TeaTree

一. 考察内容：

线段树 线段树合并 求因数

二. 题目分析：

[题目大意]

在一个树上每个点都有一个权值，并且每两个点都会给他们的LCA发送一个值为gcd(v[a],v[b])的信息，求出每个点能收到的最大权值是多少（没有收到信息的点输出0）。

[写题思路]

1. 我们求出每个节点权值的所有因数，用他们建权值线段树，如果权值线段树上i为1，则表示该节点有因数1。
2. 我们考虑递归求解，遍历整棵树，对于一个节点，我们先求出该节点收到的最大权值，这个权值显然是从两个不同的子树的节点传来的（LCA的性质），对于我们逐个考察每个子树，对于一棵子树i，我们与之前访问到的所有子树的并集求交集，很显然，交集中的最大数就是前i棵子树能贡献的最大值，需要注意的是，我们遍历到的这个节点，也被看做是自己的子树。
3. 求出一个节点能收到的最大值后，我们考虑合并以该节点为根的整棵子树，将该节点的权值设为与子树权值的并集。
4. 最后按照顺序输出答案即可。

这是我写的第一道线段树合并问题，线段树合并可以采用节点合并的方式，来合并两棵形状、大小相同的线段树，对于一些信息，将线段树合并之后可以加快查找信息的次数和速度。

```
int merge(int a, int b)
{
    if (b == 0) return a;
    if (a == 0) return b;
    int l = merge(lc[a], lc[b]), r = merge(rc[a], rc[b]);
    t = ++cnt;
    lc[t] = l, rc[t] = r;
    //合并信息
}
```

三. 代码实现：

```
#define _CRT_SECURE_NO_DEPRECATE
#pragma comment(linker, "/STACK:102400000,102400000")
/*************************
*创建时间：2018 08 26
*文件类型：源代码文件
*题目来源：HDU
*当前状态：已通过
*备忘录：线段树 线段树合并
*作者：HtBest
*************************/
#include <stdio.h>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <string.h>
#include <algorithm>
```

```
#include <queue>
#include <math.h>
// #include <sys/wait.h>
// #include <sys/types.h>
// #include <unistd.h>
using namespace std;
#define MAXN 100010
int n,m,fa[MAXN],son[MAXN],bro[MAXN],v[MAXN],cnt,ans[MAXN];
struct NODE
{
    int lson,rson,v;
    void reset()    {lson=rson=v=0;}
    void merge(NODE &a,NODE &b){v=a.v||b.v;}
}o[35000000];
struct segmentTree
{
    int root;
    void getid(int &x)//新开一个节点
    {
        x=++cnt;
        o[x].reset();
    }
    void update(int &x,int l,int r,int p)
    {
        if(!x)getid(x);
        if(l==r)    o[x].v=1;
        else
        {
            int m=(l+r)>>1;
            if(p<=m)update(o[x].lson,l,m,p);
            else update(o[x].rson,m+1,r,p);
            o[x].merge(o[o[x].lson],o[o[x].rson]);
        }
    }
    void query(int x,int l,int r,int p,NODE &q)
    {
        if(l==r)q=o[x];
        else
        {
            int m=(l+r)>>1;
            if(p<=m)query(o[x].lson,l,m,p,q);
            else query(o[x].rson,m+1,r,p,q);
        }
    }
    int qmax(int x,int l,int r)
    {
        if(l==r)return l;
        else
        {
            int m=(l+r)>>1;
            if(o[x].rson&&o[o[x].rson].v)return qmax(o[x].rson,m+1,r);
            if(o[x].lson&&o[o[x].lson].v)return qmax(o[x].lson,l,m);
            return 0;
        }
    }
    void reset()    {root=0;}
    void update(int p)  {update(root,1,100000,p);}
    bool query(int p)
    {
        NODE ans;
        query(root,1,100000,p,ans);
        return ans.v;
    }
    int qmax() {return qmax(root,1,100000);}
}seg[MAXN];
/* Variable explain:

*/
int merge(int &a,int &b)
{
    if(!o[a].v)return b;
    if(!o[b].v)return a;
```

```
    int x=++cnt;
    o[x].merge(o[a],o[b]);
    if(o[a].lson||o[a].rson||o[b].lson||o[b].rson)//不为叶子节点
    {
        o[x].lson=merge(o[a].lson,o[b].lson);
        o[x].rson=merge(o[a].rson,o[b].rson);
    }
    return x;
}
int merge2(int &a,int &b)
{
    if(!o[a].v||!o[b].v)return 0;
    int x=++cnt;
    if(o[a].lson||o[a].rson||o[b].lson||o[b].rson)//不为叶子节点
    {
        o[x].lson=merge2(o[a].lson,o[b].lson);
        o[x].rson=merge2(o[a].rson,o[b].rson);
        o[x].v=o[o[x].lson].v||o[o[x].rson].v;
    }
    else o[x].v=o[a].v&&o[b].v;
    return x;
}
void addt(int a,int b)
{
    fa[b]=a;
    bro[b]=son[a];
    son[a]=b;
}
void read()
{
    int ls1;
    // freopen(".in","r",stdin);
    // freopen(".out","w",stdout);
    scanf("%d",&n);
    for(int i=2;i<=n;++i)scanf("%d",&ls1),addt(ls1,i);
    for(int i=1;i<=n;++i)scanf("%d",&v[i]);
    return;
}
void dfs(int a)
{
    // printf("%d:\n    ",a); for(int j=1;j<=10;++j)    if(seg[a].query(j))printf("%d
",j); puts("");
    for(int i=son[a];i;i=bro[i])
    {
        dfs(i);
        segmentTree ls1;
        ls1.reset();
        ls1.root=merge2(seg[a].root,seg[i].root);
        // printf("%dn%d:\n ",a,i); for(int j=1;j<=10;++j)  if(ls1.query(j))printf("%d
",j);    puts("");
        // printf("#%d %d#\n",a,ls1.qmax());
        ans[a]=max(ans[a],ls1.qmax());//更新最大值
        cnt=ls1.root-1;//有趣的释放内存
        seg[a].root=merge(seg[a].root,seg[i].root);//合并子树和根

        // printf("%d:\n    ",a); for(int j=1;j<=10;+
+j)    if(seg[a].query(j))printf("%d ",j); puts("");
    }
}
void init()
{
    for(int i=1;i<=n;++i)
    {
        int sq=sqrt(v[i])+1;
        for(int j=1;j<=sq;++j)
            if(v[i]%j==0)
            {
                // printf("%d %d\n%d %d\n",i,j,i,v[i]/j);
                seg[i].update(j);
                seg[i].update(v[i]/j);
            }
```

```
    }
}
int main()
{
    read();
    init();
    dfs(1);
    for(int i=1;i<=n;++i)printf("%d\n",ans[i]?ans[i]:-1);
    return 0;
}
```

<题目跳转> <查看代码>