Problem D - Beauty and The Tree

You are given a rooted tree with **N** nodes numbered from **1** to **N**. Node **1** is the root. Each node **i** contains a beauty value b_i. The beauty of the **d** - th level of the tree is the sum of the beauty values of all the nodes with depth **d** and the beauty of the tree is the largest beauty value of its levels. For each node **v (2 ≤ v ≤ N)** , you have to output the beauty of the tree assuming that the subtree rooted at node **v**(including node **v**) does not exist in the tree.

Input Specification

The first line of input will contain a number **T**, the number of test cases. Each of the test cases contain 3 lines. The first line contains **N**, the number of nodes in the tree. The second line contains **N** integers $b_1, b_2, \ldots \ldots, b_n$, the beauty values of the nodes. The third line contains **N-1** integers $p_2, p_3, \ldots \ldots, p_n$ **(1<=pi<i)** where node p_i is the parent of node **i** in the tree.

Constraints

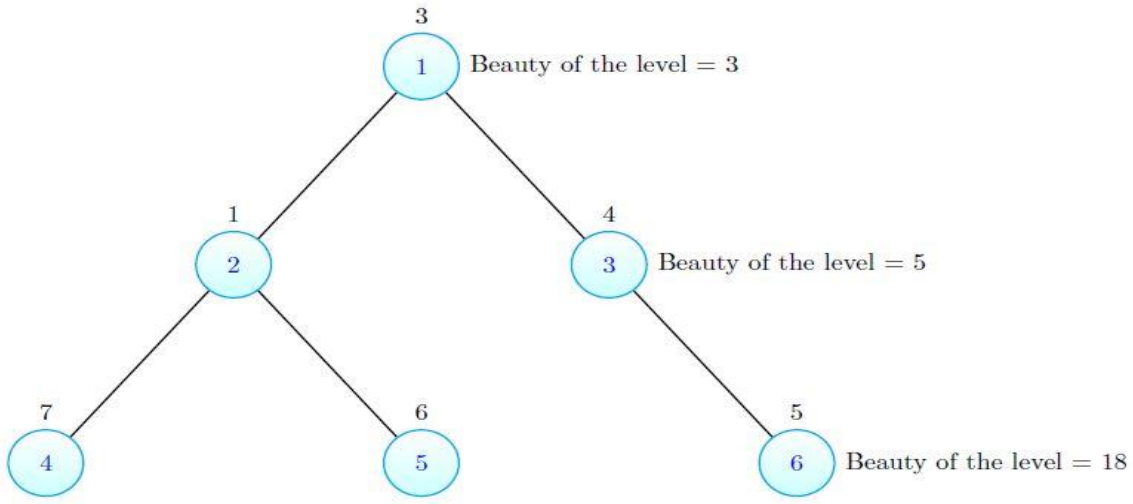
$1 \leq T \leq 30$
$1 \leq N \leq 100000$
$0 \leq b_i \leq 10000$

Output Specification

For each test case output **N-1** lines. The i_{th} line should contain the beauty value of the tree assuming that the subtree of node **i+1**(including node **i+1**) does not exist in the tree.

Input	Output
2 5 1 5 2 7 9 1 2 3 4 6 3 1 4 7 6 5 1 1 2 2 3	1 5 5 7 5 13 11 12 13

Original Tree of The Second Test Case



```cpp
/// GYM 101864D : Beauty and The Tree
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 100005, BLOCK = 320;
int dis[MAXN],fin[MAXN],nod[MAXN],lev[MAXN],wgt[MAXN];
int tree[4*MAXN],b[MAXN],ans[MAXN],tym,height;
vector<int>ed[MAXN];
struct dt{ int l,r,i; }qr[100005];
bool cmp(dt A, dt B){
  if (A.l / BLOCK != B.l / BLOCK) return A.l < B.l;
  return A.l / BLOCK % 2 ? A.r > B.r : A.r < B.r;
}
void dfs(int u,int p,int l){
    dis[u]   =  ++tym;
    nod[tym] =  u; lev[tym] =  l; wgt[tym] =  b[u];
    height = max(height,l);
    for(int i=0; i<ed[u].size(); i++){
        int v = ed[u][i];
        if(v==p)continue;
        dfs(v,u,l+1);
    }
    fin[u]=tym;
}
void update(int nd,int l,int r,int x,int v){
    if(l==r){ tree[nd]+=v; return; }
    int m = (l+r)/2;
    if(x<=m)update(2*nd,l,m,x,v);
    else update(2*nd+1,m+1,r,x,v);
    tree[nd] = max(tree[2*nd],tree[2*nd+1]);
}
void Add(int x){
    update(1,1,height,lev[x],-wgt[x]);
}
void Remove(int x){
    update(1,1,height,lev[x],wgt[x]);
}
int main(){
    int tt; scanf("%d",&tt);
    for(int ks=1; ks<=tt; ks++){
        int n; scanf("%d",&n);
        for(int i=1; i<=n; i++)scanf("%d",&b[i]);
        for(int i=2; i<=n; i++){
            int x; scanf("%d",&x);
            ed[x].push_back(i); ed[i].push_back(x);
        }
        tym = height = 0;
        dfs(1,0,1);
        for(int i=1; i<=n; i++){
            qr[i].l = dis[i];
            qr[i].r = fin[i];
            qr[i].i = i;
        }
        sort(qr+1,qr+n+1,cmp);
        int l=1,r=n;
        for(int i=1; i<=n; i++){
            while(r<qr[i].r) Add(++r);
            while(r>qr[i].r) Remove(r--);
            while(l>qr[i].l) Add(--l);
            while(l<qr[i].l) Remove(l++);
            ans[qr[i].i] = tree[1];
        }

        for(int i=2; i<=n; i++) printf("%d\n",ans[i]);

        for(int i=1; i<=n; i++)ed[i].clear();
        memset(tree,0,sizeof(tree));
    }
    return 0;
}
```