We present a linear time $O(n)$ algorithm.

2.1 analysis

Consider the directed version of the input tree T where edges are *pointing toward* the root.

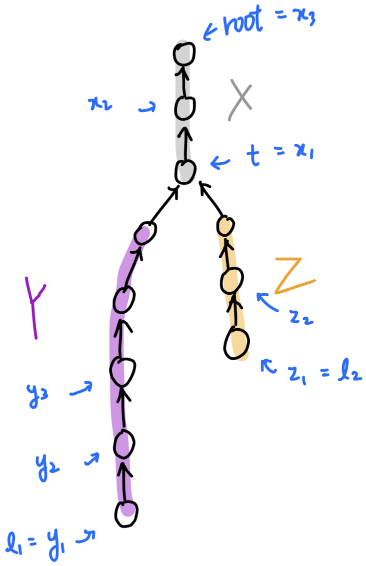
Observation 1. If there's a directed path from u to v, then $s_u \neq s_v$. This's because u is in the subtree of v and weights a_i are all positive.

First T must have at least 1 node with indegree 0 ie. a leaf.

If it has exactly 1 node with indegree 0, then it's a directed path, by observation 1 no pairs of nodes will have same s values. Thus the count is 2^n.

If it has ≥ 3 nodes with indegree 0, then the underlying undirected tree has ≥ 3 leaves, by pigeonhole principle two of those leaves will have same a values, which means same s values for leaves. Thus the count is 0.

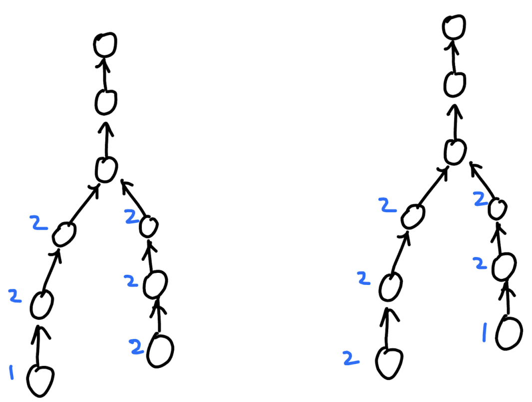
The remaining case is when T has exactly 2 nodes l_1, l_2 with indegree 0, so it'll has the form



notice there's exactly 1 node with indegree 2, call it t. Above we partitioned the nodes into 3 parts : the path $X = \{x_1 = t, ..., x_{k_1} = 1\}$ from t to root, the path $Y = \{y_1 = l_1, ..., y_{k_2}\}$ from l_1 to t excluding t, the path $Z = \{z_1 = l_2, ..., z_{k_3}\}$ from l_2 to t excluding t.

By observation 1, only pairs involving a node from Y and the other from Z may have same s value (all other pairs will never have same s value). This means nodes from X can take any a values. So below we focus on Y, Z, WLOG suppose $|Z| \leq |Y|$.

case 1 : $|Y| = |Z|$, then there are only two possible configurations for Y, Z



Let $P(w)$ be the statement a_{y_w} can only be 2 and $s_{y_w} = s_{z_w - 1} + 1$. Let $Q(w)$ be the statement a_{z_w} can only be 2 and $s_{y_w} + 1 = s_{z_w}$.

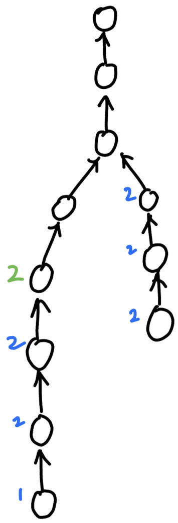
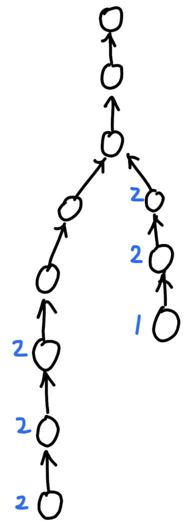
For the base case suppose suppose $a_{y_1} = 1$ (the other case ie. $a_{z_1} = 1$ is symmetric), so $Q(1)$ holds.

Now the inductive case, given $Q(w)$ holds, we show both $P(w+1)$ and $Q(w+1)$ holds. First we show $P(w+1)$ holds, notice $a_{y_{w+1}}$ can only be 2 otherwise $s_{y_{w+1}} = s_{y_w} + 1 = s_{z_w}$ a contradiction, it follows that $s_{y_{w+1}} = s_{z_w} + 1$. Then we show $Q(w+1)$ holds, as we have just shown $P(w+1)$ holds, $a_{z_{w+1}}$ can only be 2 otherwise $s_{z_{w+1}} = s_{z_w} + 1 = s_{y_{w+1}}$ a contradiction, it follows $s_{y_{w+1}} + 1 = s_{z_{w+1}}$.

Notice $P(2), ..., P(|Y|)$ and $Q(1), ..., Q(|Z|)$ hold. Therefore, given $y_1 = 1$, all other nodes in Y, Z must equal 2.

Since the other case is symmetric, the desired answer is $2 \times 2^{n-2|Y|}$.

case 2 : $|Y| > |Z|$, then we have 2 patterns

Given $a_{y_1} = 1$, notice $P(2), ..., P(|Z|+1)$ and $Q(1), ..., Q(|Z|)$ hold so the a values of $y_2, ..., y_{|Z|+1}$ and Z must equal 2. However, because $P(|Z|+1)$ holds ie. $s_{y_{|Z|+1}} = s_{z_{|Z|}} + 1$, notice the s value of $y_{|Z|+2}, ..., y_{k_2}$ will be strictly larger than any s value of Z, thus they can take any a value. The contribution to the total count in this case is $2^{n-2|Z|-1}$.

Otherwise given $a_{z_1} = 1$, let $P'(w)$ be the statement a_{z_w} can only be 2 and $s_{z_w} = s_{y_{w-1}} + 1$. Let $Q'(w)$ be the statement a_{y_w} can only be 2 and $s_{z_w} + 1 = s_{y_w}$. Clearly, $Q'(1)$ holds. Given $Q'(w)$ holds, we can show both $P'(w+1)$ and $Q'(w+1)$ holds. First we show $P'(w+1)$ holds, notice $a_{z_{w+1}}$ can only be 2 otherwise $s_{z_{w+1}} = s_{z_w} + 1 = s_{y_w}$ a contradiction, it follows that $s_{z_{w+1}} = s_{y_w} + 1$. Then we show $Q'(w+1)$ holds, as we have just shown $P'(w+1)$ holds, $a_{y_{w+1}}$ can only be 2 otherwise $s_{y_{w+1}} = s_{y_w} + 1 = s_{z_{w+1}}$ a contradiction, it follows $s_{z_{w+1}} + 1 = s_{y_{w+1}}$.

Notice $P'(2), ..., P'(|Z|)$ and $Q'(1), ..., Q'(|Z|)$ hold so the a values of $y_1, ..., y_{|Z|}$ and $Z - \{z_1\}$ must equal 2. However, because $Q'(|Z|)$ holds ie. $s_{z_{|Z|}} + 1 = s_{y_{|Z|}}$, notice the s value of $y_{|Z|+1}, ..., y_{k_2}$ will be strictly larger than any s value of Z, thus they can take any a value. The contribution to the total count in this case is $2^{n-2|Z|}$.

The total count for this case is thus $2^{n-2|Z|-1} + 2^{n-2|Z|}$.

2.2 algorithm

As described above, we present a linear time algorithm.

```
f(E,n){
        Built a tree T (using pointers) where the edge set is E.
        Make T a rooted tree by doing DFS starting at the root so now
        for each node we have node.parent and node.child

        suppose T has m nodes leaves (nodes with no child)
        if(m == 1)        return 2^n;
        if(m >= 3)        return 0;
        else {
                T has 2 leaves l_1,l_2
                size_Y, size_Z = 1
                u = l_1
                while(u.parent has exactly 1 child){
                        u = u.parent
                        size_Y  ++;
                }
                u = l_2
                while(u.parent has exactly 1 child){
                        u = u.parent
```

```
                    size_Z   ++;
                }
                if(size_Y < size_Z) swap(size_Y,size_Z);
                return (size_Y == size_Z) ? 2*2^(n-2*size_Y) : 2^(n - 2*size_Z - 1) + 2^(n - 2*size_Z)
        }
}
```