Contents

1 cdp 分治

```
#include<bits/extc++.h>
using namespace __gnu_pbds;
using namespace std;
using ll=long long;
template <class T> constexpr auto NL(T) -> T {return std
    ::numeric_limits<T>::max();}
template <class T> using Tree = tree<T, null_type, less<T
    >,rb_tree_tag, tree_order_statistics_node_update>;
#define pb push_back
constexpr int N=1e5+7;
array<int,5>a[N],tmp[N];
int n,ans[N];
template<typename T>
struct BIT
{
    int size;
    std::vector<T> c;
    BIT(int size = 0) {
        init(size);
    }
    void init(int size) {
        this->size = size;
        c.assign(size, T());
    }
    T lowbit(T x) {
        return x & (-x);
    }
    T ask(int n) {
        T sum = 0;
        for(int i = n;i;i -= i&-i){
            sum +=c[i];
        }
        return sum;
    }
    void add(int pos, T x) {
        for(int i = pos;i <= size;i += i&-i) {
            c[i]+=x;
        }
    }
    void modify(int l, int r, T x) {
        if(l>r)swap(l,r);
        add(l, x);
        add(r + 1, -x);
    }
    T getsum(T l,T r){
        if(l>r)swap(l,r);
        return ask(r)-ask(l-1);
    }
};
BIT<int>c(N);
void Solve(int l,int r){
    if(l==r)return ;
    int mid=(l+r)>>1;
    Solve(l,mid);
    Solve(mid+1,r);
    //solve 把所有的点按照（y,z)归并
    int p1=l,p2=mid+1;
    int p3=0;
    while(p1<=mid or p2<=r){
        if(p2>r || (p1<=mid and make_pair(a[p1][1],a[p1
            ][2])<=make_pair(a[p2][1],a[p2][2]))){
            c.add(a[p1][2],a[p1][3]);
            tmp[p3++]=a[p1++];
        }else{
            a[p2][4]+=c.ask(a[p2][2]);
            tmp[p3++]=a[p2++];
        }
    }
    for(int i=l;i<=mid;i++)c.add(a[i][2],-a[i][3]);
    for(int i=0;i<p3;i++)a[l+i]=tmp[i];
}
void solve(void){
    ll n,k;
    cin>>n>>k;
    for(int i=0;i<n;i++){
        cin>>a[i][0]>>a[i][1]>>a[i][2];
        a[i][3]=1;
    }
    sort(a,a+n);
    int t=0;
    for(int i=0;i<n;i++){
        if(t!=0 and (a[i][0]==a[t-1][0] and a[i][1]==a[t
            -1][1] and a[i][2]==a[t-1][2])){
            a[t-1][3]+=1;
        }else{
            a[t++]==a[i];
        }
    }
    Solve(0,n-1);
    for(int i=0;i<t;i++){
        ans[a[i][3]+a[i][4]-1]+=a[i][3];
    }

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

2 dsuontree

```
//
// Created by 墨华 on 2024/4/8.
//
constexpr int N=2e5+7;
int l[N],r[N],id[N],sz[N],hs[N],tot;
int dep1[N];
ll dep2[N];
int cnt[N];
int maxncnt[N];
ll sumcnt,ans[N];
void dfs_init(int u,int f){
    l[u]=++tot;
    id[tot]=u;
    sz[u]=1;
    hs[u]=-1;
    for(auto [v,w]:e[u])if(v!=f){
        dep1[v]=dep1[u]+1;
        dep2[v]=dep2[u]+w;
        dfs_init(v,u);
        sz[u]+=sz[v];
        if(hs[u]==-1 or sz[v]>sz[hs[u]])hs[u]=v;
    }
    r[u]=tot;
}
void dfs_solve(int u,int f,bool keep){
    for(auto v:e[u]){
        if(v!=f and v!=hs[u]){
            dfs_solve(v,u, false);
        }
    }
    if(hs[u]!=-1){
        dfs_solve(hs[u],u,true);
    }
    auto add=[&](int x){
        x=cnt[x];
        cnt[x]++;
        if(cnt[x]>maxcnt)maxcnt=cnt[x],sumcnt=0;
        if(cnt[x]==maxcnt)sumcnt+=x;
    };
    auto del=[&](int x){
        x=c[x];
        cnt[x]--;
    };
    for(auto v:e[u]){
        if(v!=f and v!=hs[u]){
            for(int x=l[v];x<=r[v];x++){
                add(id[x]);
            }
        }
    }
    add(u);
    ans[u]=sumcnt;
    if(!keep){

    }
}
```

3 jls 分块

```
template<class T>
struct Block {
    int n;
    int B;
    std::vector<T> a;
    std::vector<T> add;
    Block(int n_) : n{n_} {
        B = std::sqrt(n);
        a.resize(n);
        add.resize((n + B - 1) / B);
    }
    void rangeAdd(int l, int r, int v) {
        r++;
        if (l >= r) {
            return;
        }
        int lb = l / B;
        int rb = (r - 1) / B;
        if (lb == rb) {
            for (int i = l; i < r; i++) {
                a[i] += v;
            }
        } else {
            for (int i = l; i < (lb + 1) * B; i++) {
                a[i] += v;
            }
            for (int i = lb + 1; i < rb; i++) {
                add[i] += v;
            }
            for (int i = rb * B; i < r; i++) {
                a[i] += v;
            }
        }
    }
    T query(int x) {
        return a[x] + add[x / B];
    }
};
```

4 RMQ

```
template<class T,
        class Cmp = std::less<T>>
struct RMQ {
    const Cmp cmp = Cmp();
    static constexpr unsigned B = 64;
    using u64 = unsigned long long;
    int n;
    std::vector<std::vector<T>> a;
    std::vector<T> pre, suf, ini;
    std::vector<u64> stk;
    RMQ() {}
    RMQ(std::vector<T> v) {
        v.erase(v.begin(), v.begin() + 1);
        init(v);
    }
    void init(const std::vector<T>& v) {
        n = v.size();
        pre = suf = ini = v;
        stk.resize(n);
        if (!n) {
            return;
        }
        const int M = (n - 1) / B + 1;
        const int lg = std::__lg(M);
        a.assign(lg + 1, std::vector<T>(M));
        for (int i = 0; i < M; i++) {
            a[0][i] = v[i * B];
            for (int j = 1; j < B && i * B + j < n; j++)
                {
                a[0][i] = std::min(a[0][i], v[i * B + j],
                    cmp);
            }
        }
        for (int i = 1; i < n; i++) {
            if (i % B) {
                pre[i] = std::min(pre[i], pre[i - 1], cmp
                    );
            }
        }
        for (int i = n - 2; i >= 0; i--) {
            if (i % B != B - 1) {
                suf[i] = std::min(suf[i], suf[i + 1], cmp
                    );
            }
        }
        for (int j = 0; j < lg; j++) {
            for (int i = 0; i + (2 << j) <= M; i++) {
                a[j + 1][i] = std::min(a[j][i], a[j][i +
                    (1 << j)], cmp);
            }
        }
        for (int i = 0; i < M; i++) {
            const int l = i * B;
            const int r = std::min(1U * n, l + B);
            u64 s = 0;
            for (int j = l; j < r; j++) {
                while (s && cmp(v[j], v[std::__lg(s) + l
                    ])) {
                     s ^= 1ULL << std::__lg(s);
                }
                s |= 1ULL << (j - l);
                stk[j] = s;
            }
        }
    }
    T operator()(int l, int r) {
        --l;
        if (l / B != (r - 1) / B) {
            T ans = std::min(suf[l], pre[r - 1], cmp);
            l = l / B + 1;
            r = r / B;
            if (l < r) {
                int k = std::__lg(r - l);
                ans = std::min({ ans, a[k][l], a[k][r -
                    (1 << k)] }, cmp);
            }
            return ans;
        }
        else {
            int x = B * (l / B);
            return ini[__builtin_ctzll(stk[r - 1] >> (l -
                x)) + l];
        }
    }
};
```

5 st 表

```
template <class Info>
struct ST {
    int n, m;
    std::vector<std::vector<Info>> f;
    ST(std::vector<ll> &a) : n(a.size() - 1) {
        m = __lg(n) + 1;
        f.resize(m + 1, vector<Info>(n + 1));
        for (int i = 1; i <= n; i++) {
            f[0][i] = {a[i]};
        }
        for (int j = 1; j <= m; j++) {
            for (int i = 1; i + (1 << j) - 1 <= n; i++) {
                f[j][i] = f[j - 1][i] + f[j - 1][i + (1
                    << (j - 1))];
            }
        }
    }
    Info query(int l, int r) {
        int len = (r - l + 1);
        ll k = __lg(len);
        return f[k][l] + f[k][r - (1 << k) + 1];
    }
};
struct Info {
    ll x;
    friend Info operator+(const Info &a, const Info &b) {
        Info res;
        res.x = max(a.x, b.x);
        return res;
    }
};
```

6 zkw 线段树

```
template<class T>
struct Segment {
    T o[1 << 20]; int L;
    void upt(int x) {
        o[x] = o[x << 1] + o[x << 1 | 1];
    }
    void init(int n, const vector<T>&w) {
        L = 2 << std::__lg(n + 1);
        for (int i = 1; i <= n; ++i) o[i + L] = w[i];
        for (int i = L; i >= 1; --i) upt(i);
    }
    void change(int p, T v) {
        for (o[p += L] = v; p >>= 1; upt(p));
    }
    T query(int l, int r) {
        l += L - 1, r += L + 1;
        T ans{};
        for (; l ^ r ^ 1; l >>= 1, r >>= 1) {
            if ((l & 1) == 0) ans =ans+ o[l ^ 1];
            if ((r & 1) == 1) ans =ans+ o[r ^ 1];
        }
        return ans;
    }
};
Segment<node>seg;
struct node {
    ll val = 0;
    node () {
    }
    friend node operator+(node lhs, node rhs) {
        node now;
        if (lhs.val > rhs.val)now = lhs;
        else now = rhs;
        return now;
    }
};
```

7 主席树

```
struct Chairman_Tree {
    struct Node {int L, R, val;} tree[maxn * 500];
    void init() {
        memset(root, 0, sizeof root);
        cnt = 0;
    }
    /* 建T0空树 */
    int build(int l, int r) {
        int k = cnt++;
        tree[k].val = 0;
        if (l == r) return k;
        int mid = l + r >> 1;
        tree[k].L = build(l, mid); tree[k].R = build(mid
            + 1, r);
        return k;
    }
    /* 上一个版本节点P, 【ppos】+=del 返回新版本节点 */
    int update (int P, int l, int r, int ppos, int del) {
        int k = cnt++;
        tree[k].val = tree[P].val + del;
        if (l == r) return k;
        int mid = l + r >> 1;
        if (ppos <= mid) {
            tree[k].L = update(tree[P].L, l, mid, ppos,
                del);
            tree[k].R = tree[P].R;
        } else {
            tree[k].L = tree[P].L;
            tree[k].R = update(tree[P].R, mid + 1, r,
                ppos, del);
        }
        return k;
    }
    int query_kth(int lt, int rt, int l, int r, int k) {
        if (l == r) return a[rk[l]];
        int mid = l + r >> 1;
        if (tree[tree[rt].L].val - tree[tree[lt].L].val
            >= k) return query_kth(tree[lt].L, tree[rt].L,
             l, mid, k);
        else return query_kth(tree[lt].R, tree[rt].R, mid
             + 1, r, k + tree[tree[lt].L].val - tree[tree[
            rt].L].val);
    }
} tree;
```

8 分块

```
struct SqrtDecomposition {
    const int block_size, n;
    std::vector<int> ls, rs; // 每个块i的段位记为[ls[i],
        rs[i]), 右侧是开区间
    std::vector<bool> to_be_eval; // 该block的懒标记是否
        全部被清空

    explicit SqrtDecomposition(const int n_)
            : block_size(std::round(std::sqrt(n_))),
              n((n_ + block_size - 1) / block_size) {
        ls.resize(n); // n是block的数量 这里上取整了
        rs.resize(n);
        to_be_eval.assign(n, false);
        for (int i = 0; i < n; ++i) {
            // 每个块i的段位记为[ls[i], rs[i]), 右侧是开
                区间
            // 最后一个block里的元素的rs不一定填满了最后
                一个block，因此最后一个blocki的rs会等于数
                组长度。
            ls[i] = block_size * i;
            rs[i] = (i + 1 == n ? n_ : block_size * (i +
                1));
        }
    }

    template <typename T>
    void partial_update(const int idx, const T val); //
        这是我们需要实现的

    template <typename T>
    void total_update(const int idx, const T val); // 这
        是我们需要实现的

    template <typename T>
    void update(const int l, const int r, const T val) {
        if (r <= l) return;
        // 这里如果想debug好用点可以改成 assert(l < r);
        const int b_l = l / block_size, b_r = (r - 1) /
            block_size;
        // b_l是左端点在的block，g_r是右端点的block， -1
            是因为外部调用也是开区间。
        if (b_l < b_r) { // 不在一个区间的话，必定要分开
            讨论
            if (l == ls[b_l]) { // 如果左端点恰好是区间的
                开始
                total_update(b_l, val); // 只要更新这个区
                    间就好了
            } else {
                for (int i = l; i < rs[b_l]; ++i) {
                    partial_update(i, val); // 否则我们要
                        对左端点到下一个block之间进行暴力
                        更新
                }
            }
            for (int i = b_l + 1; i < b_r; ++i) {
                total_update(i, val); // 我们对中间的
                    block进行更新
            }
            if (r == rs[b_r]) {
                total_update(b_r, val); // 同理讨论右侧端
                    点冒出来的点，不再赘述
            } else {
                for (int i = ls[b_r]; i < r; ++i) {
                    partial_update(i, val);
                }
            }
        } else {
            for (int i = l; i < r; ++i) {
                partial_update(i, val); // 如果 l r本来就
                    在一个block，只需要直接暴力更新即可。
            }
        }
    }

    template <typename T>
    void partial_query(const int idx, T* val);

    template <typename T>
    void total_query(const int idx, T* val);

    template <typename T>
    T query(const int l, const int r, const T id) {
        const int b_l = l / block_size, b_r = (r − 1) /
            block_size;
        // b_l是左端点在的block，g_r是右端点的block， −1
            是因为外部调用也是开区间。
        T res = id;
        // 这里是我们的初始值，实现的时候会具体解释，简单
            举例，如果是求和这里就是0，然后累加
        // 之后的逻辑和update一模一样，不再赘述
        if (b_l < b_r) {
            if (l == ls[b_l]) {
                total_query(b_l, &res);
            } else {
                for (int i = l; i < rs[b_l]; ++i) {
                    partial_query(i, &res);
                }
            }
            for (int i = b_l + 1; i < b_r; ++i) {
                total_query(i, &res);
            }
            if (r == rs[b_r]) {
                total_query(b_r, &res);
            } else {
                for (int i = ls[b_r]; i < r; ++i) {
                    partial_query(i, &res);
                }
            }
        } else {
            for (int i = l; i < r; ++i) {
                partial_query(i, &res);
            }
        }
        return res;
    }
};
```

9 动态中位数

```
struct DynamicMedian {
    std::priority_queue<ll> down;
    std::priority_queue<ll, vector<ll>, greater<ll>> up;
    DynamicMedian() {}
    void insert(ll x) {
        if (down.empty() || x <= down.top()) {
            down.push(x);
        } else {
            up.push(x);
        }
        if (down.size() > 1 + up.size()) {
            up.push(down.top());
            down.pop();
        }
        if (up.size() > down.size()) {
            down.push(up.top());
            up.pop();
        }
    };
    double Ans() {
        if (up.size() + down.size() & 1) {
            return down.top();
        } else {
            return (down.top() + up.top()) / 2.0;
        }
    };
};
```

10 区间修改 +lazy 线段树

```
template<class Info, class Tag>
struct LazySegmentTree {
    int n;
    std::vector<Info> info;
    std::vector<Tag> tag;
    LazySegmentTree() : n(0) {}
    LazySegmentTree(int n_, Info v_ = Info()) {
        init(n_, v_);
    }
    template<class T>
    LazySegmentTree(std::vector<T> init_) {
        init(init_);
    }
    void init(int n_, Info v_ = Info()) {
        init(std::vector(n_, v_));
    }
    template<class T>
    void init(std::vector<T> init_) {
        n = init_.size() - 1;
        info.assign(4 << std::__lg(n) + 2, Info());
        tag.assign(4 << std::__lg(n) + 2, Tag());
        auto build = [&](auto && build, int p, int l, int
            r)->void{
            if (l == r) {
                info[p] = init_[l];
                return;
            }
            int mid = (l + r) >> 1;
            build(build, p << 1, l, mid);
            build(build, p << 1 | 1, mid + 1, r);
            push_up(p);
        };
        build(build, 1, 1, n);
    }
    void push_up(int p) {
        info[p] = (info[p << 1] + info[p << 1 | 1]);
    }
    void push_down(int p, int l, int r) {
        info[p << 1] += tag[p];
        info[p << 1 | 1] += tag[p];
        tag[p << 1] += tag[p];
        tag[p << 1 | 1] += tag[p];
        tag[p].init();
    }
    void change(int p, int l, int r, int nl, int nr, Tag
        num) {
        if (nl <= l and r <= nr) {
            info[p] += num;
            tag[p] += num;
            return;
        }
        push_down(p, l, r);
        int mid = (l + r) >> 1;
        if (nl <= mid)change(p << 1, l, mid, nl, nr, num)
            ;
        if (nr > mid)change(p << 1 | 1, mid + 1, r, nl,
            nr, num);
        push_up(p);
    }
    Info query(int p, int l, int r, int nl, int nr) {
        if (nl <= l and r <= nr) {
            return info[p];
        }
        push_down(p, l, r);
        push_up(p);
        int mid = (l + r) >> 1;
        if (nr <= mid)return query(p << 1, l, mid, nl, nr
            );
        if (nl > mid)return query(p << 1 | 1, mid + 1, r,
             nl, nr);
        return query(p << 1, l, mid, nl, nr) + query(p <<
             1 | 1, mid + 1, r, nl, nr);
    }
};
struct tag {
    void init() {

    }
    tag&operator+=(const tag &t) & {

        return *this;
    }
};
struct node {
    friend node operator+(node lhs, node rhs) {
        node res;
        return res;
    }
    node&operator+=(tag&t) {
        return *this;
        }
    };
```

11 区间修改单点查询树状数组

```
template <typename T>
struct fenwick {
    int n;
    std::vector<std::vector<T>> tr;
    fenwick(int _n) : n(_n) {
        tr.resize(2);
        for (int i = 0; i < 2; i++) {
            tr[i].resize(n + 2);
        }
    }
    void add(int i, int x, const T &v) {
        for ( ; x <= n; x += x & -x) {
            tr[i][x] += v;
        }
    }
    void modify(int l, int r, const T &v) {
        add(0, l, v);
        add(0, r + 1, -v);
        add(1, l, l * v);
        add(1, r + 1, (r + 1) * (-v));
    }
    T sum(int i, int x) {
        T ans = 0;
        for ( ; x > 0; x -= x & -x) {
            ans += tr[i][x];
        }
        return ans;
    }
    T Sum(int x) {
        return sum(0, x) * (x + 1) - sum(1, x);
    }
    T Sum(int l, int r) {
        return Sum(r) - Sum(l - 1);
    }
};
```

12 区间最小值 + 最小值出现次数

```
constexpr int N = 2e5 + 7;
struct info {
    int minv, mincnt;
    friend info operator+(const info &l, const info &r) {
        info a;
        a.minv = min(l.minv, r.minv);
        if (l.minv == r.minv)a.mincnt = l.mincnt + r.
            mincnt;
        else if (l.minv < r.minv)a.mincnt = l.mincnt;
        else a.mincnt = r.mincnt;
        return a;
    }
};
struct node {
    int t;
    info val;
} seg[N * 4];
void update(int id) {
    seg[id].val = seg[id << 1].val + seg[id << 1 | 1].val
        ;
}
void settag(int id, int t) {
    seg[id].val.minv = seg[id].val.minv + t;
    seg[id].t = seg[id].t + t;
}
void pushdown(int id) {
    if (seg[id].t != 0) {
        settag(id << 1, seg[id].t);
        settag(id << 1 | 1, seg[id].t);
        seg[id].t = 0;
    }
}
void build(int id, int l, int r) {
    seg[id].t=0;
    if (l == r) {
        seg[id].val.minv = 0;
        seg[id].val.mincnt = 1;
        return;
    } else {
        int mid = (l + r) >> 1;
        build(id << 1, l, mid);
        build(id << 1 | 1, mid + 1, r);
        update(id);
    }
}
void change(int id, int l, int r, int nl, int nr, int t)
    {
    if (l == nl and r == nr) {
        settag(id, t);
        return;
    }
    int mid = (l + r) >> 1;
    pushdown(id);
    if (nr <= mid)change(id << 1, l, mid, nl, nr, t);
    else if (nl > mid)change(id << 1 | 1, mid + 1, r, nl,
         nr, t);
    else {
        change(id << 1, l, mid, nl, mid, t);
        change(id << 1 | 1, mid + 1, r, mid + 1, nr, t);
    }
    update(id);
}
info query(int id, int l, int r, int ql, int qr) {
    if (l == ql and r == qr) {
        return seg[id].val;
    }
    int mid = (l + r) / 2;
    pushdown(id);
    update(id);
    if (qr <= mid)return query(id * 2, l, mid, ql, qr);
    else if (ql > mid)return query(id * 2 + 1, mid + 1, r
        , ql, qr);
    else {
        return query(id * 2, l, mid, ql, mid) + query(id
            * 2 + 1, mid + 1, r, mid + 1, qr);
    }
}
```

13 单点修改线段树

```
template<class info>
struct Segment {
    std::vector<info> tre;
    int n;
    Segment() : n(0) {}
    Segment(int n) : n(n), tre(4 << std::__lg(n)+2) {}
    Segment(int n, vector<info>&v) : n(n), tre(4 << std::
        __lg(n)+2) {
        init(v);
    }
    void pushup(int p) {
        tre[p] = tre[p << 1] + tre[p << 1 | 1];
    }
    void init(vector<info>&v) {
        auto build = [&](auto && build, int p, int l, int
            r) {
            if (l == r) {
                tre[p] = v[l];
                return;
            }
            int mid = (l + r) >> 1;
            build(build, p << 1, l, mid);
            build(build, p << 1 | 1, mid + 1, r);
            pushup(p);
        };
        build(build, 1, 1, n);
    }
    void change(int p, int l, int r, int pos, info x) {
        if (l == r) {
            tre[p]=x;
            return;
        }
        int mid = (l + r) >> 1;
        if (pos <= mid)change(p << 1, l, mid, pos, x);
        else change(p << 1 | 1, mid + 1, r, pos, x);
        pushup(p);
    }
    info query(int p, int l, int r, int nl, int nr) {
        if (nl <= l and r <= nr)return tre[p];
        int mid = (l + r) >> 1;
        if (nr <= mid)return query(p << 1, l, mid, nl, nr
            );
        if (nl > mid)return query(p << 1 | 1, mid + 1, r,
            nl, nr);
        return query(p << 1, l, mid, nl, nr) + query(p <<
            1 | 1, mid + 1, r, nl, nr);
    }
};
struct node {
    ll val;
    int id;
    node () {
    }
    friend  node operator+(node lhs, node rhs) {
        node now;
        if (lhs.val < rhs.val)now = lhs;
        else now = rhs;
        return now;
    }
};
```

14 扫描线求区间 mex

```
int sea
int main(){
    int q;
    cin>>q;
    vector<vector<int>>qu(n+1,vector<int>());
    for(int i=1;i<=q;i++){
        int l,r;
        cin>>l>>r;
        qu[r].push_back({l,i});
    }
    for(int r=1;r<=n;r++){
        change(1,0,n+1,a[r],r);
        for(auto que:qu[r]){
            ans[que.second]=search(1,0,n+1,que.first);
        }
    }
}
```

15 树链剖分

```
template<class SegmentTree, class Info>
struct Trh {
    std::vector<int> sz, top, dep, parent, in, out;
    int cur, n;
    SegmentTree seg;
    std::vector<std::vector<int>> e;
    Trh(int _n) : n(_n), sz(_n + 1), top(_n + 1), dep(_n
       + 1), parent(_n + 1, -1), e(_n + 1), in(_n + 1),
       cur(0), out(_n + 1) {
        seg.init(_n);
    }
    void addEdge(int u, int v) {
        e[u].push_back(v);
        e[v].push_back(u);
    }
    void init(int s) {
        dfsSz(s);
        dfsHLD(s);
    }
    void dfsSz(int u) {
        if (parent[u] != -1)
            e[u].erase(std::find(e[u].begin(), e[u].end()
               , parent[u]));
        sz[u] = 1;
        for (int &v : e[u]) {
            parent[v] = u;
            dep[v] = dep[u] + 1;
            dfsSz(v);
            sz[u] += sz[v];
            if (sz[v] > sz[e[u][0]])
                std::swap(v, e[u][0]);
        }
    }
    void dfsHLD(int u) {
        in[u] = ++cur;
        for (int v : e[u]) {
            if (v == e[u][0]) {
                top[v] = top[u];
            } else {
                top[v] = v;
            }
            dfsHLD(v);
        }
        out[u] = cur;
```

```
}
int lca(int u, int v) {
    while (top[u] != top[v]) {
        if (dep[top[u]] > dep[top[v]]) {
            u = parent[top[u]];
        } else {
            v = parent[top[v]];
        }
    }
    if (dep[u] < dep[v]) {
        return u;
    } else {
        return v;
    }
}
void change(int u, int v, ll add) {
    tag p{add};
    while (top[u] != top[v]) {
        if (dep[top[u]] > dep[top[v]]) {
            seg.change(1, 1, n, in[top[u]], in[u], p)
                ;
            u = parent[top[u]];
        } else {
            seg.change(1, 1, n, in[top[v]], in[v], p)
                ;
            v = parent[top[v]];
        }
    }
    if (dep[u] < dep[v]) {
        seg.change(1, 1, n, in[u], in[v], p);
    } else {
        seg.change(1, 1, n, in[v], in[u], p);
    }
}
node query(int u, int v) {
    node ans;
    while (top[u] != top[v]) {
        if (dep[top[u]] > dep[top[v]]) {
            ans = ans + seg.query(1, 1, n, in[top[u
                ]], in[u]);
            u = parent[top[u]];
        } else {
            ans = ans + seg.query(1, 1, n, in[top[v
                ]], in[v]);
            v = parent[top[v]];
        }
        }
        if (dep[u] < dep[v]) {
            ans = ans + seg.query(1, 1, n, in[u], in[v]);
        } else {
            ans = ans + seg.query(1, 1, n, in[v], in[u]);
        }
        return ans;
    }
};
```

16 珂朵莉树

```
struct ODT {
    struct odt {
        int l, r;
        mutable int x;
        bool operator < (const odt &a) const {
            return l < a.l;
        }
    };
    set<odt> tr;
    typedef set <odt> :: iterator IT;
    ODT(int l, int r, int x) {
        tr.insert({l, r, x});
    }
    IT split(int pos) { //将pos-1和pos之间切开，返回pos所
       在区间指针
        auto it = tr.lower_bound({pos, 0, 0});
        if (it != tr.end() && it->l == pos) return it;
        it--;
        int l = it->l, r = it->r, x = it->x;
        tr.erase(it);
        tr.insert({l, pos - 1, x});
        return tr.insert({pos, r, x}).first;
    }
    void assign(int l, int r, int x) {
        auto R = split(r + 1);
        auto L = split(l);
        tr.erase(L, R);
        tr.insert({l, r, x});
    }
    void modify(int l, int r) {
        auto R = split(r + 1);
        auto L = split(l);
        for (auto it = L; it != R; it++) {
            // 对it->x暴力修改
        }
    }
    int query() {
        int ans = 0;
        for (auto it = tr.begin(); it != tr.end(); it++)
            {

        }
        return ans;
    }
};
```

17 笛卡尔树

```
template<class T>
struct CaT{
    vector<T>l,r;
    CaT() {}
    CaT(std::vector<T>&v,int n) {
        l.assign(n+1,-1);
        r.assign(n+1,-1);
        build(v,n);
    }
    void build(std::vector<T>&v,int n) {
        vector<int> st;
        int root = 0;
        for (int i = 1; i <= n; i++) {
            int last = -1;
            while (!st.empty() && v[st.back()] > v[i]) {
                last = st.back();
                st.pop_back();
            }
            if (!st.empty())r[st.back()] = i;
            else root = i;
            l[i] = last;
            st.push_back(i);
        }
    }
};
```

18 线段树代替 set

```
struct node {
    ll val;
    int id;
    node () {
    }
    friend  node operator+(node lhs, node rhs) {
        node now;
        if (lhs.val < rhs.val)now = lhs;
        else now = rhs;
        return now;
    }
};
struct Segment {
    std::vector<node> tre;
    int n;
    Segment() : n(0) {}
    Segment(int n) : n(n), tre(4 << std::__lg(n)+2) {}
    Segment(int n, vector<node>&v) : n(n), tre(4 << std::
       __lg(n)+2) {
        init(v);
    }
    void pushup(int p) {
        tre[p] = tre[p << 1] + tre[p << 1 | 1];
    }
    void init(vector<node>&v) {
        auto build = [&](auto && build, int p, int l, int
             r) {
            if (l == r) {
                tre[p] = v[l];
                return;
            }
            int mid = (l + r) >> 1;
            build(build, p << 1, l, mid);
            build(build, p << 1 | 1, mid + 1, r);
            pushup(p);
        };
        build(build, 1, 1, n);
    }
    void change(int p, int l, int r, int pos, ll x) {
        if (l == r) {
            tre[p].val = x;
            return;
        }
        int mid = (l + r) >> 1;
        if (pos <= mid)change(p << 1, l, mid, pos, x);
        else change(p << 1 | 1, mid + 1, r, pos, x);
        pushup(p);
    }
    node query(int p, int l, int r, int nl, int nr) {
        if (nl <= l and r <= nr)return tre[p];
        int mid = (l + r) >> 1;
        if (nr <= mid)return query(p << 1, l, mid, nl, nr
            );
        if (nl > mid)return query(p << 1 | 1, mid + 1, r,
             nl, nr);
        return query(p << 1, l, mid, nl, nr) + query(p <<
             1 | 1, mid + 1, r, nl, nr);
    }
    void erase(int u) {
        change(1, 1, n, u, LNF);
    }
    void insert(int u,ll val){
        change(1,1,n,u,node{val,u});
    }
};
```

19 莫队

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn=2e5+7;
ll tmp=0;
int main(){
    int n,m;
    cin>>n>>m;
    vector<int>v(n+1);
    for(int i=1;i<=n;i++)cin>>v[i];
    vector<ll>ans(m);
    std::vector<std::array<int, 3>> que(m);
    for(int i=0;i<m;i++){
        int l,r;
        cin>>l>>r;
        que[i]={l,r,i};
    }
    const int B=400;
     vector<int>cnt(maxn,0);
    std::sort(que.begin(),que.end(),[&](array<int,3>a,
       array<int,3>b) {
        if (a[0] / B != b[0] / B) {
            return a[0]/B < b[0]/B;
        } else {
            return a[1] < b[1];
        }
    });

    ll res=0;
    auto add = [&](int x) {
        x=v[x];
        res += 1LL * cnt[x] * (cnt[x] - 1) / 2;
        cnt[x] += 1;
    };
    auto del = [&](int x) {
        x=v[x];
        cnt[x] -= 1;
        res -= 1LL * cnt[x] * (cnt[x] - 1) / 2;
    };
    int l = 1, r = 0;
    for(int i=0;i<m;i++){
        while(r<que[i][1]) r++,add(r);
        while(l>que[i][0]) l--,add(l);
        while(r>que[i][1]) del(r),r--;
```

```
            while(l<que[i][0]) del(l),l++;
            ans[que[i][2]]=res;
        }
        for(int  i=0;i<m;i++)cout<<ans[i]<<"\n";
}
```