提高组6

反转Dag图

题解

二分答案 K ，判断是否为 Dag 图。

如果边权大于 K 的边所组成的图中有环，则只修改边权小于 K 的边无法达到目的。

如果边权大于 K 的边所组成的图中无环，则一定存在对应的方案，使得修改过的图是无环的。

标准代码

C++ 11

```cpp
#include<bits/stdc++.h>
const int N = 1e5 + 5;
int x[N], y[N], w[N], d[N], P[N], vis[N];
std::vector<int> G[N], V, f;
std::queue<int> q;
int n, m, l, r = 1e9, ans, M;
inline bool check(int p) {
    for (int i = 1; i <= m; i++)
        if (w[i] > p) G[x[i]].push_back(y[i]), d[y[i]]++;
    for (int i = 1; i <= n; i++) if (!d[i]) q.push(i);
    int k = 0; V.clear(); while (!q.empty()) {
        int u = q.front(); q.pop(); P[u] = ++k;
        for (int v : G[u]) if ((--d[v]) == 0) q.push(v);
    }
    for (int i = 1; i <= m; i++)
        if (w[i] <= p && P[x[i]] > P[y[i]]) V.push_back(i);
    for (int i = 1; i <= n; i++) d[i] = 0, G[i].clear(), P[i] = 0;
    return k == n;
}
int main() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++)
        scanf("%d%d%d", &x[i], &y[i], &w[i]);
    while (l <= r)
        M = (l + r) >> 1, check(M) ? (r = M - 1, f = V, ans = M) : (l = M + 1);
    printf("%d", ans);
    return 0;
}


```

歪脖子树

题解

对于 Q x 而言，查询的子树无非 2 种可能，即最开始的 x 的子树 T_x ，或 $(T_x - x)$ 的补集。它们分别对应 dfs 序的一个区间，或一个前缀跟后缀的并。线段树维护区间最小值即可。

标准代码

C++

```cpp
#include<iostream>
#include<cstring>
#include<cstdio>
#include<algorithm>
#include<cmath>
#define rg register
#define il inline
#define maxn 200005
#define lid id << 1
#define rid (id << 1) | 1
#define ll long long
using namespace std;

il int read() { rg int x = 0, w = 1; rg char ch = getchar(); while (ch < '0' || ch > '9') { if (ch == '-') w = -1; ch = getchar(); }while (ch >= '0' && ch <= '9') { x = (x << 3) + (x << 1) + ch - '0'; ch = getchar(); }return x * w; }
struct edge {
    int to, next;
}e[maxn << 1];
struct tree {
    int l, r, min;
}t[maxn << 2];
int head[maxn], cnt, val[maxn], f[maxn][22], tot, dep[maxn], q[maxn];
int l[maxn], r[maxn];
void add(int u, int v) {
    e[++cnt].to = v;
    e[cnt].next = head[u];
    head[u] = cnt;
}
void dfs(int u, int fa) {
    l[u] = ++tot, q[tot] = u;
    dep[u] = dep[fa] + 1;
    f[u][0] = fa;
    for (rg int i = 1; i <= 20; ++i)
        f[u][i] = f[f[u][i - 1]][i - 1];
    for (rg int i = head[u]; i; i = e[i].next) {
        rg int to = e[i].to;
        if (to != fa) {
            dfs(to, u);
        }
    }
    r[u] = tot;
}
void pushup(int id) {
    t[id].min = min(t[lid].min, t[rid].min);
}
```

```
void build(int id, int l, int r) {
    t[id].l = l, t[id].r = r;
    if (l == r) { t[id].min = val[q[l]]; return; }
    rg int mid = (l + r) >> 1;
    build(lid, l, mid);
    build(rid, mid + 1, r);
    pushup(id);
}
void update(int id, int pos, int x) {
    if (pos == t[id].l && t[id].l == t[id].r) { t[id].min = x; return; }
    rg int mid = (t[id].l + t[id].r) >> 1;
    if (pos <= mid) update(lid, pos, x);
    else update(rid, pos, x);
    pushup(id);
}
int query(int id, int l, int r) {
    if (l > r) return 1000000000;
    if (t[id].l == l && t[id].r == r) return t[id].min;
    rg int mid = (t[id].l + t[id].r) >> 1;
    if (r <= mid) return query(lid, l, r);
    else if (l > mid) return query(rid, l, r);
    else return min(query(lid, l, mid), query(rid, mid + 1, r));
}
char id[20];
int main() {
    int n = read(), m = read(), fa, x, y;
    for (rg int i = 1; i <= n; ++i) {
        fa = read(), val[i] = read();
        if (fa) add(fa, i);
    }
    int rt = 1;
    dfs(1, 0);
    build(1, 1, n);
    for (rg int i = 1; i <= m; ++i) {
        scanf("%s", id);
        if (id[0] == 'E') rt = read();
        else if (id[0] == 'V') {
            x = read(), y = read();
            update(1, l[x], y);
        }
        else {
            x = read();
            if (x == rt) printf("%d\n", t[1].min);
            else if (l[x] <= l[rt] && r[rt] <= r[x]) {
```

```
                rg int depth = dep[rt] - dep[x] - 1, y = rt;
                for (rg int i = 0; i <= 20; ++i) if (depth & (1 << i)) y = f[y][i];
                printf("%d\n", min(query(1, 1, l[y] - 1), query(1, r[y] + 1, n)));
            }
            else printf("%d\n", query(1, l[x], r[x]));
        }
    }
    return 0;
}


```

倒水问题

题解

首先倒水次数是 $Log(n)$ 级别的。我们先对所有水杯按照水量排序。

然后考虑 DP ， $DP[i][j]$ 表示将前 i 杯水用 j 次倒空，所需要提前移除的最少的水杯数量。

$DP[i][j]$ 从 $DP[i-1][j]$ 转移的话，则需要提前多移除一个水杯。

$DP[i][j]$ 从 $DP[k][j-1]$ 转移的话，则第 k 杯水的水量小于第 i 杯水的一半，且第 $k+1$ 杯水的水量大于等于第 i 杯的一半。 也就是说我们多花费一次倒水的机会，将 $k+1$ 到 i 的水杯全部倒空。

因此完整的状态转移是：

$$DP[i][j] = min(DP[k][j-1], DP[i-1][j])$$

最终我们找出 $DP[n][j] <= k$ 的最大的 j ， 时间复杂度为 $O(nlog(a_i))$ 。

标准代码

C++ 11

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N = 2e5 + 10;
int n, k, a[N];
int dp[N][40], mx[N];//移走i个,答案为j
int find(int x) {
    return upper_bound(a + 1, a + n + 1, a[x] / 2) - a;
}
int main() {
    scanf("%d%d", &n, &k);
    for (int i = 1; i <= n; i++)
        scanf("%d", &a[i]);
    sort(a + 1, a + n + 1);
    for (int i = 0; i <= k; i++)dp[i][0] = n - i + 1;
    for (int j = 1; j <= 40; j++)
    {
        for (int i = 0; i <= k; i++)
            if (dp[i][j - 1] == 1) {
                printf("%d %d", j - 1, i);
                return 0;
            }
        dp[0][j] = find(dp[0][j - 1] - 1);
        for (int i = 1; i <= k; i++)
            dp[i][j] = min(find(dp[i][j - 1] - 1), dp[i - 1][j] - 1);
    }
    return 0;
}


```

树的颜色

题解

首先把整棵树树链剖分，对于每条链使用线段树维护两个标记，第一个标记表示该节点与其父亲的这条边的改变次数的奇偶性，若为 0则改变次数为偶，若为1 则改变次数为奇，记为tag1 ； 第二个标记表示该节点与其非重儿子的边的改变次数的奇偶性，记为tag2 。

容易发现，若一条边为重边，该边所连儿子的 tag1 为1 ，则该边为黑边，否则为白边，若一条边为轻边，该边所连儿子的 tag1 与该边所连父亲的 tag2 相加在对2进行取模，若为1 则该边为黑边，否则为白边

当t=1时，由于只将该条路径的颜色改变，那么只跟 tag1有关，只需把除lca外在此路径上的所有点的tag1异或1。

当 t=2时，我们每跳一条重链，先将该重链上的所有 tag2 异或 1 ，这样就把这条路径上以及该路径所连轻边的颜色都给改变，路径上重链与重链之间相连的轻边同样会改变颜色，所以再使用tag1将颜色改变回来，lca到其父亲的连边，使用tag1改变。

当 t=3时，每跳到一条重链直接把链上除了链顶之外的tag1加起来，链顶单独判断其是否为黑边。

总时间复杂度 $O(n\log^2 n)$

标准代码

C++

```cpp
#include<iostream>
#include<cstdio>
#define N 210000
using namespace std;
int n;
int st[N+1],tot;
struct SEG
{
    int sum[N<<2|1];
    bool tag[N<<2|1];
    void pushup(int p){sum[p]=sum[p<<1]+sum[p<<1|1];}
    void pushdown(int p,int l,int r)
    {
        if(tag[p])
        {
            int mid=(l+r)>>1;
            sum[p<<1]=mid-l+1-sum[p<<1];
            sum[p<<1|1]=r-mid-sum[p<<1|1];
            tag[p<<1]^=1,tag[p<<1|1]^=1;
            tag[p]=false;
        }
    }
    void update(int p,int l,int r,int L,int R)
    {
        if(L<=l&&r<=R)
        {
            sum[p]=r-l+1-sum[p];
            tag[p]^=1;
            return;
        }
        pushdown(p,l,r);
        int mid=(l+r)>>1;
        if(L<=mid)
            update(p<<1,l,mid,L,R);
        if(R>mid)
            update(p<<1|1,mid+1,r,L,R);
        pushup(p);
    }
    int query(int p,int l,int r,int L,int R)
    {
        if(L<=l&&r<=R)
            return sum[p];
        pushdown(p,l,r);
        int mid=(l+r)>>1,res=0;
```

```
        if(L<=mid)
            res+=query(p<<1,l,mid,L,R);
        if(R>mid)
            res+=query(p<<1|1,mid+1,r,L,R);
        return res;
    }
};
SEG t1,t2;
struct edge
{
        int to,last;
}e[N<<1|1];
void add(int a,int b)
{
    e[++tot].to=b;
    e[tot].last=st[a];
    st[a]=tot;
}
int siz[N+1],depth[N+1],fa[N+1],son[N+1],top[N+1];
void dfs1(int u,int f)
{
    fa[u]=f,depth[u]=depth[f]+1,siz[u]=1;
    for(int i=st[u];i!=0;i=e[i].last)
    {
        int v=e[i].to;
        if(v==f)
            continue;
        dfs1(v,u);
        siz[u]+=siz[v];
        if(siz[v]>siz[son[u]])
            son[u]=v;
    }
}
int id[N+1],dfn;
void dfs2(int u,int topf)
{
    top[u]=topf,id[u]=++dfn;
    if(!son[u])
        return;
    dfs2(son[u],topf);
    for(int i=st[u];i!=0;i=e[i].last)
    {
        int v=e[i].to;
        if(v==fa[u]||v==son[u])
```

```
            continue;
        dfs2(v,v);
    }
}
void update1(int x,int y)
{
    while(top[x]!=top[y])
    {
        if(depth[top[x]]<depth[top[y]])
            swap(x,y);
        t1.update(1,1,n,id[top[x]],id[x]);
        x=fa[top[x]];
    }
    if(depth[x]>depth[y])
        swap(x,y);
    if(id[x]!=id[y])
        t1.update(1,1,n,id[x]+1,id[y]);
}
void update2(int x,int y)
{
    while(top[x]!=top[y])
    {
        if(depth[top[x]]<depth[top[y]])
            swap(x,y);
        t2.update(1,1,n,id[top[x]],id[x]);
        t1.update(1,1,n,id[top[x]],id[top[x]]);
        if(son[x])
            t1.update(1,1,n,id[son[x]],id[son[x]]);
        x=fa[top[x]];
    }
    if(depth[x]>depth[y])
        swap(x,y);
    t1.update(1,1,n,id[x],id[x]);
    t2.update(1,1,n,id[x],id[y]);
    if(son[y])
        t1.update(1,1,n,id[son[y]],id[son[y]]);
}
int query(int x,int y)
{
    int res=0;
    while(top[x]!=top[y])
    {
        if(depth[top[x]]<depth[top[y]])
            swap(x,y);
```

```
            if(top[x]!=x)
                res+=t1.query(1,1,n,id[top[x]]+1,id[x]);

        res+=(t1.query(1,1,n,id[fa[top[x]]],id[top[x]])+t2.query(1,1,n,id[fa[top[x]]],id[fa[top[x]]]))&1
        }
        if(depth[x]>depth[y])
            swap(x,y);
        if(id[x]!=id[y])
            res+=t1.query(1,1,n,id[x]+1,id[y]);
        return res;
    }
    int main()
    {
        scanf("%d",&n);
        for(int i=1;i<n;i++)
        {
            int a,b;
            scanf("%d %d",&a,&b);
            add(a,b),add(b,a);
        }
        dfs1(1,0),dfs2(1,1);
        int m;
        scanf("%d",&m);
        for(;m--;)
        {
            int t,a,b;
            scanf("%d %d %d",&t,&a,&b);
            if(t==1)
                update1(a,b);
            else
                if(t==2)
                    update2(a,b);
                else
                    printf("%d\n",query(a,b));
        }
        return 0;
    }
```