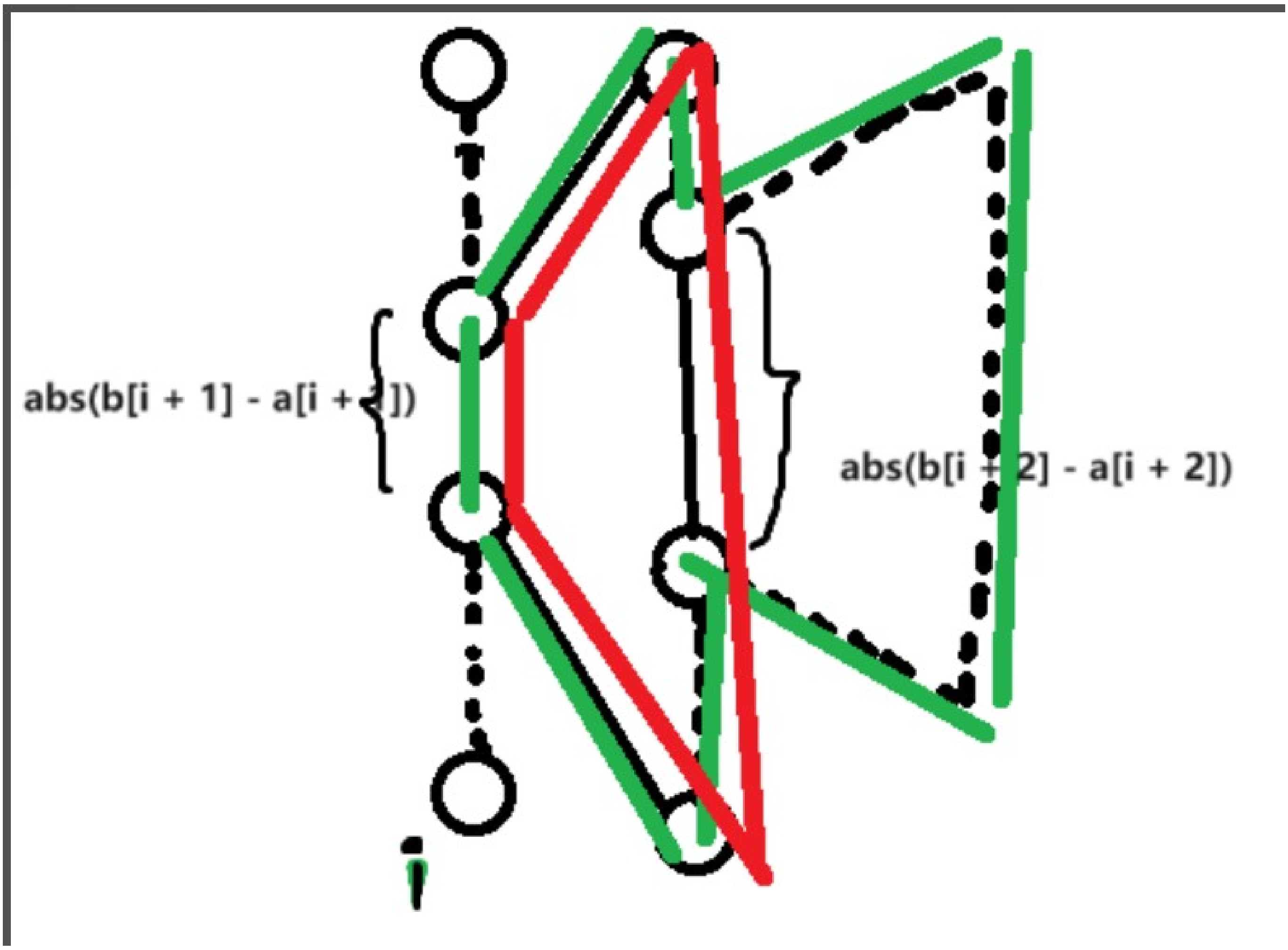
I 为什么不来看看我

应该能看出来是个 dp。

用 dp_i 表示从末尾到第 i 条链为止（指第 i 条链不往第 $i-1$ 条链连接），所以显然 dp 范围是从 $n-1$ 到 1 。

则考虑 dp_i 如何从 dp_{i+1} 转移，存在两种情况： 如图中红色和绿色两种路径，其中注意绿色路径只有当 $a_{i+1} \neq b_{i+1}$ 时才有效

~~写题解的水平有限，凑活看吧~~



于是转移方程为：

```
ll ans1 = 0;
if(a[i + 2] != b[i + 2])
    ans1 = abs(a[i + 1] - b[i + 1]) + 2 + dp[i + 1] + c[i + 1] - 1 - 2*abs(a[i + 2] - b[i + 2]);
ll ans2 = abs(a[i + 1] - b[i + 1]) + 2 + c[i + 1] - 1;
dp[i] = max(ans1, ans2);
```

参考代码：

```
// Created by qyy on 2024/8/4.

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

#define PII pair<int, int>
#define endl "\n"

const long long inf = 0x3f3f3f3f3f3f3f3f;
const int N = 2e5 + 10;
const int mod = 1e9 + 7;

int n;
ll dp[N];
ll a[N], b[N], c[N], d[N];

void init(){
    for(int i = 1; i <= n + 3; i++){
        dp[i] = a[i] = c[i] = b[i] = 0;
    }
}
void solve() {
    cin >> n;
    init();
    for(int i = 1; i <= n; i++){
        cin >> c[i];
    }
    for(int i = 1; i <= n; i++){
        cin >> a[i];
    }
    for(int i = 1; i <= n; i++){
        cin >> b[i];
    }
    ll ans = 0;
    for(int i = n - 1; i >= 1; i--){
        ll ans1 = 0;
        if(a[i + 2] != b[i + 2]){
            ans1 = abs(a[i + 1] - b[i + 1]) + 2 + dp[i + 1] + c[i + 1] - 1 -
2*abs(a[i + 2] - b[i + 2]);
        }
        ll ans2 = abs(a[i + 1] - b[i + 1]) + 2 + c[i + 1] - 1;
        dp[i] = max(ans1, ans2);
    }
    for(int i = 1; i <= n; i++){
        ans = max(ans, dp[i]);
    }
    cout << ans << endl;
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    int t = 1;
    cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

M 重生之我是树论高手

题目大意

给出一个n个点，m条边组成的森林，有q组询问： 1.给出点x,求点x所在的树的直径 2.给出点x,y,要求将x,y所在的树之间连一条边并构成一棵新的树，满足这个新的树的直径最小

解题思路

首先，我们用树形DP (或bfs)求出每棵树的直径，并用并查集维护连通情况 维护c数组：对于每棵树的根节点x，$c[x]$ =该树的直径长度 接下来，对于每个询问 2(如果给出的两点在同一棵树内则忽略),利用并查集找出两棵树的根节点x,y ,并用并查集合并两棵树；合并后的树的直径则为$max\lceil\frac{c[x]}{2}\rceil+\lceil\frac{c[y]}{2}\rceil+1,c[x],c[y]$,这里讲一下原因 要想直径最短，我们选择加边的点一定要在直径上，因为其他的点走到直径还要一段距离，从而增长了路径 那么直径就被选择的点分成了两段。因为我们要最小化较长的那一段，所以要让选择的点尽量靠近直径的 中点。最后的答案就是 直径长度的一半向上取整

```
// Created by qyy on 2024/5/31.

#include <bits/stdc++.h>
using namespace std;

typedef long long ll;

#define PII pair<int, int>
#define endl "\n"

const long long inf = 0x3f3f3f3f3f3f3f3f;
const int N = 4e5 + 10;
const int mod = 1e9 + 7;

vector<int> tmp;
int n, m, q, dist[N], fa[N], zhi[N];
bool vis[N];

int head[N], cnt;
struct Edge{
    int from, to, nxt;
}e[N << 1];
```

```
void add(int u, int v){
    e[++cnt].from = u;
    e[cnt].to = v;
    e[cnt].nxt = head[u];
    head[u] = cnt;
}

void dfs(int u, int father, int d){
    tmp.push_back(u);
    dist[u] = d;
    for(int i = head[u]; i != 0; i = e[i].nxt){
        int v = e[i].to;
        if(v != father)
            dfs(v, u, d + 1);
    }
}

int find_set(int x){
    if(fa[x] != x) fa[x] = find_set(fa[x]);
    return fa[x];
}
void merge_set(int x, int y){
    x = find_set(x);
    y = find_set(y);
    if(x != y){
        fa[x] = y;
    }
}



void solve() {
    cin >> n >> m >> q;
    for(int i = 1; i <= n; i++){
        fa[i] = i;
    }
    for(int i = 1; i <= m; i++){
        int u, v;
        cin >> u >> v;
        add(u, v);
        add(v, u);
        merge_set(u, v);
    }
    for(int i = 1; i <= n; i++){
        int fi = find_set(i);
        int root = fi;
        if(!vis[root]){
            tmp.clear();
            dfs(root, -1, 0);

            for(auto i:tmp){
                if(dist[i] > dist[root]) root = i;
                vis[i] = true;
            }
            dfs(root, -1, 0);
            int maxzhi = 0;
            for(auto i:tmp){
                maxzhi = max(maxzhi, dist[i]);
            }
            zhi[fi] = maxzhi;
        }
    }

    for(int i = 1; i <= q; i++){
        int flag;
        cin >> flag;
        if(flag == 1){
            int x;
            cin >> x;
            cout << zhi[find_set(x)] << endl;
        }else{
            int x, y;
            cin >> x >> y;
            int fx = find_set(x), fy = find_set(y);
            if(fx == fy){
                continue;
            }else{
                int new_zhi = max({zhi[fx], zhi[fy], 1 + (zhi[fx]+1)/2 +
(zhi[fy]+1)/2});
                merge_set(x, y);
                int fx = find_set(x);
                zhi[fx] = new_zhi;
            }
        }
    }
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```