目录

1 数据结构

1.1 单调栈

```
int s[N], top=0;
for(int i=1;i<=n;i++) {
    while(top&&a[s[top]]>a[i]) top--;
    if(s[top]==a[i]) s[top]=i;//去重复操作，视情况而定
    s[++top]=i;
}
```

1.2 单调队列

```
int l=0,r=1;
int q[N];
for(int i=1;i<=n;i++) {
    while(l<=r&&a[q[r]]>a[i]) r--;
    q[++r]=i;
    while(l<=r&&q[l]<i-m+1) l++;//判断条件看情况
    ans[i]=a[q[l]];
}
```

1.3 线段树套平衡树

1 l r k 查询 k 在区间内的排名 $O(\log^2 N)$
2 l r k 查询区间内排名为 k 的值 $O(\log^3 N)$
3 pos k 修改某一位值上的数值 $O(\log^2 N)$
4 l r k 查询 k 在区间内的前驱 (前驱定义为严格小于 x，且最大的数，若不存在输出-2147483647) $O(\log^3 N)$
5 l r k 查询 k 在区间内的后继 (后继定义为严格大于 x，且最小的数，若不存在输出 2147483647) $O(\log^3 N)$
后面两个也有 $O(\log^2 N)$ 的做法，求前驱/后继后取 max, min。

```
int a[MAXN];
class DS { public:
    // treap begin
    struct treapnode {
        int ch[2], val, dat, siz, cnt;
    } tnd[MAXN*30];
    int tot;
    inline int New_treapnode(int v) {
        tot++;
        tnd[tot].ch[0] = tnd[tot].ch[1] = 0;
        tnd[tot].val = v, tnd[tot].dat = rand(), tnd[tot].siz = 1, tnd[tot].cnt = 1;
        return tot;
    }
#define lson tnd[rt].ch[0]
#define rson tnd[rt].ch[1]
    inline void push_up(int rt) {
        tnd[rt].siz = tnd[lson].siz + tnd[rson].siz + tnd[rt].cnt;
    }
    inline void rotate(int &rt, int d) {
        int tmp = tnd[rt].ch[d ^ 1];
        tnd[rt].ch[d ^ 1] = tnd[tmp].ch[d], tnd[tmp].ch[d] = rt, rt = tmp;
```

```
        push_up(tnd[rt].ch[d]), push_up(rt);
    }
    void insert(int &rt, int v) {
        if (!rt) {
            rt = New_treapnode(v);
            return;
        }
        if (v == tnd[rt].val) tnd[rt].cnt++;
        else {
            int d = v < tnd[rt].val ? 0 : 1;
            insert(tnd[rt].ch[d], v);
            if (tnd[rt].dat < tnd[tnd[rt].ch[d]].dat) rotate(rt, d ^ 1);
        }
        push_up(rt);
    }
    void remove(int &rt, int v) {
        if (!rt) return;
        if (v == tnd[rt].val) {
            if (tnd[rt].cnt > 1) {
                tnd[rt].cnt--, push_up(rt);
                return;
            }
            if (lson || rson) {
                if (!rson || tnd[lson].dat > tnd[rson].dat) {
                    rotate(rt, 1), remove(rson, v);
                } else rotate(rt, 0), remove(lson, v);
                push_up(rt);
            } else
                rt = 0;
        }
        v < tnd[rt].val ? remove(lson, v) : remove(rson, v);
        push_up(rt);
    }

    int get_rank(int rt, int v) {
        if (!rt) return 0;
        if (v == tnd[rt].val)
            return tnd[lson].siz;
        else if (v < tnd[rt].val)
            return get_rank(lson, v);
        else
            return tnd[lson].siz + tnd[rt].cnt + get_rank(rson, v);
    }

    bool isexit(int rt, int v) {
        if (!rt) return 0;
        if (v == tnd[rt].val)
            return 1;
        else if (v < tnd[rt].val)
            return isexit(lson, v);
        else
            return isexit(rson, v);
    }

    void init() {
        tot = 0;
```

```
    }
    // treap end

    struct segnode {
        int l, r, root;
    } T[MAXN << 2];

    void build(int rt, int l, int r) {
        T[rt].l = l, T[rt].r = r;
        for (int i = l; i <= r; i++)
            insert(T[rt].root, a[i]);
        if (l == r) return;
        int mid = (l + r) >> 1;
        build(rt << 1, l, mid), build(rt << 1 | 1, mid + 1, r);
    }

    void modify(int rt, int pos, int v) { // a[pos] -> v;
        remove(T[rt].root, a[pos]), insert(T[rt].root, v);
        if (T[rt].l == T[rt].r) {
            a[pos] = v;
            return;
        }
        int mid = (T[rt].l + T[rt].r) >> 1;
        if (pos <= mid) modify(rt << 1, pos, v);
        else modify(rt << 1 | 1, pos, v);
    }

    int get_rank(int rt, int L, int R, int v) {  // op1
        if (L <= T[rt].l && T[rt].r <= R) return get_rank(T[rt].root, v);
        int mid = (T[rt].l + T[rt].r) >> 1;
        int ans = 0;
        if (L <= mid) ans += get_rank(rt << 1, L, R, v);
        if (R > mid) ans += get_rank(rt << 1 | 1, L, R, v);
        return ans;
    }

    bool isexit(int rt, int L, int R, int v) {
        if (L <= T[rt].l && T[rt].r <= R) return isexit(T[rt].root, v);
        int mid = (T[rt].l + T[rt].r) >> 1;
        bool ans = false;
        if (L <= mid) ans |= isexit(rt << 1, L, R, v);
        if (R > mid) ans |= isexit(rt << 1 | 1, L, R, v);
        return ans;
    }
} tree;

int main() {
    int n, m; scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
    tree.build(1, 1, n);
    while (m--) {
        int opt; scanf("%d", &opt);
        if (opt == 1) {
            int l, r, k; scanf("%d%d%d", &l, &r, &k);
            printf("%d\n", tree.get_rank(1, l, r, k) + 1);
        } else if (opt == 2) {
```

```
            int l, r, k; scanf("%d%d%d", &l, &r, &k);
            int L = 0, R = 1e8;
            while (L < R) {
                int mid = (L + R + 1) >> 1;
                if (tree.get_rank(1, l, r, mid) >= k) R = mid - 1;
                else L = mid;
            }
            printf("%d\n", L);
        } else if (opt == 3) {
            int pos, k; scanf("%d%d", &pos, &k);
            tree.modify(1, pos, k);
        } else if (opt == 4) {
            int l, r, k;
            scanf("%d%d%d", &l, &r, &k);
            int num = tree.get_rank(1, l, r, k); // the rank of k
            if (num == 0) printf("-2147483647\n");
            else {
                int L = 0, R = 1e8;
                while (L < R) {
                    int mid = (L + R + 1) >> 1;
                    if (tree.get_rank(1, l, r, mid) >= num) R = mid - 1;
                    else L = mid;
                }
                printf("%d\n", L);
            }
        } else if (opt == 5) {
            int l, r, k; scanf("%d%d%d", &l, &r, &k);
            int num = tree.get_rank(1, l, r, k + 1);
            if (num == r - l + 1) printf("2147483647\n");
            else {
                num++;
                int L = 0, R = 1e8;
                while (L < R) {
                    int mid = (L + R + 1) >> 1;
                    if (tree.get_rank(1, l, r, mid) >= num) R = mid - 1;
                    else L = mid;
                }
                printf("%d\n", L);
            }
        }
    }
}
```

2 图论

2.1 点分治

2.1.1 两点间的距离是否为 3 的倍数（[国家集训队] 聪聪可可）

```
/*  5
    1 2 1
    1 3 2
    1 4 1
```

```
    2 5 3
    res = 13/25
    (1,1), (2,2), (2,3), (2,5), (3,2), (3,3), (3,4), (3,5), (4,3), (4,4), (5,2), (5,3), (5,5)*/
struct Edge {
    int to, nex, w;
} e[MAXN << 1];
int head[MAXN], tol;
void addEdge(int u, int v, int w) {
    e[tol].to = v, e[tol].w = w, e[tol].nex = head[u], head[u] = tol, tol++;
}
int son[MAXN], f[MAXN], vis[MAXN];
int t[5], dis[MAXN];
int main() {
    int n; scanf("%d", &n);
    for (int i = 1; i <= n; i++) head[i] = -1;
    tol = 0;
    for (int i = 2; i <= n; i++) {
        int u, v, w;
        scanf("%d%d%d", &u, &v, &w);
        w %= 3;
        addEdge(u, v, w), addEdge(v, u, w);
    }
    int root = 0, sum = n;
    f[0] = n;
    function<void(int, int)> get_root = [&](int u, int fa) {
        son[u] = 1, f[u] = 0;
        for (int i = head[u]; ~i; i = e[i].nex) {
            int v = e[i].to;
            if (vis[v] || v == fa) continue;
            get_root(v, u);
            son[u] += son[v];
            f[u] = max(f[u], son[v]);
        }
        f[u] = max(f[u], sum - son[u]);
        if (f[u] < f[root]) root = u;
    };
    get_root(1, 0);
    function<void(int, int)> get_dis = [&](int u, int fa) {
        t[dis[u]]++;
        for (int i = head[u]; ~i; i = e[i].nex) {
            int v = e[i].to, w = e[i].w;
            if (v == fa || vis[v]) continue;
            dis[v] = (dis[u] + w) % 3;
            get_dis(v, u);
        }
    };
    auto cal = [&](int u, int x) {
        t[0] = t[1] = t[2] = 0;
        dis[u] = x;
        get_dis(u, 0);
        return t[1] * t[2] * 2 + t[0] * t[0];
    };
    int res = 0;
    function<void(int)> solve = [&](int u) {
        res += cal(u, 0);
        vis[u] = 1;
```

```
        for (int i = head[u]; ~i; i = e[i].nex) {
            int v = e[i].to, w = e[i].w;
            if (vis[v]) continue;
            res -= cal(v, w);
            root = 0, sum = son[v];
            get_root(v, 0);
            solve(root);
        }
    };
    solve(root);
    int gcdd = __gcd(res, n * n);
    printf("%d/%d", res / gcdd, n * n / gcdd);
}
```

2.1.2 询问树上距离为 k 的点对是否存在

时间复杂度：$O(n\log^2 n + nm\log n)$

```
const int MAXN = 1e4 + 5;
const int MAXM = 105;
struct Edge {
    int to, w, nex;
} e[MAXN << 1];
int head[MAXN], tol;
void addEdge(int u, int v, int w) {
    e[tol].to = v, e[tol].w = w, e[tol].nex = head[u], head[u] = tol, tol++;
}
int son[MAXN], f[MAXN], vis[MAXN];
int dis[MAXN], top[MAXN];
int qs[MAXM], res[MAXM];
int main() {
    int n, m; scanf("%d%d", &n, &m);
    tol = 0;
    for (int i = 1; i <= n; i++) head[i] = -1;
    for (int i = 2; i <= n; i++) {
        int u, v, w;
        scanf("%d%d%d", &u, &v, &w);
        addEdge(u, v, w), addEdge(v, u, w);
    }
    for (int i = 1; i <= m; i++) scanf("%d", &qs[i]);

    int root = 0, sum = n;
    f[0] = n;
    function<void(int, int)> get_root = [&](int u, int fa) {
        son[u] = 1, f[u] = 0;
        for (int i = head[u]; ~i; i = e[i].nex) {
            int v = e[i].to;
            if (vis[v] || v == fa) continue;
            get_root(v, u);
            son[u] += son[v], f[u] = max(f[u], son[v]);
        }
        f[u] = max(f[u], sum - son[u]);
        if (f[u] < f[root]) root = u;
    };
    get_root(1, 0);

```

```
    vector<int> vec;
    function<void(int, int, int)> get_dis = [&](int u, int fa, int topf) {
        vec.pb(u), top[u] = topf;
        for (int i = head[u]; ~i; i = e[i].nex) {
            int v = e[i].to, w = e[i].w;
            if (vis[v] || v == fa) continue;
            dis[v] = dis[u] + w;
            get_dis(v, u, topf);
        }
    };
    auto cal = [&](int u) {
        vec.clear(), vec.pb(u);
        dis[u] = 0, top[u] = u;
        for (int i = head[u]; ~i; i = e[i].nex) {
            int v = e[i].to, w = e[i].w;
            if (vis[v]) continue;
            dis[v] = w;
            get_dis(v, u, v);
        }
        sort(vec.begin(), vec.end(), [&](int ta, int tb) {
            return dis[ta] < dis[tb];
        });
        for (int i = 1; i <= m; i++) {
            int L = 0, R = SZ(vec) - 1;
            if (res[i] || qs[i] == 0) {
                res[i] = 1;
            } else {
                while (L < R) {
                    if (dis[vec[L]] + dis[vec[R]] > qs[i]) R--;
                    else if (dis[vec[L]] + dis[vec[R]] < qs[i]) L++;
                    else if (top[vec[L]] == top[vec[R]]) {
                        if (dis[vec[R]] == dis[vec[R - 1]]) R--;
                        else L++;
                    } else {
                        res[i] = 1;
                        break;
                    }
                }
            }
        }
    };

    function<void(int)> solve = [&](int u) {
        vis[u] = 1;
        cal(u);
        for (int i = head[u]; ~i; i = e[i].nex) {
            int v = e[i].to, w = e[i].w;
            if (vis[v]) continue;
            root = 0, sum = son[v];
            get_root(v, 0);
            solve(root);
        }
    };
    solve(root);
    for (int i = 1; i <= m; i++) {
        printf("%s\n", res[i] ? "AYE" : "NAY");
```

```
    }
}
```

2.1.3 两点间距离不超过 l 距离权重和不超过 w

```
/*  4 4 6
    1 3
    1 4
    1 3
    res = 4
    6 2 17
    1 3
    2 5
    2 13
    1 6
    5 9
    res = 9     */
class BIT { public:
    int val[MAXN], n;
    void init(int _n) {
        n = _n;
        for (int i = 1; i <= n; i++) val[i] = 0;
    }
    inline int lowbit(int x) { return x & (-x); }

    void add(int pos, int v) {
        for (int i = pos; i <= n; i += lowbit(i)) val[i] += v;
    }
    int query(int pos) {
        int ans = 0;
        for (int i = pos; i >= 1; i -= lowbit(i)) ans += val[i];
        return ans;
    }
} tree;
struct Edge {
    int to, nex;
    ll w;
} e[MAXN << 1];
int head[MAXN], tol;
void addEdge(int u, int v, ll w) {
    e[tol].to = v, e[tol].w = w, e[tol].nex = head[u], head[u] = tol, tol++;
}
int son[MAXN], f[MAXN], vis[MAXN];
int dis[MAXN];
ll wis[MAXN];

int main() {
    int n, l; ll w; scanf("%d%d%lld", &n, &l, &w);
    tol = 0;
    for (int i = 1; i <= n; i++) head[i] = -1;
    for (int i = 2; i <= n; i++) {
        int v; ll w; scanf("%d%lld", &v, &w);
        addEdge(i, v, w), addEdge(v, i, w);
    }
```

```
    int root = 0, sum = n;
    f[0] = n;
    function<void(int, int)> get_root = [&](int u, int fa) {
        son[u] = 1, f[u] = 0;
        for (int i = head[u]; ~i; i = e[i].nex) {
            int v = e[i].to;
            if (vis[v] || v == fa) continue;
            get_root(v, u);
            son[u] += son[v], f[u] = max(f[u], son[v]);
        }
        f[u] = max(f[u], sum - son[u]);
        if (f[u] < f[root]) root = u;
    };
    get_root(1, 0);

    vector<int> vec;
    function<void(int, int)> get_dis = [&](int u, int fa) {
        vec.pb(u);
        for (int i = head[u]; ~i; i = e[i].nex) {
            int v = e[i].to;
            ll w = e[i].w;
            if (v == fa || vis[v]) continue;
            dis[v] = dis[u] + 1, wis[v] = wis[u] + w;
            get_dis(v, u);
        }
    };
    auto cal = [&](int u, int x1, ll x2) {
        vec.clear();
        dis[u] = x1, wis[u] = x2, get_dis(u, 0);
        sort(vec.begin(), vec.end(), [&](int ta, int tb) {
            return wis[ta] < wis[tb];
        });
        tree.init(n+1);
        for (int i = 0; i < SZ(vec); i++) tree.add(dis[vec[i]] + 1, 1);
        ll ans = 0;
        int L = 0, R = SZ(vec) - 1;
        while (L < R) {
            if (wis[vec[L]] + wis[vec[R]] <= w) {
                tree.add(dis[vec[L]] + 1, -1);
                ans += tree.query(l - dis[vec[L]] + 1);
                L++;
            } else {
                tree.add(dis[vec[R]] + 1, -1);
                R--;
            }
        }
        tree.add(dis[vec[L]] + 1, -1);
        return ans;
    };

    ll res = 0;
    function<void(int)> solve = [&](int u) {
        res += cal(u, 0, 0);
        vis[u] = 1;
        for (int i = head[u]; ~i; i = e[i].nex) {
            int v = e[i].to;
```

```
            ll w = e[i].w;
            if (vis[v]) continue;
            res -= cal(v, 1, w);
            root = 0, sum = son[v];
            get_root(v, 0);
            solve(root);
        }
    };
    solve(root);
    printf("%lld\n", res);
}
```

2.2 DSU

2.2.1 询问子树颜色种类数

```
const int MAXN = 1e5 + 5;
struct Edge {
    int to, nex;
} e[MAXN << 1];
int head[MAXN], tol;
void addEdge(int u, int v) {
    e[tol].to = v, e[tol].nex = head[u], head[u] = tol, tol++;
}
int son[MAXN], siz[MAXN];
int col[MAXN], cnt[MAXN], res[MAXN];
int main() {
    int n; scanf("%d", &n);
    tol = 0;
    for (int i = 1; i <= n; i++) head[i] = -1;
    for (int i = 2; i <= n; i++) {
        int u, v; scanf("%d%d", &u, &v);
        addEdge(u, v), addEdge(v, u);
    }
    for (int i = 1; i <= n; i++) scanf("%d", &col[i]);
    function<void(int, int)> dfs1 = [&](int u, int f) {
        siz[u] = 1;
        int maxson = -1;
        for (int i = head[u]; ~i; i = e[i].nex) {
            int v = e[i].to;
            if (v == f) continue;
            dfs1(v, u);
            siz[u] += siz[v];
            if (siz[v] > maxson) son[u] = v, maxson = siz[v];
        }
    };
    dfs1(1, 0);
    int ans = 0, son_son;
    function<void(int, int, int)> dfs3 = [&](int u, int f, int val) {
        cnt[col[u]] += val;
        if (val == 1 && cnt[col[u]] == 1) ans++;
        else if (val == -1 && cnt[col[u]] == 0) ans--;
        for (int i = head[u]; ~i; i = e[i].nex) {
            int v = e[i].to;
```

```
            if (v == f || v == son_son) continue;
            dfs3(v, u, val);
        }
    };
    function<void(int, int, bool)> dfs2 = [&](int u, int f, bool kp) {
        for (int i = head[u]; ~i; i = e[i].nex) {
            int v = e[i].to;
            if (v == son[u] || v == f) continue;
            dfs2(v, u, 0);
        }
        if (son[u]) dfs2(son[u], u, 1), son_son = son[u];
        dfs3(u, f, 1), son_son = -1;
        res[u] = ans;
        if (!kp) dfs3(u, f, -1);
    };
    dfs2(1, 0, 0);
    int m; scanf("%d", &m);
    while (m--) {
        int x; scanf("%d", &x);
        printf("%d\n", res[x]);
    }
}
```

2.3 无向图相邻边成对

```
vector<pii > res;
int vis[MAXN];
int dfs(int u) {
    vis[u] = 1;
    int cur = -1;
    for (int i = head[u]; ~i; i = e[i].nex) {
        int v = e[i].to, id = e[i].id;
        if (vis[v] == 1) continue;
        int tx1 = id;
        if (!vis[v]) {
            int tx2 = dfs(v);
            if (tx2 != -1) {
                res.pb(mp(tx1, tx2));
                tx1 = -1;
            }
        }
        if (tx1 != -1) {
            if (cur != -1) {
                res.pb(mp(tx1, cur));
                cur = -1;
            } else cur = tx1;
        }
    }
    vis[u] = 2;
    return cur;
}

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

```
    int cnt = 0;
    for (int i = 1; i <= n; i++) {
        ll a, b, c, d;
        scanf("%lld%lld%lld%lld", &a, &b, &c, &d);
        ll gcdd1 = __gcd(c * b, d * (a + b)), gcdd2 = __gcd(b * (c + d), d * a);
        pll ks1 = mp(c * b / gcdd1, d * (a + b) / gcdd1);
        pll ks2 = mp(b * (c + d) / gcdd2, d * a / gcdd2);

        if (ma.find(ks1) == ma.end()) {
            ma[ks1] = ++cnt, head[cnt] = -1;
        }
        if (ma.find(ks2) == ma.end()) {
            ma[ks2] = ++cnt, head[cnt] = -1;
        }
        addEdge(ma[ks1], ma[ks2], i), addEdge(ma[ks2], ma[ks1], i);
    }
    for (int i = 1; i <= cnt; i++) {
        if (!vis[i]) dfs(i);
    }
    printf("%d\n", SZ(res));
    for (auto e: res) { printf("%d %d\n", e.first, e.second); }

}
```

2.4 BFS TREE(CF1496F)

给定一张无向图，取任意两点进行如下操作：

以这两点 x,y 为源构造生成树，使得对于任意点 k，有 $dis[x][k]=min(dis[x][k])\&\&dis[y][k]=min(dis[y][k])$

即 x,y 点与 k 点的距离即为 x,y 与 k 的最短路径（之一），对于每一对 x,y 求能构造的生成树有多少

记 $dist(x,y)$ 为从点 x 到点 y 所经过的点的个数。

有两点性质：

1. 对于点 z，当 $dist(x,z)+dist(y,z)-1=dist(x,y)$，那么点 z 应当是在从 x 到 y 的最短路上。特别的，当这样的点的个树超过 $dist(x,y)$ 个时，那么 x 和 y 作为根节点的 BFS 树同构必定不存在。

2. 对于其他不在从 x 到 y 的最短路的点 u，要存在相邻的点 v，使得 $dist(x,v)=dist(x,u)-1$ 并且 $dist(y,v)=dist(y,u)-1$。

```
int dis[MAXN][MAXN];
void bfs(int s) {
    queue<int> q; dis[s][s] = 1; q.push(s);
    while (!q.empty()) {
        int u = q.front(); q.pop();
        for (int i = head[u]; ~i; i = e[i].nex) {
            int v = e[i].to;
            if (!dis[s][v]) {
                dis[s][v] = dis[s][u] + 1;
                q.push(v);
            }
        }
    }
}
ll res[MAXN][MAXN];
int main() {
    int n, m; scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++) head[i] = -1;
    while (m--) {
        int u, v;
```

```
        scanf("%d%d", &u, &v);
        addEdge(u, v), addEdge(v, u);
    }
    for (int i = 1; i <= n; i++) bfs(i);
    for (int x = 1; x <= n; x++) {
        for (int y = x; y <= n; y++) {
            int cnt = 0;
            for (int i = 1; i <= n; i++) {
                if (dis[x][i] + dis[y][i] - 1 == dis[x][y]) cnt++; // i int the path of x to y
            }
            ll ans = 1;
            if (cnt > dis[x][y]) ans = 0;
            for (int u = 1; u <= n; u++) {
                if (dis[x][u] + dis[y][u] - 1 != dis[x][y]) {
                    int fg = 0;
                    for (int i = head[u]; ~i; i = e[i].nex) {
                        int v = e[i].to;
                        if (dis[x][v] == dis[x][u] - 1 && dis[y][v] == dis[y][u] - 1) fg++;
                    }
                    ans = ans * fg % mod;
                    if (!ans) break;
                }
            }
            res[x][y] = res[y][x] = ans;
        }
    }
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            printf("%d ", res[i][j]);
        }
        printf("\n");
    }
}
```

3 整体二分

3.1 每次询问一个子矩阵的第 k 小数

```
/*  [input]         [output]
    2 2
    2 1
    3 4
    1 2 1 2 1       1
    1 1 2 2 3       3           */
class BIT { public:
    int val[MAXN][MAXN]; int n, m;
    void init(int _n, int _m) {
        n = _n, m = _m;
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= m; j++)
                val[i][j] = 0;
    }
    inline int lowbit(int x) { return x & (-x); }
```

```
    void add(int x, int y, int v) {
        for (int i = x; i <= n; i += lowbit(i))
            for (int j = y; j <= m; j += lowbit(j))
                val[i][j] += v;
    }
    inline int query(int x, int y) {
        int ans = 0;
        for (int i = x; i >= 1; i -= lowbit(i))
            for (int j = y; j >= 1; j -= lowbit(j))
                ans += val[i][j];
        return ans;
    }
    int query(int x1, int y1, int x2, int y2) {
        return query(x2, y2) - query(x1 - 1, y2) - query(x2, y1 - 1) + query(x1 - 1, y1 - 1);
    }
} bit;

struct Query {
    int x1, y1, x2, y2, k, val, id, type;
} q[MAXM], q1[MAXM], q2[MAXM];
int res[MAXM];
int a[MAXN][MAXN];

int main() {
    int n, m; scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++)
            scanf("%d", &a[i][j]);
    }
    int qcnt = 0;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            q[++qcnt] = {i, j, i, j, 0, a[i][j], 0, 1};
        }
    }
    for (int i = 1; i <= m; i++) {
        int x1, y1, x2, y2, k;
        scanf("%d%d%d%d%d", &x1, &y1, &x2, &y2, &k);
        q[++qcnt] = {x1, y1, x2, y2, k, 0, i, 2};
    }
    bit.init(n, n);
    function<void(int, int, int, int)> solve = [&](int l, int r, int ql, int qr) {
        if (ql > qr) return;
        if (l == r) {
            for (int i = ql; i <= qr; i++) {
                if (q[i].type == 2) res[q[i].id] = l;
            }
            return;
        }
        int mid = (l + r) >> 1;
        int cnt1 = 0, cnt2 = 0;
        for (int i = ql; i <= qr; i++) {
            if (q[i].type == 1) {
                if (q[i].val <= mid) {
                    bit.add(q[i].x1, q[i].y1, 1);
                    q1[++cnt1] = q[i];
```

```
                } else q2[++cnt2] = q[i];
            } else {
                int d = bit.query(q[i].x1, q[i].y1, q[i].x2, q[i].y2);
                if (q[i].k <= d) q1[++cnt1] = q[i];
                else {
                    q[i].k -= d;
                    q2[++cnt2] = q[i];
                }
            }
        }
        for (int i = 1; i <= cnt1; i++) {
            if (q1[i].type == 1)
                bit.add(q1[i].x1, q1[i].y1, -1);
        }
        for (int i = 1; i <= cnt1; i++) q[ql + i - 1] = q1[i];
        for (int i = 1; i <= cnt2; i++) q[ql + cnt1 + i - 1] = q2[i];
        solve(l, mid, ql, ql + cnt1 - 1);
        solve(mid + 1, r, ql + cnt1, qr);

    };
    solve(0, 1e9, 1, qcnt);
    for (int i = 1; i <= m; i++) printf("%d\n", res[i]);
}
```

3.2 带修主席树

```
class BIT { public:
    int val[MAXN], n;
    void init(int _n) {
        n = _n;
        for (int i = 1; i <= n; i++) val[i] = 0;
    }
    inline int lowbit(int x) { return x & (-x); }
    void add(int pos, int v) {
        for (int i = pos; i <= n; i += lowbit(i)) val[i] += v;
    }
    int query(int pos) {
        int ans = 0;
        for (int i = pos; i >= 1; i -= lowbit(i)) ans += val[i];
        return ans;
    }
    int query(int l, int r) { return query(r) - query(l - 1); }
} tree;
struct Query {
    int l, r, k, val, id, type;
} q[MAXN << 1], q1[MAXN << 1], q2[MAXN << 1];
int a[MAXN], res[MAXN], type[MAXN];
int main() {
    int n, m; scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
    int qcnt = 0;
    for (int i = 1; i <= n; i++) {
        q[++qcnt] = {0, 0, 0, a[i], i, 1};
    }
```

```
    for (int i = 1; i <= m; i++) {
        char opt[3];
        scanf("%s", opt);
        if (opt[0] == 'Q') {
            int l, r, k;
            scanf("%d%d%d", &l, &r, &k);
            q[++qcnt] = {l, r, k, 0, i, 2};
            type[i] = 1;
        } else {
            int x, y;
            scanf("%d%d", &x, &y);
            q[++qcnt] = {0, 0, 0, a[x], x, 3};
            q[++qcnt] = {0, 0, 0, y, x, 1};
            a[x] = y;
        }
    }

    tree.init(n);
    function<void(int, int, int, int)> solve = [&](int l, int r, int ql, int qr) {
        if (ql > qr) return;
        if (l == r) {
            for (int i = ql; i <= qr; i++) {
                if (q[i].type == 2) res[q[i].id] = l;
            }
            return;
        }
        int mid = (l + r) >> 1;
        int cnt1 = 0, cnt2 = 0;
        for (int i = ql; i <= qr; i++) {
            if (q[i].type == 1) {
                if (q[i].val <= mid) {
                    tree.add(q[i].id, 1);
                    q1[++cnt1] = q[i];
                } else {
                    q2[++cnt2] = q[i];
                }
            } else if (q[i].type == 3) {
                if (q[i].val <= mid) {
                    tree.add(q[i].id, -1);
                    q1[++cnt1] = q[i];
                } else {
                    q2[++cnt2] = q[i];
                }
            } else {
                int d = tree.query(q[i].l, q[i].r);
                if (q[i].k <= d) {
                    q1[++cnt1] = q[i];
                } else {
                    q[i].k -= d;
                    q2[++cnt2] = q[i];
                }
            }
        }
        for (int i = 1; i <= cnt1; i++) {
            if (q1[i].type == 1) tree.add(q1[i].id, -1);
            else if (q1[i].type == 3) tree.add(q1[i].id, 1);
```

```
        }
        for (int i = 1; i <= cnt1; i++) q[ql + i - 1] = q1[i];
        for (int i = 1; i <= cnt2; i++) q[ql + cnt1 + i - 1] = q2[i];
        solve(l, mid, ql, ql + cnt1 - 1);
        solve(mid + 1, r, ql + cnt1, qr);
    };
    solve(0, 1e9, 1, qcnt);
    for (int i = 1; i <= m; i++) {
        if (type[i]) printf("%d\n", res[i]);
    }
}
```

3.3 在右半边的整体二分

```
class BIT_LL { public:
    __int128 val[MAXN], n;
    void init(int _n) {
        n = _n;
        for (int i = 1; i <= n; i++) val[i] = 0;
    }
    inline int lowbit(int x) { return x & (-x); }
    void add(int pos, __int128 v) {
        for (int i = pos; i <= n; i += lowbit(i)) val[i] += v;
    }
    __int128 query(int pos) {
        __int128 ans = 0;
        for (int i = pos; i >= 1; i -= lowbit(i)) ans += val[i];
        return ans;
    }
    __int128 query(int L, int R) {
        return query(R) - query(L - 1);
    }
} bit1, bit2;
struct Query {
    int d, p, l;
    ll g, L;
    int id;
} q[MAXN << 1], q1[MAXN << 1], q2[MAXN << 1];
int res[MAXN], type[MAXN];
int main() {
    int n, m; scanf("%d%d", &n, &m);
    bit1.init(1e5), bit2.init(1e5);
    int qcnt = 0;
    for (int i = 1; i <= n; i++) {
        int d, p, l;
        scanf("%d%d%d", &d, &p, &l);
        q[++qcnt] = {d, p, l, 0, 0, 0};
        bit1.add(q[i].p, (ll) l);
        bit2.add(q[i].p, (ll) l * p);
    }
    for (int i = 1; i <= m; i++) {
        ll g, L;
        scanf("%lld%lld", &g, &L);
        ll tmp = bit1.query(1e5);
```

```
        if (L > tmp) {
            type[i] = 0;
        } else {
            int posL = 1, posR = 1e5;
            while (posL < posR) {
                int mid = (posL + posR) >> 1;
                if (bit1.query(mid) < L) posL = mid + 1;
                else posR = mid;
            }
            int pos = posL;

            ll momo = bit2.query(pos - 1) +
                      (L - bit1.query(pos - 1)) * (bit2.query(pos, pos) / bit1.query(pos,
                          pos));
            if (momo > g) type[i] = 0;
            else type[i] = 1, q[++qcnt] = {0, 0, 0, g, L, i};
        }
    }
    bit1.init(1e5), bit2.init(1e5);
    function<void(int, int, int, int)> solve = [&](int l, int r, int ql, int qr) {
        if (ql > qr) return;
        if (l == r) {
            for (int i = ql; i <= qr; i++) {
                if (q[i].id && type[q[i].id]) {
                    res[q[i].id] = l;
                }
            }
            return;
        }
        int mid = (l + r + 1) >> 1;
        int cnt1 = 0, cnt2 = 0;
        for (int i = ql; i <= qr; i++) {
            if (q[i].id == 0) {
                if (q[i].d >= mid) {
                    bit1.add(q[i].p, (ll) q[i].l);
                    bit2.add(q[i].p, (ll) q[i].l * q[i].p);
                    q2[++cnt2] = q[i];
                } else q1[++cnt1] = q[i];
            } else {
                ll tmp = bit1.query(1e5);
                if (q[i].L > tmp) {
                    q1[++cnt1] = q[i];
                } else {
                    int posL = 1, posR = 1e5;
                    while (posL < posR) {
                        int mid = (posL + posR) >> 1;
                        if (bit1.query(mid) < q[i].L) posL = mid + 1;
                        else posR = mid;
                    }
                    int pos = posL;
                    __int128 momo = (__int128) bit2.query(pos - 1) +
                                    ((__int128) q[i].L - (__int128) bit1.query(pos - 1)) *
                                    (__int128) ((__int128) bit2.query(pos, pos) / (__int128)
                                        bit1.query(pos, pos));
                    if (momo > q[i].g) q1[++cnt1] = q[i];
                    else q2[++cnt2] = q[i];
```

```
                }
            }
        }
        for (int i = 1; i <= cnt1; i++) q[ql + i - 1] = q1[i];
        for (int i = 1; i <= cnt2; i++) q[ql + cnt1 + i - 1] = q2[i];
        solve(l, mid - 1, ql, ql + cnt1 - 1);
        for (int i = 1; i <= cnt2; i++) {
            if (q[ql + cnt1 + i - 1].id == 0) {
                bit1.add(q[ql + cnt1 + i - 1].p, 0ll - (ll) q[ql + cnt1 + i - 1].l);
                bit2.add(q[ql + cnt1 + i - 1].p, 0ll - (ll) q[ql + cnt1 + i - 1].l * q[ql +
                    cnt1 + i - 1].p);
            }
        }
        solve(mid, r, ql + cnt1, qr);
    };
    solve(1, 1e5, 1, qcnt);
    for (int i = 1; i <= m; i++)
        if (type[i]) printf("%d\n", res[i]);
        else printf("-1\n");
}
```

4 计算几何

4.1 凸包

```
struct Point {
    double x, y;
    Point(double _x = 0, double _y = 0) : x(_x), y(_y) {}
    Point operator-(const Point &tb) { return Point(x - tb.x, y - tb.y); }
    double dis(const Point &tb) {
        return sqrt((x - tb.x) * (x - tb.x) + (y - tb.y) * (y - tb.y));
    }
};
double cross(const Point &ta, const Point &tb) {
    return ta.x * tb.y - ta.y * tb.x;
}
namespace Convex {
    vector<Point> GetConvexHull(const vector<Point> &P) {
        vector<Point> ans, p(P);
        int n = SZ(p), t = -1;
        sort(p.begin(), p.end(), [&](const Point &ta, const Point &tb) {
            if (ta.x != tb.x) return ta.x < tb.x;
            else return ta.y < tb.y;
        }), ans.assign(n * 2, Point(0, 0));
        auto ins = [&](int pre, const Point &u) {
            while (t > pre && cross(ans[t - 1] - u, ans[t] - u) >= 0) t--;
            ans[++t] = u;
        };
        for (auto u:p) ins(0, u);
        for (int i = n - 2, pre = t; i >= 0; i--) ins(pre, p[i]);
        ans.resize(t);
        return ans;
    }
```

```
}
vector<Point> vec;
int main() {
    int n; scanf("%d", &n);
    vec.resize(n);
    for (int i = 0; i < n; i++) {
        scanf("%lf%lf", &vec[i].x, &vec[i].y);
    }
    vector<Point> v = Convex::GetConvexHull(vec);
    double res = 0.0;
    for (int i = 0; i < SZ(v); i++) {
        if (i == 0) res += v[i].dis(v[SZ(v) - 1]);
        else res += v[i].dis(v[i - 1]);
    }
    printf("%.2lf\n", res);
}
```

4.2 四边形

4.2.1 四边形计数

```
struct Point {
    ll x, y;
    Point() {}
    Point(ll _x, ll _y) : x(_x), y(_y) {}
    Point operator-(const Point &tb) const {
        return Point(x - tb.x, y - tb.y);
    }
    ll operator^(const Point &tb) const {
        return x * tb.y - y * tb.x;
    }
} p[MAXN];

struct node {
    Point p; int a, b;
    node() {}
    node(Point _p, int _a, int _b) : p(_p), a(_a), b(_b) {}
} nd[MAXN * MAXN];

int rk[MAXN], id[MAXN];
int main() {
    int T; scanf("%d", &T);
    while (T--) {
        int n; scanf("%d", &n);
        for (int i = 1; i <= n; i++) scanf("%lld%lld", &p[i].x, &p[i].y);
        sort(p + 1, p + 1 + n, [&](const Point &ta, const Point &tb) {
            if (ta.x != tb.x) return ta.x < tb.x;
            else return ta.y < tb.y;
        });
        int nd_tot = 0;
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j < i; j++) nd[++nd_tot] = node(p[i] - p[j], i, j);
        }
        sort(nd + 1, nd + 1 + nd_tot, [&](const node &ta, const node &tb) {
```

```
            ll tmp = ta.p ^tb.p;
            return tmp < 0;
        });
        ll uconv = (ll) n * (n - 1) * (n - 2) * (n - 3) / 6;
        for (int i = 1; i <= n; i++) rk[i] = id[i] = i;
        for (int i = 1; i <= nd_tot; i++) {
            int a = nd[i].a, b = nd[i].b;
            if (rk[a] > rk[b]) swap(a, b);
            int na = rk[a] - 1, nb = n - rk[b];
            uconv -= (ll) na * (na - 1) / 2;
            uconv -= (ll) nb * (nb - 1) / 2;
            swap(rk[a], rk[b]);
            swap(id[rk[a]], id[rk[b]]);
        }
        ll conv = (ll) n * (n - 1) * (n - 2) * (n - 3) / 24 - uconv;
        printf("%lld\n", conv);
    }

}
```

4.2.2 四边形最小面积计数

```
struct Point {
    ll x, y;
    Point() {}
    Point(ll _x, ll _y) : x(_x), y(_y) {}
    Point operator-(const Point &tb) const {
        return Point(x - tb.x, y - tb.y);
    }
    ll operator^(const Point &tb) const {
        return x * tb.y - y * tb.x;
    }
} p[MAXN];
struct node {
    Point p; int a, b;
    node() {}
    node(Point _p, int _a, int _b) : p(_p), a(_a), b(_b) {}
} nd[MAXN * MAXN];
int rk[MAXN], id[MAXN];
int main() {
    int n; scanf("%d", &n);
    for (int i = 1; i <= n; i++) scanf("%lld%lld", &p[i].x, &p[i].y);
    sort(p + 1, p + 1 + n, [&](const Point &ta, const Point &tb) {
        if (ta.x != tb.x) return ta.x < tb.x;
        else return ta.y < tb.y;
    });
    int nd_tot = 0;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j < i; j++) nd[++nd_tot] = node(p[i] - p[j], i, j);
    }
    sort(nd + 1, nd + 1 + nd_tot, [&](const node &ta, const node &tb) {
        ll tmp = ta.p ^tb.p;
        return tmp < 0;
    });
```

```
    ll uconv = (ll) n * (n - 1) * (n - 2) * (n - 3) / 6;
    ll area = LLONG_MAX; // llinf?  dog never use!
    ll cnt_uconv = 0, cnt_conv = 0;
    for (int i = 1; i <= n; i++) rk[i] = id[i] = i;
    for (int i = 1; i <= nd_tot; i++) {
        Point cen = nd[i].p;
        int a = nd[i].a, b = nd[i].b;
        if (rk[a] > rk[b]) swap(a, b);
        int na = rk[a] - 1, nb = n - rk[b];
        uconv -= (ll) na * (na - 1) / 2;
        uconv -= (ll) nb * (nb - 1) / 2;
        if (1 <= rk[a] - 1 && rk[b] + 1 <= n) {
            ll left = abs(cen ^ (p[a] - p[id[rk[a] - 1]])), right = abs(cen ^ (p[b] -
                p[id[rk[b] + 1]]));
            ll cur_area = left + right;
            if (cur_area < area) area = cur_area, cnt_uconv = cnt_conv = 0;
            if (cur_area == area) {
                vi l, r;
                l.pb(id[rk[a] - 1]), r.pb(id[rk[b] + 1]);
                for (int j = rk[a] - 2; j >= 1; j--) {
                    if (abs(cen ^ (p[id[j]] - p[a])) == left) l.pb(id[j]);
                }
                for (int j = rk[b] + 2; j <= n; j++) {
                    if (abs(cen ^ (p[id[j]] - p[b])) == right) r.pb(id[j]);
                }
                for (auto &ea:l) {
                    for (auto &eb: r) {
                        vector<Point> tmp{p[ea], p[a], p[eb], p[b]};
                        int s = signbit((tmp[1] - tmp[0]) ^ (tmp[2] - tmp[0]));
                        bool f = 0;
                        for (int j = 1; j < 4; j++) {
                            if (s != signbit((tmp[(j + 1) % 4] - tmp[j]) ^ (tmp[(j + 2) % 4] -
                                tmp[j]))) {
                                f = 1; break;
                            }
                        }
                        if (f) cnt_uconv++;
                        else cnt_conv++;
                    }
                }
            }
        }
        swap(rk[a], rk[b]); swap(id[rk[a]], id[rk[b]]);
    }
    ll conv = (ll) n * (n - 1) * (n - 2) * (n - 3) / 24 - uconv;
    cnt_conv /= 2;
    ll res = 4ll * cnt_conv + 3ll * cnt_uconv + 2ll * (conv - cnt_conv) + 1ll * (uconv * 3 -
        cnt_uconv);
    printf("%lld\n", res);
}
```

5 习题整理

5.1 dfs+2019 银川 A

```
const int MAXN = 1e5 + 5;
string name[MAXN], color[MAXN], bo_name[6], bo_color;
int val[MAXN];
vector<int> vec[4], v2[4];
int main() {
    cin.tie(0), cin.sync_with_stdio(0);
    int T; cin >> T;
    while (T--) {
        int n; cin >> n;
        for (int i = 1; i <= n; i++) {
            cin >> name[i] >> color[i] >> val[i];
        }
        for (int i = 1; i <= 5; i++) cin >> bo_name[i];
        cin >> bo_color;
        for (int i = 0; i < 4; i++) vec[i].clear(), v2[i].clear();
        for (int i = 1; i <= n; i++) {
            int fg1 = 0, fg2 = 0;
            for (int j = 1; j <= 5; j++) {
                if (name[i] == bo_name[j]) {
                    fg1 = 1;
                    break;
                }
            }
            if (color[i] == bo_color) fg2 = 1;
            if (fg1 == 0 && fg2 == 0) vec[0].pb(i);
            else if (fg1 == 0 && fg2 == 1) vec[1].pb(i);
            else if (fg1 == 1 && fg2 == 0) vec[2].pb(i);
            else vec[3].pb(i);
        }
        for (int i = 0; i < 4; i++) {
            sort(vec[i].begin(), vec[i].end(), [&](int ta, int tb) {
                if (name[ta] != name[tb]) return name[ta] < name[tb];
                return val[ta] > val[tb];
            });
            string pre = "";
            for (auto e: vec[i]) {
                if (name[e] != pre) {
                    v2[i].pb(e);
                    pre = name[e];
                }
            }
            sort(v2[i].begin(), v2[i].end(), [&](int ta, int tb) {
                return val[ta] > val[tb];
            });
        }
        int res = 0;
        set<string> st;
        function<void(int, int, int, int, int, int, int, int)> dfs = [&](int pos, int ri1, int
            ri2, int base, int t0,
```

```
                                                                                int t1, int t2, int
                                                                                    t3) {
            if (pos == 6) {
                double ri = 1.0 + 0.1 * ri1 + 0.2 * ri2;
                res = max(res, (int) floor(ri * base));
                return;
            }
            if (t0 < SZ(v2[0])) {
                if (!st.count(name[v2[0][t0]])) {
                    st.insert(name[v2[0][t0]]);
                    dfs(pos + 1, ri1, ri2, base + val[v2[0][t0]], t0 + 1, t1, t2, t3);
                    st.erase(name[v2[0][t0]]);
                } else dfs(pos, ri1, ri2, base, t0 + 1, t1, t2, t3);
            }
            if (t1 < SZ(v2[1])) {
                if (!st.count(name[v2[1][t1]])) {
                    st.insert(name[v2[1][t1]]);
                    dfs(pos + 1, ri1, ri2 + 1, base + val[v2[1][t1]], t0, t1 + 1, t2, t3);
                    st.erase(name[v2[1][t1]]);
                } else dfs(pos, ri1, ri2, base, t0, t1 + 1, t2, t3);
            }
            if (t2 < SZ(v2[2])) {
                if (!st.count(name[v2[2][t2]])) {
                    st.insert(name[v2[2][t2]]);
                    dfs(pos + 1, ri1 + 1, ri2, base + val[v2[2][t2]], t0, t1, t2 + 1, t3);
                    st.erase(name[v2[2][t2]]);
                } else dfs(pos, ri1, ri2, base, t0, t1, t2 + 1, t3);
            }
            if (t3 < SZ(v2[3])) {
                if (!st.count(name[v2[3][t3]])) {
                    st.insert(name[v2[3][t3]]);
                    dfs(pos + 1, ri1 + 1, ri2 + 1, base + val[v2[3][t3]], t0, t1, t2, t3 + 1);
                    st.erase(name[v2[3][t3]]);
                } else dfs(pos, ri1, ri2, base, t0, t1, t2, t3 + 1);
            }
        };
        dfs(1, 0, 0, 0, 0, 0, 0, 0);
        cout << res << '\n';
    }
}
```

5.2 链哈 LCA 倍增

```
struct LCA {
    int fa[MAXN][22], dep[MAXN], n, limt, bin[22], len[MAXN];
    ll sd[2], p[2], ha[MAXN][2], tmp[MAXN][2];
    vector<int> mp[MAXN];
    void init(int _n) {
        n = _n;
        for (limt = 1; 1 << (limt - 1) <= n; limt++);
        for (int i = bin[0] = 1; 1 << (i - 1) <= n; i++) bin[i] = (bin[i - 1] << 1);
        sd[0] = 13331, sd[1] = 23333;
        p[0] = 1e9 + 7, p[1] = 998244353;
        mem(ha[0], 0);
```

```
        tmp[0][0] = tmp[0][1] = 1;
        mem(fa[0], 0);
        for (int i = 1; i <= n; i++) {
            mp[i].clear();
            mem(fa[i], 0);
            tmp[i][0] = tmp[i - 1][0] * sd[0] % p[0];
            tmp[i][1] = tmp[i - 1][1] * sd[1] % p[1];
        }
    }
    void add_edge(int a, int b) { mp[a].pb(b); }
    void dfs(int x, int pre) {
        ha[x][0] = (ha[pre][0] * sd[0] + len[x]) % p[0];
        ha[x][1] = (ha[pre][1] * sd[1] + len[x]) % p[1];
        for (int i = 1; bin[i] <= dep[x]; i++) fa[x][i] = fa[fa[x][i - 1]][i - 1];
        for (int i = 0; i < sz(mp[x]); i++) {
            int to = mp[x][i];
            if (to == pre) continue;
            dep[to] = dep[x] + 1;
            fa[to][0] = x;
            dfs(to, x);
        }
    }
    void work(int rt) {
        dep[rt] = 0;
        dfs(rt, 0);
    }
    int find(int a, int L) {
        if (len[a] == L) return a;
        for (int i = limt; ~i; i--) {
            if (len[fa[a][i]] > L) a = fa[a][i];
        }
        assert(len[fa[a][0]] == L);
        return fa[a][0];
    }
    ll get(int l, int r, int f) {
        int LEN = dep[r] - dep[l] + 1;
        if (l == 0) return ha[r][f];
        return ((ha[r][f] - ha[fa[l][0]][f] * tmp[LEN][f]) % p[f] + p[f]) % p[f];
    }
    void comp(int x, int y) {
        int tx, ty;
        tx = x, ty = y;
        if (len[tx] != len[ty]) {
            if (len[tx] < len[ty]) puts("sjfnb");
            else if (len[tx] > len[ty]) puts("cslnb");
            return;
        }
        for (int i = limt; ~i; i--) {
            if (get(fa[tx][i], tx, 0) == get(fa[ty][i], ty, 0) &&
                get(fa[tx][i], tx, 1) == get(fa[ty][i], ty, 1)) {
                tx = fa[tx][i];
                ty = fa[ty][i];
            }
        }
        tx = fa[tx][0];
        ty = fa[ty][0];
```

```
        if (len[tx] < len[ty]) puts("sjfnb");
        else if (len[tx] > len[ty]) puts("cslnb");
        else puts("draw");
    }
} lca;
int pos[MAXN];
struct Palindrome_Tree {
    struct node {
        int ch[MAXC];
        int fail, len;
    } T[MAXN];
    int las, tol;
    int c[MAXN];  // cnt 计数, pos记录位置
    int get_fail(int x, int pos) {
        while (c[pos - T[x].len - 1] != c[pos]) {
            x = T[x].fail;
        }
        return x;
    }
    void init() { // 传入字符串长度
        memset(T[0].ch, 0, sizeof(T[0].ch));
        memset(T[1].ch, 0, sizeof(T[1].ch));
        T[0].len = 0, T[1].len = -1;
        T[0].fail = 1, T[1].fail = 0;
        las = 0; tol = 1;
    }
    void insert(char s[], int len) {
        c[0] = -1;
        for (int i = 1; i <= len; i++) {
            c[i] = s[i] - 'a';
            int p = get_fail(las, i);
            if (!T[p].ch[c[i]]) {
                T[++tol].len = T[p].len + 2;
                memset(T[tol].ch, 0, sizeof(T[tol].ch));
                int u = get_fail(T[p].fail, i);
                T[tol].fail = T[u].ch[c[i]];
                T[p].ch[c[i]] = tol;
            }
            las = T[p].ch[c[i]];
            pos[i] = las;
        }
    }
    void dfs(int u) {
        printf("u = %d T[u].fail = %d T[u].len = %d\n", u, T[u].fail, T[u].len);
        for (int i = 0; i < 26; i++) {
            if (T[u].ch[i]) {
                printf("%d %d %d\n", u, i, T[u].ch[i]);
                dfs(T[u].ch[i]);
            }
        }
    }
    void build() {
        lca.init(tol + 1);
        for (int i = 1; i <= tol; i++) lca.add_edge(T[i].fail + 1, i + 1);
        for (int i = 0; i <= tol; i++) lca.len[i + 1] = T[i].len;
        lca.work(1);
```

```
    }
} tree;
char s[MAXN];
int main() {
    int T; scanf("%d", &T);
    while (T--) {
        int len;
        scanf("%d", &len);
        scanf("%s", s + 1);
        tree.init();
        tree.insert(s, len);
        tree.build();
        int q; scanf("%d", &q);
        while (q--) {
            int a, b, c, d;
            scanf("%d%d%d%d", &a, &b, &c, &d);
            int x = lca.find(pos[b] + 1, b - a + 1);
            int y = lca.find(pos[d] + 1, d - c + 1);
            lca.comp(x, y);
        }
    }
}
```

5.3 后缀自动机 +set

给定一个长度为 n 的字符串，对于前缀 $1..i$，找到最短的字符串，使其在整个长度为 n 的字符串中只出现一次，输出长度。

对于后缀自动机，其实是有三种情况进行分类讨论的：

1. 直接连到起始节点，这种情况不会出现重复。
2. 直连，类似 aa，这种情况会出现重复。
3. 之前出现过的状态，需要复制节点信息，这种情况会出现重复。

```
struct node {
    int id, len;
    node(int _id = 0, int _len = 0) : id(_id), len(_len) {}
    bool operator<(const node &tb) const {
        if (len != tb.len) return len < tb.len;
        else return id < tb.id;
    }
};
set<node> st;

class SAM {
public:
    int rt, link[MAXN], maxlen[MAXN], trans[MAXN][MAXC];
    int val[MAXN];
    void init() {
        rt = 1;
        link[1] = maxlen[1] = 0;
        memset(trans[1], 0, sizeof(trans[1]));
    }
    int insert(int ch, int last) {
        int z = ++rt, p = last;
        val[z] = 1;
        memset(trans[z], 0, sizeof(trans[z]));
        maxlen[z] = maxlen[last] + 1;
```

```
        while (p && !trans[p][ch])trans[p][ch] = z, p = link[p];
        if (!p) link[z] = 1;
        else {
            int x = trans[p][ch];
            if (maxlen[p] + 1 == maxlen[x]) {
                link[z] = x;
                if (val[x]) val[x] = 0, st.erase(node(x, maxlen[link[x]] + 1));
            } else {
                int y = ++rt;
                maxlen[y] = maxlen[p] + 1;
                if (val[x]) st.erase(node(x, maxlen[link[x]] + 1));
                for (int i = 0; i < MAXC; i++) trans[y][i] = trans[x][i];
                while (p && trans[p][ch] == x) trans[p][ch] = y, p = link[p];
                link[y] = link[x], link[z] = link[x] = y;
                if (val[x]) st.insert(node(x, maxlen[link[x]] + 1));
            }
        }
        st.insert(node(z, maxlen[link[z]] + 1));
        return z;
    }
} sa;

char s[MAXN];

int main() {
    int n;
    scanf("%d", &n);
    sa.init();
    scanf("%s", s + 1);
    int last = 1;
    for (int i = 1; i <= n; i++) {
        last = sa.insert(s[i] - 'a', last);
        set<node>::iterator it = st.begin();
        printf("%d\n", it->len);
    }
}
```

5.4 CF840D Destiny

给定 n 个元素，m 次询问。

每次给出三个参数 l, r, k，询问区间 $[l, r]$ 内是否存在出现次数严格大于 $\frac{r-l+1}{k}$ 的数。如果存在就输出最小的那个 ans，否则输出 -1.

时间复杂度: $O(nk \log n)$

```
class HJT { public:
    int ch[MAXN * 70][2], sum[MAXN * 70];
    int tot = 0;
    inline void push_up(int rt) {
        sum[rt] = sum[ch[rt][0]] + sum[ch[rt][1]];
    }
    int change(int rt, int pos, int val, int be, int en) {
        int nrt = ++tot;
        ch[nrt][0] = ch[nrt][1] = sum[nrt] = 0;
        if (be == en) {
            sum[nrt] = sum[rt] + val;
```

```
            return nrt;
        }
        int mid = (be + en) >> 1;
        if (pos <= mid) {
            ch[nrt][0] = change(ch[rt][0], pos, val, be, mid);
            ch[nrt][1] = ch[rt][1];
        } else {
            ch[nrt][0] = ch[rt][0];
            ch[nrt][1] = change(ch[rt][1], pos, val, mid+1, en);
        }
        push_up(nrt);
        return nrt;
    }
    int query(int lrt, int rrt, int k, int be, int en) {
        if (be >= en) return be;
        int delta = sum[ch[rrt][0]] - sum[ch[lrt][0]];
        int mid = (be + en) >> 1;
        int ans = -1;
        if (delta > k) ans = query(ch[lrt][0], ch[rrt][0], k, be, mid);
        delta = sum[ch[rrt][1]] - sum[ch[lrt][1]];
        if (ans == -1 && delta > k) ans = query(ch[lrt][1], ch[rrt][1], k, mid + 1, en);
        return ans;
    }
} tree;
int ai[MAXN], root[MAXN];
int main() {
    int n, q; scanf("%d%d", &n, &q);
    for (int i = 1; i <= n; i++) scanf("%d", &ai[i]);
    root[0] = 0;
    for (int i = 1; i <= n; i++) {
        root[i] = tree.change(root[i - 1], ai[i], 1, 1, n);
    }
    while (q--) {
        int l, r, k; scanf("%d%d%d", &l, &r, &k);
        int K = (r - l + 1) / k;
        int res = tree.query(root[l-1], root[r], K, 1, n);
        printf("%d\n", res);
    }
}
```