ICPC Nanjing 2021 H, Crystalfly

ICPC Nanjing 2021 H, Crystalfly - 题目 - Daimayuan Online Judge

题目简介

从根节点往下走，当到达一个节点时，可以把该节点上地所有蝴蝶抓走。

但是当到达一个节点时，相邻节点上的蝴蝶就会被惊动，并且在t_i秒后飞走（1<=t_i<=3）

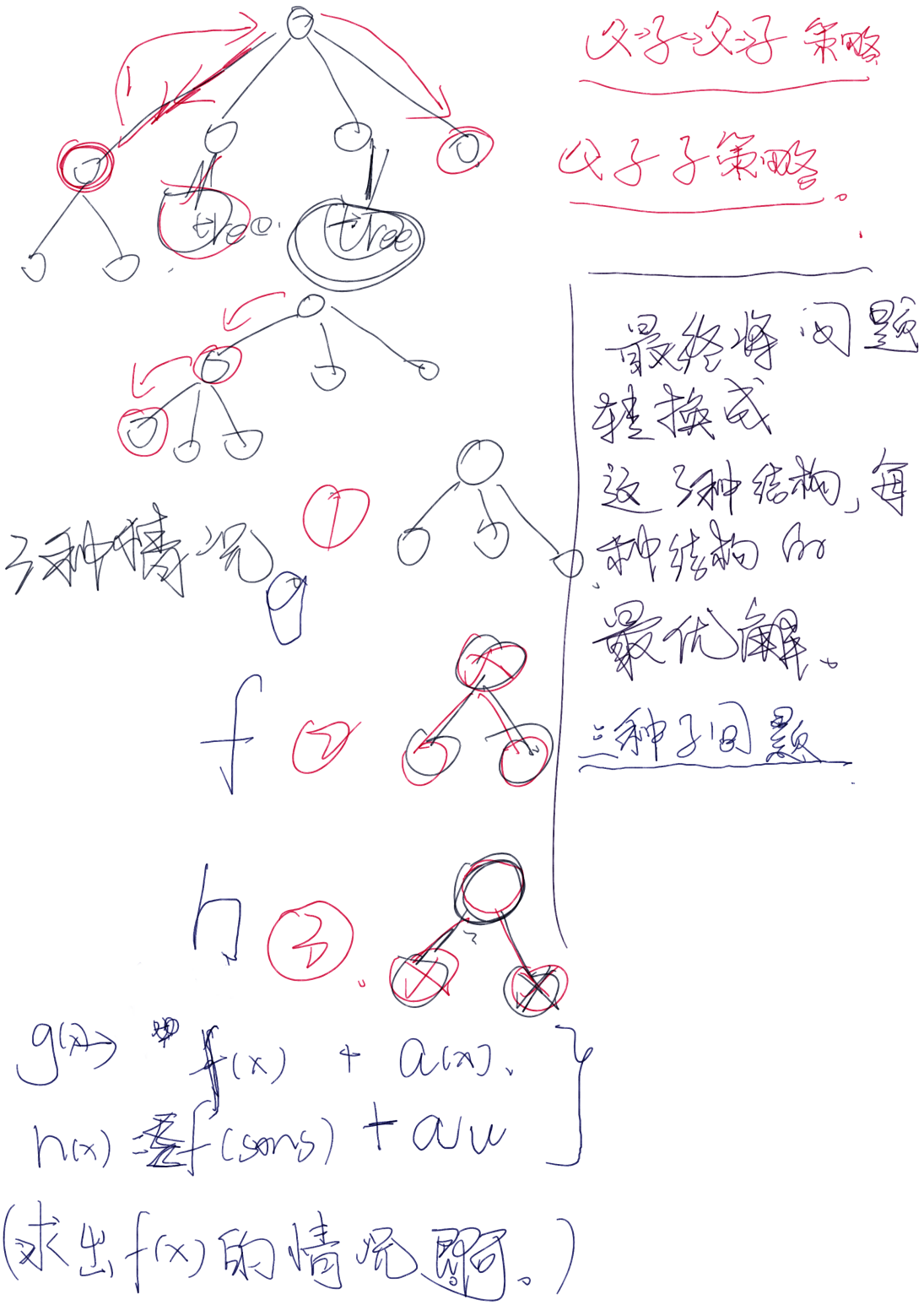
求出最大的可抓蝴蝶个数（假设时间无限）

solve

考察从根部往下走：

1. 可以采取两种策略：
    1. 1 - 》 2 - 》 3
    2. 1 - 》 2 —》 1 - 》2

    如果选择了1.那么2子树全部处理完再回来才是更优策略。因为回到1之后 1 之后，那些儿子的蝴蝶都已经飞走了。利用返回的时间，不如在2子树中考虑其它的策略。
2.



求出每种子树作为这三种结构之一的最优解。

由于三种子树之间的关系是互相转移的。所以只需要求出f的结构即可。

第一种方案：

$$g_v + \sum f_{son!=v} \\ = \sum(f_{son}) + max(a_{sons}) \tag{1}$$

第二种方案：

$$h_i + g_j + \sum f_{sons} - f_i - f_j \tag{2}$$
$$= \sum f_{sons} + max(h_i - f_i + max(\text{剩下的，} t = 3\text{的最大的权值。}))\text{；}$$

转移实现

1. 第一种方案：每个节点贡献一次。记。记录两个变量。总花费是$O(n)$
2. 第二种方案：
    1. 暴力枚举a_i, a_j。如果节点的度数非常的大，花费的复杂度将会高达$O(n^2)$
    2. 优化的角度是。
        1. 记录下最大与次大的a_i。
        2. 所有的方案都是从这两个之中选的。
        3. 枚举i， 然后看是否为最大，如果是就选择次打。否则为最大。

code

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;

const int oo = 0x0fffffff;
const int N = 1E6 + 10;

ll a[N];
int t[N];

void work(int testNo)
{
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++) cin >> a[i];
    for (int i = 1; i <= n; i++) cin >> t[i];

    vector<vector<int>> son(n + 1);
    for (int i = 1; i < n; i++) {
        int u , v; cin >> u >> v;
        son[u].push_back(v); son[v].push_back(u);
    }
    vector<ll> f(n + 1 , 0)  , h(n + 1 , 0) ;

    function<void(int , int)> dfs = [&](int u , int fa) -> void{
        ll sum = 0;
        ll ma = 0;
        for (auto v : son[u]) if (v != fa) {
                dfs(v , u);
                sum += f[v];
                ma = max(ma , a[v]);
            }

        f[u] = sum + ma;
        pair<ll , int> mx1{0 , 0} , mx2{0 , 0};
```

```
        for (auto v : son[u]) if (v != fa && t[v] == 3) {
                pair<ll , int> mx3 = {a[v] , v};
                if (mx3 > mx2) mx2 = mx3;
                if (mx2 > mx1) swap(mx1 , mx2);
            }

        for (auto v : son[u])if (v != fa) {
                ll temp = sum + h[v] - f[v] ;
                if (v == mx1.second) temp += mx2.first;
                else temp += mx1.first;
                f[u] = max(f[u] , temp);
            }
        h[u] = sum + a[u];
    };
    dfs(1 , 0);
    cout << f[1] + a[1] << '\n';

}


int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);

    int t; cin >> t;
    for (int i = 1; i <= t; i++)work(i);
}

/* stuff you should look for
* int overflow, array bounds
* special cases (n=1?)
* do smth instead of nothing and stay organized
* WRITE STUFF DOWN
* DON'T GET STUCK ON ONE APPROACH
*/
```

生长思考：

1. 对于最大位，次位置。
    1. 仅仅记录大小是错误的。同时应该记录位置。因为最大次大这两个，在儿子集合中的（t = 3）子集中选取。子集之外也有等大的。会造成影响。充要的方法是，记录儿子标号，利用儿子标号来记录。
2. dls是怎么探求问题的解空间的?
    1. 从最小的状态开始。枚举出状态