Std Code Library(Qinhuangdao)

tingyx

Nanjing University of Science and Technology

October 3, 2023

Contents

一切的开始

Codeforces/XCPC

- 需要 C++17/C++20

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 2e5 + 10;
const ll mod = 998244353;
const ll inf32 = 1e9;
const ll inf64 = 1e18;


void solve(){

}

int main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
// --------------------------------------------------------------------------------
```

int128

- 不要使用 cin/cout，记得关同步流

```cpp
typedef __int128 i128;

i128 read()
{
   i128 x = 0; bool f = 0;
   char c = getchar();
   while (c < '0' || c > '9')
   {
       if (c == '-')
           f = 1;
       c = getchar();
   }
   while (c >= '0' && c <= '9')
   {
       x = (x << 1) + (x << 3) + (c ^ 48);
       c = getchar();
   }
   return f ? -x : x;
}

inline void write(i128 x)
{
   if (x < 0)
       putchar('-'), x = -x;
   if (x > 9)
```

```
        write(x / 10);
    putchar(x % 10 + '0');
}
```

数据结构

二维数点

- 逆序对

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 500010;
ll m;
ll a[maxn], b[maxn], c[maxn];
int lowbit(int x){return x & (-x);}
void add(int x, ll y){
    for (int i = x; i <= m; i += lowbit(i)) c[i] += y;
}
ll sum(int x){
    ll res = 0;
    for (int i = x; i; i -= lowbit(i)) res += c[i];
    return res;
}
int main(){
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i){
        cin >> a[i];
        b[i] = a[i];
    }
    sort(b + 1, b + n + 1);
    m = unique(b + 1, b + n + 1) - b - 1;
    ll ans = 0;
    for (int i = n; i; i--){
        int k = lower_bound(b + 1, b + m + 1, a[i]) - b;
        ans += sum(k - 1);
        add(k, 1);
    }
    cout << ans;
    return 0;
}
```

- 园丁的烦恼 (矩阵内点的个数)

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define pii pair<int, int>
#define vi vector<int>
#define vl vector<ll>
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 1e7 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

struct BIT{
    int tr[maxn];
    int lowbit(int x){return x & -x;}
    void add(int p, int x){
        for (; p < maxn; p += lowbit(p)) tr[p] += x;
    }
    ll query(int p){
        ll sum = 0;
```

```
        for (; p > 0; p -= lowbit(p))
            sum += tr[p];
        return sum;
    }
}Tr;

void solve(){
    int n, m;
    cin >> n >> m;
    vector<pii> pos;
    vector<tuple<int, int, int, int>> q;
    vector<ll> ans(m + 1);
    rep(i, 1, n){
        int tx, ty;
        cin >> tx >> ty;
        tx++, ty++;
        pos.push_back({tx, ty});
    }
    sort(pos.begin(), pos.end());
    rep(i, 1, m){
        int x1, y1, x2, y2;
        cin >> x1 >> y1 >> x2 >> y2;
        x1++, y1++, x2++, y2++;
        q.push_back({x1 - 1, y1 - 1, 1, i});
        q.push_back({x1 - 1, y2, -1, i});
        q.push_back({x2, y1 - 1, -1, i});
        q.push_back({x2, y2, 1, i});
    }
    sort(q.begin(), q.end());
    int cur = 0;
    for (auto [x, y, c, id] : q){
        while (cur < n && pos[cur].first <= x) Tr.add(pos[cur++].second, 1);
        ans[id] += c * Tr.query(y);
    }
    rep(i, 1, m) cout << ans[i] << endl;
}

int main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

- HH 的项链（区间元素种类）照常把 x 所在一维降掉后，发现 y 轴并没有明显的偏序关系。可以这样考虑，我们只计每个元素第一次在区间中出现时有贡献，设 pre[i] 表示位置 i 的元素前一次出现的位置，在整个序列中第一次出现时记为 0

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 1e6 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

struct BIT{
    ll tr[maxn];
    int lowbit(int x){return x & -x;}
    void add(int p, ll x){
```

```
        for (; p < maxn; p += lowbit(p)) tr[p] += x;
    }
    ll query(int p){
        ll sum = 0;
        for (; p > 0; p -= lowbit(p))
            sum += tr[p];
        return sum;
    }
}Tr;

ll pre[maxn], ans[maxn];
void solve(){
    int n, m;
    cin >> n;
    vector<pll> pos;
    vector<tuple<int, int, int, int>> q;
    for (int i = 3; i <= n + 2; ++i){
        int a;
        cin >> a;
        pos.push_back({i, pre[a] ? pre[a] : 2}), pre[a] = i;
    }
    sort(pos.begin(), pos.end());
    cin >> m;
    for (int i = 1; i <= m; ++i){
        int l, r;
        cin >> l >> r;
        l += 2, r += 2;
        q.push_back({l - 1, 1, 1, i});
        q.push_back({l - 1, l - 1, -1, i});
        q.push_back({r, 1, -1, i});
        q.push_back({r, l - 1, 1, i});
    }
    sort(q.begin(), q.end());
    int cur = 0;
    for (auto [x, y, c, id] : q)
    {
        while (cur < n && pos[cur].first <= x)
            Tr.add(pos[cur++].second, 1);
        ans[id] += c * Tr.query(y);
    }
    for (int i = 1; i <= m; i++) cout << ans[i] << endl;
}

int main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

- 矩阵内权值之和

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 3e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

```

```cpp
struct BIT{
    ll tr[maxn];
    int lowbit(int x){return x & -x;}
    void add(int p, ll x){
        for (; p < maxn; p += lowbit(p)) tr[p] += x;
    }
    ll query(int p){
        ll sum = 0;
        for (; p > 0; p -= lowbit(p))
            sum += tr[p];
        return sum;
    }
}Tr;

void solve(){
    int n, m;
    cin >> n >> m;
    vector<tuple<int, int, int>> pos;
    vector<tuple<int, int, int, int>> q;
    vector<ll> ans(m + 1);
    vector<int> yy;
    rep(i, 1, n){
        int x, y, p;
        cin >> x >> y >> p;
        yy.push_back(y);
        pos.push_back({x, y, p});
    }
    sort(pos.begin(), pos.end());
    rep(i, 1, m){
        int x1, y1, x2, y2;
        cin >> x1 >> y1 >> x2 >> y2;
        yy.push_back(y1 - 1), yy.push_back(y2);
        q.push_back({x1 - 1, y1 - 1, 1, i});
        q.push_back({x2, y1 - 1, -1, i});
        q.push_back({x1 - 1, y2, -1, i});
        q.push_back({x2, y2, 1, i});
    }
    sort(q.begin(), q.end());
    sort(yy.begin(), yy.end());
    yy.erase(unique(yy.begin(), yy.end()), yy.end());
    int cur = 0;
    for (auto [x, y, c, id] : q){
        y = lower_bound(yy.begin(), yy.end(), y) - yy.begin() + 1;
        while (cur < n){
            auto [_x, _y, p] = pos[cur];
            if (_x > x) break;
            _y = lower_bound(yy.begin(), yy.end(), _y) - yy.begin() + 1;
            Tr.add(_y, p), ++cur;
        }
        ans[id] += c * Tr.query(y);
    }
    for (int i = 1; i <= m; ++i) cout << ans[i] << endl;
}

int main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

可持续化线段树

- HH 的项链

求区间内不重复的数的个数。扫描数列建立可持久化线段树，第 i 个数若第一次出现，则在线段树中的位置 i 加 1；若不是第一次出现，将上次出现的位置减 1，在本次位置加 1。对于每个询问的区间 [L,R]，在第 R 个版本上的线段树只有前 R 个数，在线段树上查询位置 L，对经过的区间中的和进行累计即可。

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 1e6 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

struct node{
    int ls, rs;
    int cnt;
}tr[maxn << 5];
int idx = 0, rt[maxn];

void push_up(int u){
    tr[u].cnt = tr[tr[u].ls].cnt + tr[tr[u].rs].cnt;
}

int build(int l, int r){
    int u = idx++;
    if (l == r){
        tr[u].cnt = 0;
        return u;
    }
    int mid = l + r >> 1;
    tr[u].ls = build(l, mid);
    tr[u].rs = build(mid + 1, r);
    push_up(u);
    return u;
}

int update(int old, int l, int r, int pos, int val){
    int u = idx++;
    tr[u] = tr[old];
    if (l == pos && r == pos){
        tr[u].cnt += val;
        return u;
    }
    int mid = l + r >> 1;
    if (pos <= mid) tr[u].ls = update(tr[old].ls, l, mid, pos, val);
    else tr[u].rs = update(tr[old].rs, mid + 1, r, pos, val);
    push_up(u);
    return u;
}

int query(int l, int r, int ver, int pos){
    if (l == r) return tr[ver].cnt;
    int mid = l + r >> 1;
    if (pos <= mid) return tr[tr[ver].rs].cnt + query(l, mid, tr[ver].ls, pos);
    else return query(mid + 1, r, tr[ver].rs, pos);
}

int b[maxn];
map<int, int> mp;
void solve(){
    int n, m;
    cin >> n;
    rt[0] = build(1, n);
```

```
    for (int i = 1; i <= n; ++i) cin >> b[i];
    for (int i = 1; i <= n; ++i){
        if (mp.find(b[i]) == mp.end()){
            mp[b[i]] = i;
            rt[i] = update(rt[i - 1], 1, n, i, 1);
        }else {
            int tmp = update(rt[i - 1], 1, n, mp[b[i]], -1);
            rt[i] = update(tmp, 1, n, i, 1);
        }
        mp[b[i]] = i;
    }
    cin >> m;
    for (int i = 1; i <= m; ++i){
        int l, r;
        cin >> l >> r;
        cout << query(1, n, rt[r], l) << endl;
    }
}

int main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

区间问题

莫队

- 区间取两个数相同概率

```
#include <algorithm>
#include <cmath>
#include <cstdio>
using namespace std;
const int N = 50005;
int n, m, maxn;
int c[N];
long long sum;
int cnt[N];
long long ans1[N], ans2[N];

struct query {
  int l, r, id;

  bool operator<(const query &x) const {  // 重载 < 运算符
    if (l / maxn != x.l / maxn) return l < x.l;
    return (l / maxn) & 1 ? r < x.r : r > x.r;
  }
} a[N];

void add(int i) {
  sum += cnt[i];
  cnt[i]++;
}

void del(int i) {
  cnt[i]--;
  sum -= cnt[i];
}

long long gcd(long long a, long long b) { return b ? gcd(b, a % b) : a; }

int main() {
```

```
  scanf("%d%d", &n, &m);
  maxn = sqrt(n);
  for (int i = 1; i <= n; i++) scanf("%d", &c[i]);
  for (int i = 0; i < m; i++) scanf("%d%d", &a[i].l, &a[i].r), a[i].id = i;
  sort(a, a + m);
  for (int i = 0, l = 1, r = 0; i < m; i++) {  // 具体实现
    if (a[i].l == a[i].r) {
      ans1[a[i].id] = 0, ans2[a[i].id] = 1;
      continue;
    }
    while (l > a[i].l) add(c[--l]);
    while (r < a[i].r) add(c[++r]);
    while (l < a[i].l) del(c[l++]);
    while (r > a[i].r) del(c[r--]);
    ans1[a[i].id] = sum;
    ans2[a[i].id] = (long long)(r - l + 1) * (r - l) / 2;
  }
  for (int i = 0; i < m; i++) {
    if (ans1[i] != 0) {
      long long g = gcd(ans1[i], ans2[i]);
      ans1[i] /= g, ans2[i] /= g;
    } else
      ans2[i] = 1;
    printf("%lld/%lld\n", ans1[i], ans2[i]);
  }
  return 0;
}
```

CDQ

- 逆序对

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 2e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

void solve(){
    int n;
    cin >> n;
    vi a(n + 1), temp(n + 1);
    ll ans = 0;
    rep(i, 1, n) cin >> a[i];
    function<void(int, int)> cdq = [&](int l, int r){
        if (l == r) return;
        int mid = l + r >> 1;
        cdq(l, mid);
        cdq(mid + 1, r);
        int p1 = l, p2 = mid + 1, idx = l;
        while (p1 <= mid && p2 <= r){
            if (a[p1] > a[p2]) temp[idx++] = a[p1++];
            else temp[idx++] = a[p2++], ans += p1 - l;
        }
        while (p1 <= mid) temp[idx++] = a[p1++];
        while (p2 <= r) temp[idx++] = a[p2++], ans += p1 - l;
        for (int i = l; i <= r; ++i) a[i] = temp[i];
    };
    cdq(1, n);
    cout << ans << endl;
```

```
}

int main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

- 求最长不上升子序列和最长上升子序列

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 100005;
int n, x, dp[MAXN], a[MAXN], ans;
pair<int, int> temp[MAXN][20]; //val, pos

bool cmp(const pair<int, int> &A, const pair<int, int> &B, const int &type) {
    return type ? A.first != B.first ? A.first > B.first : A.second < B.second : A.first != B.first ? A.first <
    ↪ B.first: A.second > B.second;
}

void mergeSort(int l, int r, int deep, const int &cmptype) {
    if (l == r) {
        temp[l][deep].first = a[l];
        temp[l][deep].second = l;
        return;
    }
    int mid = (l + r) >> 1;
    mergeSort(l, mid, deep + 1, cmptype);
    mergeSort(mid + 1, r, deep + 1, cmptype);
    int p1 = l, p2 = mid + 1;
    while (p1 <= mid && p2 <= r) {
        if (cmp(temp[p1][deep + 1], temp[p2][deep + 1], cmptype)) {
            temp[l++][deep] = temp[p1++][deep + 1];
        } else {
            temp[l++][deep] = temp[p2++][deep + 1];
        }
    }
    while (p1 <= mid) {
        temp[l++][deep] = temp[p1++][deep + 1];
    }
    while (p2 <= r) {
        temp[l++][deep] = temp[p2++][deep + 1];
    }
}

void cdqDivAlgorithm(int l, int r, int deep, const int &cmptype) {
    if (l == r) {
        dp[l] = max(dp[l], 1);
        ans = max(ans, dp[l]);
        return;
    }
    int mid = (l + r) >> 1;
    cdqDivAlgorithm(l, mid, deep + 1, cmptype);
    int p1 = l, p2 = mid + 1, premax = 0;
    while (p1 <= mid && p2 <= r) {
        if (cmp(temp[p1][deep + 1], temp[p2][deep + 1], cmptype)) {
            premax = max(premax, dp[temp[p1++][deep + 1].second]);
        } else {
            dp[temp[p2][deep + 1].second] = max(premax + 1, dp[temp[p2][deep + 1].second]);
            p2++;
        }
    }
    while (p2 <= r) {
        dp[temp[p2][deep + 1].second] = max(premax + 1, dp[temp[p2][deep + 1].second]);
        p2++;
```

```
    }
    cdqDivAlgorithm(mid + 1, r, deep + 1, cmptype);
}

int main()
{
    while (scanf("%d", &x) != EOF)a[++n] = x;
    mergeSort(1, n, 0, 1);
    cdqDivAlgorithm(1, n, 0, 1);
    printf("%d\n", ans);
    memset(dp, 0, sizeof(dp));
    ans = 0;
    mergeSort(1, n, 0, 0);
    cdqDivAlgorithm(1, n, 0, 0);
    printf("%d\n", ans);
    return 0;
}
```

- 求地毯覆盖（最多取多少个不相互覆盖）

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 1000005;
int n, L[MAXN], R[MAXN], id[MAXN], dp[MAXN], ans;
int temp[MAXN];
void cdqDivAlgorithm(int l, int r) {
    if (l == r) {
        dp[id[l]] = max(1, dp[id[l]]);
        ans = max(ans, dp[id[l]]);
        return;
    }
    int mid = (l + r) >> 1;
    cdqDivAlgorithm(l, mid);
    int p1 = l, pl, p2 = mid + 1, premax = 0;
    while (p1 <= mid && p2 <= r) {
        if (R[id[p1]] <= L[id[p2]]) {
            premax = max(premax, dp[id[p1++]]);
        } else {
            dp[id[p2]] = max(premax + 1, dp[id[p2]]);
            ++p2;
        }
    }
    while (p2 <= r) {
        dp[id[p2]] = max(premax + 1, dp[id[p2]]);
        ++p2;
    }
    cdqDivAlgorithm(mid + 1, r);
    p1 = l, pl = l, p2 = mid + 1;
    while (p1 <= mid && p2 <= r) {
        if (R[id[p1]] < R[id[p2]]) {
            temp[pl++] = id[p1++];
        } else {
            temp[pl++] = id[p2++];
        }
    }
    while (p1 <= mid) {
        temp[pl++] = id[p1++];
    }
    while (p2 <= r) {
        temp[pl++] = id[p2++];
    }
    for (int i = l; i <= r; ++i) {
        id[i] = temp[i];
    }
}
int main()
{
    scanf("%d", &n);
    for (int i = 1; i <= n; ++i) {
        scanf("%d %d", &L[i], &R[i]);
        id[i] = i;
    }
```

```
    sort(id + 1, id + 1 + n, [](const int &A, const int &B) {
        return L[A] < L[B];
    });
    cdqDivAlgorithm(1, n);
    printf("%d\n", ans);
    return 0;
}
```

- 动态凸包

第一行：一个整数 N，表示方案和询问的总数。接下来 N 行，每行开头一个单词 “Query” 或 “Project”。若单词为 Query，则后接一个整数 T，表示 Blue Mary 询问第 T 天的最大收益。若单词为 Project，则后接两个实数 S，P，表示该种设计方案第一天的收益 S，以及以后每天比上一天多出的收益 P。对于每一个 Query，输出一个整数，表示询问的答案，并精确到整百元 1 <= N <= 100000 1 <= T <=50000 0 < P < 100，|S|<= 10^6

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 100005;
const double eps = 1e-6;
int m, n, id[MAXN], qid[MAXN], type[MAXN], x[MAXN], temp[MAXN], top;
double k[MAXN], b[MAXN], ans[MAXN];
char op[55];
inline bool cmp(const int &A, const int &B) {
    return type[A] != type[B] ? type[A] < type[B] : type[A] ? x[A] < x[B] : k[A] < k[B];
}
inline int dcmp(double x) {
    return x > eps ? 1 : x < -eps ? -1 : 0;
}
inline double getCross(const double &k1, const double &b1, const double &k2, const double &b2) {
    return (b2 - b1) / (k1 - k2);
}
inline double getVal(const double &k, const double &b, const int &x)
{
    return k * x + b;
}
pair<double, double>stk[MAXN];
void stkClear() {
    top = 0;
    stk[++top] = make_pair(0, 0);
}
void stkInsert(double k, double b) {
    if (dcmp(stk[top].first - k) == 0 && dcmp(stk[top].second - b) < 0)top--;
    if (dcmp(stk[top].first - k) == 0 && dcmp(stk[top].second - b) >= 0)return;
    while (top >= 2 && dcmp(getCross(stk[top].first, stk[top].second, stk[top - 1].first, stk[top - 1].second) -
 ↪  getCross(stk[top].first, stk[top].second, k, b)) > 0)top--;
    stk[++top] = make_pair(k, b);
}
double stkQuery(int x) {
    while (top >= 2 && dcmp(getVal(stk[top].first, stk[top].second, x) - getVal(stk[top - 1].first, stk[top -
 ↪  1].second, x)) < 0)--top;
    return getVal(stk[top].first, stk[top].second, x);
}
void cdqDivAlgorithm(int l, int r) {
    if (l == r)return;
    int mid = (l + r) >> 1;
    cdqDivAlgorithm(l, mid);
    cdqDivAlgorithm(mid + 1, r);
    stkClear();
    for (int i = l; i <= mid && !type[id[i]]; ++i) {
        stkInsert(k[id[i]], b[id[i]]);
    }
    for (int i = r; i > mid && type[id[i]]; --i) {
        ans[qid[id[i]]] = max(ans[qid[id[i]]], stkQuery(x[id[i]]));
    }
    int p1 = l, pl = l, p2 = mid + 1;
    while (p1 <= mid && p2 <= r) {
        if (cmp(id[p1], id[p2])) {
            temp[pl++] = id[p1++];
        } else {
            temp[pl++] = id[p2++];
        }
```

```
    }
    while (p1 <= mid) {
        temp[pl++] = id[p1++];
    }
    while (p2 <= r) {
        temp[pl++] = id[p2++];
    }
    for (int i = l; i <= r; ++i) {
        id[i] = temp[i];
    }
}
int main() {
    scanf("%d", &n);
    for (int i = 1; i <= n; ++i) {
        id[i] = i;
        scanf("%s", op);
        if (*op == 'P') {
            type[i] = 0;
            scanf("%lf %lf", &b[i], &k[i]);
            b[i] -= k[i];
        }
        else {
            type[i] = 1;
            qid[i] = ++m;
            scanf("%d", &x[i]);
        }
    }
    cdqDivAlgorithm(1, n);
    for (int i = 1; i <= m; ++i) {
        printf("%d\n", (int)ans[i] / 100);
    }
    return 0;
}

```

树上问题

树剖

- 2018ICPC 青岛网络赛（多测时候用来剖的）

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j, k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 1e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

void solve()
{
    int n, m, q, k, cnt = 0;
    cin >> n >> m >> q;
    vi red(n + 1);
    vector<vector<pll>> G(n + 1);
    vl dis(n + 1), dep(n + 1), v(n + 1);
    vi dfn(n + 1), idx(n + 1);
    vi son(n + 1, -1), sz(n + 1), fa(n + 1), top(n + 1);
    function<void(int, int)> dfs1 = [&](int u, int f) {
        son[u] = -1;
        sz[u] = 1;
```

```
        if(!red[u])
            red[u] = red[f];
        for(auto [v, w] : G[u]) {
            if(v == f)
                continue;
            dep[v] = dep[u] + 1;
            dis[v] = dis[u] + w;
            fa[v] = u;
            dfs1(v, u);
            sz[u] += sz[v];
            if(son[u] == -1 || sz[v] > sz[son[u]])
                son[u] = v;
        }
    };
    function<void(int, int)> dfs2 = [&](int u, int t) {
        top[u] = t;
        dfn[u] = ++cnt;
        idx[cnt] = u;
        if(son[u] == -1)
            return;
        dfs2(son[u], t);
        for(auto [v, w] : G[u])
            if(v != son[u] && v != fa[u])
                dfs2(v, v);
    };
    auto lca = [&](int u, int v) {
        while(top[u] != top[v]) {
            if(dep[top[u]] > dep[top[v]])
                u = fa[top[u]];
            else
                v = fa[top[v]];
        }
        return dep[u] > dep[v] ? v : u;
    };
    for(int i = 1, x; i <= m; ++i)
        cin >> x, red[x] = x;
    for(int i = 1; i < n; ++i) {
        int u, v, w;
        cin >> u >> v >> w;
        G[u].push_back({v, w});
        G[v].push_back({u, w});
    }
    dfs1(1, 0);
    dfs2(1, 1);
    for(int i = 1; i <= n; ++i)
        v[i] = dis[i] - dis[red[i]];
    while(q--) {
        cin >> k;
        vector<int> p(k + 1);
        auto check = [&](ll st) {
            vector<int> q;
            for(int i = 1; i <= k; ++i)
                if(v[p[i]] > st)
                    q.push_back(p[i]);
            if(q.size() == 0)
                return true;
            int mnd = n + 1, mxd = 0;
            for(int i = 0; i < q.size(); ++i) {
                mnd = min(mnd, dfn[q[i]]);
                mxd = max(mxd, dfn[q[i]]);
            }
            int ca = lca(idx[mnd], idx[mxd]);
            for(int i = 0; i < q.size(); ++i)
                if(dis[q[i]] - dis[ca] > st)
                    return false;
            return true;
        };
        ll mx = 0;
        for(int i = 1; i <= k; ++i) {
            cin >> p[i];
            mx = max(mx, v[p[i]]);
```

```
        }
        ll l = 0, r = mx;
        while(l < r) {
            ll mid = (l + r) >> 1;
            if(check(mid))
                r = mid;
            else
                l = mid + 1;
        }
        cout << l << endl;
    }
}

int main()
{
    ios;
    // freopen("sample.txt", "r", stdin);
    // freopen("resout.txt", "w", stdout);
    int t = 1;
    cin >> t;
    while(t--) {
        solve();
    }
    return 0;
}
```

- 树上操作

1. 节点 x 加上 a
2. 节点 x 的子树中所有点的点权加 a
3. 询问某个点 x 到根节点

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 2e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;
const int N = 1e5 + 10, M = N * 2;

int n, m;
// w 为节点权值
int h[N], w[N], e[M], ne[M], idx;
// id[x] 为节点 x 的新编号, nw[x] 是新编号为 x 的节点的权值
int id[N], nw[N], cnt;
// dep 为深度, sz 为子树大小, top[x] 是 x 所在重链的头结点,
// fa[x] 为 x 父亲, son[x] 为 x 的重儿子
int dep[N], sz[N], top[N], fa[N], son[N];
struct Tree {
    int l, r;
    ll sum, add;
} tr[N << 2];

void add(int a, int b) {
    e[idx] = b, ne[idx] = h[a], h[a] = idx++;
}

// 第一次 dfs, 求节点深度、父亲、子树大小和重儿子
void dfs1(int u, int from, int depth) {
    dep[u] = depth, fa[u] = from, sz[u] = 1;
```

```
    for (int i = h[u]; ~i; i = ne[i]) {
        int v = e[i];
        if (v == from) continue;
        dfs1(v, u, depth + 1);
        sz[u] += sz[v];
        if (sz[son[u]] < sz[v]) son[u] = v;
    }
}

// 第二次 dfs, t 为 u 重链头结点
void dfs2(int u, int t) {
    id[u] = ++cnt, nw[cnt] = w[u], top[u] = t;
    // 到叶子了, 直接返回
    if (!son[u]) return;
    // 先遍历重儿子
    dfs2(son[u], t);
    // 遍历轻儿子
    for (int i = h[u]; ~i; i = ne[i]) {
        int v = e[i];
        if (v == fa[u] || v == son[u]) continue;
        dfs2(v, v);
    }
}

void pushup(int u) {
    tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
}

void pushdown(int u) {
    auto &root = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
    if (root.add) {
        left.sum += root.add * (left.r - left.l + 1);
        left.add += root.add;
        right.sum += root.add * (right.r - right.l + 1);
        right.add += root.add;
        root.add = 0;
    }
}

void build(int u, int l, int r) {
    tr[u] = {l, r, nw[l], 0};
    if (l == r) return;
    int mid = l + r >> 1;
    build(u << 1, l, mid), build(u << 1 | 1, mid + 1, r);
    pushup(u);
}

void update(int u, int l, int r, ll k) {
    if (l <= tr[u].l && tr[u].r <= r) {
        tr[u].add += k;
        tr[u].sum += k * (tr[u].r - tr[u].l + 1);
        return;
    }
    pushdown(u);
    int mid = tr[u].l + tr[u].r >> 1;
    if (l <= mid) update(u << 1, l, r, k);
    if (r > mid) update(u << 1 | 1, l, r, k);
    pushup(u);
}

ll query(int u, int l, int r) {
    if (l <= tr[u].l && tr[u].r <= r) return tr[u].sum;
    pushdown(u);
    int mid = tr[u].l + tr[u].r >> 1;
    ll res = 0;
    if (l <= mid) res += query(u << 1, l, r);
    if (r > mid) res += query(u << 1 | 1, l, r);
    return res;
}

void update_path(int u, int v, ll k) {
```

```cpp
    while (top[u] != top[v]) {
        if (dep[top[u]] < dep[top[v]]) swap(u, v);
        // u 的重链头更深，并且 u 重链头在 dfs 序里下标更小，直接更新 u 重链头到 u 这段区间
        update(1, id[top[u]], id[u], k);
        // u 跳到重链头上面
        u = fa[top[u]];
    }
    if (dep[u] < dep[v]) swap(u, v);
    update(1, id[v], id[u], k);
}

ll query_path(int u, int v) {
    ll res = 0;
    while (top[u] != top[v]) {
        if (dep[top[u]] < dep[top[v]]) swap(u, v);
        res += query(1, id[top[u]], id[u]);
        u = fa[top[u]];
    }
    if (dep[u] < dep[v]) swap(u, v);
    res += query(1, id[v], id[u]);
    return res;
}

void update_tree(int u, ll k) {
    update(1, id[u], id[u] + sz[u] - 1, k);
}

ll query_tree(int u) {
    return query(1, id[u], id[u] + sz[u] - 1);
}

void solve() {
    int n, q;
    memset(h, -1, sizeof h);
    cin >> n >> q;
    int cnt = 0;
    for (int i = 1; i <= n; ++i) cin >> w[i];
    for (int i = 1; i <= n - 1; ++i) {
        int u, v;
        cin >> u >> v;
        add(u, v);
        add(v, u);
    }
    dfs1(1, 1, 0);
    dfs2(1, 1);
    build(1, 1, n);
    while (q--) {
        int t, u;
        ll k;
        cin >> t >> u;
        if (t == 1) {
            cin >> k;
            update_path(u, u, k);
        } else if (t == 2) {
            cin >> k;
            update_tree(u, k);
        } else cout << query_path(1, u) << endl;
    }
}

int main() {
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

- 树上路径

1. 将以 u 为根的子树内节点 (包括 u) 的权值加 val
2. 将 (u, v) 路径上的节点权值加 val
3. 询问 (u, v) 路径上节点的权值两两相乘的和

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define int ll
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll mod = 1e9 + 7;
const ll inf = 0x3f3f3f3f;
const int N = 1e5 + 10, M = N * 2;

int n, m;
int h[N], a[N], e[M], ne[M], idx;
int id[N], cnt, rnk[N];
int dep[N], sz[N], top[N], fa[N], son[N];
ll inv2;

void add(int u, int v) {
    e[idx] = v, ne[idx] = h[u], h[u] = idx++;
}
ll qmi(ll x, ll k) {
    ll res = 1;
    while (k) {
        if (k & 1) res = res * x % mod;
        x = x * x % mod;
        k >>= 1;
    }
    return res;
}
struct Segment {
    struct Node {
        int l, r;
        ll sum, psum, add;
    } tr[N * 4];
    void pushup(int u) {
        tr[u].sum = (tr[u << 1].sum + tr[u << 1 | 1].sum) % mod;
        tr[u].psum = (tr[u << 1].psum + tr[u << 1 | 1].psum) % mod;
        return;
    }
    void pushdown(Node& u, Node& l, Node& r) {
        if (u.add) {
            ll x = u.add;
            l.psum = (l.psum + 2 * l.sum * x % mod + (ll)x * x % mod * (l.r - l.l + 1) % mod) % mod;
            r.psum = (r.psum + 2 * r.sum * x % mod + (ll)x * x % mod * (r.r - r.l + 1) % mod) % mod;
            l.sum = (l.sum + (ll)x * (l.r - l.l + 1) % mod) % mod;
            r.sum = (r.sum + (ll)x * (r.r - r.l + 1) % mod) % mod;
            l.add = (l.add + x) % mod;
            r.add = (r.add + x) % mod;
            u.add = 0;
        }
        return;
    }
    void pushdown(int u) {
        pushdown(tr[u], tr[u << 1], tr[u << 1 | 1]);
    }
    void build(int u, int l, int r) {
        tr[u] = {l, r};
        if (l == r) {
```

```
            tr[u].sum = a[rnk[l]];
            tr[u].psum = (ll)a[rnk[l]] * a[rnk[l]] % mod;
            return;
        }
        int mid = (l + r) >> 1;
        build(u << 1, l, mid);
        build(u << 1 | 1, mid + 1, r);
        pushup(u);
        return;
    }
    void update(int u, int l, int r, ll x) {
        if (l <= tr[u].l && tr[u].r <= r) {
           tr[u].psum = (tr[u].psum + 2 * tr[u].sum * x % mod + (ll)x * x % mod * (tr[u].r - tr[u].l + 1) % mod) % mod;
            tr[u].sum = (tr[u].sum + (ll)(tr[u].r - tr[u].l + 1) * x % mod) % mod;
            tr[u].add = (tr[u].add + x) % mod;
            return;
        }
        pushdown(u);
        int mid = (tr[u].l + tr[u].r) >> 1;
        if (l <= mid) update(u << 1, l, r, x);
        if (mid < r) update(u << 1 | 1, l, r, x);
        pushup(u);
        return;
    }
    ll query_sum(int u, int l, int r) {
        if (l <= tr[u].l && tr[u].r <= r) return tr[u].sum;
        pushdown(u);
        int mid = (tr[u].l + tr[u].r) >> 1;
        ll res = 0;
        if (l <= mid) res = (res + query_sum(u << 1, l, r)) % mod;
        if (mid < r) res = (res + query_sum(u << 1 | 1, l, r)) % mod;
        return res;
    }
    ll query_psum(int u, int l, int r) {
        if (l <= tr[u].l && tr[u].r <= r) return tr[u].psum;
        pushdown(u);
        int mid = (tr[u].l + tr[u].r) >> 1;
        ll res = 0;
        if (l <= mid) res = (res + query_psum(u << 1, l, r)) % mod;
        if (mid < r) res = (res + query_psum(u << 1 | 1, l, r)) % mod;
        return res;
    }
} Tr;

//Tree
void dfs1(int u, int from, int depth) {
    dep[u] = depth, fa[u] = from, sz[u] = 1;
    for (int i = h[u]; ~i; i = ne[i]) {
        int v = e[i];
        if (v == from) continue;
        dfs1(v, u, depth + 1);
        sz[u] += sz[v];
        if (sz[son[u]] < sz[v]) son[u] = v;
    }
}
void dfs2(int u, int t) {
    id[u] = ++cnt, top[u] = t;
    rnk[cnt] = u;
    if (!son[u]) return;
    dfs2(son[u], t);
    for (int i = h[u]; ~i; i = ne[i]) {
        int v = e[i];
        if (v == fa[u] || v == son[u]) continue;
        dfs2(v, v);
    }
}
void update_path(int u, int v, ll k) {//更新路径
    while (top[u] != top[v]) {
        if (dep[top[u]] < dep[top[v]]) swap(u, v);
        Tr.update(1, id[top[u]], id[u], k);
        u = fa[top[u]];
```

```
    }
    if (dep[u] < dep[v]) swap(u, v);
    Tr.update(1, id[v], id[u], k);
}
ll query_path(int u, int v) {
    ll res_sum = 0, res_psum = 0;
    while (top[u] != top[v]) {
        if (dep[top[u]] < dep[top[v]]) swap(u, v);
        res_sum = (res_sum + Tr.query_sum(1, id[top[u]], id[u])) % mod;
        res_psum = (res_psum + Tr.query_psum(1, id[top[u]], id[u])) % mod;
        u = fa[top[u]];
    }
    if (dep[u] < dep[v]) swap(u, v);
    res_sum = (res_sum + Tr.query_sum(1, id[v], id[u])) % mod;
    res_psum = (res_psum + Tr.query_psum(1, id[v], id[u])) % mod;
    return (res_sum * res_sum % mod - res_psum + mod) % mod * inv2 % mod;
}
//Tree

void solve() {
    inv2 = qmi(2, mod - 2);
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> a[i];
    memset(h, -1, sizeof h);
    for (int i = 1; i <= n - 1; ++i) {
        int u, v;
        cin >> u >> v;
        add(u, v);
        add(v, u);
    }
    dfs1(1, 0, 1);
    dfs2(1, 1);
    Tr.build(1, 1, n);

    while (m--) {
        int op;
        cin >> op;
        ll u, v, k;
        if (op == 1) {
            cin >> u >> k;
            Tr.update(1, id[u], id[u] + sz[u] - 1, k);
        } else if (op == 2) {
            cin >> u >> v >> k;
            update_path(u, v, k);
        } else {
            cin >> u >> v;
            cout << query_path(u, v) << endl;
        }
    }
}

signed main() {
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

dsu

- 树上数颜色

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

const int N = 2e5 + 5;
```

```cpp

int n;

// g[u]: 存储与 u 相邻的结点
vector<int> g[N];

// sz: 子树大小
// big: 重儿子
// col: 结点颜色
// L[u]: 结点 u 的 DFS 序
// R[u]: 结点 u 子树中结点的 DFS 序的最大值
// Node[i]: DFS 序为 i 的结点
// ans: 存答案
// cnt[i]: 颜色为 i 的结点个数
// totColor: 目前出现过的颜色个数
int sz[N], big[N], col[N], L[N], R[N], Node[N], totdfn;
int ans[N], cnt[N], totColor;

void add(int u) {
  if (cnt[col[u]] == 0) ++totColor;
  cnt[col[u]]++;
}

void del(int u) {
  cnt[col[u]]--;
  if (cnt[col[u]] == 0) --totColor;
}

int getAns() { return totColor; }

void dfs0(int u, int p) {
  L[u] = ++totdfn;
  Node[totdfn] = u;
  sz[u] = 1;
  for (int v : g[u])
    if (v != p) {
      dfs0(v, u);
      sz[u] += sz[v];
      if (!big[u] || sz[big[u]] < sz[v]) big[u] = v;
    }
  R[u] = totdfn;
}

void dfs1(int u, int p, bool keep) {
  // 计算轻儿子的答案
  for (int v : g[u])
    if (v != p && v != big[u]) {
      dfs1(v, u, false);
    }
  // 计算重儿子答案并保留计算过程中的数据（用于继承）
  if (big[u]) {
    dfs1(big[u], u, true);
  }
  for (int v : g[u])
    if (v != p && v != big[u]) {
      // 子树结点的 DFS 序构成一段连续区间，可以直接遍历
      for (int i = L[v]; i <= R[v]; i++) {
        add(Node[i]);
      }
    }
  add(u);
  ans[u] = getAns();
  if (keep == false) {
    for (int i = L[u]; i <= R[u]; i++) {
      del(Node[i]);
    }
  }
}

int main() {
  scanf("%d", &n);
```

```
  for (int i = 1; i <= n; i++) scanf("%d", &col[i]);
  for (int i = 1; i < n; i++) {
    int u, v;
    scanf("%d%d", &u, &v);
    g[u].push_back(v);
    g[v].push_back(u);
  }
  dfs0(1, 0);
  dfs1(1, 0, false);
  for (int i = 1; i <= n; i++) printf("%d%c", ans[i], " \n"[i == n]);
  return 0;
}
```

- 子树权值不大于 k 的数量

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll N = 1e6 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;
int tr[N];
int h[N], to[2 * N], ne[2 * N], cnt;
int sz[N], dep[N], fa[N], son[N];
int top[N], dfn, L[N], R[N], idx[N], skp;
int a[N], sum, ans[N];
int n, m;

vector<pll> q[N];
int lowbit(int x) {return x & -x;}
void add(int p, int k){for (int i = p; i < N; i += lowbit(i)) tr[i] += k;}
int query(int p){int res = 0; for (int i = p; i; i -= lowbit(i)) res += tr[i]; return res;}

void addedge(int u, int v){
    to[++cnt] = v;
    ne[cnt] = h[u];
    h[u] = cnt;
}

void dfs1(int u, int f){
    sz[u] = 1;
    dep[u] = dep[f] + 1;
    fa[u] = f;
    for (int i = h[u]; i; i = ne[i]){
        int v = to[i];
        if (v == f) continue;
        dfs1(v, u);
        sz[u] += sz[v];
        if (!son[u] || sz[son[u]] < sz[v]) son[u] = v;
    }
}

void dfs2(int u, int t){
    L[u] = ++dfn;
    idx[dfn] = u;
    top[u] = t;
    if (son[u]) dfs2(son[u], t);
    for (int i = h[u]; i; i = ne[i]){
        int v = to[i];
        if (v != fa[u] && v != son[u])
            dfs2(v, v);
    }
```

```
    R[u] = dfn;
}

void get(int u, int op){
    for (int i = L[u]; i <= R[u]; ++i){
        if (idx[i] == skp){i = R[idx[i]]; continue;}
        add(a[idx[i]], op);
    }
    if (op == -1) return;
    for (auto x : q[u]) ans[x.second] = query(x.first);
}

void dsu(int u){
    for (int i = h[u]; i; i = ne[i]){
        int v = to[i];
        if (v == fa[u] || v == son[u]) continue;
        dsu(v);
    }
    if (son[u]) {dsu(son[u]), skp = son[u];}
    get(u, 1);
    if (u == top[u]){
        skp = 0;
        get(u, -1);
    }
}
void solve() {
    cin >> n;
    rep(i, 1, n) cin >> a[i];
    int u, v;
    for (int i = 1; i <= n - 1; ++i){
        cin >> u >> v;
        addedge(u, v);
        addedge(v, u);
    }
    cin >> m;
    int x, k;
    rep(i, 1, m){
        cin >> x >> k;
        q[x].push_back({k, i});
    }
    dfs1(1, 0);
    dfs2(1, 1);
    dsu(1);
    for (int i = 1; i <= m; ++i) cout << ans[i] << endl;
}

int main() {
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

- 子树查询类问题

现在将会问你 m 个问题。对于每个问题，它将会给你三个参数 x,l,r 表示询问以 x 为根的子树中，节点深度在该子树中不小于 l 且不大于 r 的所有节点。你需要告诉智乃酱三个信息，所有符合条件节点的最小值，最大值，以及它们的和。

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
```

```cpp
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 1e5 + 10;
const ll mod = 998244353;
const ll inf32 = 1e9;
const ll inf64 = 2e18;

int tot, h[maxn], len[maxn], L[maxn], R[maxn], fa[maxn], son[maxn], dfn, n, m, x, l, r, u, v;
ll val[maxn];

struct node {
    ll Sum, Max, Min;
} ans[maxn];

struct qnode {
    int id;
    int l, r;
    qnode(int _id = 0, int _l = 0, int _r = 0) {id = _id, l = _l, r = _r;}
};

struct edges {
    int to, next;
} e[2 * maxn];
vector<qnode> lis[maxn];

struct tnode
{
    ll Sum, Max, Min;
    int l, r;
};
tnode operator + (const tnode &a, const tnode &b)
{
    tnode c;
    c.l = a.l;
    c.r = b.r;
    c.Sum = a.Sum + b.Sum;
    c.Max = max(a.Max, b.Max);
    c.Min = min(a.Min, b.Min);
    return c;
}

struct Segment_Tree
{
    tnode t[4 * maxn];
    int mp[maxn];
    void update (int root)
    {
        int ch = root << 1;
        t[root] = t[ch] + t[ch + 1];
    }
    void buildt(int root, int l, int r)
    {
        t[root].l = l;
        t[root].r = r;
        if (l != r)
        {
            int mid = (l + r) >> 1;
            int ch = root << 1;
            buildt(ch, l, mid);
            buildt(ch + 1, mid + 1, r);
            update(root);
        }
        else
        {
            mp[l] = root;
            t[root].Max = -inf64;
            t[root].Min = inf64;
            t[root].Sum = 0;
```

```
        }
    }
    void change(int pos, long long delta, long long nmax, long long nmin)
    {
        int root = mp[pos];
        t[root].Sum += delta;
        t[root].Max = max(t[root].Max, nmax);
        t[root].Min = min(t[root].Min, nmin);
        while (root >>= 1)update(root);
    }
    tnode getdata(int pos)
    {
        return t[mp[pos]];
    }
    tnode getseg(int root, int l, int r)
    {
        if (t[root].l == l && t[root].r == r)
        {
            return t[root];
        }
        int mid = (t[root].l + t[root].r) >> 1;
        int ch = root << 1;
        if (r <= mid)return getseg(ch, l, r);
        else if (l > mid)return getseg(ch + 1, l, r);
        else return getseg(ch, l, mid) + getseg(ch + 1, mid + 1, r);
    }
};
Segment_Tree ST;

void add_edge(int u, int to)
{
    e[++tot].to = to;
    e[tot].next = h[u];
    h[u] = tot;
    return;
}


void dfs1(int x, int father)
{
    fa[x] = father;
    for (int i = h[x]; i; i = e[i].next)
    {
        if (e[i].to != father)
        {
            dfs1(e[i].to, x);
            if (!son[x] || len[son[x]] < len[e[i].to])son[x] = e[i].to;
        }
    }
    len[x] = len[son[x]] + 1;
    return;
}

void dfs2(int x)
{
    L[x] = ++dfn;
    R[x] = L[x] + len[x] - 1;
    if (son[x])dfs2(son[x]);
    for (int i = h[x]; i; i = e[i].next)
    {
        if (e[i].to != fa[x] && e[i].to != son[x])
        {
            dfs2(e[i].to);
        }
    }
    return;
}

void dfs(int x)
{
    if (son[x])
```

```
    {
        dfs(son[x]);
    }
    for (int i = h[x]; i; i = e[i].next)
    {
        if (e[i].to != fa[x] && e[i].to != son[x])
        {
            dfs(e[i].to);
            for (int j = L[e[i].to], k = 1; j <= R[e[i].to]; ++j, ++k)
            {
                tnode temp = ST.getdata(j);
                ST.change(L[x] + k, temp.Sum, temp.Max, temp.Min);
            }
        }
    }
    ST.change(L[x], val[x], val[x], val[x]);
    for (auto &i : lis[x])
    {
        tnode temp = ST.getseg(1, L[x] + i.l, L[x] + i.r);
        ans[i.id].Sum = temp.Sum;
        ans[i.id].Max = temp.Max;
        ans[i.id].Min = temp.Min;
    }
    return;
}

void solve() {
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> val[i];
    for (int i = 1; i <= n - 1; ++i) {
        cin >> u >> v;
        add_edge(u, v);
        add_edge(v, u);
    }
    dfs1(1, 0);
    dfs2(1);
    ST.buildt(1, 1, n);
    cin >> m;
    for (int i = 1; i <= m; ++i) {
        cin >> x >> l >> r;
        lis[x].push_back(qnode(i, l, r));
    }
    dfs(1);
    for (int i = 1; i <= m; ++i)
    {
        cout << ans[i].Min << " " <<  ans[i].Max << " " << ans[i].Sum << endl;
    }
}

int main() {
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

- 小 Q 与树

u∑v∑min(a[u], a[v]) * dis(u, v)

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
```

```cpp
#define int ll
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 2e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

int n, h[maxn], to[maxn << 1], nxt[maxn << 1], cnt = 0;
int sz[maxn], son[maxn], dep[maxn], L[maxn], R[maxn], f[maxn], idx[maxn], top[maxn], dfn = 0;
ll sum[maxn], ans = 0;
struct node {
    int x, id;
} a[maxn];

void add(int u, int v) {
    to[++cnt] = v;
    nxt[cnt] = h[u];
    h[u] = cnt;
}

void dfs1(int u, int fa) {
    f[u] = fa;
    dep[u] = dep[fa] + 1;
    sz[u] = 1;
    for (int i = h[u]; i; i = nxt[i]) {
        int v = to[i];
        if (v == fa) continue;
        dfs1(v, u);
        sz[u] += sz[v];
        if (sz[v] > sz[son[u]])
            son[u] = v;
    }
}

void dfs2(int u, int t) {
    top[u] = t;
    L[u] = ++dfn;
    idx[dfn] = u;
    if (son[u]) dfs2(son[u], t);
    for (int i = h[u]; i; i = nxt[i]) {
        int v = to[i];
        if (v == f[u] || v == son[u]) continue;
        dfs2(v, v);
    }
    R[u] = dfn;
}

struct Segment {
    struct Node {
        int l, r;
        int sum, add;
    } tr[maxn * 4];
    void pushup(int u) {
        tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
    }
    void pushdown(int u) {
        if (tr[u].add) {
            int x = tr[u].add;
            tr[u << 1].sum += (tr[u << 1].r - tr[u << 1].l + 1) * x;
            tr[u << 1 | 1].sum += (tr[u << 1 | 1].r - tr[u << 1 | 1].l + 1) * x;
            tr[u << 1].add += x;
            tr[u << 1 | 1].add += x;
            tr[u].add = 0;
        }
        return;
    }
    void build(int u, int l, int r) {
        tr[u] = {l, r};
```

```
        if (l == r) return;
        int mid = (l + r) >> 1;
        build(u << 1, l, mid);
        build(u << 1 | 1, mid + 1, r);
        return;
    }
    void modify(int u, int l, int r, int x) {
        if (l <= tr[u].l && tr[u].r <= r) {
            tr[u].add += x;
            tr[u].sum += (tr[u].r - tr[u].l + 1) * x;
            return;
        }
        pushdown(u);
        int mid = (tr[u].l + tr[u].r) >> 1;
        if (l <= mid) modify(u << 1, l, r, x);
        if (mid < r) modify(u << 1 | 1, l, r, x);
        pushup(u);
        return;
    }
    int query(int u, int l, int r) {
        if (l <= tr[u].l && tr[u].r <= r) return tr[u].sum;
        pushdown(u);
        int mid = (tr[u].l + tr[u].r) >> 1;
        int res = 0;
        if (l <= mid) res += query(u << 1, l, r);
        if (mid < r) res += query(u << 1 | 1, l, r);
        return res;
    }
} Tr;

void tree_add(int x, int y) {
    while (top[x] != top[y]) {
        if (dep[top[x]] < dep[top[y]]) swap(x, y);
        Tr.modify(1, L[top[x]], L[x], 1);
        x = f[top[x]];
    }
    if (dep[x] < dep[y]) swap(x, y);
    Tr.modify(1, L[y], L[x], 1);
    return;
}

int tree_sum(int x, int y) {
    int res = 0;
    while (top[x] != top[y]) {
        if (dep[top[x]] < dep[top[y]]) swap(x, y);
        res += Tr.query(1, L[top[x]], L[x]);
        x = f[top[x]];
    }
    if (dep[x] < dep[y]) swap(x, y);
    res += Tr.query(1, L[y], L[x]);
    return res;
}

void solve() {
    cin >> n;
    rep(i, 1, n) {
        cin >> a[i].x;
        a[i].id = i;
    }
    sort(a + 1, a + n + 1, [&](node p, node q) {return p.x > q.x;});
    rep(i, 1, n - 1) {
        int u, v;
        cin >> u >> v;
        add(u, v);
        add(v, u);
    }
    dfs1(1, 0);
    dfs2(1, 1);
    Tr.build(1, 1, n);
    rep(i, 1, n) sum[i] = sum[i - 1] + dep[a[i].id];
    rep(i, 1, n){
```

```
        int x = a[i].id;
        ans = (ans + (ll)dep[x] * (i - 1) * a[i].x % mod + sum[i - 1] * a[i].x % mod - 2ll * a[i].x * tree_sum(1, x) %
    ↪ mod + mod) % mod;
        tree_add(1, x);
    }
    cout << ans * 2ll % mod << endl;
}
signed main() {
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

计算几何

二维几何：点与向量

```
#define y1 yy1
#define nxt(i) ((i + 1) % s.size())
typedef double LD;
const LD PI = 3.14159265358979323846;
const LD eps = 1E-10;
int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
struct L;
struct P;
typedef P V;
struct P {
    LD x, y;
    explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
    explicit P(const L& l);
};
struct L {
    P s, t;
    L() {}
    L(P s, P t): s(s), t(t) {}
};

P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
inline bool operator < (const P& a, const P& b) {
    return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
}
bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
P::P(const L& l) { *this = l.t - l.s; }
ostream &operator << (ostream &os, const P &p) {
    return (os << "(" << p.x << "," << p.y << ")");
}
istream &operator >> (istream &is, P &p) {
    return (is >> p.x >> p.y);
}

LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
// ---------------------------------------------
```

字符串

KMP

- KMP 模板

```cpp
#include <bits/stdc++.h>

using namespace std;

const int N = 1e6 + 10;

vector<int> prefix_function(string s)
{
    int n = (int)s.length();
    vector<int> pi(n);
    for (int i = 2; i < n; i++)
    {
        pi[i] = pi[i - 1];
        while (pi[i] && s[i] != s[pi[i] + 1])
            pi[i] = pi[pi[i]];
        pi[i] += (s[i] == s[pi[i] + 1]);
    }
    return pi;
}

int main(void)
{
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    string s1, s2;
    cin >> s1 >> s2;
    s1 = " " + s1;
    s2 = " " + s2;
    auto nxt = prefix_function(s2);
    for (int i = 1, j = 0; i < s1.size(); i++)
    {
        while (j && s1[i] != s2[j + 1])
            j = nxt[j];
        if (s1[i] == s2[j + 1])
            j++;
        if (j == s2.size() - 1)
        {
            cout << i - j + 1 << "\n";
            j = nxt[j];
        }
    }
    for (int i = 1; i < s2.size(); i++)
        cout << nxt[i] << " ";

    return 0;
}
```

- carpet(二维 KMP) 有一个 n*m 的地毯，aij 表示地毯每格的元素，bij 表示地毯每格的价格，要求选取一块价格最大值最小的地毯，并且这块地毯无限铺开之后，原地毯是其子矩阵

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define x first
#define y second
#define int ll
#define rep(i, j, k) for (int i = (j); i <= (k); i++)
#define per(i, j, k) for (int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 1e6 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

```

```cpp
vector<int> prefix_function(string s)
{
    int n = (int)s.length();
    vector<int> pi(n);
    for (int i = 2; i < n; i++)
    {
        pi[i] = pi[i - 1];
        while (pi[i] && s[i] != s[pi[i] + 1])
            pi[i] = pi[pi[i]];
        pi[i] += (s[i] == s[pi[i] + 1]);
    }
    return pi;
}

int get_length(vector<string> s)
{
    int len = s[1].size() - 1;
    int ret = len;
    vector<int> cnt(len + 1);
    for (int i = 1; i < s.size(); ++i)
    {
        string tmp = s[i];
        auto nxt = prefix_function(tmp);
        int j = len;
        while (j)
        {
            cnt[len - nxt[j]]++;
            j = nxt[j];
        }
    }
    for (int i = 1; i <= len; ++i)
        if (cnt[i] == s.size() - 1)
        {
            ret = i;
            break;
        }
    return ret;
}

void solve()
{
    int n, m;
    cin >> n >> m;
    vector<string> s1(n + 1);
    for (int i = 1; i <= n; ++i)
        cin >> s1[i], s1[i] = " " + s1[i];
    vector<string> s2(m + 1);
    for (int i = 1; i <= m; ++i)
    {
        string tmp = " ";
        for (int j = 1; j <= n; ++j)
            tmp += s1[j][i];
        s2[i] = tmp;
    }
    vector<vector<int>> a(n + 1, vector<int>(m + 1, 0));
    for (int i = 1; i <= n; ++i)
        for (int j = 1; j <= m; ++j)
            cin >> a[i][j];
    int p = get_length(s1), q = get_length(s2);
    ll ans = 1e9;
    deque<int> dq;
    auto b = a;
    for (int i = 1; i <= n; ++i){
        while (dq.size()) dq.pop_back();
        for (int j = 1; j <= m; ++j){
            while (dq.size() && j - dq.front() + 1 > p) dq.pop_front();
            while (dq.size() && a[i][dq.back()] <= a[i][j]) dq.pop_back();
            dq.push_back(j);
            b[i][j] = a[i][dq.front()];
        }
    }
```

```
    for (int j = 1; j <= m; ++j){
        while (dq.size()) dq.pop_back();
        for (int i = 1; i <= n; ++i){
            while (dq.size() && i - dq.front() + 1 > q) dq.pop_front();
            while (dq.size() && b[dq.back()][j] <= b[i][j]) dq.pop_back();
            dq.push_back(i);
            if (i >= q && j >= p)
                ans = min(ans, 1ll * b[dq.front()][j]);
        }
    }
    ans = ans * (p + 1) * (q + 1);
    cout << ans << endl;
}

signed main()
{
    ios;
    // freopen("sample.txt", "r", stdin);
    // freopen("resout.txt", "w", stdout);
    int t = 1;
    // cin >> t;
    while (t--)
    {
        solve();
    }
    return 0;
}

```

杂项

线性基

- 线性基模板 (总异或最大值)

```
ll p[64];
void insert(ll x){
    for (int i = 63; ~i; --i){
        if (!(x >> i)) continue;
        if (!p[i]){
            p[i] = x;
            break;
        }
        x ^= p[i];
    }
}
```

- 区间线性基 (区间异或最大值, 强制在线)

```
#include<bits/stdc++.h>
#define M 500009
using namespace std;
int read() {
    int f = 1, re = 0; char ch;
    for (ch = getchar(); !isdigit(ch) && ch != '-'; ch = getchar());
    if (ch == '-') {f = -1, ch = getchar();}
    for (; isdigit(ch); ch = getchar()) re = (re << 3) + (re << 1) + ch - '0';
    return re * f;
}
int pos[M][32], p[M][32], t, n, m, lastans;
void insert(int val, int num, int po) {
    for (int i = 30; i >= 0; i--) {
        if (val & (1ll << i)) {
            if (!p[num][i]) {p[num][i] = val, pos[num][i] = po; return;}
            else if (pos[num][i] < po) {
                swap(val, p[num][i]);
                swap(po, pos[num][i]);
            } val ^= p[num][i];
        }
    } return;
```

```
}
int query(int l, int r) {
    int ans = 0;
    for (int i = 30; i >= 0; i--)
        if (pos[r][i] >= l && (p[r][i]^ans) > ans) ans ^= p[r][i];
    return ans;
}
signed main() {
    t = read();
    while (t--) {
        n = read(), m = read(); lastans = 0;
        memset(p, 0, sizeof(p));
        memset(pos, 0, sizeof(pos));
        for (int i = 1; i <= n; i++) {
            int x = read();
            for (int j = 0; j <= 30; j++)
                p[i][j] = p[i - 1][j], pos[i][j] = pos[i - 1][j];
            insert(x, i, i);
        }
        for (int i = 1; i <= m; i++) {
            int opt = read();
            if (opt) {
                int x = read()^lastans; n++;
                for (int j = 0; j <= 30; j++)
                    p[n][j] = p[n - 1][j], pos[n][j] = pos[n - 1][j];
                insert(x, n, n);
            }
            else {
                int l = (read()^lastans) % n + 1;
                int r = (read()^lastans) % n + 1;
                if (l > r) swap(l, r);
                printf("%d\n", lastans = query(l, r));
            }
        }
    } return 0;
}
```

- 区间问题 (异或和, 区间内是否存在异或和为 x)

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
constexpr ll maxn = 4e5 + 5;
int pos[65];
ll p[65], t, n, m;
bool ans[maxn];
void insert(ll val, int P)
{
    for (int i = 59; i >= 0; i--)
    {
        if (val & (1ll << i))
        {
            if (!p[i])
            {
                p[i] = val, pos[i] = P;
                return;
            }
            else if (pos[i] < P)
            {
                swap(val, p[i]);
                swap(P, pos[i]);
            }
            val ^= p[i];
        }
    }
    return;
}
bool query(int l, ll val)
{
    for (int i = 59; i >= 0; i--)
    {
        if (val & (1ll << i))
```

```
        {
            if (!p[i])
                return false;
            if (pos[i] < l)
                return false;
            val ^= p[i];
        }
    }
    return true;
}
signed main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n >> m;
    vector<ll> a(n + 1);
    vector<tuple<int, int, ll, int>> q(m);
    for (int i = 1; i <= n; i++)
        cin >> a[i];
    for (int i = 0; i < m; i++)
    {
        auto &[l, r, val, id] = q[i];
        cin >> l >> r >> val, id = i;
    }
    sort(q.begin(), q.end(), [&](auto x, auto y)
         {
        auto &[l1,r1,val1,id1] = x;
        auto &[l2,r2,val2,id2] = y;
        return (r1==r2)?(l1<l2):(r1<r2); });
    int R = 0;
    for (int i = 0; i < m; i++)
    {
        auto &[l, r, val, id] = q[i];
        while (R < r)
            insert(a[R + 1], R + 1),R++;
        ans[id] = query(l, val);
    }
    for (int i = 0; i < m; i++)
    {
        cout << (ans[i] ? "Yes\n" : "No\n");
    }

    return 0;
}

```

Tarjan

- 缩点

```
//Tarjan 缩点（删去一个点，有多少点对不能互通）
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define x first
#define y second
#define int ll
#define rep(i, j, k) for (int i = (j); i <= (k); i++)
#define per(i, j, k) for (int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
typedef __int128 i128;
const ll maxn = 1e6 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

ll n, m;
ll head[maxn], nxt[maxn], to[maxn], tot = 1;
ll dfn[maxn], low[maxn];
```

```cpp
bool vis[maxn];
ll cnt;
ll deg[maxn];
ll ans[maxn];
ll sz[maxn];

void addedge(int u, int v)
{
    nxt[++tot] = head[u];
    to[head[u] = tot] = v;
    nxt[++tot] = head[v];
    to[head[v] = tot] = u;
}

void tarjan(int u, int lst)
{
    dfn[u] = low[u] = ++cnt;
    ll sum = 0;
    sz[u] = 1;
    for (int i = head[u]; i; i = nxt[i])
    {
        if (i != (lst ^ 1))
        {
            int v = to[i];
            if (!dfn[v])
            {
                tarjan(v, i);
                sz[u] += sz[v];
                low[u] = min(low[u], low[v]);
                if (low[v] >= dfn[u])
                {
                    // 找到新的双连通分量
                    ans[u] += 1ll * sz[v] * (n - sz[v]);
                    sum += sz[v];
                    ++deg[u];
                    if (deg[u] > 1 || u != 1)
                        vis[u] = 1;
                }
            }
            else
                low[u] = min(low[u], dfn[v]);
        }
    }
    if (vis[u])
    {
        ans[u] += 1ll * (n - (sum + 1)) * (sum + 1) + n - 1;
    }else
        ans[u] = 2 * (n - 1);
}
void solve()
{
    cin >> n >> m;
    for (int i = 1; i <= m; ++i)
    {
        int u, v;
        cin >> u >> v;
        addedge(u, v);
    }
    tarjan(1, -1);
    for (int i = 1; i <= n; ++i)
    {
        if (vis[i])
        {
            cout << ans[i] << endl;
        }
        else
        {
            cout << 2ll * (n - 1) << endl;
        }
    }
}
```

```
93
94  signed main()
95  {
96      ios;
97      //freopen("sample.txt", "r", stdin);
98      //freopen("res.txt", "w", stdout);
99      int t = 1;
100     // cin >> t;
101     while (t--)
102     {
103         solve();
104     }
105
106     return 0;
107 }
```