Platelet

Team Reference Material

(25-page version)

凌皓煜
陈　彤
顾　逸

Shanghai Jiao Tong University, 2018

目录

1. Graph Theory

1.1 2-SAT (ct)

```
struct Edge {
    Edge *next;
    int to;
} *last[maxn << 1], e[maxn << 2], *ecnt = e;
inline void link(int a, int b)
{
    *++ecnt = (Edge) {last[a], b}; last[a] = ecnt;
}
int dfn[maxn], low[maxn], timer, st[maxn], top, id[maxn], colcnt, n;
bool fail, used[maxn];
void tarjan(int x, int fa)
{
    dfn[x] = low[x] = ++timer; st[++top] = x;
    for (R Edge *iter = last[x]; iter; iter = iter -> next)
        if (iter -> to != fa)
        {
            if (!dfn[iter -> to])
            {
                tarjan(iter -> to, x);
                cmin(low[x], low[iter -> to]);
            }
            else if (!id[iter -> to]) cmin(low[x], dfn[iter -> to]);
        }
    if (dfn[x] == low[x])
    {
        ++colcnt; bool flag = 1;
        for (; ;)
        {
            int now = st[top--];
            id[now] = colcnt;
            if (now <= 2 * n)
            {
                flag &= !used[id[now <= n ? now + n : now - n]];
                now <= n ? fail |= (id[now + n] == id[now]) : fail |= (id[now - n] == id[now]);
            }
            if (now == x) break;
        }
        used[colcnt] = flag;
    }
}
int ans[maxn], tot;
int main()
{
    /*
        build your graph here.
    */
    for (R int i = 1; !fail && i <= n; ++i) if (!dfn[i]) tarjan(i, 0);
    if (fail)
    {
        puts("Impossible");
        return 0;
    }
    for (R int i = 1; i <= n; ++i) if (used[id[i]]) ans[++tot] = i;
    printf("%d\n", tot);
    std::sort(ans + 1, ans + tot + 1);
    for (R int i = 1; i <= tot; ++i) printf("%d ", ans[i]);
    return 0;
}
```

1.2 割点与桥 (ct)

割点

```
int dfn[maxn], low[maxn], timer, ans, num;
void tarjan(int x, int fa)
{
    dfn[x] = low[x] = ++timer;
    for (Edge *iter = last[x]; iter; iter = iter -> next)
        if (iter -> to != fa)
        {
            if (!dfn[iter -> to])
            {
                tarjan(iter -> to, x);
                cmin(low[x], low[iter -> to]);
                if (dfn[x] <= low[iter -> to])
                {
                    cut[x] = 1;
                    if (!fa && dfn[x] < low[iter -> to]) num = 233;
                    else if (!fa) ++num;
                }
            }
            else cmin(low[x], dfn[iter -> to]);
        }
}
int main()
{
    for (int i = 1; i <= n; ++i)
        if (!dfn[i])
        {
            num = 0;
            tarjan(i, 0);
            if (num == 1) cut[i] = 0;
        }
}
```

桥

```
int dfn[maxn], low[maxn], timer;
void tarjan(int x, int fa)
{
    dfn[x] = low[x] = ++timer;
    for (R Edge *iter = last[x]; iter; iter = iter -> next)
        if (iter -> to != fa)
        {
            if (!dfn[iter -> to])
            {
                dfs(iter -> to, x);
                cmin(low[x], low[iter -> to]);
                if (dfn[x] < low[iter -> to]) ans[x][iter -> to] = ans[iter -> to][x] = 1;
            }
            else cmin(low[x], dfn[iter -> to]);
```

```
        }
}
```

1.3 Steiner tree (lhy)

```
void Steiner_Tree()
{
    memset(f, 0x3f, sizeof(f));
    for(int i = 1; i <= n; i++)
        f[0][i] = 0;
    for(int i = 1; i <= p; i++)
        f[1 << (i - 1)][idx[i]] = 0;
    int S = 1 << p;
    for(int s = 1; s < S; s++)
    {
        for(int i = 1; i <= n; i++)
        {
            for(int k = (s - 1) & s; k; k = (k - 1) & s)
                f[s][i] = min(f[s][i], f[k][i] + f[s ^ k][i]);
        }
        SPFA(f[s]);
    }
    int ans = inf;
    for(int i = 1; i <= n; i++)
        ans = min(ans, f[S - 1][i]);
}
```

1.4 K 短路 (lhy)

```
const int MAXNODE = MAXN + MAXM * 2;

bool used[MAXN];
int n, m, cnt, S, T, Kth, N, TT;
int rt[MAXN], seq[MAXN], adj[MAXN], from[MAXN], dep[MAXN];
LL dist[MAXN], w[MAXM], ans[MAXK];

struct GivenEdge{
    int u, v, w;
    GivenEdge() {};
    GivenEdge(int _u, int _v, int _w) : u(_u), v(_v), w(_w){};
}edge[MAXM];

struct Edge{
    int v, nxt, w;
    Edge() {};
    Edge(int _v, int _nxt, int _w) : v(_v), nxt(_nxt), w(_w) {};
}e[MAXM];

inline void addedge(int u, int v, int w)
{
    e[++cnt] = Edge(v, adj[u], w); adj[u] = cnt;
}

void dij(int S)
{
    for(int i = 1; i <= N; i++)
    {
        dist[i] = INF;
        dep[i] = 0x3f3f3f3f;
        used[i] = false;
        from[i] = 0;
    }
    static priority_queue<pair<LL, int>, vector<pair<LL, int> >, greater<pair<LL, int> > > hp;
    while(!hp.empty())hp.pop();
    hp.push(make_pair(dist[S] = 0, S));
    dep[S] = 1;
    while(!hp.empty())
    {
        pair<LL, int> now = hp.top();
        hp.pop();
        int u = now.second;
        if(used[u])continue;
        else used[u] = true;
        for(int p = adj[u]; p; p = e[p].nxt)
        {
            int v = e[p].v;
            if(dist[u] + e[p].w < dist[v])
            {
                dist[v] = dist[u] + e[p].w;
                dep[v] = dep[u] + 1;
                from[v] = p;
                hp.push(make_pair(dist[v], v));
            }
        }
    }
    for(int i = 1; i <= m; i++)    w[i] = 0;
    for(int i = 1; i <= N; i++)
        if(from[i])w[from[i]] = -1;
    for(int i = 1; i <= m; i++)
    {
        if(~w[i] && dist[edge[i].u] < INF && dist[edge[i].v] < INF)
        {
            w[i] = -dist[edge[i].u] + (dist[edge[i].v] + edge[i].w);
        }
        else
        {
            w[i] = -1;
        }
    }
}

inline bool cmp_dep(int p, int q)
{
    return dep[p] < dep[q];
}

struct Heap{
    LL key;
    int id, lc, rc, dist;
    Heap() {};
    Heap(LL k, int i, int l, int r, int d) : key(k), id(i), lc(l), rc(r), dist(d) {};
    inline void clear()
    {
        key = 0;
        id = lc = rc = dist = 0;
    }
}hp[MAXNODE];

inline int merge_simple(int u, int v)
{
    if(!u)return v;
    if(!v)return u;
    if(hp[u].key > hp[v].key)
    {
        swap(u, v);
    }
```

```cpp
    hp[u].rc = merge_simple(hp[u].rc, v);
    if(hp[hp[u].lc].dist < hp[hp[u].rc].dist)
    {
        swap(hp[u].lc, hp[u].rc);
    }
    hp[u].dist = hp[hp[u].rc].dist + 1;
    return u;
}

inline int merge_full(int u, int v)
{
    if(!u)return v;
    if(!v)return u;
    if(hp[u].key > hp[v].key)
    {
        swap(u, v);
    }
    int nownode = ++cnt;
    hp[nownode] = hp[u];
    hp[nownode].rc = merge_full(hp[nownode].rc, v);
    if(hp[hp[nownode].lc].dist < hp[hp[nownode].rc].dist)
    {
        swap(hp[nownode].lc, hp[nownode].rc);
    }
    hp[nownode].dist = hp[hp[nownode].rc].dist + 1;
    return nownode;
}

priority_queue<pair<LL, int>, vector<pair<LL, int> >, greater<pair<LL, int> > > Q;

int main()
{
    while(scanf("%d%d", &n, &m) != EOF)
    {
        scanf("%d%d%d%d", &S, &T, &Kth, &TT);
        for(int i = 1; i <= m; i++)
        {
            int u, v, w;
            scanf("%d%d%d", &u, &v, &w);
            edge[i] = {u, v, w};
        }
        N = n;
        memset(adj, 0, sizeof(*adj) * (N + 1));
        cnt = 0;
        for(int i = 1; i <= m; i++)
            addedge(edge[i].v, edge[i].u, edge[i].w);
        dij(T);
        if(dist[S] > TT)
        {
            puts("Whitesnake!");
            continue;
        }
        for(int i = 1; i <= N; i++)
            seq[i] = i;
        sort(seq + 1, seq + N + 1, cmp_dep);

        cnt = 0;
        memset(adj, 0, sizeof(*adj) * (N + 1));
        memset(rt, 0, sizeof(*rt) * (N + 1));
        for(int i = 1; i <= m; i++)
            addedge(edge[i].u, edge[i].v, edge[i].w);
        rt[T] = cnt = 0;
        hp[0].dist = -1;
        for(int i = 1; i <= N; i++)
        {
            int u = seq[i], v = edge[from[u]].v;
            rt[u] = 0;
            for(int p = adj[u]; p; p = e[p].nxt)
            {
                if(~w[p])
                {
                    hp[++cnt] = Heap(w[p], p, 0, 0, 0);
                    rt[u] = merge_simple(rt[u], cnt);
                }
            }
            if(i == 1)continue;
            rt[u] = merge_full(rt[u], rt[v]);
        }
        while(!Q.empty())Q.pop();
        Q.push(make_pair(dist[S], 0));
        edge[0].v = S;
        for(int kth = 1; kth <= Kth; kth++)
        {
            if(Q.empty())
            {
                ans[kth] = -1;
                continue;
            }
            pair<LL, int> now = Q.top(); Q.pop();
            ans[kth] = now.first;
            int p = now.second;
            if(hp[p].lc)
            {
                Q.push(make_pair(+hp[hp[p].lc].key + now.first - hp[p].key, hp[p].lc));
            }
            if(hp[p].rc)
            {
                Q.push(make_pair(+hp[hp[p].rc].key + now.first - hp[p].key, hp[p].rc));
            }
            if(rt[edge[hp[p].id].v])
            {
                Q.push(make_pair(hp[rt[edge[hp[p].id].v]].key + now.first, rt[edge[hp[p].id].v]));
            }
        }
        if(ans[Kth] == -1 || ans[Kth] > TT)
        {
            puts("Whitesnake!");
        }
        else
        {
            puts("yareyaredawa");
        }
    }
}
```

1.5 最大团 (Nightfall)

时间复杂度建议 $n \le 150$

```cpp
typedef bool BB[N];
struct Maxclique {
    const BB *e; int pk, level; const float Tlimit;
    struct Vertex { int i, d; Vertex(int i) : i(i), d(0) {}};
    typedef vector<Vertex> Vertices; Vertices V;
    typedef vector<int> ColorClass; ColorClass QMAX, Q;
    vector<ColorClass> C;
    static bool desc_degree(const Vertex &vi,const Vertex &vj)
    { return vi.d > vj.d; }
```

```cpp
    void init_colors(Vertices &v) {
        const int max_degree = v[0].d;
        for (int i = 0; i < (int)v.size(); i++)
            v[i].d = min(i, max_degree) + 1; }
    void set_degrees(Vertices &v) {
        for (int i = 0, j; i < (int)v.size(); i++)
            for (v[i].d = j = 0; j < (int)v.size(); j++)
                v[i].d += e[v[i].i][v[j].i]; }
    struct StepCount{ int i1, i2; StepCount(): i1(0),i2(0){}};
    vector<StepCount> S;
    bool cut1(const int pi, const ColorClass &A) {
        for (int i = 0; i < (int)A.size(); i++)
            if (e[pi][A[i]]) return true; return false; }
    void cut2(const Vertices &A, Vertices & B) {
        for (int i = 0; i < (int)A.size() - 1; i++)
            if (e[A.back().i][A[i].i])
                B.push_back(A[i].i); }
    void color_sort(Vertices & R) { int j=0, maxno=1;
        int min_k=max((int)QMAX.size()-(int)Q.size()+1,1);
        C[1].clear(), C[2].clear();
        for (int i = 0; i < (int)R.size(); i++) {
            int pi = R[i].i, k = 1; while (cut1(pi, C[k])) k++;
            if (k > maxno) maxno = k, C[maxno + 1].clear();
            C[k].push_back(pi); if (k < min_k) R[j++].i = pi; }
        if (j > 0) R[j - 1].d = 0;
        for (int k = min_k; k <= maxno; k++)
            for (int i = 0; i < (int)C[k].size(); i++)
                R[j].i = C[k][i], R[j++].d = k; }

    void expand_dyn(Vertices &R) {
        S[level].i1 = S[level].i1 + S[level-1].i1 - S[level].i2;
        S[level].i2 = S[level - 1].i1;
        while ((int)R.size()) {
            if ((int)Q.size() + R.back().d > (int)QMAX.size()) {
                Q.push_back(R.back().i); Vertices Rp; cut2(R, Rp);
                if ((int)Rp.size()) {
                    if((float)S[level].i1/++pk<Tlimit) degree_sort(Rp);
                    color_sort(Rp); S[level].i1++, level++;
                    expand_dyn(Rp); level--;
                } else if ((int)Q.size() > (int)QMAX.size()) QMAX=Q;
                Q.pop_back(); } else return; R.pop_back(); }}
    void mcqdyn(int *maxclique, int &sz) {
        set_degrees(V); sort(V.begin(), V.end(), desc_degree);
        init_colors(V);
        for (int i=0; i<(int)V.size()+1; i++) S[i].i1=S[i].i2=0;
        expand_dyn(V); sz = (int)QMAX.size();
        for(int i=0;i<(int)QMAX.size();i++) maxclique[i]=QMAX[i];}
    void degree_sort(Vertices & R) {
        set_degrees(R); sort(R.begin(), R.end(), desc_degree); }
    Maxclique(const BB *conn,const int sz,const float tt=.025)
        : pk(0), level(1), Tlimit(tt){
            for(int i = 0; i < sz; i++) V.push_back(Vertex(i));
            e = conn, C.resize(sz + 1), S.resize(sz + 1); }};
BB e[N]; int ans, sol[N]; for (...) e[x][y]=e[y][x]=true;
Maxclique mc(e, n); mc.mcqdyn(sol, ans); // 全部 0 下标
for (int i = 0; i < ans; ++i) cout << sol[i] << endl;
```

1.6 极大团计数 (Nightfall)

0-based, 需删除自环

极大团计数, 最坏情况 $O(3^{n/3})$

```cpp
ll ans; ull E[64];
#define bit(i) (1ULL << (i))
void dfs(ull P, ull X, ull R) { // 不需要方案时可去掉 R 相关语句
    if (!P && !X) { ++ans; sol.pb(R); return; }
    ull Q = P & ~E[__builtin_ctzll(P | X)];
    for (int i; i = __builtin_ctzll(Q), Q; Q &= ~bit(i)) {
        dfs(P & E[i], X & E[i], R | bit(i));
        P &= ~bit(i), X |= bit(i); }}
    ans = 0; dfs(n == 64 ? ~0ULL : bit(n) - 1, 0, 0);
```

1.7 二分图最大匹配 (lhy)

左侧 n 个点, 右侧 m 个点, 1-based, 初始化将 $matx$ 和 $maty$ 置为 0

```cpp
int BFS()
{
    int flag = 0, h = 0, l = 0;
    for(int i = 1; i <= k; i++)
        dy[i] = 0;
    for(int i = 1; i <= n; i++)
    {
        dx[i] = 0;
        if(!matx[i])q[++l] = i;
    }
    while(h < l)
    {
        int x = q[++h];
        for(int i = son[x]; i; i = edge[i].next)
        {
            int y = edge[i].y;
            if(!dy[y])
            {
                dy[y] = dx[x] + 1;
                if(!maty[y])flag = 1;
                else
                {
                    dx[maty[y]] = dx[x] + 2;
                    q[++l] = maty[y];
                }
            }
        }
    }
    return flag;
}

int DFS(int x)
```

```
{
    for(int i = son[x]; i; i = edge[i].next)
    {
        int y = edge[i].y;
        if(dy[y] == dx[x] + 1)
        {
            dy[y] = 0;
            if(!maty[y] || DFS(maty[y]))
            {
                matx[x] = y, maty[y] = x;
                return 1;
            }
        }
    }
    return 0;
}

void Hopcroft()
{
    for(int i = 1; i <= n; i++)
        matx[i] = maty[i] = 0;
    while(BFS())
        for(int i = 1; i <= n; i++)
            if(!matx[i])DFS(i);
}
```

1.8 一般图最大匹配 (lhy)

```
struct blossom{

    struct Edge{
        int x, y, next;
    }edge[M];

    int n, W, tot, h, l, son[N];
    int mat[N], pre[N], tp[N], q[N], vis[N], F[N];

    void Prepare(int n_)
    {
        n = n_;
        W = tot = 0;
        for(int i = 1; i <= n; i++)
            son[i] = mat[i] = vis[i] = 0;
    }

    void add(int x, int y)
    {
        edge[++tot].x = x; edge[tot].y = y; edge[tot].next = son[x]; son[x] = tot;
    }

    int find(int x)
    {
        return F[x] ? F[x] = find(F[x]) : x;
    }

    int lca(int u, int v)
    {
        for(++W;; u = pre[mat[u]], swap(u, v))
            if(vis[u = find(u)] == W)return u;
            else vis[u] = u ? W : 0;
    }

    void aug(int u, int v)
    {
        for(int w; u; v = pre[u = w])
            w = mat[v], mat[mat[u] = v] = u;
    }

    void blo(int u, int v, int f)
    {
        for(int w; find(u) ^ f; u = pre[v = w])
            pre[u] = v, F[u] ? 0 : F[u] = f, F[w = mat[u]] ? 0 : F[w] = f, tp[w] ^ 1 ? 0 : tp[q[++l] = w] = -1;
    }

    int bfs(int x)
    {
        for(int i = 1; i <= n; i++)
            tp[i] = F[i] = 0;
        h = l = 0;
        q[++l] = x;
        tp[x]--;
        while(h < l)
        {
            x = q[++h];
            for(int i = son[x]; i; i = edge[i].next)
            {
                int y = edge[i].y, Lca;
                if(!tp[y])
                {
                    if(!mat[y])return aug(y, x), 1;
                    pre[y] = x, ++tp[y], --tp[q[++l] = mat[y]];
                }
                else if(tp[y] ^ 1 && find(x) ^ find(y))
                    blo(x, y, Lca = lca(x, y)), blo(y, x, Lca);
            }
        }
        return 0;
    }

    int solve()
    {
        int ans = 0;
        for(int i = 1; i <= n; i++)
            if(!mat[i])ans += bfs(i);
        return ans;
    }
}G;
```

1.9 KM 算法 (Nightfall)

$O(n^3)$，1-based，最大权匹配
不存在的边权值开到 $-n\times(|MAXV|)$，∞ 为 $3n\times(|MAXV|)$
匹配为 (lk_i, i)

```
long long KM(int n, long long w[N][N])
{
    long long ans = 0;
    int x, py, p;
    long long d;
    for(int i = 1; i <= n; i++)
        lx[i] = ly[i] = 0, lk[i] = -1;
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
            lx[i] = max(lx[i], w[i][j]);
    for(int i = 1; i <= n; i++)
    {
        for(int j = 1; j <= n; j++)
            slk[j] = inf, vy[j] = 0;
        for(lk[py = 0] = i; lk[py]; py = p)
        {
```

```
            vy[py] = 1; d = inf; x = lk[py];
            for(int y = 1; y <= n; y++)
                if(!vy[y])
                {
                    if(lx[x] + ly[y] - w[x][y] < slk[y])
                        slk[y] = lx[x] + ly[y] - w[x][y], pre[y] = py;
                    if(slk[y] < d)d = slk[y], p = y;
                }
            for(int y = 0; y <= n; y++)
                if(vy[y])lx[lk[y]] -= d, ly[y] += d;
                else slk[y] -= d;
        }
        for(; py; py = pre[py])lk[py] = lk[pre[py]];
    }
    for(int i = 1; i <= n; i++)
        ans += lx[i] + ly[i];
    return ans;
}
```

1.10 最小树形图 (Nightfall)

```
using Val = long long;
#define nil mem
struct Node { Node *l,*r; int dist;int x,y;Val val,laz; }
mem[M] = {{nil, nil, -1}}; int sz = 0;
#define NEW(arg...) (new(mem + ++ sz)Node{nil,nil,0,arg})
void add(Node *x, Val o) {if(x!=nil){x->val+=o, x->laz+=o;}}
void down(Node *x){add(x->l,x->laz);add(x->r,x->laz);x->laz=0;}
Node *merge(Node *x, Node *y) {
    if (x == nil) return y; if (y == nil) return x;
    if (y->val < x->val) swap(x, y); //smalltop heap
    down(x); x->r = merge(x->r, y);
    if (x->l->dist < x->r->dist) swap(x->l, x->r);
    x->dist = x->r->dist + 1; return x; }
    Node *pop(Node *x){down(x); return merge(x->l, x->r);}
    struct DSU { int f[N]; void clear(int n) {
        for (int i=0; i<=n; ++i) f[i]=i; }
    int fd(int x) { if (f[x]==x) return x;
        return f[x]=fd(f[x]); }
    int& operator[](int x) {return f[fd(x)];}};
DSU W, S; Node *H[N], *pe[N];
vector<pair<int, int>> G[N]; int dist[N], pa[N];
// addedge(x, y, w) : NEW(x, y, w, 0)
Val chuliu(int s, int n) { // O(ElogE)
    for (int i = 1; i <= n; ++ i) G[i].clear();
    Val re=0; W.clear(n); S.clear(n); int rid=0;
    fill(H, H + n + 1, (Node*) nil);
    for (auto i = mem + 1; i <= mem + sz; ++ i)
        H[i->y] = merge(i, H[i->y]);
    for (int i = 1; i <= n; ++ i) if (i != s)
        for (;;) {
            auto in = H[S[i]]; H[S[i]] = pop(H[S[i]]);
            if (in == nil) return INF; // no solution
            if (S[in -> x] == S[i]) continue;
            re += in->val; pe[S[i]] = in;
            // if (in->x == s) true root = in->y
            add(H[S[i]], -in->val);
            if (W[in->x]!=W[i]) {W[in->x]=W[i];break;}
            G[in -> x].push_back({in->y,++rid});
            for (int j=S[in->x]; j!=S[i]; j=S[pe[j]->x]) {
                G[pe[j]->x].push_back({pe[j]->y, rid});
                H[j] = merge(H[S[i]], H[j]); S[i]=S[j]; }}
    ++ rid; for (int i=1; i<=n; ++ i) if(i!=s && S[i]==i)
        G[pe[i]->x].push_back({pe[i]->y, rid});
    return re;}
void makeSol(int s, int n) {
    fill(dist, dist + n + 1, n + 1); pa[s] = 0;
    for (multiset<pair<int, int>> h = {{0,s}}; !h.empty();){
        int x=h.begin()->second;
        h.erase(h.begin()); dist[x]=0;
        for (auto i : G[x]) if (i.second < dist[i.first]) {
            h.erase({dist[i.first], i.first});
            h.insert({dist[i.first] = i.second, i.first});
            pa[i.first] = x; }}}
```

1.11 支配树 (Nightfall,ct)

DAG (ct)

```
struct Edge {
    Edge *next;
    int to;
} ;
Edge *last[maxn], e[maxm], *ecnt = e; // original graph
Edge *rlast[maxn], re[maxm], *recnt = re; // reversed-edge graph
Edge *tlast[maxn], te[maxn << 1], *tecnt = te; // dominate tree graph
int deg[maxn], q[maxn], fa[maxn][20], all_fa[maxn], fa_cnt, size[maxn], dep[maxn];
inline void link(int a, int b)
{
    *++ecnt = (Edge) {last[a], b}; last[a] = ecnt; ++deg[b];
}
inline void link_rev(int a, int b)
{
    *++recnt = (Edge) {rlast[a], b}; rlast[a] = recnt;
}
inline void link_tree(int a, int b)
{
    *++tecnt = (Edge) {tlast[a], b}; tlast[a] = tecnt;
}
inline int getlca(int a, int b)
{
    if (dep[a] < dep[b]) std::swap(a, b);
    int temp = dep[a] - dep[b];
    for (int i; temp; temp -= 1 << i)
        a = fa[a][i = __builtin_ctz(temp)];
    for (int i = 16; ~i; --i)
        if (fa[a][i] != fa[b][i])
            a = fa[a][i], b = fa[b][i];
    if (a == b) return a;
    return fa[a][0];
}
void dfs(int x)
{
    size[x] = 1;
    for (Edge *iter = tlast[x]; iter; iter = iter -> next)
        dfs(iter -> to), size[x] += size[iter -> to];
```

```
}
int main()
{
    q[1] = 0;
    int head = 0, tail = 1;
    while (head < tail)
    {
        int now = q[++head];
        fa_cnt = 0;
        for (Edge *iter = rlast[now]; iter; iter = iter -> next)
            all_fa[++fa_cnt] = iter -> to;
        for (; fa_cnt > 1; --fa_cnt)
            all_fa[fa_cnt - 1] = getlca(all_fa[fa_cnt], all_fa[fa_cnt - 1]);
        fa[now][0] = all_fa[fa_cnt];
        dep[now] = dep[all_fa[fa_cnt]] + 1;
        if (now) link_tree(fa[now][0], now);

        for (int i = 1; i <= 16; ++i)
            fa[now][i] = fa[fa[now][i - 1]][i - 1];
        for (Edge *iter = last[now]; iter; iter = iter -> next)
            if (--deg[iter -> to] == 0) q[++tail] = iter -> to;
    }
    dfs(0);
    for (int i = 1; i <= n; ++i) printf("%d\n", size[i] - 1 );
    return 0;
}
```

一般图 (Nightfall)

```
struct Dominator_Tree{
    int n, s, cnt;
    int dfn[N], id[N], pa[N], semi[N], idom[N], p[N], mn[N];
    vector<int> e[N], dom[N], be[N];

    void ins(int x, int y){e[x].push_back(y);}

    void dfs(int x)
    {
        dfn[x] = ++cnt; id[cnt] = x;
        for(auto i:e[x])
        {
            if(!dfn[i])dfs(i), pa[dfn[i]] = dfn[x];
            be[dfn[i]].push_back(dfn[x]);
        }
    }

    int get(int x)
    {
        if(p[x] != p[p[x]])
        {
            if(semi[mn[x]] > semi[get(p[x])])mn[x] = get(p[x]);
            p[x] = p[p[x]];
        }
        return mn[x];
    }

    void LT()
    {
        for(int i = cnt; i > 1; i--)
        {
            for(auto j:be[i])semi[i] = min(semi[i], semi[get(j)]);
            dom[semi[i]].push_back(i);
            int x = p[i] = pa[i];
            for(auto j:dom[x])
                idom[j] = (semi[get(j)] < x ? get(j) : x);
            dom[x].clear();
        }
        for(int i = 2; i <= cnt; i++)
        {
            if(idom[i] != semi[i])idom[i] = idom[idom[i]];
            dom[id[idom[i]]].push_back(id[i]);
        }
    }
    void build()
    {
        for(int i = 1; i <= n; i++)
            dfn[i] = 0, dom[i].clear(), be[i].clear(), p[i] = mn[i] = semi[i] = i;
        cnt = 0, dfs(s), LT();
    }
};
```

1.12 虚树 (ct)

```
struct Edge {
    Edge *next;
    int to;
} *last[maxn], e[maxn << 1], *ecnt = e;
inline void link(int a, int b)
{
    *++ecnt = (Edge) {last[a], b}; last[a] = ecnt;
    *++ecnt = (Edge) {last[b], a}; last[b] = ecnt;
}
int a[maxn], n, dfn[maxn], pos[maxn], timer, inv[maxn], st[maxn];
int fa[maxn], size[maxn], dep[maxn], son[maxn], top[maxn];
bool vis[maxn];
void dfs1(int x); // 树剖
void dfs2(int x);
inline int getlca(int a, int b);
inline bool cmp(int a, int b)
{
    return dfn[a] < dfn[b];
}
inline bool isson(int a, int b)
{
    return dfn[a] <= dfn[b] && dfn[b] <= inv[a];
}
typedef long long ll;
bool imp[maxn];
struct sEdge {
    sEdge *next;
    int to, w;
} *slast[maxn], se[maxn << 1], *secnt = se;
inline void slink(int a, int b, int w)
{
    *++secnt = (sEdge) {slast[a], b, w}; slast[a] = secnt;
}
int main()
{
    scanf("%d", &n);
    for (int i = 1; i < n; ++i)
    {
        int a, b; scanf("%d%d", &a, &b);
```

```cpp
        link(a, b);
    }
    int m; scanf("%d", &m);
    dfs1(1); dfs2(1);
    memset(size, 0, (n + 1) << 2);
    for (; m; --m)
    {
        int top = 0; scanf("%d", &k);
        for (int i = 1; i <= k; ++i) scanf("%d", &a[i]), vis[a[i]] = imp[a[i]] = 1;
        std::sort(a + 1, a + k + 1, cmp);
        int p = k;
        for (int i = 1; i < k; ++i)
        {
            int lca = getlca(a[i], a[i + 1]);
            if (!vis[lca]) vis[a[++p] = lca] = 1;
        }
        std::sort(a + 1, a + p + 1, cmp);
        st[++top] = a[1];
        for (int i = 2; i <= p; ++i)
        {
            while (!isson(st[top], a[i])) --top;
            slink(st[top], a[i], dep[a[i]] - dep[st[top]]);
            st[++top] = a[i];
        }
        /*
            write your code here.
        */
        for (int i = 1; i <= p; ++i) vis[a[i]] = imp[a[i]] = 0, slast[a[i]] = 0;
        secnt = se;
    }
    return 0;
}
```

1.13 点分治 (ct)

```cpp
int root, son[maxn], size[maxn], sum;
bool vis[maxn];
void dfs_root(int x, int fa)
{
    size[x] = 1; son[x] = 0;
    for (Edge *iter = last[x]; iter; iter = iter -> next)
    {
        if (iter -> to == fa || vis[iter -> to]) continue;
        dfs_root(iter -> to, x);
        size[x] += size[iter -> to];
        cmax(son[x], size[iter -> to]);
    }
    cmax(son[x], sum - size[x]);
    if (!root || son[x] < son[root]) root = x;
}
void dfs_chain(int x, int fa)
{
    /*
        write your code here.
    */
    for (Edge *iter = last[x]; iter; iter = iter -> next)
    {
        if (vis[iter -> to] || iter -> to == fa) continue;
        dfs_chain(iter -> to, x);
    }
}
void calc(int x)
{
    for (Edge *iter = last[x]; iter; iter = iter -> next)
    {
        if (vis[iter -> to]) continue;
        dfs_chain(iter -> to, x);
        /*
            write your code here.
        */
    }
}
void work(int x)
{
    vis[x] = 1;
    calc(x);
    for (Edge *iter = last[x]; iter; iter = iter -> next)
    {
        if (vis[iter -> to]) continue;
        root = 0;
        sum = size[iter -> to];
        dfs_root(iter -> to, 0);
        work(root);
    }
}
int main()
{
    root = 0; sum = n;
    dfs_root(1, 0);
    work(root);
    return 0;
}
```

1.14 树上倍增 (ct)

```cpp
int fa[maxn][17], mn[maxn][17], dep[maxn];
bool vis[maxn];
void dfs(int x)
{
    vis[x] = 1;
    for (int i = 1; i <= 16; ++i)
    {
        if (dep[x] < (1 << i)) break;
        fa[x][i] = fa[fa[x][i - 1]][i - 1];
        mn[x][i] = dmin(mn[x][i - 1], mn[fa[x][i - 1]][i - 1]);
    }
    for (Edge *iter = last[x]; iter; iter = iter -> next)
        if (!vis[iter -> to])
        {
            fa[iter -> to][0] = x;
            mn[iter -> to][0] = iter -> w;
            dep[iter -> to] = dep[x] + 1;
            dfs(iter -> to);
        }
}
inline int getlca(int x, int y)
{
    if (dep[x] < dep[y]) std::swap(x, y);
    int t = dep[x] - dep[y];
    for (int i = 0; i <= 16 && t; ++i)
        if ((1 << i) & t)
            x = fa[x][i], t ^= 1 << i;
    for (int i = 16; i >= 0; --i)
        if (fa[x][i] != fa[y][i])
        {
```

```cpp
            x = fa[x][i];
            y = fa[y][i];
        }
    if (x == y) return x;
    return fa[x][0];
}
inline int getans(int x, int f)
{
    int ans = inf, t = dep[x] - dep[f];
    for (int i = 0; i <= 16 && t; ++i)
        if (t & (1 << i))
        {
            cmin(ans, mn[x][i]);
            x = fa[x][i];
            t ^= 1 << i;
        }
    return ans;
}
```

1.15 Link-Cut Tree (ct)

LCT 常见应用

- 动态维护边双

可以通过 LCT 来解决一类动态边双连通分量问题。即静态的询问可以用边双连通分量来解决，而树有加边等操作的问题。
把一个边双连通分量缩到 LCT 的一个点中，然后在 LCT 上求出答案。缩点的方法为加边时判断两点的连通性，如果已经联通则把两点在目前 LCT 路径上的点都缩成一个点。

- 动态维护基环森林

通过 LCT 可以动态维护基环森林，即每个点有且仅有一个出度的图。有修改操作，即改变某个点的出边。对于每颗基环森林记录一个点为根，并把环上额外的一条边单独记出，剩下的边用 LCT 维护。一般使用有向 LCT 维护。
修改时分以下几种情况讨论：

– 修改的点是根，如果改的父亲在同一个连通块中，直接改额外边，否则删去额外边，在 LCT 上加边。
– 修改的点不是根，那么把这个点和其父亲的联系切除。如果该点和根在一个环上，那么把多的那条边加到 LCT 上。最后如果改的那个父亲和修改的点在一个联通块中，记录额外边，否则 LCT 上加边。

- 子树询问

通过记录轻边信息可以快速地维护出整颗 LCT 的一些值。如子树和，子树最大值等。在 Access 时要进行虚实边切换，这时减去实边的贡献，并加上新加虚边的贡献即可。有时需要套用数据结构，如 Set 来维护最值等问题。
模板：

– $x \to y$ 链 $+z$
– $x \to y$ 链变为 z
– 在以 x 为根的树对 y 子树的点权求和
– $x \to y$ 链取 max
– $x \to y$ 链求和
– 连接 x, y
– 断开 x, y

V 单点值，sz 平衡树的 size，mv 链上最大，S 链上和，sm 区间相同标记，lz 区间加标记，B 虚边之和，ST 子树信息和，SM 子树和链上信息和。更新时：
$S[x] = S[c[x][0]] + S[c[x][1]] + V[x]$
$ST[x] = B[x] + ST[c[x][0]] + ST[c[x][1]]$
$SM[x] = S[x] + ST[x]$

```cpp
struct Node *null;
struct Node {
    Node *ch[2], *fa, *pos;
    int val, mn, l, len; bool rev;
    // min_val in chain
    inline bool type()
    {
        return fa -> ch[1] == this;
    }
    inline bool check()
    {
        return fa -> ch[type()] == this;
    }
    inline void pushup()
    {
        pos = this; mn = val;
        ch[0] -> mn < mn ? mn = ch[0] -> mn, pos = ch[0] -> pos : 0;
        ch[1] -> mn < mn ? mn = ch[1] -> mn, pos = ch[1] -> pos : 0;
        len = ch[0] -> len + ch[1] -> len + l;
    }
    inline void pushdown()
    {
        if (rev)
        {
            ch[0] -> rev ^= 1;
            ch[1] -> rev ^= 1;
            std::swap(ch[0], ch[1]);
            rev ^= 1;
        }
    }
    inline void pushdownall()
    {
        if (check()) fa -> pushdownall();
        pushdown();
    }
    inline void rotate()
    {
        bool d = type(); Node *f = fa, *gf = f -> fa;
        (fa = gf, f -> check()) ? fa -> ch[f -> type()] = this : 0;
        (f -> ch[d] = ch[!d]) != null ? ch[!d] -> fa = f : 0;
        (ch[!d] = f) -> fa = this;
        f -> pushup();
    }
    inline void splay(bool need = 1)
    {
        if (need) pushdownall();
        for (; check(); rotate())
            if (fa -> check())
                (type() == fa -> type() ? fa : this) -> rotate();
        pushup();
    }
    inline Node *access()
    {
        Node *i = this, *j = null;
        for (; i != null; i = (j = i) -> fa)
        {
            i -> splay();
            i -> ch[1] = j;
            i -> pushup();
        }
        return j;
    }
    inline void make_root()
    {
```

```
        access();
        splay();
        rev ^= 1;
    }
    inline void link(Node *that)
    {
        make_root();
        fa = that;
        splay(0);
    }
    inline void cut(Node *that)
    {
        make_root();
        that -> access();
        that -> splay(0);
        that -> ch[0] = fa = null;
        that -> pushup();
    }
} mem[maxn];
inline Node *query(Node *a, Node *b)
{
    a -> make_root(); b -> access(); b -> splay(0);
    return b -> pos;
}
inline int dist(Node *a, Node *b)
{
    a -> make_root(); b -> access(); b -> splay(0);
    return b -> len;
}
```

1.16 圆方树 (ct)

```
int dfn[maxn], low[maxn], timer, st[maxn], top, id[maxn], scc;
void dfs(int x)
{
    dfn[x] = low[x] = ++timer; st[++top] = x;
    for (Edge *iter = last[x]; iter; iter = iter -> next)
        if (!dfn[iter -> to])
        {
            dfs(iter -> to);
            cmin(low[x], low[iter -> to]);
            if (dfn[x] == low[iter->to])
            {
                int now, elder = top, minn = c[x];
                ++scc;
                do
                {
                    now = st[top--];
                    cmin(minn, c[now]);
                }
                while (iter -> to != now);
                for (int i = top + 1; i <= elder; ++i)
                    add(scc, st[i], minn);
                add(scc, x, minn);
            }
        }
        else if (!id[iter -> to]) cmin(low[x], dfn[iter -> to]);
}
```

1.17 无向图最小割 (Nightfall)

```
int d[N];bool v[N],g[N];
int get(int&s,int&t){
    CL(d);CL(v);int i,j,k,an,mx;
    for(i=1;i<=n;i++){ k=mx=-1;
        for(j=1;j<=n;j++)if(!g[j]&&!v[j]&&d[j]>mx) k=j,mx=d[j];
        if(k==-1)return an;
        s=t;t=k;an=mx;v[k]=1;
        for(j=1;j<=n;j++)if(!g[j]&&!v[j])d[j]+=w[k][j];
    }return an;}
int mincut(int n,int w[N][N]){
    //n 为点数, w[i][j] 为 i 到 j 的流量, 返回无向图所有点对最小割之和
    int ans=0,i,j,s,t,x,y,z;
    for(i=1;i<=n-1;i++){
        ans=min(ans,get(s,t));
        g[t]=1;if(!ans)break;
        for(j=1;j<=n;j++)if(!g[j])w[s][j]=(w[j][s]+=w[j][t]);
    }return ans;}
// 无向图最小割树
void fz(int l,int r){// 左闭右闭, 分治建图
    if(l==r)return;S=a[l];T=a[r];
    reset();// 将所有边权复原
    flow(S,T);// 做网络流
    dfs(S);// 找割集, v[x]=1 属于 S 集, 否则属于 T 集
    ADD(S,T,fl);// 在最小割树中建边
    L=l,R=r;for(i=l;i<=r;i++) if(v[a[i]])q[L++]=a[i]; else q[R--]=a[i];
    for(i=l;i<=r;i++)a[i]=q[i];fz(l,L-1);fz(R+1,r);}
```

1.18 最大流 (lhy,ct)

Dinic (ct)

```
struct Edge {
    Edge *next, *rev;
    int to, cap;
} *last[maxn], *cur[maxn], e[maxm], *ecnt = e;
inline void link(R int a, R int b, R int w)
{
    *++ecnt = (Edge) {last[a], ecnt + 1, b, w}; last[a] = ecnt;
    *++ecnt = (Edge) {last[b], ecnt - 1, a, 0}; last[b] = ecnt;
}
int ans, s, t, q[maxn], dep[maxn];
inline bool bfs()
{
    memset(dep, -1, (t + 1) << 2);
    dep[q[1] = t] = 0; int head = 0, tail = 1;
    while (head < tail)
    {
        int now = q[++head];
        for (Edge *iter = last[now]; iter; iter = iter -> next)
            if (dep[iter -> to] == -1 && iter -> rev -> cap)
                dep[q[++tail] = iter -> to] = dep[now] + 1;
    }
    return dep[s] != -1;
}
int dfs(int x, int f)
{
    if (x == t) return f;
    int used = 0;
    for (Edge* &iter = cur[x]; iter; iter = iter -> next)
        if (iter -> cap && dep[iter -> to] + 1 == dep[x])
```

```cpp
        {
            int v = dfs(iter -> to, dmin(f - used, iter -> cap));
            iter -> cap -= v;
            iter -> rev -> cap += v;
            used += v;
            if (used == f) return f;
        }
    return used;
}
inline void dinic()
{
    while (bfs())
    {
        memcpy(cur, last, sizeof cur);
        ans += dfs(s, inf);
    }
}
```

SAP (lhy)

```cpp
void SAP(int n, int st, int ed)
{
    for(int i = 1; i <= n; i++)
        now[i] = son[i];
    sumd[0] = n;
    int flow = inf, x = st;
    while(dis[st] < n)
    {
        back[x] = flow;
        int flag = 0;
        for(int i = now[x]; i != -1; i = edge[i].next)
        {
            int y = edge[i].y;
            if(edge[i].f && dis[y] + 1 == dis[x])
            {
                flag = 1;
                now[x] = i;
                pre[y] = i;
                flow = min(flow, edge[i].f);
                x = y;
                if(x == ed)
                {
                    ans += flow;
                    while(x != st)
                    {
                        edge[pre[x]].f -= flow;
                        edge[pre[x] ^ 1].f += flow;
                        x = edge[pre[x]].x;
                    }
                    flow = inf;
                }
                break;
            }
        }
        if(flag)continue;
        int minn = n - 1, tmp;
        for(int i = son[x]; i != -1; i = edge[i].next)
        {
            int y = edge[i].y;
            if(edge[i].f && dis[y] < minn)
            {
                minn = dis[y];
                tmp = i;
            }
        }
        now[x] = tmp;
        if(!(--sumd[dis[x]]))return;
        sumd[dis[x] = minn + 1]++;
        if(x != st)flow = back[x = edge[pre[x]].x];
    }
}
```

1.19 费用流 (ct)

SPFA(ct)

```cpp
struct Edge {
    Edge *next, *rev;
    int from, to, cap, cost;
} *last[maxn], *prev[maxn], e[maxm], *ecnt = e;
inline void link(int a, int b, int w, int c)
{
    *++ecnt = (Edge) {last[a], ecnt + 1, a, b, w, c}; last[a] = ecnt;
    *++ecnt = (Edge) {last[b], ecnt - 1, b, a, 0, -c}; last[b] = ecnt;
}
int s, t, q[maxn << 2], dis[maxn];
ll ans;
bool inq[maxn];
#define inf 0x7fffffff
inline bool spfa()
{
    for (int i = 1; i <= t; ++i) dis[i] = inf;
    int head = 0, tail = 1; dis[q[1] = s] = 0;
    while (head < tail)
    {
        int now = q[++head]; inq[now] = 0;
        for (Edge *iter = last[now]; iter; iter = iter -> next)
            if (iter -> cap && dis[iter -> to] > dis[now] + iter -> cost)
            {
                dis[iter -> to] = dis[now] + iter -> cost;
                prev[iter -> to] = iter;
                !inq[iter -> to] ? inq[q[++tail] = iter -> to] = 1 : 0;
            }
    }
    return dis[t] != inf;
}
inline void mcmf()
{
    int x = inf;
    for (Edge *iter = prev[t]; iter; iter = prev[iter -> from]) cmin(x, iter -> cap);
    for (Edge *iter = prev[t]; iter; iter = prev[iter -> from])
    {
        iter -> cap -= x;
        iter -> rev -> cap += x;
        ans += 1ll * x * iter -> cost;
    }
}
```

zkw(lhy)

```cpp
int aug(int no, int res)
{
    if(no == ED)return mincost += 1ll * pil * res, res;
    v[no] = 1;
    int flow = 0;
```

```
    for(int i = son[no]; i != -1; i = edge[i].next)
        if(edge[i].f && !v[edge[i].y] && !edge[i].c)
        {
            int d = aug(edge[i].y, min(res, edge[i].f));
            edge[i].f -= d, edge[i ^ 1].f += d, flow += d, res -= d;
            if(!res)return flow;
        }
    return flow;
}

bool modlabel()
{
    long long d = 0x3f3f3f3f3f3f3f3fll;
    for(int i = 1; i <= cnt; i++)
        if(v[i])
        {
            for(int j = son[i]; j != -1; j = edge[j].next)
                if(edge[j].f && !v[edge[j].y] && edge[j].c < d)d = edge[j].c;
        }
    if(d == 0x3f3f3f3f3f3f3f3fll)return 0;
    for(int i = 1; i <= cnt; i++)
        if(v[i])
        {
            for(int j = son[i]; j != -1; j = edge[j].next)
                edge[j].c -= d, edge[j ^ 1].c += d;
        }
    pil += d;
    return 1;
}

void minimum_cost_flow_zkw()
{
    pil = 0;
    int nowans = 0;
    nowf = 0;
    do{
        do{
            for(int i = 1; i <= cnt; i++)
                v[i] = 0;
            nowans = aug(ST, inf);
            nowf += nowans;
        }while(nowans);
    }while(modlabel());
}
```

1.20 图论知识 (gy,lhy)

Hall theorem

二分图 $G=(X,Y,E)$ 有完备匹配的充要条件是：对于 X 的任意一个子集 S 都满足 $|S| \le |A(S)|$，$A(S)$ 是 Y 的子集，是 S 的邻集（与 S 有边的边集）。

Prufer 编码

树和其 prufer 编码一一对应，一颗 n 个点的树，其 prufer 编码长度为 $n-2$，且度数为 d_i 的点在 prufer 编码中出现 d_i-1 次。

由树得到序列：总共需要 $n-2$ 步，第 i 步在当前的树中寻找具有最小标号的叶子节点，将与其相连的点的标号设为 Prufer 序列的第 i 个元素 p_i，并将此叶子节点从树中删除，直到最后得到一个长度为 $n-2$ 的 Prufer 序列和一个只有两个节点的树。

由序列得到树：先将所有点的度赋初值为 1，然后加上它的编号在 Prufer 序列中出现的次数，得到每个点的度；执行 $n-2$ 步，第 i 步选取具有最小标号的度为 1 的点 u 与 $v=p_i$ 相连，得到树中的一条边，并将 u 和 v 的度减 1。最后再把剩下的两个度为 1 的点连边，加入到树中。

相关结论：

- n 个点完全图，每个点度数依次为 $d_1, d_2, \ldots, dn$，这样生成树的棵树为：$\frac{(n-2)!}{(d_1-1)!(d_2-1)!\ldots(d_n-1)!}$
- 左边有 n_1 个点，右边有 n_2 个点的完全二分图的生成树棵树为：$n_1^{n_2-1}+n_2^{n_1-1}$
- m 个连通块，每个连通块有 c_i 个点，把他们全部连通的生成树方案数：$(\sum c_i)^{m-2}\prod c_i$

差分约束

若要使得所有量两两的值最接近，则将如果将源点到各点的距离初始化为 0。若要使得某一变量与其余变量的差最大，则将源点到各点的距离初始化为 ∞，其中之一为 0。若求最小方案则跑最长路，否则跑最短路。

弦图

弦图：任意点数 ≥ 4 的环皆有弦的无向图
单纯点：与其相邻的点的诱导子图为完全图的点
完美消除序列：每次选择一个单纯点删去的序列
弦图必有完美消除序列
$O(m+n)$ 求弦图的完美消除序列：每次选择未选择的标号最大的点，并将与其相连的点标号 +1，得到完美消除序列的反序
最大团数 = 最小染色数：按完美消除序列从后往前贪心地染色
最小团覆盖 = 最大点独立集：按完美消除序列从前往后贪心地选点加入点独立集

计数问题

- **有根树计数**

$a_1 = 1$

$$a_{n+1} = \frac{\sum_{j=1}^{n} j \cdot a_j \cdot S_{n,j}}{n}$$

$$S_{n,j} = \sum_{i=1}^{n/j} a_{n+1-ij} = S_{n-j,j} + a_{n+1-j}$$

- **无根树计数**

$$\begin{cases} a_n - \sum_{i=1}^{n/2} a_i a_{n-i} & n \text{ is odd} \\ a_n - \sum_{i=1}^{n/2} a_i a_{n-i} + \frac{1}{2} a_{\frac{n}{2}} (a_{\frac{n}{2}} + 1) & n \text{ is even} \end{cases}$$

- **生成树计数**

Kirchhoff Matrix $T = Deg - A$，Deg 是度数对角阵，A 是邻接矩阵。无向图度数矩阵是每个点度数；有向图度数矩阵是每个点入度。邻接矩阵 $A[u][v]$ 表示 $u \to v$ 边个数，重边按照边数计算，自环不计入度数。
无向图生成树计数：$c = |K\text{的任意1个}n-1\text{阶主子式}|$
有向图外向树计数：$c = |\text{去掉根所在的那阶得到的主子式}|$

- **Edmonds Matrix**

Edmonds matrix A of a balanced ($|U| = |V|$) bipartite graph $G = (U, V, E)$:

$$A_{ij} = \begin{cases} x_{ij} & (u_i, v_j) \in E \\ 0 & (u_i, v_j) \notin E \end{cases}$$

where the x_{ij} are indeterminates.
G 有完备匹配当且仅当关于 x_{ij} 的多项式 $\det(A_{ij})$ 不恒为 0。
完备匹配的个数等于多项式中单项式的个数

- **偶数点完全图完备匹配计数**

$(n-1)!!$

- **无根二叉树计数**

$(2n-5)!!$

- **有根二叉树计数**

$(2n-3)!!$

上下界网络流

$B(u,v)$ 表示边 (u,v) 流量的下界，$C(u,v)$ 表示边 (u,v) 流量的上界，设 $F(u,v)$ 表示边 (u,v) 的实际流量
设 $G(u,v)=F(u,v)-B(u,v)$，则 $0 \le G(u,v) \le C(u,v)-B(u,v)$

- 无源汇的上下界可行流

建立超级源点 S^* 和超级汇点 T^*，对于原图每一条边 (u,v) 在新网络中连如下三条边：$S^* \to v$，容量为 $B(u,v)$；$u \to T^*$，容量为 $B(u,v)$；$u \to v$，容量为 $C(u,v)-B(u,v)$。最后求新网络的最大流，判断从超级源点 S^* 出发的边是否都满流即可，边 (u,v) 的最终解中的实际流量为 $G(u,v)+B(u,v)$。

- 有源汇的上下界可行流

从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边。按照无源汇的上下界可行流一样做即可，流量即为 $T \to S$ 边上的流量。

- 有源汇的上下界最大流

– 在有源汇的上下界可行流中，从汇点 T 到源点 S 的边改为连一条上界为 ∞，下界为 x 的边。x 满足二分性质，找到最大的 x 使得新网络存在有源汇的上下界可行流即为原图的最大流。

– 从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边，变成无源汇的网络。按照无源汇的上下界可行流的方法，建立超级源点 S^* 与超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，再将从汇点 T 到源点 S 的这条边拆掉，求一次 $S \to T$ 的最大流即可。

- 有源汇的上下界最小流

– 在有源汇的上下界可行流中，从汇点 T 到源点 S 的边改为连一条上界为 x，下界为 0 的边。x 满足二分性质，找到最小的 x 使得新网络存在有源汇的上下界可行流即为原图的最大流。

– 按照无源汇的上下界可行流的方法，建立超级源点 S^* 与超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，但是注意不加上汇点 T 到源点 S 的这条边，即不使之改为无源汇的网络去求解。求完后，再加上那条汇点 T 到源点 S 的边，上界为 ∞ 的边。因为这条边的下界为 0，所以 S^*, T^* 无影响，再求一次 $S^* \to T^*$ 的最大流。若超级源点 S^* 出发的边全部满流，则 $T \to S$ 边上的流量即为原图的最小流，否则无解。

- 上下界费用流

求无源汇上下界最小费用可行流或有源汇上下界最小费用最大可行流，用相应构图方法，给边加上费用即可。
求有源汇上下界最小费用最小可行流，先按相应构图方法建图，求出一个保证必要边满流情况下的最小费用。如果费用全部非负，那么此时的费用即为答案。如果费用有负数，继续做从 S 到 T 的流量任意的最小费用流，加上原来的费用就是答案。

费用流消负环

新建超级源 S^* 和超级汇 T^*，对于所有流量非空的负权边 e，先满流（ans+=$e.f$*$e.c$，$e.rev.f$+=$e.f$，$e.f$=0），再连边 $S^* \to e.to$，$e.from \to T*$，流量均为 $e.f(>0)$，费用均为 0。再连边 $T \to S$，流量为 ∞，费用为 0。跑一遍 $S^* \to T^*$ 的最小费用最大流，将费用累加 ans，拆掉 $T \to S$ 那条边（此边的流量为残量网络中 $S \to T$ 的流量。此时负环已消，再继续跑最小费用最大流。

二物流

水源 S_1，水汇 T_1，油源 S_2，油汇 T_2，每根管道流量共用，使流量和最大。
建超级源 S_1^*，超级汇 T_1^*，连边 $S_1^* \to S_1$，$S_1^* \to S_2$，$T_1 \to T_1^*$，$T_2 \to T_1^*$，设最大流为 x_1。
建超级源 S_2^*，超级汇 T_2^*，连边 $S_2^* \to S_1$，$S_2^* \to T_2$，$T_1 \to T_2^*$，$S_2 \to T_2^*$，设最大流为 x_2。则最大流中水流量 $\frac{x_1+x_2}{2}$，油流量 $\frac{x_1-x_2}{2}$。

最大权闭合子图

给定一个带点权的有向图，求其最大权闭合子图。
从源点 S 向每一条正权点连一条容量为权值的边，每个负权点向汇点 T 连一条容量为权值绝对值的边，有向图原来的边容量为 ∞。求它的最小割，与源点 S 连通的点构成最大权闭合子图，权值为正权值和 − 最小割。

最大密度子图

给定一个无向图，求其一个子图，使得子图的边数 $|E|$ 和点数 $|V|$ 满足 $\frac{|E|}{|V|}$ 最大。
二分答案 k，使得 $|E|-k|V| \ge 0$ 有解，将原图边和点都看作点，边 (u,v) 分别向 u 和 v 连边求最大权闭合子图。

2. Math

2.1 int64 相乘取模 (Durandal)

```
int64_t mul(int64_t x, int64_t y, int64_t p) {
    int64_t t = (x * y - (int64_t) ((long double) x / p
    ↪ * y + 1e-3) * p) % p;
```

```
    return t < 0 ? t + p : t;
}
```

2.2 ex-Euclid (gy)

```
// return gcd(a, b)
// ax+by=gcd(a,b)
int extend_gcd(int a, int b, int &x, int &y) {
    if (b == 0) {
        x = 1, y = 0;
        return a;
    }
    int res = extend_gcd(b, a % b, x, y);
    int t = y;
    y = x - a / b * y;
    x = t;
    return res;
}

// return minimal positive integer x so that ax+by=c
// or -1 if such x does not exist
int solve_equ(int a, int b, int c) {
    int x, y, d;
    d = extend_gcd(a, b, x, y);
    if (c % d)
        return -1;
    int t = c / d;
    x *= t;
    y *= t;
    int k = b / d;
    x = (x % k + k) % k;
    return x;
}

// return minimal positive integer x so that ax==b(mod
 ↪ p)
// or -1 if such x does not exist
int solve(int a, int b, int p) {
    a = (a % p + p) % p;
    b = (b % p + p) % p;
    return solve_equ(a, p, b);
}
```

2.3 中国剩余定理 (Durandal)

返回是否可行，余数和模数结果为 r_1, m_1

```
bool CRT(int &r1, int &m1, int r2, int m2) {
    int x, y, g = extend_gcd(m1, m2, x, y);
    if ((r2 - r1) % g != 0) return false;
    x = 1ll * (r2 - r1) * x % m2;
    if (x < 0) x += m2;
    x /= g;
    r1 += m1 * x;
    m1 *= m2 / g;
    return true;
}
```

2.4 线性同余不等式 (Durandal)

必须满足 $0 \le d < m$, $0 \le l \le r < m$，返回 $\min\{x \ge 0 \mid l \le x \cdot d \bmod m \le r\}$，无解返回 -1

```
int64_t calc(int64_t d, int64_t m, int64_t l, int64_t
 ↪ r) {
    if (l == 0) return 0;
    if (d == 0) return -1;
    if (d * 2 > m) return calc(m - d, m, m - r, m -
l);
    if ((l - 1) / d < r / d) return (l - 1) / d + 1;
    int64_t k = calc((-m % d + d) % d, d, l % d, r %
     ↪ d);
    if (k == -1) return -1;
    return (k * m + l - 1) / d + 1;
}
```

2.5 平方剩余 (Nightfall)

$x^2 \equiv a \pmod p, 0 \le a < p$

返回是否存在解

p 必须是质数，若是多个单次质数的乘积可以分别求解再用 CRT 合并

复杂度为 $O(\log n)$

```
void multiply(ll &c, ll &d, ll a, ll b, ll w) {
    int cc = (a * c + b * d % MOD * w) % MOD;
    int dd = (a * d + b * c) % MOD; c = cc, d = dd; }
bool solve(int n, int &x) {
    if (n==0) return x=0,true; if (MOD==2) return
     ↪ x=1,true;
    if (power(n, MOD / 2, MOD) == MOD - 1) return
     ↪ false;
    ll c = 1, d = 0, b = 1, a, w;
    // finding a such that a^2 - n is not a square
    do { a = rand() % MOD; w = (a * a - n + MOD) % MOD;
        if (w == 0) return x = a, true;
    } while (power(w, MOD / 2, MOD) != MOD - 1);
    for (int times = (MOD + 1) / 2; times; times >>=
     ↪ 1) {
        if (times & 1) multiply(c, d, a, b, w);
        multiply(a, b, a, b, w); }
    // x = (a + sqrt(w)) ^ ((p + 1) / 2)
    return x = c, true; }
```

2.6 组合数 (Nightfall)

```
int l,a[33],p[33],P[33];
U fac(int k,LL n){// 求 n! mod pk^tk, 返回值 U{ 不包含
 ↪ pk 的值, pk 出现的次数 }
    if (!n)return U{1,0};LL x=n/p[k],y=n/P[k],ans=1;int
     ↪ i;
    if(y){// 求出循环节的答案
        for(i=2;i<P[k];i++)if(i%p[k])ans=ans*i%P[k];
        ans=Pw(ans,y,P[k]);
    }for(i=y*P[k];i<=n;i++) if(i%p[k])ans=ans*i%M;// 求
     ↪ 零散部分
    U z=fac(k,x);return U{ans*z.x%M,x+z.z};
}LL get(int k,LL n,LL m){// 求 C(n,m) mod pk^tk
    U a=fac(k,n),b=fac(k,m),c=fac(k,n-m);// 分三部分求
     ↪ 解
    return Pw(p[k],a.z-b.z-c.z,P[k])*a.x%P[k]*inv(
     ↪ b.x,P[k])%P[k]*inv(c.x,P[k])%P[k];
}LL CRT(){// CRT 合并答案
    LL d,w,y,x,ans=0;
    fr(i,1,l)w=M/P[i],exgcd(w,P[i],x,y),
        ans=(ans+w*x%M*a[i])%M;
    return (ans+M)%M;
}LL C(LL n,LL m){// 求 C(n,m)
    fr(i,1,l)a[i]=get(i,n,m);
    return CRT();
}LL exLucas(LL n,LL m,int M){
    int jj=M,i; // 求 C(n,m)mod M,M=prod(pi^ki), 时间
     ↪ O(pi^kilg^2n)
    for(i=2;i*i<=jj;i++)if(jj%i==0)
     ↪ for(p[++l]=i,P[l]=1;jj%i==0;P[l]*=p[l])jj/=i;
```

```
    if(jj>1)l++,p[l]=P[l]=jj;
    return C(n,m);}
```

2.7 高斯消元 (ct)

增广矩阵大小为 $m \times (n+1)$

```
db a[maxn][maxn], x[maxn];
int main()
{
    int rank = 0;

    for (int i = 1, now = 1; i <= n && now <= m; ++now)
    {
        int tmp = i;
        for (int j = i + 1; j <= n; ++j)
            if (fabs(a[j][now]) > fabs(a[tmp][now]))tmp = j;
        for (int k = now; k <= m; ++k)
            std::swap(a[i][k], a[tmp][k]);
        if (fabs(a[i][now]) < eps) continue;

        for (int j = i + 1; j <= n; ++j)
        {
            db tmp = a[j][now] / a[i][now];
            for (int k = now; k <= m; ++k)
                a[j][k] -= tmp * a[i][k];
        }
        ++i; ++rank;
    }

    if (rank == n)
    {
        x[n] = a[n][n + 1] / a[n][n];
        for (int i = n - 1; i; --i)
        {
            for (int j = i + 1; j <= n; ++j)
                a[i][n + 1] -= x[j] * a[i][j];
            x[i] = a[i][n + 1] / a[i][i];
        }
    }
    else puts("Infinite Solution!");
    return 0;
}
```

2.8 Miller Rabin & Pollard Rho (gy)

In Java, use BigInteger.isProbablePrime(int certainty) to replace miller_rabin(BigInteger number)

Test Set	First Wrong Answer
$2, 3, 5, 7$	(INT32_MAX)
$2, 7, 61$	$4, 759, 123, 141$
$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37$	(INT64_MAX)

```
const int test_case_size = 12;
const int test_cases[test_case_size] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};

int64_t multiply_mod(int64_t x, int64_t y, int64_t p) {
    int64_t t = (x * y - (int64_t) ((long double) x / p * y + 1e-3) * p) % p;
    return t < 0 ? t + p : t;
}

int64_t add_mod(int64_t x, int64_t y, int64_t p) {
    return (0ull + x + y) % p;
}

int64_t power_mod(int64_t x, int64_t exp, int64_t p) {
    int64_t ans = 1;
    while (exp) {
        if (exp & 1)
            ans = multiply_mod(ans, x, p);
        x = multiply_mod(x, x, p);
        exp >>= 1;
    }
    return ans;
}

bool miller_rabin_check(int64_t prime, int64_t base) {
    int64_t number = prime - 1;
    for (; ~number & 1; number >>= 1)
        continue;
    int64_t result = power_mod(base, number, prime);
    for (; number != prime - 1 && result != 1 && result != prime - 1; number <<= 1)
        result = multiply_mod(result, result, prime);
    return result == prime - 1 || (number & 1) == 1;
}

bool miller_rabin(int64_t number) {
    if (number < 2)
        return false;
    if (number < 4)
        return true;
    if (~number & 1)
        return false;
    for (int i = 0; i < test_case_size && test_cases[i] < number; i++)
        if (!miller_rabin_check(number, test_cases[i]))
            return false;
    return true;
}

int64_t gcd(int64_t x, int64_t y) {
    return y == 0 ? x : gcd(y, x % y);
}

int64_t pollard_rho_test(int64_t number, int64_t seed) {
    int64_t x = rand() % (number - 1) + 1, y = x;
    int head = 1, tail = 2;
    while (true) {
        x = multiply_mod(x, x, number);
        x = add_mod(x, seed, number);
        if (x == y)
            return number;
        int64_t answer = gcd(std::abs(x - y), number);
        if (answer > 1 && answer < number)
            return answer;
        if (++head == tail) {
            y = x;
            tail <<= 1;
        }
    }
}

void factorize(int64_t number, std::vector<int64_t> &divisor) {
    if (number > 1) {
        if (miller_rabin(number)) {
            divisor.push_back(number);
        } else {
            int64_t factor = number;
```

```cpp
            while (factor >= number)
                factor = pollard_rho_test(number, rand() % (number - 1) + 1);
            factorize(number / factor, divisor);
            factorize(factor, divisor);
        }
    }
}
```

2.9 $O(m^2 \log n)$ 线性递推 (lhy)

```cpp
typedef vector<int> poly;
//{1, 3} {2, 1} an = 2an-1 + an-2, calc(3) = 7
struct LinearRec{
    int n, LOG;
    poly first, trans;
    vector<poly> bin;
    poly add(poly &a, poly &b)
    {
        poly res(n * 2 + 1, 0);
        for(int i = 0; i <= n; i++)
            for(int j = 0; j <= n; j++)
                (res[i + j] += 1ll * a[i] * b[j] % mo) %= mo;
        for(int i = 2 * n; i > n; i--)
        {
            for(int j = 0; j < n; j++)
                (res[i - 1 - j] += 1ll * res[i] * trans[j] % mo) %= mo;
            res[i] = 0;
        }
        res.erase(res.begin() + n + 1, res.end());
        return res;
    }
    LinearRec(poly &first, poly &trans, int LOG): LOG(LOG), first(first), trans(trans)
    {
        n = first.size();
        poly a(n + 1, 0);
        a[1] = 1;
        bin.push_back(a);
        for(int i = 1; i < LOG; i++)
            bin.push_back(add(bin[i - 1], bin[i - 1]));
    }
    int calc(long long k)
    {
        poly a(n + 1, 0);
        a[0] = 1;
        for(int i = 0; i < LOG; i++)
            if((k >> i) & 1)a = add(a, bin[i]);
        int ret = 0;
        for(int i = 0; i < n; i++)
            if((ret += 1ll * a[i + 1] * first[i] % mo) >= mo)ret -= mo;
        return ret;
    }
};
```

2.10 线性基 (ct)

```cpp
int main()
{
    for (int i = 1; i <= n; ++i)
    {
        ull x = F();
        cmax(m, 63 - __builtin_clzll(x));
        for ( ; x; )
        {
            tmp = __builtin_ctzll(x);
            if (!b[tmp])
            {
                b[tmp] = x;
                break;
            }
            x ^= b[tmp];
        }
    }
}
```

2.11 FFT NTT FWT (lhy,ct,gy)

FFT (ct)

0-based

```cpp
typedef double db;
const db pi = acos(-1);

struct Complex {
    db x, y;
    inline Complex operator * (const Complex &that) const {return (Complex) {x * that.x - y * that.y, x * that.y + y * that.x};}
    //inline Complex operator + (const Complex &that) const {return (Complex) {x + that.x, y + that.y};}
    inline Complex operator += (const Complex &that){x+=that.x;y+=that.y;}
    inline Complex operator - (const Complex &that) const {return (Complex) {x - that.x, y - that.y};}
} buf_a[maxn], buf_b[maxn], buf_c[maxn], w[maxn], c[maxn], a[maxn], b[maxn];

int n;
void bit_reverse(Complex *x, Complex *y)
{
    for (int i = 0; i < n; ++i) y[i] = x[i];
    Complex tmp;
    for (int i = 0, j = 0; i < n; ++i)
    {
        (i>j)?tmp=y[i],y[i]=y[j],y[j]=tmp,0:1;
        for (int l = n >> 1; (j ^= l) < l; l >>= 1);
    }
}
void init()
{
    int h=n>>1;
    for (int i = 0; i < h; ++i) w[i+h] = (Complex) {cos(2 * pi * i / n), sin(2 * pi * i / n)};
    for (int i = h; i--; )w[i]=w[i<<1];
}
void dft(Complex *a)
{
    Complex tmp;
    for(int p = 2, m = 1; m != n; p = (m = p) << 1)
        for(int i = 0; i != n; i += p) for(int j = 0; j != m; ++j)
        {
            tmp = a[i + j + m] * w[j + m];
            a[i + j + m] = a[i + j] - tmp;
            a[i + j] += tmp;
        }
}

```

```cpp
int main()
{
    fread(S, 1, 1 << 20, stdin);
    int na = F(), nb = F(), x;
    for (int i = 0; i <= na; ++i) a[i].x=F();
    for (int i = 0; i <= nb; ++i) b[i].x=F();
    for (n = 1; n < na + nb + 1; n <<= 1) ;
    bit_reverse(a, buf_a);
    bit_reverse(b, buf_b);
    init();
    dft(buf_a);
    dft(buf_b);
    for (int i = 0; i < n; ++i) c[i] = buf_a[i] * buf_b[i];
    std::reverse(c + 1, c + n);
    bit_reverse(c, buf_c);
    dft(buf_c);
    for (int i = 0; i <= na + nb; ++i) printf("%d%c", int(buf_c[i].x / n + 0.5), " \n"[i==na+nb]);
    return 0;
}
```

NTT (gy)

0-based

```cpp
const int N = 1e6 + 10;
const int64_t MOD = 998244353, G = 3;
int rev[N];

int64_t powMod(int64_t a, int64_t exp) {
    int64_t ans = 1;
    while (exp) {
        if (exp & 1)
            (ans *= a) %= MOD;
        (a *= a) %= MOD;
        exp >>= 1;
    }
    return ans;
}

void number_theoretic_transform(int64_t *p, int n, int idft) {
    for (int i = 0; i < n; i++)
        if (i < rev[i])
            std::swap(p[i], p[rev[i]]);
    for (int j = 1; j < n; j <<= 1) {
        static int64_t wn1, w, t0, t1;
        wn1 = powMod(G, (MOD - 1) / (j << 1));
        if (idft == -1)
            wn1 = powMod(wn1, MOD - 2);
        for (int i = 0; i < n; i += j << 1) {
            w = 1;
            for (int k = 0; k < j; k++) {
                t0 = p[i + k];
                t1 = w * p[i + j + k] % MOD;
                p[i + k] = (t0 + t1) % MOD;
                p[i + j + k] = (t0 - t1 + MOD) % MOD;
                (w *= wn1) %= MOD;
            }
        }
    }
    if (idft == -1) {
        int nInv = powMod(n, MOD - 2);
        for (int i = 0; i < n; i++)
            (p[i] *= nInv) %= MOD;
    }
}

int64_t *ntt_main(int64_t *a, int64_t *b, int n, int m) {
    static int64_t aa[N], bb[N];
    static int nn, len;
    len = 0;
    for (nn = 1; nn < m + n; nn <<= 1)
        len++;
    for (int i = 0; i < nn; i++) {
        aa[i] = a[i];
        bb[i] = b[i];
    }
    rev[0] = 0;
    for (int i = 1; i < nn; i++)
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (len - 1));
    number_theoretic_transform(aa, nn, 1);
    number_theoretic_transform(bb, nn, 1);
    for (int i = 0; i < nn; i++)
        (aa[i] *= bb[i]) %= MOD;
    number_theoretic_transform(aa, nn, -1);
    return aa;
}
```

FWT (lhy)

0-based

```cpp
void fwt(int n, int *x, bool inv = false)
{
    for(int i = 0; i < n; i++)
        for(int j = 0; j < (1 << n); j++)
            if((j >> i) & 1)
            {
                int p = x[j ^ (1 << i)], q = x[j];
                if(!inv)
                {
                    //xor
                    x[j ^ (1 << i)] = p - q;
                    x[j] = p + q;
                    //or
                    x[j ^ (1 << i)] = p;
                    x[j] = p + q;
                    //and
                    x[j ^ (1 << i)] = p + q;
                    x[j] = q;
                }
                else
                {
                    //xor
                    x[j ^ (1 << i)] = (p + q) >> 1;
                    x[j] = (q - p) >> 1;
                    //or
                    x[j ^ (1 << i)] = p;
                    x[j] = q - p;
                    //and
                    x[j ^ (1 << i)] = p - q;
                    x[j] = q;
                }
            }
}

void solve(int n, int *a, int *b, int *c)
{
    fwt(n, a);
    fwt(n, b);
    for(int i = 0; i < (1 << n); i++)
        c[i] = a[i] * b[i];
    fwt(n, c, 1);
}
```

2.12 Lagrange 插值 (ct)

求解 $\sum\limits_{i=1}^{n} i^k \bmod (10^9+7)$

```cpp
const int mod = 1e9 + 7;
int f[maxn], pre[maxn], suf[maxn], inp[maxn], p[maxn];
inline int qpow(int base, int power)
{
    int ret = 1;
    for (; power; power >>= 1, base = 1ll * base * base % mod)
        power & 1 ? ret = 1ll * ret * base % mod : 0;
    return ret;
}
bool vis[maxn];
int pr[maxn], prcnt, fpow[maxn];
int main()
{
    int n = F(), k = F();
    // ************
    fpow[1] = 1;
    for (int i = 2; i <= k + 2; ++i)
    {
        if (!vis[i]) pr[++prcnt] = i, fpow[i] = qpow(i, k);
        for (int j = 1; j <= prcnt && i * pr[j] <= k + 2; ++j)
        {
            vis[i * pr[j]] = 1;
            fpow[i * pr[j]] = 1ll * fpow[i] * fpow[pr[j]] % mod;
            if (i % pr[j] == 0) break;
        }
    }
    // ************* pre-processing
    for (int i = 1; i <= k + 2; ++i) f[i] = (f[i - 1] + fpow[i]) % mod;
    if (n <= k + 2) return !printf("%d\n", f[n] );
    pre[0] = 1;
    for (int i = 1; i <= k + 3; ++i) pre[i] = 1ll * pre[i - 1] * (n - i) % mod;
    suf[k + 3] = 1;
    for (int i = k + 2; i >= 0; --i) suf[i] = 1ll * suf[i + 1] * (n - i) % mod;

    p[0] = 1;
    for (int i = 1; i <= k + 2; ++i) p[i] = (1ll * p[i - 1] * i) % mod;

    inp[k + 2] = qpow(p[k + 2], mod - 2);

    for (int i = k + 1; i >= 0; --i) inp[i] = (1ll * inp[i + 1] * (i + 1)) % mod;

    int ans = 0;
    for (int i = 1; i <= k + 2; ++i)
    {
        int temp = inp[k + 2 - i]; if ((k + 2 - i) & 1) temp = mod - temp;
        int tmp = 1ll * pre[i - 1] * suf[i + 1] % mod * temp % mod * inp[i - 1] % mod * f[i] % mod;
        ans = (ans + tmp) % mod;
    }
    printf("%d\n", ans );
    return 0;
}
```

2.13 杜教筛 (ct)

Dirichlet 卷积：$(f*g)(n)=\sum\limits_{d|n} f(d)g(\frac{n}{d})$

对于积性函数 $f(n)$，求其前缀和 $S(n)=\sum\limits_{i=1}^{n} f(i)$

寻找一个恰当的积性函数 $g(n)$，使得 $g(n)$ 和 $(f*g)(n)$ 的前缀和都容易计算

则 $g(1)S(n)=\sum\limits_{i=1}^{n}(f*g)(i)-\sum\limits_{i=2} ng(i)S(\lfloor\frac{n}{i}\rfloor)$

$\mu(n)$ 和 $\phi(n)$ 取 $g(n)=1$

两种常见形式：

- $S(n)=\sum\limits_{i=1}^{n}(f\cdot g)(i)$ 且 $g(i)$ 为完全积性函数

$S(n)=\sum\limits_{i=1}^{n}((f*1)\cdot g)(i)-\sum\limits_{i=2}^{n} S(\lfloor\frac{n}{i}\rfloor)g(i)$

- $S(n)=\sum\limits_{i=1}^{n}(f*g)(i)$

$S(n)=\sum\limits_{i=1}^{n} g(i)\sum\limits_{ij\le n}(f*1)(j)-\sum\limits_{i=2}^{n} S(\lfloor\frac{n}{i}\rfloor)$

```cpp
int phi[maxn], pr[maxn / 10], prcnt;
ll sph[maxn];
bool vis[maxn];
const int moha = 3333331;
struct Hash {
    Hash *next;
    int ps; ll ans;
} *last1[moha], mem[moha], *tot = mem;
inline ll S1(int n)
{
    if (n < maxn) return sph[n];
    for (R Hash *iter = last1[n % moha]; iter; iter = iter -> next)
        if (iter -> ps == n) return iter -> ans;

    ll ret = 1ll * n * (n + 1ll) / 2;
    for (ll i = 2, j; i <= n; i = j + 1)
    {
        j = n / (n / i);
        ret -= S1(n / i) * (j - i + 1);
    }
    *++tot = (Hash) {last1[n % moha], n, ret}; last1[n % moha] = tot;
    return ret;
}
int main()
{
    int T; scanf("%d", &T);
    phi[1] = sph[1] = 1;
    for (int i = 2; i < maxn; ++i)
    {
        if (!vis[i]) pr[++prcnt] = i, phi[i] = i - 1;
        sph[i] = sph[i - 1] + phi[i];
        for (int j = 1; j <= prcnt && 1ll * i * pr[j] < maxn; ++j)
        {
            vis[i * pr[j]] = 1;
            if (i % pr[j])
                phi[i * pr[j]] = phi[i] * (pr[j] - 1);
            else
            {
                phi[i * pr[j]] = phi[i] * pr[j];
                break;
            }
        }
    }
    for (; T; --T)
```

```
    {
        int N; scanf("%d", &N);
        printf("%lld\n", S1(N));
    }
    return 0;
}
```

2.14 Extended Eratosthenes Sieve (Nightfall)

一般积性函数的前缀和，要求：$f(p)$ 为多项式

```
struct poly { LL a[2]; poly() {} int size() const
  ↪ {return 2;}
    poly(LL x, LL y) {a[0] = x; a[1] = y;} };
poly operator * (poly a, int p) {
    return poly(a.a[0], a.a[1] * p);
}
poly operator - (const poly &a, const poly &b){
    return poly(a.a[0]-b.a[0], a.a[1]-b.a[1]);
}
poly sum_fp(LL l, LL r) { // f(p) = 1 + p
    return poly(r-l+1, (l+r) * (r-l+1) / 2);
}
LL fpk(LL p, LL k) { // f(p^k) = sum{i in 0..k | p^i}
    LL res = 0, q = 1;
    for (int i = 0; i <= k; ++ i) { res += q; q *= p; }
    return res;
}
LL Value(poly p) { return p.a[0] + p.a[1]; }
LL n; int m; vector<poly> A, B; vector<int> P;
//need w = n/k , about O(w^0.7)
LL calc(LL w, int id, LL f) {
    LL T = w>m ? Value(B[n/w]) : Value(A[w]);
    if (id) T -= Value(A[P[id - 1]]); LL ret = T * f;
    for (int i = id; i < P.size(); ++ i) {
        int p = P[i], e = 1; LL q = (LL) p*p; if
          ↪ (q>w) break;
        ret += calc(w/p, i+1, f * fpk(p, 1));
        while (1) {
            ++ e; LL f2 = f * fpk(p, e); ret+=f2; LL qq
              ↪ = q*p;
            if (qq <= w) {
                ret += calc(w/q, i+1, f2); q = qq;
            } else break; } }
    return ret;
}
void prepare(LL N) { // about O(n^0.67)
    n = N; m = (int) sqrt(n + .5L);
    A.resize(m + 1); B.resize(m + 1);
    P.clear(); vector<int> isp; isp.resize(m + 1, 1);
    for (int i = 1; i <= m; ++ i) {
        A[i] = sum_fp(2, i); B[i] = sum_fp(2, n / i); }
    for (int p = 2; p <= m; ++ p) {
        if (isp[p]) P.push_back(p);
        for (int j : P) { if (j * p > m) break;
            isp[j * p] = 0; if (j % p == 0) break; }
        if (!isp[p]) continue;
        poly d = A[p - 1]; LL p2 = (LL) p * p;
        int to = (int) min(n / p2, (LL) m);
        for (int i = 1; i <= m / p; ++ i)
            B[i] = B[i] - (B[i * p] - d) * p;
        for (int i = m / p + 1; i <= to; ++ i)
            B[i] = B[i] - (A[n / p / i] - d) * p;
        for (int i = m; i >= p2; -- i)
            A[i] = A[i] - (A[i / p] - d) * p; }
}
main() : prepare(n); LL ans = calc(n, 0, 1);
```

2.15 BSGS (ct,Durandal)

2.15.1 BSGS (ct)

p 是素数，返回 $\min\{x \geq 0 \mid y^x \equiv z \pmod p\}$

```
const int mod = 19260817;
struct Hash
{
    Hash *next;
    int key, val;
} *last[mod], mem[100000], *tot = mem;
inline void insert(R int x, R int v)
{
    *++tot = (Hash) {last[x % mod], x, v}; last[x %
      ↪ mod] = tot;
}
inline int query(R int x)
{
    for (R Hash *iter = last[x % mod]; iter; iter =
      ↪ iter -> next)
        if (iter -> key == x) return iter -> val;
    return -1;
}
inline void del(R int x)
{
    last[x % mod] = 0;
}
int main()
{
    for (; T; --T)
    {
        R int y, z, p; scanf("%d%d%d", &y, &z, &p);
        R int m = (int) sqrt(p * 1.0);
        y %= p; z %= p;
        if (!y && !z) {puts("0"); continue;}
        if (!y) {puts("Orz, I cannot find x!");
          ↪ continue;}
        R int pw = 1;
        for (R int i = 0; i < m; ++i, pw = 1ll * pw * y
          ↪ % p) insert(1ll * z * pw % p, i);
        R int ans = -1;
        for (R int i = 1, t, pw2 = pw; i <= p / m + 1;
          ↪ ++i, pw2 = 1ll * pw2 * pw % p)
            if ((t = query(pw2)) != -1) {ans = i * m -
              ↪ t; break;}
        if (ans == -1) puts("Orz, I cannot find x!");
        else printf("%d\n", ans );
        tot = mem; pw = 1;
        for (R int i = 0; i < m; ++i, pw = 1ll * pw * y
          ↪ % p) del(1ll * z * pw % p);
    }
    return 0;
}
```

2.15.2 ex-BSGS (Durandal)

必须满足 $0 \leq a < p$, $0 \leq b < p$，返回 $\min\{x \geq 0 \mid a^x \equiv b \pmod p\}$

```
int64_t ex_bsgs(int64_t a, int64_t b, int64_t p) {
    if (b == 1)
        return 0;
    int64_t t, d = 1, k = 0;
    while ((t = std::__gcd(a, p)) != 1) {
        if (b % t) return -1;
        k++, b /= t, p /= t, d = d * (a / t) % p;
        if (b == d) return k;
```

```cpp
    }
    map.clear();
    int64_t m = std::ceil(std::sqrt((long double) p));
    int64_t a_m = pow_mod(a, m, p);
    int64_t mul = b;
    for (int j = 1; j <= m; j++) {
        (mul *= a) %= p;
        map[mul] = j;
    }
    for (int i = 1; i <= m; i++) {
        (d *= a_m) %= p;
        if (map.count(d))
            return i * m - map[d] + k;
    }
    return -1;
}

int main() {
    int64_t a, b, p;
    while (scanf("%lld%lld%lld", &a, &b, &p) != EOF)
        printf("%lld\n", ex_bsgs(a, b, p));
    return 0;
}
```

2.16 直线下整点个数 (gy)

必须满足 $a \geq 0, b \geq 0, m > 0$，返回 $\sum_{i=0}^{n-1} \frac{a+bi}{m}$

```cpp
int64_t count(int64_t n, int64_t a, int64_t b, int64_t m) {
    if (b == 0)
        return n * (a / m);
    if (a >= m)
        return n * (a / m) + count(n, a % m, b, m);
    if (b >= m)
        return (n - 1) * n / 2 * (b / m) + count(n, a, b % m, m);
    return count((a + b * n) / m, (a + b * n) % m, m, b);
}
```

2.17 Pell equation (gy)

$x^2 - ny^2 = 1$ 有解当且仅当 n 不为完全平方数
求其特解 (x_0, y_0)
其通解为 $(x_{k+1}, y_{k+1}) = (x_0x_k + ny_0y_k, x_0y_k + y_0x_k)$

```cpp
std::pair<int64_t, int64_t> pell(int64_t n) {
    static int64_t p[N], q[N], g[N], h[N], a[N];
    p[1] = q[0] = h[1] = 1;
    p[0] = q[1] = g[1] = 0;
    a[2] = std::sqrt(n) + 1e-7L;
    for (int i = 2; true; i++) {
        g[i] = -g[i - 1] + a[i] * h[i - 1];
        h[i] = (n - g[i] * g[i]) / h[i - 1];
        a[i + 1] = (g[i] + a[2]) / h[i];
        p[i] = a[i] * p[i - 1] + p[i - 2];
        q[i] = a[i] * q[i - 1] + q[i - 2];
        if (p[i] * p[i] - n * q[i] * q[i] == 1)
            return std::make_pair(p[i], q[i]);
    }
}
```

2.18 单纯形 (gy)

返回 $x_{m\times1}$ 使得 $\max\{c_{1\times m} \cdot x_{m\times1} \mid x_{m\times1} \geq 0_{m\times1}, A_{n\times m} \cdot x_{m\times1} \leq b_{n\times1}\}$

```cpp
const double eps = 1e-8;

std::vector<double> simplex(const std::vector<
    std::vector<double> > &A, const std::vector<double>
    &b, const std::vector<double> &c) {
    int n = A.size(), m = A[0].size() + 1, r = n, s = m
        - 1;
    std::vector< std::vector<double> > D(n + 2,
        std::vector<double>(m + 1));
    std::vector<int> ix(n + m);
    for (int i = 0; i < n + m; i++) {
        ix[i] = i;
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m - 1; j++) {
            D[i][j] = -A[i][j];
        }
        D[i][m - 1] = 1;
        D[i][m] = b[i];
        if (D[r][m] > D[i][m]) {
            r = i;
        }
    }

    for (int j = 0; j < m - 1; j++) {
        D[n][j] = c[j];
    }
    D[n + 1][m - 1] = -1;
    for (double d; true; ) {
        if (r < n) {
            std::swap(ix[s], ix[r + m]);
            D[r][s] = 1. / D[r][s];
            for (int j = 0; j <= m; j++) {
                if (j != s) {
                    D[r][j] *= -D[r][s];
                }
            }
            for (int i = 0; i <= n + 1; i++) {
                if (i != r) {
                    for (int j = 0; j <= m; j++) {
                        if (j != s) {
                            D[i][j] += D[r][j] *
                                D[i][s];
                        }
                    }
                    D[i][s] *= D[r][s];
                }
            }
        }
        r = -1, s = -1;
        for (int j = 0; j < m; j++) {
            if (s < 0 || ix[s] > ix[j]) {
                if (D[n + 1][j] > eps || D[n + 1][j] >
                    -eps && D[n][j] > eps) {
                    s = j;
                }
            }
        }
        if (s < 0) {
            break;
        }
        for (int i = 0; i < n; i++) {
            if (D[i][s] < -eps) {
                if (r < 0 || (d = D[r][m] / D[r][s] -
                    D[i][m] / D[i][s]) < -eps || d <
                    eps && ix[r + m] > ix[i + m]) {
```

```cpp
                    r = i;
                }
            }
        }

        if (r < 0) {
            return /* solution unbounded */ std::vector<double>();
        }
    }
    if (D[n + 1][m] < -eps) {
        return /* no solution */ std::vector<double>();
    }

    std::vector<double> x(m - 1);
    for (int i = m; i < n + m; i++) {
        if (ix[i] < m - 1) {
            x[ix[i]] = D[i - m][m];
        }
    }
    return x;
}
```

2.19 数学知识 (gy)

原根

当 $\gcd(a,m)=1$ 时，使 $a^x \equiv 1 \pmod m$ 成立的最小正整数 x 称为 a 对于模 m 的阶，计为 $\mathrm{ord}_m(a)$。
阶的性质：$a^n \equiv 1 \pmod m$ 的充要条件是 $\mathrm{ord}_m(a) \mid n$，可推出 $\mathrm{ord}_m(a) \mid \psi(m)$。
当 $\mathrm{ord}_m(g) = \psi(m)$ 时，则称 g 是模 n 的一个原根，$g^0, g^1, \ldots, g^{\psi(m)-1}$ 覆盖了 m 以内所有与 m 互素的数。
原根存在的充要条件：$m = 2, 4, p^k, 2p^k$，其中 p 为奇素数，$k \in \mathbb{N}^*$

求和公式

- $\sum\limits_{k=1}^{n}(2k-1)^2 = \frac{1}{3}n(4n^2-1)$
- $\sum\limits_{k=1}^{n}k^3 = \frac{1}{4}n^2(n+1)^2$
- $\sum\limits_{k=1}^{n}(2k-1)^3 = n^2(2n^2-1)$
- $\sum\limits_{k=1}^{n}k^4 = \frac{1}{30}n(n+1)(2n+1)(3n^2+3m-1)$
- $\sum\limits_{k=1}^{n}k^5 = \frac{1}{12}n^2(n+1)^2(2n^2+2n-1)$
- $\sum\limits_{k=1}^{n}k(k+1) = \frac{1}{3}n(n+1)(n+2)$
- $\sum\limits_{k=1}^{n}k(k+1)(k+2) = \frac{1}{4}n(n+1)(n+2)(n+3)$
- $\sum\limits_{k=1}^{n}k(k+1)(k+2)(k+3) = \frac{1}{5}n(n+1)(n+2)(n+3)(n+4)$

错排公式

D_n 表示 n 个元素错位排列的方案数
$D_1 = 0, D_2 = 1$
$D_n = (n-1)(D_{n-2} + D_{n-1}), n \geq 3$
$D_n = n! \cdot (1 - \frac{1}{1!} + \frac{1}{2!} - \cdots + (-1)^n \frac{1}{n!})$

Fibonacci sequence

$F_0 = 0, F_1 = 1$
$F_n = F_{n-1} + F_{n-2}$
$F_{n+1} \cdot F_{n-1} - F_n^2 = (-1)^n$
$F_{-n} = (-1)^n F_n$
$F_{n+k} = F_k \cdot F_{n+1} + F_{k-1} \cdot F_n$
$\gcd(F_m, F_n) = F_{\gcd(m,n)}$
$F_m \mid F_n^2 \Leftrightarrow nF_n \mid m$
$F_n = \frac{\varphi^n - \Psi^n}{\sqrt{5}}, \varphi = \frac{1+\sqrt{5}}{2}, \Psi = \frac{1-\sqrt{5}}{2}$
$F_n = \lfloor \frac{\varphi^n}{\sqrt{5}} + \frac{1}{2} \rfloor, n \geq 0$
$n(F) = \lfloor \log_\varphi (F \cdot \sqrt{5} + \frac{1}{2}) \rfloor$

Stirling number (1st kind)

用 $\begin{bmatrix} n \\ k \end{bmatrix}$ 表示 Stirling number (1st kind)，为将 n 个元素分成 k 个环的方案数
$\begin{bmatrix} n+1 \\ k \end{bmatrix} = n\begin{bmatrix} n \\ k \end{bmatrix} + \begin{bmatrix} n \\ k-1 \end{bmatrix}, k > 0$
$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = 1, \begin{bmatrix} n \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ n \end{bmatrix} = 0, n > 0$
$\begin{bmatrix} n \\ k \end{bmatrix}$ 为将 n 个元素分成 k 个环的方案数
$\begin{bmatrix} x \\ x-n \end{bmatrix} = \sum\limits_{k=0}^{n} \left\langle\!\!\left\langle {n \atop k} \right\rangle\!\!\right\rangle \binom{x+k}{2n}$

Stirling number (2nd kind)

用 $\begin{Bmatrix} n \\ k \end{Bmatrix}$ 表示 Stirling number (2nd kind)，为将 n 个元素划分成 k 个非空集合的方案数
$\begin{Bmatrix} n+1 \\ k \end{Bmatrix} = k\begin{Bmatrix} n \\ k \end{Bmatrix} + \begin{Bmatrix} n \\ k-1 \end{Bmatrix}, k > 0$
$\begin{Bmatrix} 0 \\ 0 \end{Bmatrix} = 1, \begin{Bmatrix} n \\ 0 \end{Bmatrix} = \begin{Bmatrix} 0 \\ n \end{Bmatrix} = 0, n > 0$
$\begin{Bmatrix} n \\ k \end{Bmatrix} = \frac{1}{k!}\sum\limits_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n$
$\begin{Bmatrix} n \\ k \end{Bmatrix}$
$\begin{Bmatrix} x \\ x-n \end{Bmatrix} = \sum\limits_{k=0}^{n} \left\langle\!\!\left\langle {n \atop k} \right\rangle\!\!\right\rangle \binom{x+n-k-1}{2n}$

Catalan number

c_n 表示长度为 $2n$ 的合法括号序的数量
$c_1 = 1, c_{n+1} = \sum\limits_{i=1}^{n} c_i \times c_{n+1-i}$
$c_n = \frac{\binom{2n}{n}}{n+1}$

Bell number

B_n 表示基数为 n 的集合的划分方案数
$$B_i = \begin{cases} 1 & i = 0 \\ \sum\limits_{k=0}^{n} \binom{n}{k} B_k & i > 0 \end{cases}$$
$B_n = \sum\limits_{k=0}^{n} \begin{Bmatrix} n \\ k \end{Bmatrix}$
$B_{p^m+n} \equiv mB_n + B_{n+1} \pmod p$

五边形数定理

$p(n)$ 表示将 n 划分为若干个正整数之和的方案数
$p(n) = \sum\limits_{k \in \mathbb{N}^*} (-1)^{k-1} p(n - \frac{k(3k-1)}{2})$

Bernoulli number

$\sum\limits_{j=0}^{m}\binom{m+1}{j}B_j=0, m>0$

$B_i=\begin{cases}1 & i=0\\ -\frac{\sum\limits_{j=0}^{i-1}\binom{i+1}{j}B_j}{i+1} & i>0\end{cases}$

$\sum\limits_{k=1}^{n}k^m=\frac{1}{m+1}\sum\limits_{k=0}^{m}\binom{m+1}{k}B_k n^{m+1-k}$

Stirling permutation

$1,1,2,2\ldots,n,n$ 的排列中，对于每个 i，都有两个 i 之间的数大于 i

排列方案数为 $(2n-1)!!$

Eulerian number

$\left\langle\begin{matrix}n\\k\end{matrix}\right\rangle$ 表示 1 到 n 的排列中，恰有 k 个数比前一个大的方案数

$\left\langle\begin{matrix}n\\0\end{matrix}\right\rangle=\left\langle\begin{matrix}n\\n-1\end{matrix}\right\rangle=1$

$\left\langle\begin{matrix}0\\m\end{matrix}\right\rangle=[m=0]$

$\left\langle\begin{matrix}n\\m\end{matrix}\right\rangle=\left\langle\begin{matrix}n\\n-1-m\end{matrix}\right\rangle$

$\left\langle\begin{matrix}n\\m\end{matrix}\right\rangle=(m+1)\left\langle\begin{matrix}n-1\\m\end{matrix}\right\rangle+(n-m)\left\langle\begin{matrix}n-1\\m-1\end{matrix}\right\rangle$

$\left\langle\begin{matrix}n\\m\end{matrix}\right\rangle=\sum\limits_{k=0}^{m}(-1)^k\binom{n+1}{k}(m+1-k)^n$

Eulerian number (2nd kind)

$\left\langle\!\!\left\langle\begin{matrix}n\\k\end{matrix}\right\rangle\!\!\right\rangle$ 表示 Stirling permutation 中，恰有 k 个数比前一个大的方案数

$\left\langle\!\!\left\langle\begin{matrix}n\\m\end{matrix}\right\rangle\!\!\right\rangle=(2n-m-1)\left\langle\!\!\left\langle\begin{matrix}n-1\\m-1\end{matrix}\right\rangle\!\!\right\rangle+(m+1)\left\langle\!\!\left\langle\begin{matrix}n-1\\m\end{matrix}\right\rangle\!\!\right\rangle$

$\left\langle\!\!\left\langle\begin{matrix}n\\0\end{matrix}\right\rangle\!\!\right\rangle=1$

$\left\langle\!\!\left\langle\begin{matrix}0\\m\end{matrix}\right\rangle\!\!\right\rangle=[m=0]$

Burnside lemma

Let G be a finite group that acts on a set X. For each g in G let X^g denote the set of elements in X that are fixed by g (also said to be left invariant by g), i.e. $X^g=\{x\in X\mid g.x=x\}$. Burnside's lemma asserts the following formula for the number of orbits, denoted $|X/G|$:

$|X/G|=\frac{1}{|G|}\sum\limits_{g\in G}|X^g|$

Example application: The number of rotationally distinct colorings of the faces of a cube using n colors

Let X be the set of n^6 possible face colour combinations that can be applied to a cube in one particular orientation, and let the rotation group G of the cube act on X in the natural manner. Then two elements of X belong to the same orbit precisely when one is simply a rotation of the other. The number of rotationally distinct colourings is thus the same as the number of orbits and can be found by counting the sizes of the fixed sets for the 24 elements of G.

- one identity element which leaves all n^6 elements of X unchanged
- six 90-degree face rotations, each of which leaves n^3 of the elements of X unchanged
- three 180-degree face rotations, each of which leaves n^4 of the elements of X unchanged
- eight 120-degree vertex rotations, each of which leaves n^2 of the elements of X unchanged
- six 180-degree edge rotations, each of which leaves n^3 of the elements of X unchanged

The average fix size is thus $\frac{1}{24}(n^6+6\cdot n^3+3\cdot n^4+8\cdot n^2+6\cdot n^3)$

Hence there are 57 rotationally distinct colorings of the faces of a cube in 3 colours.

Pólya theorem

设 $\overline{G}$ 是 n 个对象的置换群，用 m 种颜色对 n 个对象染色，则不同染色方案为：

$L=\frac{1}{|\overline{G}|}(m^{c(\overline{P_1})}+m^{c(\overline{P_2})}+\cdots+m^{c(\overline{P_g})})$

其中 $\overline{G}=\{\overline{P_1},\overline{P_2},\ldots,\overline{P_g}\}$，$c(\overline{P_k})$ 为 $\overline{P_k}$ 的循环节数

Möbius function

$\mu(n)=\begin{cases}1 & n\text{ square-free, even number of prime factors}\\ -1 & n\text{ square-free, odd number of prime factors}\\ 0 & n\text{ has a squared prime factor}\end{cases}$

$\sum\limits_{d|n}\mu(d)=\begin{cases}1 & n=1\\ 0 & n>1\end{cases}$

$g(n)=\sum\limits_{d|n}f(d)\Leftrightarrow f(n)=\sum\limits_{d|n}\mu(d)g(\frac{n}{d})$

Lagrange polynomial

给定次数为 n 的多项式函数 $L(x)$ 上的 $n+1$ 个点 $(x_0,y_0),(x_1,y_1),\ldots,(x_n,y_n)$

则 $L(x)=\sum\limits_{j=0}^{n}y_j\prod\limits_{0\le m\le n,m\ne j}\frac{x-x_m}{x_j-x_m}$

3. Geometry

3.1 点、直线、圆 (gy)

```cpp
using number = long double;
const number eps = 1e-8;

number _sqrt(number x) {
    return std::sqrt(std::max(x, (number) 0));
}
number _asin(number x) {
    x = std::min(x, (number) 1), x = std::max(x, (number) -1);
    return std::asin(x);
}
```

```cpp
number _acos(number x) {
    x = std::min(x, (number) 1), x = std::max(x, (number) -1);
    return std::acos(x);
}

int sgn(number x) {
    return (x > eps) - (x < -eps);
}
int cmp(number x, number y) {
    return sgn(x - y);
}

struct point {
    number x, y;
    point() {}
    point(number x, number y) : x(x), y(y) {}

    number len2() const {
        return x * x + y * y;
    }
    number len() const {
        return _sqrt(len2());
    }
    point unit() const {
        return point(x / len(), y / len());
    }
    point rotate90() const {
        return point(-y, x);
    }

    friend point operator+(const point &a, const point &b) {
        return point(a.x + b.x, a.y + b.y);
    }
    friend point operator-(const point &a, const point &b) {
        return point(a.x - b.x, a.y - b.y);
    }
    friend point operator*(const point &a, number b) {
        return point(a.x * b, a.y * b);
    }
    friend point operator/(const point &a, number b) {
        return point(a.x / b, a.y / b);
    }
    friend number dot(const point &a, const point &b) {
        return a.x * b.x + a.y * b.y;
    }
    friend number det(const point &a, const point &b) {
        return a.x * b.y - a.y * b.x;
    }
    friend number operator==(const point &a, const point &b) {
        return cmp(a.x, b.x) == 0 && cmp(a.y, b.y) == 0;
    }
};

number dis2(const point &a, const point &b) {
    return (a - b).len2();
}
number dis(const point &a, const point &b) {
    return (a - b).len();
}

struct line {
    point a, b;
    line() {}
    line(point a, point b) : a(a), b(b) {}
    point value() const {
        return b - a;
    }
};

bool point_on_line(const point &p, const line &l) {
    return sgn(det(p - l.a, p - l.b)) == 0;
}
// including endpoint
bool point_on_ray(const point &p, const line &l) {
    return sgn(det(p - l.a, p - l.b)) == 0 &&
        sgn(dot(p - l.a, l.b - l.a)) >= 0;
}
// including endpoints
bool point_on_seg(const point &p, const line &l) {
    return sgn(det(p - l.a, p - l.b)) == 0 &&
        sgn(dot(p - l.a, l.b - l.a)) >= 0 &&
        sgn(dot(p - l.b, l.a - l.b)) >= 0;
}
bool seg_has_intersection(const line &a, const line &b) {
    if (point_on_seg(a.a, b) || point_on_seg(a.b, b) ||
            point_on_seg(b.a, a) || point_on_seg(b.b, a))
        return /* including endpoints */ true;
    return sgn(det(a.a - b.a, b.b - b.a)) * sgn(det(a.b - b.a, b.b - b.a)) < 0
        && sgn(det(b.a - a.a, a.b - a.a)) * sgn(det(b.b - a.a, a.b - a.a)) < 0;
}
point intersect(const line &a, const line &b) {
    number s1 = det(a.b - a.a, b.a - a.a);
    number s2 = det(a.b - a.a, b.b - a.a);
    return (b.a * s2 - b.b * s1) / (s2 - s1);
}
point projection(const point &p, const line &l) {
    return l.a + (l.b - l.a) * dot(p - l.a, l.b - l.a) / (l.b - l.a).len2();
}
number dis(const point &p, const line &l) {
    return std::abs(det(p - l.a, l.b - l.a)) / (l.b - l.a).len();
}
point symmetry_point(const point &a, const point &o) {
    return o + o - a;
}
point reflection(const point &p, const line &l) {
    return symmetry_point(p, projection(p, l));
}

struct circle {
    point o;
    number r;
    circle() {}
    circle(point o, number r) : o(o), r(r) {}
};

bool intersect(const line &l, const circle &a, point &p1, point &p2) {
    number x = dot(l.a - a.o, l.b - l.a);
    number y = (l.b - l.a).len2();
    number d = x * x - y * ((l.a - a.o).len2() - a.r * a.r);
    if (sgn(d) < 0) return false;
    point p = l.a - (l.b - l.a) * (x / y), delta = (l.b - l.a) * (_sqrt(d) / y);
    p1 = p + delta, p2 = p - delta;
```

```cpp
    return true;
}
bool intersect(const circle &a, const circle &b, point &p1, point &p2) {
    if (a.o == b.o && cmp(a.r, b.r) == 0)
        return /* value for coincident circles */ false;
    number s1 = (b.o - a.o).len();
    if (cmp(s1, a.r + b.r) > 0 || cmp(s1, std::abs(a.r - b.r)) < 0)
        return false;
    number s2 = (a.r * a.r - b.r * b.r) / s1;
    number aa = (s1 + s2) / 2, bb = (s1 - s2) / 2;
    point p = (b.o - a.o) * (aa / (aa + bb)) + a.o;
    point delta = (b.o - a.o).unit().rotate90() * _sqrt(a.r * a.r - aa * aa);
    p1 = p + delta, p2 = p - delta;
    return true;
}
bool tangent(const point &p0, const circle &c, point &p1, point &p2) {
    number x = (p0 - c.o).len2();
    number d = x - c.r * c.r;
    if (sgn(d) < 0) return false;
    if (sgn(d) == 0)
        return /* value for point_on_line */ false;
    point p = (p0 - c.o) * (c.r * c.r / x);
    point delta = ((p0 - c.o) * (-c.r * _sqrt(d) / x)).rotate90();
    p1 = c.o + p + delta;
    p2 = c.o + p - delta;
    return true;
}
bool ex_tangent(const circle &a, const circle &b, line &l1, line &l2) {
    if (cmp(std::abs(a.r - b.r), (b.o - a.o).len()) == 0) {
        point p1, p2;
        intersect(a, b, p1, p2);
        l1 = l2 = line(p1, p1 + (a.o - p1).rotate90());
        return true;
    } else if (cmp(a.r, b.r) == 0) {
        point dir = b.o - a.o;
        dir = (dir * (a.r / dir.len())).rotate90();
        l1 = line(a.o + dir, b.o + dir);
        l2 = line(a.o - dir, b.o - dir);
        return true;
    } else {
        point p = (b.o * a.r - a.o * b.r) / (a.r - b.r);
        point p1, p2, q1, q2;
        if (tangent(p, a, p1, p2) && tangent(p, b, q1, q2)) {
            l1 = line(p1, q1);
            l2 = line(p2, q2);
            return true;
        } else {
            return false;
        }
    }
}
bool in_tangent(const circle &a, const circle &b, line &l1, line &l2) {
    if (cmp(a.r + b.r, (b.o - a.o).len()) == 0) {
        point p1, p2;
        intersect(a, b, p1, p2);
        l1 = l2 = line(p1, p1 + (a.o - p1).rotate90());
        return true;
    } else {
        point p = (b.o * a.r + a.o * b.r) / (a.r + b.r);
        point p1, p2, q1, q2;
        if (tangent(p, a, p1, p2) && tangent(p, b, q1, q2)) {
            l1 = line(p1, q1);
            l2 = line(p2, q2);
            return true;
        } else {
            return false;
        }
    }
}
```

3.2 平面最近点对 (Grimoire)

```cpp
bool byY(P a,P b){return a.y<b.y;}
LL solve(P *p,int l,int r){
    LL d=1LL<<62;
    if(l==r)
        return d;
    if(l+1==r)
        return dis2(p[l],p[r]);
    int mid=(l+r)>>1;
    d=min(solve(l,mid),d);
    d=min(solve(mid+1,r),d);
    vector<P>tmp;
    for(int i=l;i<=r;i++)
        if(sqr(p[mid].x-p[i].x)<=d)
            tmp.push_back(p[i]);
    sort(tmp.begin(),tmp.end(),byY);
    for(int i=0;i<tmp.size();i++)
        for(int j=i+1;j<tmp.size()&&j-i<10;j++)
            d=min(d,dis2(tmp[i],tmp[j]));
    return d;
}
```

3.3 凸包游戏 (Grimoire)

给定凸包，$O(n \log n)$ 完成询问：

- 点在凸包内
- 凸包外的点到凸包的两个切点
- 向量关于凸包的切点
- 直线与凸包的交点

传入凸包要求 1 号点为 $Pair(x, y)$ 最小的

```cpp
const int INF = 1000000000;
struct Convex
{
    int n;
    vector<Point> a, upper, lower;
    Convex(vector<Point> _a) : a(_a) {
        n = a.size();
        int ptr = 0;
        for(int i = 1; i < n; ++ i) if (a[ptr] < a[i]) ptr = i;
        for(int i = 0; i <= ptr; ++ i) lower.push_back(a[i]);
        for(int i = ptr; i < n; ++ i) upper.push_back(a[i]);
        upper.push_back(a[0]);
```

```cpp
    }
    int sign(long long x) { return x < 0 ? -1 : x > 0; }
    pair<long long, int> get_tangent(vector<Point> &convex, Point vec) {
        int l = 0, r = (int)convex.size() - 2;
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            if (sign((convex[mid + 1] - convex[mid]).det(vec)) > 0) r = mid;
            else l = mid;
        }
        return max(make_pair(vec.det(convex[r]), r)
            , make_pair(vec.det(convex[0]), 0));
    }
    void update_tangent(const Point &p, int id, int &i0, int &i1) {
        if ((a[i0] - p).det(a[id] - p) > 0) i0 = id;
        if ((a[i1] - p).det(a[id] - p) < 0) i1 = id;
    }
    void binary_search(int l, int r, Point p, int &i0, int &i1) {
        if (l == r) return;
        update_tangent(p, l % n, i0, i1);
        int sl = sign((a[l % n] - p).det(a[(l + 1) % n] - p));
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            int smid = sign((a[mid % n] - p).det(a[(mid + 1) % n] - p));
            if (smid == sl) l = mid;
            else r = mid;
        }
        update_tangent(p, r % n, i0, i1);
    }
    int binary_search(Point u, Point v, int l, int r) {
        int sl = sign((v - u).det(a[l % n] - u));
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            int smid = sign((v - u).det(a[mid % n] - u));
            if (smid == sl) l = mid;
            else r = mid;
        }
        return l % n;
    }
    // 判定点是否在凸包内，在边界返回 true
    bool contain(Point p) {
        if (p.x < lower[0].x || p.x > lower.back().x) return false;
        int id = lower_bound(lower.begin(), lower.end()
            , Point(p.x, -INF)) - lower.begin();
        if (lower[id].x == p.x) {
            if (lower[id].y > p.y) return false;
        } else if ((lower[id - 1] - p).det(lower[id] - p) < 0) return false;
        id = lower_bound(upper.begin(), upper.end(), Point(p.x, INF)
            , greater<Point>()) - upper.begin();
        if (upper[id].x == p.x) {
            if (upper[id].y < p.y) return false;
        } else if ((upper[id - 1] - p).det(upper[id] - p) < 0) return false;
        return true;
    }
    // 求点 p 关于凸包的两个切点，如果在凸包外则有序返回编号
    // 共线的多个切点返回任意一个，否则返回 false
    bool get_tangent(Point p, int &i0, int &i1) {
        if (contain(p)) return false;
        i0 = i1 = 0;
        int id = lower_bound(lower.begin(), lower.end(), p) - lower.begin();
        binary_search(0, id, p, i0, i1);
        binary_search(id, (int)lower.size(), p, i0, i1);
        id = lower_bound(upper.begin(), upper.end(), p
            , greater<Point>()) - upper.begin();
        binary_search((int)lower.size() - 1, (int)lower.size() - 1 + id, p, i0, i1);
        binary_search((int)lower.size() - 1 + id
            , (int)lower.size() - 1 + (int)upper.size(), p, i0, i1);
        return true;
    }
    // 求凸包上和向量 vec 叉积最大的点，返回编号，共线的多个切点返回任意一个
    int get_tangent(Point vec) {
        pair<long long, int> ret = get_tangent(upper, vec);
        ret.second = (ret.second + (int)lower.size() - 1) % n;
        ret = max(ret, get_tangent(lower, vec));
        return ret.second;
    }
    // 求凸包和直线 u,v 的交点，如果无严格相交返回 false.
    //如果有则是和 (i,next(i)) 的交点，两个点无序，交在点上不确定返回前后两条线段其中之一
    bool get_intersection(Point u, Point v, int &i0, int &i1) {
        int p0 = get_tangent(u - v), p1 = get_tangent(v - u);
        if (sign((v - u).det(a[p0] - u)) * sign((v - u).det(a[p1] - u)) < 0) {
            if (p0 > p1) swap(p0, p1);
            i0 = binary_search(u, v, p0, p1);
            i1 = binary_search(u, v, p1, p0 + n);
            return true;
        } else {
            return false;
        }
    }
};
```

3.4 半平面交 (Grimoire)

```cpp
struct P{
    int quad() const { return sgn(y) == 1 || (sgn(y) == 0 && sgn(x) >= 0);}
};
struct L{
    bool onLeft(const P &p) const { return sgn((b - a)*( p - a)) > 0; }
    L push() const{ // push out eps
        const double eps = 1e-10;
        P delta = (b - a).turn90().norm() * eps;
        return L(a - delta, b - delta);
    }
};
bool sameDir(const L &l0, const L &l1) {
    return parallel(l0, l1) && sgn((l0.b - l0.a)^(l1.b - l1.a)) == 1;
}
bool operator < (const P &a, const P &b) {
```

```cpp
    if (a.quad() != b.quad())
        return a.quad() < b.quad();
    else
        return sgn((a*b)) > 0;
}
bool operator < (const L &l0, const L &l1) {
    if (sameDir(l0, l1))
        return l1.onLeft(l0.a);
    else
        return (l0.b - l0.a) < (l1.b - l1.a);
}
bool check(const L &u, const L &v, const L &w) {
    return w.onLeft(intersect(u, v));
}
vector<P> intersection(vector<L> &l) {
    sort(l.begin(), l.end());
    deque<L> q;
    for (int i = 0; i < (int)l.size(); ++i) {
        if (i && sameDir(l[i], l[i - 1])) {
            continue;
        }
        while (q.size() > 1
            && !check(q[q.size() - 2], q[q.size() - 1], l[i]))
                q.pop_back();
        while (q.size() > 1
            && !check(q[1], q[0], l[i]))
                q.pop_front();
        q.push_back(l[i]);
    }
    while (q.size() > 2
        && !check(q[q.size() - 2], q[q.size() - 1], q[0]))
            q.pop_back();
    while (q.size() > 2
        && !check(q[1], q[0], q[q.size() - 1]))
            q.pop_front();
    vector<P> ret;
    for (int i = 0; i < (int)q.size(); ++i)
    ret.push_back(intersect(q[i], q[(i + 1) % q.size()]));
    return ret;
}
```

3.5 点在多边形内 (Grimoire)

```cpp
bool inPoly(P p,vector<P>poly){
    int cnt=0;
    for(int i=0;i<poly.size();i++){
        P a=poly[i],b=poly[(i+1)%poly.size()];
        if(onSeg(p,L(a,b)))
            return false;
        int x=sgn(det(a,p,b));
        int y=sgn(a.y-p.y);
        int z=sgn(b.y-p.y);
        cnt+=(x>0&&y<=0&&z>0);
        cnt-=(x<0&&z<=0&&y>0);
    }
    return cnt;
}
```

3.6 最小圆覆盖 (Grimoire)

```cpp
struct line{
    point p,v;
};
point Rev(point v){return point(-v.y,v.x);}
point operator*(line A,line B){
    point u=B.p-A.p;
    double t=(B.v*u)/(B.v*A.v);
    return A.p+A.v*t;
}
point get(point a,point b){
    return (a+b)/2;
}
point get(point a,point b,point c){
    if(a==b)return get(a,c);
    if(a==c)return get(a,b);
    if(b==c)return get(a,b);
    line ABO=(line){(a+b)/2,Rev(a-b)};
    line BCO=(line){(c+b)/2,Rev(b-c)};
    return ABO*BCO;
}
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++)scanf("%lf%lf",&p[i].x,&p[i].y);
    random_shuffle(p+1,p+1+n);
    O=p[1];r=0;
    for(int i=2;i<=n;i++){
        if(dis(p[i],O)<r+1e-6)continue;
        O=get(p[1],p[i]);r=dis(O,p[i]);
        for(int j=1;j<i;j++){
            if(dis(p[j],O)<r+1e-6)continue;
            O=get(p[i],p[j]);r=dis(O,p[i]);
            for(int k=1;k<j;k++){
                if(dis(p[k],O)<r+1e-6)continue;
                O=get(p[i],p[j],p[k]);r=dis(O,p[i]);
            }
        }
    }printf("%.2lf %.2lf %.2lf\n",O.x,O.y,r);
    return 0;
}
```

3.7 最小球覆盖 (Grimoire)

```cpp
bool equal(const double & x, const double & y) {
    return x + eps > y and y + eps > x;
}
double operator % (const Point & a, const Point & b) {
    return a.x * b.x + a.y * b.y + a.z * b.z;
}
Point operator * (const Point & a, const Point & b) {
    return Point(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
}
struct Circle {
    double r; Point o;
};
struct Plane {
    Point nor;
    double m;
    Plane(const Point & nor, const Point & a) : nor(nor){
        m = nor % a;
    }
};
Point intersect(const Plane & a, const Plane & b, const Plane & c) {
    Point c1(a.nor.x, b.nor.x, c.nor.x), c2(a.nor.y, b.nor.y, c.nor.y), c3(a.nor.z, b.nor.z, c.nor.z), c4(a.m, b.m, c.m);
    return 1 / ((c1 * c2) % c3) * Point((c4 * c2) % c3, (c1 * c4) % c3, (c1 * c2) % c4);
}
```

```cpp
bool in(const Point & a, const Circle & b) {
    return sign((a - b.o).len() - b.r) <= 0;
}
bool operator < (const Point & a, const Point & b) {
    if(!equal(a.x, b.x)) {
        return a.x < b.x;
    }
    if(!equal(a.y, b.y)) {
        return a.y < b.y;
    }
    if(!equal(a.z, b.z)) {
        return a.z < b.z;
    }
    return false;
}
bool operator == (const Point & a, const Point & b) {
    return equal(a.x, b.x) and equal(a.y, b.y) and
     equal(a.z, b.z);
}
vector<Point> vec;
Circle calc() {
    if(vec.empty()) {
        return Circle(Point(0, 0, 0), 0);
    }else if(1 == (int)vec.size()) {
        return Circle(vec[0], 0);
    }else if(2 == (int)vec.size()) {
        return Circle(0.5 * (vec[0] + vec[1]), 0.5 *
         (vec[0] - vec[1]).len());
    }else if(3 == (int)vec.size()) {
        double r((vec[0] - vec[1]).len() * (vec[1] -
         vec[2]).len() * (vec[2] - vec[0]).len() / 2
         / fabs(((vec[0] - vec[2]) * (vec[1] -
         vec[2])).len()));
        return Circle(intersect(Plane(vec[1] - vec[0],
         0.5 * (vec[1] + vec[0])),
                                Plane(vec[2] - vec[1], 0.5 *
                                 (vec[2] + vec[1])),
                            Plane((vec[1] - vec[0]) * (vec[2] -
                             vec[0]), vec[0])), r);
    }else {
        Point o(intersect(Plane(vec[1] - vec[0], 0.5 *
         (vec[1] + vec[0])),
                    Plane(vec[2] - vec[0], 0.5 * (vec[2]
                     + vec[0])),
                    Plane(vec[3] - vec[0], 0.5 * (vec[3]
                     + vec[0]))));
        return Circle(o, (o - vec[0]).len());
    }
}
Circle miniBall(int n) {
    Circle res(calc());
    for(int i(0); i < n; i++) {
        if(!in(a[i], res)) {
            vec.push_back(a[i]);
            res = miniBall(i);
            vec.pop_back();
            if(i) {
                Point tmp(a[i]);
                memmove(a + 1, a, sizeof(Point) * i);
                a[0] = tmp;
            }
        }
    }
    return res;
}
int main() {
    int n;
    sort(a, a + n);
    n = unique(a, a + n) - a;
    vec.clear();
    printf("%.10f\n", miniBall(n).r);
}
```

3.8 圆并 (Grimoire)

```cpp
double ans[2001];
struct Point {
    double x, y;
    Point(){}
    Point(const double & x, const double & y) : x(x),
     y(y) {}
    void scan() {scanf("%lf%lf", &x, &y);}
    double sqrlen() {return sqr(x) + sqr(y);}
    double len() {return sqrt(sqrlen());}
    Point rev() {return Point(y, -x);}
    void print() {printf("%f %f\n", x, y);}
    Point zoom(const double & d) {double lambda = d /
     len(); return Point(lambda * x, lambda * y);}
} dvd, a[2001];
Point centre[2001];
double atan2(const Point & x) {
    return atan2(x.y, x.x);
}
Point operator - (const Point & a, const Point & b) {
    return Point(a.x - b.x, a.y - b.y);
}
Point operator + (const Point & a, const Point & b) {
    return Point(a.x + b.x, a.y + b.y);
}
double operator * (const Point & a, const Point & b) {
    return a.x * b.y - a.y * b.x;
}
Point operator * (const double & a, const Point & b) {
    return Point(a * b.x, a * b.y);
}
double operator % (const Point & a, const Point & b) {
    return a.x * b.x + a.y * b.y;
}
struct circle {
    double r; Point o;
    circle() {}
    void scan() {
        o.scan();
        scanf("%lf", &r);
    }
} cir[2001];
struct arc {
    double theta;
    int delta;
    Point p;
    arc() {};
    arc(const double & theta, const Point & p, int d) :
     theta(theta), p(p), delta(d) {}
} vec[4444];
int nV;
inline bool operator < (const arc & a, const arc & b) {
    return a.theta + eps < b.theta;
}
int cnt;
inline void psh(const double t1, const Point p1, const
 double t2, const Point p2) {
    if(t2 + eps < t1)
        cnt++;
    vec[nV++] = arc(t1, p1, 1);
    vec[nV++] = arc(t2, p2, -1);
}
```

```cpp
inline double cub(const double & x) {
    return x * x * x;
}
inline void combine(int d, const double & area, const Point & o) {
    if(sign(area) == 0) return;
    centre[d] = 1 / (ans[d] + area) * (ans[d] * centre[d] + area * o);
    ans[d] += area;
}
bool equal(const double & x, const double & y) {
    return x + eps>  y and y + eps > x;
}
bool equal(const Point & a, const Point & b) {
    return equal(a.x, b.x) and equal(a.y, b.y);
}
bool equal(const circle & a, const circle & b) {
    return equal(a.o, b.o) and equal(a.r, b.r);
}
bool f[2001];
int main() {
    int n, m, index;
    while(EOF != scanf("%d%d%d", &m, &n, &index)) {
        index--;
        for(int i(0); i < m; i++) {
            a[i].scan();
        }
        for(int i(0); i < n; i++) {
            cir[i].scan();//n 个圆
        }
        for(int i(0); i < n; i++) {//这一段在去重圆 能加速 删掉不会错
            f[i] = true;
            for(int j(0); j < n; j++) if(i != j) {
                if(equal(cir[i], cir[j]) and i < j or !equal(cir[i], cir[j]) and cir[i].r < cir[j].r + eps and (cir[i].o - cir[j].o).sqrlen() < sqr(cir[i].r - cir[j].r) + eps) {
                    f[i] = false;
                    break;
                }
            }
        }
        int n1(0);
        for(int i(0); i < n; i++)
            if(f[i])
                cir[n1++] = cir[i];
        n = n1;//去重圆结束
        fill(ans, ans + n + 1, 0);//ans[i] 表示被圆覆盖至少 i 次的面积
        fill(centre, centre + n + 1, Point(0, 0));//centre[i] 表示上面 ans[i] 部分的重心
        for(int i(0); i < m; i++)
            combine(0, a[i] * a[(i + 1) % m] * 0.5, 1. / 3 * (a[i] + a[(i + 1) % m]));
        for(int i(0); i < n; i++) {
            dvd = cir[i].o - Point(cir[i].r, 0);
            nV = 0;
            vec[nV++] = arc(-pi, dvd, 1);
            cnt = 0;
            for(int j(0); j < n; j++) if(j != i) {
                double d = (cir[j].o - cir[i].o).sqrlen();
                if(d < sqr(cir[j].r - cir[i].r) + eps) {
                    if(cir[i].r + i * eps < cir[j].r + j * eps)
                        psh(-pi, dvd, pi, dvd);
                }else if(d + eps < sqr(cir[j].r + cir[i].r)) {
                    double lambda = 0.5 * (1 + (sqr(cir[i].r) - sqr(cir[j].r)) / d);
                    Point cp(cir[i].o + lambda * (cir[j].o - cir[i].o));
                    Point nor((cir[j].o - cir[i].o).rev().zoom(sqrt(sqr(cir[i].r) - (cp - cir[i].o).sqrlen())));
                    Point frm(cp + nor);
                    Point to(cp - nor);
                    psh(atan2(frm - cir[i].o), frm, atan2(to - cir[i].o), to);
                }
            }
            sort(vec + 1, vec + nV);
            vec[nV++] = arc(pi, dvd, -1);
            for(int j = 0; j + 1 < nV; j++) {
                cnt += vec[j].delta;
                //if(cnt == 1) {//如果只算 ans[1] 和 centre[1], 可以加这个 if 加速.
                    double theta(vec[j + 1].theta - vec[j].theta);
                    double area(sqr(cir[i].r) * theta * 0.5);
                    combine(cnt, area, cir[i].o + 1. / area / 3 * cub(cir[i].r) * Point(sin(vec[j + 1].theta) - sin(vec[j].theta), cos(vec[j].theta) - cos(vec[j + 1].theta)));
                    combine(cnt, -sqr(cir[i].r) * sin(theta) * 0.5, 1. / 3 * (cir[i].o + vec[j].p + vec[j + 1].p));
                    combine(cnt, vec[j].p * vec[j + 1].p * 0.5, 1. / 3 * (vec[j].p + vec[j + 1].p));
                //}
            }
        }
        combine(0, -ans[1], centre[1]);
        for(int i = 0; i < m; i++) {
            if(i != index)
                (a[index] - Point((a[i] - a[index]) * (centre[0] - a[index]), (a[i] - a[index]) % (centre[0] - a[index])).zoom((a[i] - a[index]).len())).print();
            else
                a[i].print();
        }
    }
    return 0;
}
```

3.9 圆与多边形并 (Grimoire)

```cpp
double form(double x){
    while(x>=2*pi)x-=2*pi;
    while(x<0)x+=2*pi;
    return x;
}
double calcCir(C cir){
    vector<double>ang;
```

```
    ang.push_back(0);
    ang.push_back(pi);
    double ans=0;
    for(int i=1;i<=n;i++){
        if(cir==c[i])continue;
        P p1,p2;
        if(intersect(cir,c[i],p1,p2)){
            ang.push_back(form(cir.ang(p1)));
            ang.push_back(form(cir.ang(p2)));
        }
    }

    for(int i=1;i<=m;i++){
        vector<P>tmp;
        tmp=intersect(poly[i],cir);
        for(int j=0;j<tmp.size();j++){
            ang.push_back(form(cir.ang(tmp[j])));
        }
    }
    sort(ang.begin(),ang.end());
    for(int i=0;i<ang.size();i++){
        double t1=ang[i],t2=(i+1==ang.size()
            ?ang[0]+2*pi:ang[i+1]);
        P p=cir.at((t1+t2)/2);
        int ok=1;
        for(int j=1;j<=n;j++){
            if(cir==c[j])continue;
            if(inC(p,c[j],true)){
                ok=0;
                break;
            }
        }
        for(int j=1;j<=m&&ok;j++){
            if(inPoly(p,poly[j],true)){
                ok=0;
                break;
            }
        }
        if(ok){
            double r=cir.r,x0=cir.o.x,y0=cir.o.y;
            ans+=(r*r*(t2-t1)+r*x0*(sin(t2)-sin(t1))
                -r*y0*(cos(t2)-cos(t1)))/2;

        }
    }
    return ans;
}
P st;
bool bySt(P a,P b){
    return dis(a,st)<dis(b,st);
}
double calcSeg(L l){
    double ans=0;
    vector<P>pt;
    pt.push_back(l.a);
    pt.push_back(l.b);
    for(int i=1;i<=n;i++){
        P p1,p2;
        if(intersect(c[i],l,p1,p2)){
            if(onSeg(p1,l))
                pt.push_back(p1);
            if(onSeg(p2,l))
                pt.push_back(p2);
        }
    }
    st=l.a;
    sort(pt.begin(),pt.end(),bySt);
    for(int i=0;i+1<pt.size();i++){
        P p1=pt[i],p2=pt[i+1];
        P p=(p1+p2)/2;
        int ok=1;
        for(int j=1;j<=n;j++){
            if(sgn(dis(p,c[j].o),c[j].r)<0){
                ok=0;
                break;
            }
        }
        if(ok){
            double x1=p1.x,y1=p1.y,x2=p2.x,y2=p2.y;
            double res=(x1*y2-x2*y1)/2;
            ans+=res;
        }
    }
    return ans;
}
```

3.10 三角剖分 (Grimoire)

Triangulation::find 返回包含某点的三角形
Triangulation::add_point 将某点加入三角剖分
某个 *Triangle* 在三角剖分中当且仅当它的 *has_children* 为 0
如果要找到三角形 u 的邻域，则枚举它的所有 u.edge[i].tri，该条边的两个点为 u.p[(i + 1) % 3], u.p[(i + 2) % 3]
通过三角剖分构造 V 图：连接相邻三角形外接圆圆心
注意初始化内存池和 *Triangulation* :: *LOTS*
复杂度 $O(n \log n)$

```
const int N = 100000 + 5, MAX_TRIS = N * 6;
const double eps = 1e-6, PI = acos(-1.0);
struct P {
    double x,y; P():x(0),y(0){}
    P(double x, double y):x(x),y(y){}
    bool operator ==(P const& that)const {return
        x==that.x&&y==that.y;}
};
inline double sqr(double x) { return x*x; }
double dist_sqr(P const& a, P const& b){return
    sqr(a.x-b.x)+sqr(a.y-b.y);}
bool in_circumcircle(P const& p1, P const& p2, P
    const& p3, P const& p4) {//p4 in C(p1,p2,p3)
    double u11 = p1.x - p4.x, u21 = p2.x - p4.x, u31 =
        p3.x - p4.x;
    double u12 = p1.y - p4.y, u22 = p2.y - p4.y, u32 =
        p3.y - p4.y;
    double u13 = sqr(p1.x) - sqr(p4.x) + sqr(p1.y) -
        sqr(p4.y);
    double u23 = sqr(p2.x) - sqr(p4.x) + sqr(p2.y) -
        sqr(p4.y);
    double u33 = sqr(p3.x) - sqr(p4.x) + sqr(p3.y) -
        sqr(p4.y);
    double det = -u13*u22*u31 + u12*u23*u31 +
        u13*u21*u32 - u11*u23*u32 - u12*u21*u33 +
        u11*u22*u33;
    return det > eps;
}
double side(P const& a, P const& b, P const& p) {
    return (b.x-a.x)*(p.y-a.y) - (b.y-a.y)*(p.x-a.x);}
typedef int SideRef; struct Triangle; typedef Triangle*
    TriangleRef;
struct Edge {
    TriangleRef tri; SideRef side; Edge() : tri(0),
        side(0) {}
    Edge(TriangleRef tri, SideRef side) : tri(tri),
        side(side) {}
};
```

```cpp
struct Triangle {
    P p[3]; Edge edge[3]; TriangleRef children[3]; Triangle() {}
    Triangle(P const& p0, P const& p1, P const& p2) {
        p[0] = p0; p[1] = p1; p[2] = p2;
        children[0] = children[1] = children[2] = 0;
    }
    bool has_children() const { return children[0] != 0; }
    int num_children() const {
        return children[0] == 0 ? 0
            : children[1] == 0 ? 1
            : children[2] == 0 ? 2 : 3;
    }
    bool contains(P const& q) const {
        double a=side(p[0],p[1],q), b=side(p[1],p[2],q), c=side(p[2],p[0],q);
        return a >= -eps && b >= -eps && c >= -eps;
    }
} triange_pool[MAX_TRIS], *tot_triangles;
void set_edge(Edge a, Edge b) {
    if (a.tri) a.tri->edge[a.side] = b;
    if (b.tri) b.tri->edge[b.side] = a;
}
class Triangulation {
    public:
        Triangulation() {
            const double LOTS = 1e6;//初始为极大三角形
            the_root = new(tot_triangles++) Triangle(P(-LOTS,-LOTS),P(+LOTS,-LOTS),P(0,+LOTS));
        }
        TriangleRef find(P p) const { return find(the_root,p); }
        void add_point(P const& p) { add_point(find(the_root,p),p); }
    private:
        TriangleRef the_root;
        static TriangleRef find(TriangleRef root, P const& p) {
            for( ; ; ) {
                if (!root->has_children()) return root;
                else for (int i = 0; i < 3 && root->children[i] ; ++i)
                        if (root->children[i]->contains(p))
                            {root = root->children[i]; break;}
            }
        }
        void add_point(TriangleRef root, P const& p) {
            TriangleRef tab,tbc,tca;
            tab = new(tot_triangles++) Triangle(root->p[0], root->p[1], p);
            tbc = new(tot_triangles++) Triangle(root->p[1], root->p[2], p);
            tca = new(tot_triangles++) Triangle(root->p[2], root->p[0], p);
            set_edge(Edge(tab,0),Edge(tbc,1)); set_edge(Edge(tbc,0),Edge(tca,1));
            set_edge(Edge(tca,0),Edge(tab,1)); set_edge(Edge(tab,2),root->edge[2]);
            set_edge(Edge(tbc,2),root->edge[0]); set_edge(Edge(tca,2),root->edge[1]);
            root->children[0]=tab; root->children[1]=tbc; root->children[2]=tca;
            flip(tab,2); flip(tbc,2); flip(tca,2);
        }
        void flip(TriangleRef tri, SideRef pi) {
            TriangleRef trj = tri->edge[pi].tri; int pj = tri->edge[pi].side;
            if(!trj || !in_circumcircle(tri->p[0],tri->p[1],tri->p[2],trj->p[pj])) return;
            TriangleRef trk = new(tot_triangles++) Triangle(tri->p[(pi+1)%3], trj->p[pj], tri->p[pi]);
            TriangleRef trl = new(tot_triangles++) Triangle(trj->p[(pj+1)%3], tri->p[pi], trj->p[pj]);
            set_edge(Edge(trk,0), Edge(trl,0));
            set_edge(Edge(trk,1), tri->edge[(pi+2)%3]); set_edge(Edge(trk,2), trj->edge[(pj+1)%3]);
            set_edge(Edge(trl,1), trj->edge[(pj+2)%3]); set_edge(Edge(trl,2), tri->edge[(pi+1)%3]);
            tri->children[0]=trk; tri->children[1]=trl; tri->children[2]=0;
            trj->children[0]=trk; trj->children[1]=trl; trj->children[2]=0;
            flip(trk,1); flip(trk,2); flip(trl,1); flip(trl,2);
        }
};
int n; P ps[N];
void build(){
    tot_triangles = triange_pool; cin >> n;
    for(int i = 0; i < n; ++ i) scanf("%lf%lf",&ps[i].x,&ps[i].y);
    random_shuffle(ps, ps + n); Triangulation tri;
    for(int i = 0; i < n; ++ i) tri.add_point(ps[i]);
}
```

3.11 三维几何基础 (Grimoire)

```cpp
struct P {
    double x, y, z;
    P(){}
    P(double _x,double _y,double _z):x(_x),y(_y),z(_z){}
    double len2(){
        return (x*x+y*y+z*z);
    }
    double len(){
        return sqrt(x*x+y*y+z*z);
    }
};
bool operator==(P a,P b){
    return sgn(a.x-b.x)==0 && sgn(a.y-b.y)==0 && sgn(a.z-b.z)==0 ;
}
bool operator<(P a,P b){
    return sgn(a.x-b.x) ? a.x<b.x :(sgn(a.y-b.y)?a.y<b.y :a.z<b.z);
}
P operator+(P a,P b){
    return P(a.x+b.x,a.y+b.y,a.z+b.z);
}
P operator-(P a,P b){
    return P(a.x-b.x,a.y-b.y,a.z-b.z);
```

```cpp
}
P operator*(P a,double b){
    return P(a.x*b,a.y*b,a.z*b);
}
P operator/(P a,double b){
    return P(a.x/b,a.y/b,a.z/b);
}
P operator*(const P &a, const P &b) {
    return P(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
}
double operator^(const P &a, const P &b) {
    return a.x*b.x+a.y*b.y+a.z*b.z;
}

double dis(P a,P b){return (b-a).len();}
double dis2(P a,P b){return (b-a).len2();}

// 3D line intersect
P intersect(const P &a0, const P &b0, const P &a1, const P &b1) {
    double t = ((a0.x - a1.x) * (a1.y - b1.y) - (a0.y - a1.y) * (a1.x - b1.x)) / ((a0.x - b0.x) * (a1.y - b1.y) - (a0.y - b0.y) * (a1.x - b1.x));
    return a0 + (b0 - a0) * t;
}
// area-line intersect
P intersect(const P &a, const P &b, const P &c, const P &l0, const P &l1) {

    P p = (b-a)*(c-a); // 平面法向量
    double t = (p^(a-l0)) / (p^(l1-l0));
    return l0 + (l1 - l0) * t;
}
```

3.12 三维凸包 (Grimoire)

```cpp
int mark[1005][1005],n, cnt;;
double mix(const P &a, const P &b, const P &c) {
    return a^(b*c);
}
double area(int a, int b, int c) {
    return ((info[b] - info[a])*(info[c] - info[a])).len();
}
double volume(int a, int b, int c, int d) {
    return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]);
}
struct Face {
    int a, b, c; Face() {}
    Face(int a, int b, int c): a(a), b(b), c(c) {}
    int &operator [](int k) {
        if (k == 0) return a; if (k == 1) return b; return c;
    }
};
vector <Face> face;
inline void insert(int a, int b, int c) {
    face.push_back(Face(a, b, c));
}
void add(int v) {
    vector <Face> tmp; int a, b, c; cnt++;
    for (int i = 0; i < SIZE(face); i++) {
        a = face[i][0]; b = face[i][1]; c = face[i][2];
        if (sgn(volume(v, a, b, c)) < 0)
        mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] = mark[a][c] = cnt;
        else tmp.push_back(face[i]);
    } face = tmp;
    for (int i = 0; i < SIZE(tmp); i++) {
        a = face[i][0]; b = face[i][1]; c = face[i][2];
        if (mark[a][b] == cnt) insert(b, a, v);
        if (mark[b][c] == cnt) insert(c, b, v);
        if (mark[c][a] == cnt) insert(a, c, v);
    }
}
int Find() {
    for (int i = 2; i < n; i++) {
        P ndir = (info[0] - info[i])*(info[1] - info[i]);
        if (ndir == P()) continue; swap(info[i], info[2]);
        for (int j = i + 1; j < n; j++) if (sgn(volume(0, 1, 2, j)) != 0) {
            swap(info[j], info[3]); insert(0, 1, 2); insert(0, 2, 1); return 1;
        }
    }
    return 0;
}
//find the weight center
double calcDist(const P &p, int a, int b, int c) {
    return fabs(mix(info[a] - p, info[b] - p, info[c] - p) / area(a, b, c));
}
//compute the minimal distance of center of any faces
P findCenter() { //compute center of mass
    double totalWeight = 0;
    P center(.0, .0, .0);
    P first = info[face[0][0]];
    for (int i = 0; i < SIZE(face); ++i) {
        P p = (info[face[i][0]]+info[face[i][1]]+info[face[i][2]])*.25;
        double weight = mix(info[face[i][0]] - first, info[face[i][1]] - first, info[face[i][2]] - first);
        totalWeight += weight; center = center + p * weight;
    }
    center = center / totalWeight;
    return center;
}
double minDis(P p) {
    double res = 1e100; //compute distance
    for (int i = 0; i < SIZE(face); ++i)
        res = min(res, calcDist(p, face[i][0], face[i][1], face[i][2]));
    return res;
}

void findConvex(P *info,int n) {
    sort(info, info + n); n = unique(info, info + n) - info;
    face.clear(); random_shuffle(info, info + n);
    if(!Find())return abort();
    memset(mark, 0, sizeof(mark)); cnt = 0;
    for (int i = 3; i < n; i++) add(i);
}
```

3.13 三维绕轴旋转 (gy)

右手大拇指指向 $axis$ 方向，四指弯曲方向旋转 w 弧度

```cpp
P rotate(const P& s, const P& axis, double w) {
    double x = axis.x, y = axis.y, z = axis.z;
    double s1 = x * x + y * y + z * z, ss1 = msqrt(s1),
           cosw = cos(w), sinw = sin(w);
    double a[4][4];
    memset(a, 0, sizeof a);
    a[3][3] = 1;
    a[0][0] = ((y * y + z * z) * cosw + x * x) / s1;
    a[0][1] = x * y * (1 - cosw) / s1 + z * sinw / ss1;
    a[0][2] = x * z * (1 - cosw) / s1 - y * sinw / ss1;
    a[1][0] = x * y * (1 - cosw) / s1 - z * sinw / ss1;
    a[1][1] = ((x * x + z * z) * cosw + y * y) / s1;
    a[1][2] = y * z * (1 - cosw) / s1 + x * sinw / ss1;
    a[2][0] = x * z * (1 - cosw) / s1 + y * sinw / ss1;
    a[2][1] = y * z * (1 - cosw) / s1 - x * sinw / ss1;
    a[2][2] = ((x * x + y * y) * cos(w) + z * z) / s1;
    double ans[4] = {0, 0, 0, 0}, c[4] = {s.x, s.y, s.z, 1};
    for (int i = 0; i < 4; ++ i)
        for (int j = 0; j < 4; ++ j)
            ans[i] += a[j][i] * c[j];
    return P(ans[0], ans[1], ans[2]);
}
```

3.14 几何知识 (gy)

Pick theorem

顶点为整点的简单多边形，其面积 A，内部格点数 i，边上格点数 b 满足：
$A = i + \frac{b}{2} - 1$

欧拉示性数

- 三维凸包的顶点个数 V，边数 E，面数 F 满足：
$V - E + F = 2$

- 平面图的顶点个数 V，边数 E，平面被划分的区域数 F，组成图形的连通部分的数目 C 满足：
$V - E + F = C + 1$

几何公式

- **三角形**

半周长 $p = \frac{a+b+c}{2}$
面积 $S = \frac{1}{2}aH_a = \frac{1}{2}ab \cdot \sin C = \sqrt{p(p-a)(p-b)(p-c)} = pr = \frac{abc}{4R}$
中线长 $M_a = \frac{1}{2}\sqrt{2(b^2+c^2)-a^2} = \frac{1}{2}\sqrt{b^2+c^2+2bc\cdot\cos A}$
角平分线长 $T_a = \frac{\sqrt{bc((b+c)^2-a^2)}}{b+c} = \frac{2bc}{b+c}\cos\frac{A}{2}$
高 $H_a = b\sin C = \sqrt{b^2 - (\frac{a^2+b^2-c^2}{2a})^2}$
内切圆半径 $r = \frac{S}{p} = 4R\sin\frac{A}{2}\sin\frac{B}{2}\sin\frac{C}{2} = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}} = p\tan\frac{A}{2}\tan\frac{B}{2}\tan\frac{C}{2}$
外接圆半径 $R = \frac{abc}{4S} = \frac{a}{2\sin A}$
旁切圆半径 $r_A = \frac{2S}{-a+b+c}$
重心 $(\frac{x_1+x_2+x_3}{3}, \frac{y_1+y_2+y_3}{3})$
外心 $\left(\frac{\begin{vmatrix} x_1^2+y_1^2 & y_1 & 1 \\ x_2^2+y_2^2 & y_2 & 1 \\ x_3^2+y_3^2 & y_3 & 1 \end{vmatrix}}{2\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}, \frac{\begin{vmatrix} x_1 & x_1^2+y_1^2 & 1 \\ x_2 & x_2^2+y_2^2 & 1 \\ x_3 & x_3^2+y_3^2 & 1 \end{vmatrix}}{2\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}\right)$
内心 $(\frac{ax_1+bx_2+cx_3}{a+b+c}, \frac{ay_1+by_2+cy_3}{a+b+c})$
垂心 $\left(\frac{\begin{vmatrix} x_2x_3+y_2y_3 & 1 & y_1 \\ x_3x_1+y_3y_1 & 1 & y_2 \\ x_1x_2+y_1y_2 & 1 & y_3 \end{vmatrix}}{2\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}, \frac{\begin{vmatrix} x_2x_3+y_2y_3 & x_1 & 1 \\ x_3x_1+y_3y_1 & x_2 & 1 \\ x_1x_2+y_1y_2 & x_3 & 1 \end{vmatrix}}{2\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}\right)$
旁心 $(\frac{-ax_1+bx_2+cx_3}{-a+b+c}, \frac{-ay_1+by_2+cy_3}{-a+b+c})$

- **圆**

弧长 $l = rA$
弦长 $a = 2\sqrt{2hr-h^2} = 2r\cdot\sin\frac{A}{2}$
弓形高 $h = r - \sqrt{r^2 - \frac{a^2}{4}} = r(1-\cos\frac{A}{2})$
扇形面积 $S_1 = \frac{1}{2}lr = \frac{1}{2}Ar^2$
弓形面积 $S_2 = \frac{1}{2}r^2(A - \sin A)$

- **Circles of Apollonius**

已知三个两两相切的圆，半径为 r_1, r_2, r_3
与它们外切的圆半径为 $\left|\frac{r_1r_2r_3}{r_1r_2+r_2r_3+r_3r_1-2\sqrt{r_1r_2r_3(r_1+r_2+r_3)}}\right|$
与它们内切的圆半径为 $\frac{r_1r_2r_3}{r_1r_2+r_2r_3+r_3r_1+2\sqrt{r_1r_2r_3(r_1+r_2+r_3)}}$

- **棱台**

体积 $V = \frac{1}{3}h(A_1 + A_2 + \sqrt{A_1A_2})$
正棱台侧面积 $S = \frac{1}{2}(p_1+p_2)l$，$l$ 为侧高

- **球**

体积 $V = \frac{4}{3}\pi r^3$
表面积 $S = 4\pi r^2$

- **球台**

侧面积 $S = 2\pi rh$
体积 $V = \frac{1}{6}\pi h(3(r_1^2+r_2^2)+h_h)$

- **球扇形**

球面面积 $S = 2\pi rh$
体积 $V = \frac{2}{3}\pi r^2 h = \frac{2}{3}\pi r^3 h(1-\cos\varphi)$

- **球面三角形**

考虑单位球上的球面三角形，a, b, c 表示三边长（弧所对球心角），A, B, C 表示三角大小（切线夹角）
余弦定理 $\cos a = \cos b \cdot \cos c + \sin a \cdot \sin b \cdot \cos A$
正弦定理 $\frac{\sin A}{\sin a} = \frac{\sin B}{\sin b} = \frac{\sin C}{\sin c}$
球面面积 $S = A + B + C - \pi$

- **四面体**

体积 $V = \frac{1}{6}\left|\overrightarrow{AB}\cdot(\overrightarrow{AC}\times\overrightarrow{AD})\right|$

4. String

4.1 KMP (ct)

KMP

```cpp
int main()
```

```cpp
    for (int i = 2, j = 0; i <= n; ++i)
    {
        for ( ; j && s[j + 1] != s[i]; j = fail[j]) ;
        s[i] == s[j + 1] ? ++j : 0;
        fail[i] = j;
    }
    return 0;
}
```

exKMP

$extend_i$ 表示 T 与 $S_{i,n}$ 的最长公共前缀

```cpp
int next[maxn], extend[maxn], fail[maxn];
void getnext(R char *s, R int len)
{
    fail[1] = 0;
    R int p = 0;
    memset(next, 0, (len + 2) << 2);
    for (R int i = 2; i <= len; ++i)
    {
        while (p && s[p + 1] != s[i]) p = fail[p];
        s[p + 1] == s[i] ? ++p : 0;
        fail[i] = p;
        p ? cmax(next[i - p + 1], p) : 0;
    }
}
void getextend(R char *s, R int lens, R char *t, R int lent)
{
    getnext(t, lent);
    R int a = 1, p = 0;

    for (R int i = 1; i <= lens; ++i)
    {
        if (i + next[i - a + 1] - 1 >= p)
        {
            cmax(p, i - 1);
            while (p < lens && p - i + 1 < lent && s[p + 1] == t[p - i + 2]) ++p;
            a = i;
            extend[i] = p - i + 1;
        }
        else extend[i] = next[i - a + 1];
    }
}
```

4.2 AC 自动机 (ct)

```cpp
struct Trie {
    Trie *next[26], *fail;
    int end;
} mem[maxn * maxl], *tot = mem, *q[maxn * maxl];
char s[maxl];
inline void insert(int v)
{
    Trie *now = mem; int n = strlen(s + 1);
    for (int i = 1; i <= n; ++i)
    {
        int v = s[i] - 'a';
        if (!now -> next[v])
        {
            now -> next[v] = ++tot;
            for (int i = 0; i < 26; ++i) tot -> next[i] = 0;
            tot -> fail = 0;
            tot -> end = 0;
        }
        now = now -> next[v];
    }
    now -> end |= v;
}
inline void build()
{
    int head = 0, tail = 0;
    for (int i = 0; i < 26; ++i)
        if (mem -> next[i]) (q[++tail] = mem -> next[i]) -> fail = mem;
        else mem -> next[i] = mem;
    while (head < tail)
    {
        Trie *now = q[++head];
        now -> end |= now -> fail -> end;
        for (int i = 0; i < 26; ++i)
            if (!now -> next[i])
                now -> next[i] = now -> fail -> next[i];
            else
                (q[++tail] = now -> next[i]) -> fail = now -> fail -> next[i];
    }
}
```

4.3 Lydon Word Decomposition (Nightfall)

满足 s 的最小后缀等于 s 本身的串称为 Lyndon 串.
等价于: s 是它自己的所有循环移位中唯一最小的一个.
任意字符串 s 可以分解为 $s = \overline{s_1 s_2 \ldots s_k}$, 其中 s_i 是 Lyndon 串, $s_i \geq s_i + 1$。且这种分解方法是唯一的。

```cpp
void mnsuf(char *s, int *mn, int n) { // 每个前缀的最小后缀
    for (int i = 0; i < n; ) {
        int j = i, k = i + 1; mn[i] = i;
        for (; k < n && s[j] <= s[k]; ++ k)
            if (s[j] == s[k]) mn[k] = mn[j] + k - j, ++j;
            else mn[k] = j = i;
        for (; i <= j; i += k - j) {} } } // lyn+=s[i..i+k-j-1]
        void mxsuf(char *s, int *mx, int n) { // 每个前缀的最大后缀
            fill(mx, mx + n, -1);
            for (int i = 0; i < n; ) {
                int j = i, k = i + 1; if (mx[i] == -1) mx[i] = i;
                for (; k < n && s[j] >= s[k]; ++k) {
                    j = s[j] == s[k] ? j + 1 : i;
                    if (mx[k] == -1) mx[k] = i; }
                for (; i <= j; i += k - j) {} }
```

4.4 后缀数组 (ct)

```cpp
char s[maxn];
int sa[maxn], rank[maxn], wa[maxn], wb[maxn], cnt[maxn], height[maxn];
inline void build(int n, int m)
{
    int *x = wa, *y = wb, *t;
    for (int i = 1; i <= n; ++i) cnt[x[i] = s[i] - 'a' + 1]++;
    for (int i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
    for (int i = n; i; --i) sa[cnt[x[i]]--] = i;

```

```
    for (int j = 1; j < n || (j == 1 && m < n); j <<= 1, t = x, x = y, y = t)
    {
        memset(cnt + 1, 0, m << 2);
        int p = 0;
        for (int i = n - j + 1; i <= n; ++i) y[++p] = i;
        for (int i = 1; i <= n; ++i)
        {
            ++cnt[x[i]];
            sa[i] > j ? y[++p] = sa[i] - j : 0;
        }
        for (int i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
        for (int i = n; i; --i) sa[cnt[x[y[i]]]--] = y[i];
            m = 0;
        for (int i = 1; i <= n; ++i)
            y[sa[i]] = (i == 1 || x[sa[i]] != x[sa[i - 1]] || x[sa[i - 1] + j] != x[sa[i] + j]) ? ++m : m;
    }
    for (int i = 1; i <= n; ++i) rank[sa[i]] = i;
    for (int i = 1, j, k = 0; i <= n; height[rank[i++]] = k)
        for (k ? --k : 0, j = sa[rank[i] - 1]; s[i + k] == s[j + k]; ++k);
}
```

4.5 后缀自动机 (ct,lhy)

后缀自动机 (lhy)

```
struct Sam{
    Sam *fa, *go[26];
    int val;

    void clear()
    {
        fa = 0;
        val = 0;
        memset(go, 0, sizeof(go));
    }
}*now, *root, *last, *cur, Pool[N << 1];

void Prepare()
{
    cur = Pool;
    cur -> clear();
    root = last = cur;
}

Sam *Insert(Sam *last, int now)
{
    Sam *p = last;
    if(p -> go[now])
    {
        Sam *q = p -> go[now];
        if(q -> val == p -> val + 1)return q;
        Sam *nt = ++cur;
        nt -> clear();
        nt -> val = p -> val + 1;
        memcpy(nt -> go, q -> go, sizeof(q -> go));
        nt -> fa = q -> fa;
        q -> fa = nt;
        while(p && p -> go[now] == q)p -> go[now] = nt, p = p -> fa;
        return nt;
    }
    Sam *np = ++cur;
    np -> clear();
    np -> val = p -> val + 1;
    while(p && !p -> go[now])p -> go[now] = np, p = p -> fa;
    if(!p)np -> fa = root;
    else
    {
        Sam *q = p -> go[now];
        if(q -> val == p -> val + 1)
        {
            np -> fa = q;
        }
        else
        {
            Sam *nt = ++cur;
            nt -> clear();
            nt -> val = p -> val + 1;
            memcpy(nt -> go, q -> go, sizeof q -> go);
            nt -> fa = q -> fa;
            q -> fa = nt;
            np -> fa = nt;
            while(p && p -> go[now] == q)p -> go[now] = nt, p = p -> fa;
        }
    }
    return np;
}
```

后缀自动机 (ct)

```
struct SAM {
    SAM *next[26], *fa;
    int val;
} mem[maxn], *last = mem, *tot = mem;
void extend(int c)
{
    R SAM *p = last, *np;
    last = np = ++tot; np -> val = p -> val + 1;
    for (; p && !p -> next[c]; p = p -> fa) p -> next[c] = np;
    if (!p) np -> fa = rt[id];
    else
    {
        SAM *q = p -> next[c];
        if (q -> val == p -> val + 1) np -> fa = q;
        else
        {
            SAM *nq = ++tot;
            memcpy(nq -> next, q -> next, sizeof nq -> next);
            nq -> val = p -> val + 1;
            nq -> fa = q -> fa;
            q -> fa = np -> fa = nq;
            for (; p && p -> next[c] == q; p = p -> fa) p -> next[c] = nq;
        }
    }
}
```

广义后缀自动机 (ct)

```
struct sam {
    sam *next[26], *fa;
    int val;
} mem[maxn << 1], *tot = mem;
inline sam *extend(sam *p, int c)
```

```cpp
{
    if (p -> next[c])
    {
        sam *q = p -> next[c];
        if (q -> val == p -> val + 1)
            return q;
        else
        {
            sam *nq = ++tot;
            memcpy(nq -> next, q -> next, sizeof nq -> next);
            nq -> val = p -> val + 1;
            nq -> fa = q -> fa;
            q -> fa = nq;
            for ( ; p && p -> next[c] == q; p = p -> fa)
                p -> next[c] = nq;
            return nq;
        }
    }
    sam *np = ++tot;
    np -> val = p -> val + 1;
    for ( ; p && !p -> next[c]; p = p -> fa) p -> next[c] = np;
    if (!p)
        np -> fa = mem;
    else
    {
        sam *q = p -> next[c];
        if (q -> val == p -> val + 1)
            np -> fa = q;
        else
        {
            sam *nq = ++tot;
            memcpy(nq -> next, q -> next, sizeof nq -> next);
            nq -> val = p -> val + 1;
            nq -> fa = q -> fa;
            q -> fa = np -> fa = nq;
            for ( ; p && p -> next[c] == q; p = p -> fa)
                p -> next[c] = nq;
        }
    }
    return np;
}
```

4.6 Manacher (ct)

```cpp
char str[maxn];
int p1[maxn], p2[maxn], n;
void manacher1()
{
    int mx = 0, id;
    for(int i = 1; i <= n; ++i)
    {
        if (mx >= i) p1[i] = dmin(mx - i, p1[(id << 1) - i]);
        else p1[i] = 1;
        for (; str[i + p1[i]] == str[i - p1[i]]; ++p1[i]) ;
        if (p1[i] + i - 1 > mx) id = i, mx = p1[i] + i - 1;
    }
}
void manacher2()
{
    int mx = 0, id;
    for(int i = 1; i <= n; i++)
    {
        if (mx >= i) p2[i] = dmin(mx - i, p2[(id << 1) - i]) ;
        else p2[i] = 0;
        for (; str[i + p2[i] + 1] == str[i - p2[i]]; ++p2[i]);
        if (p2[i] + i > mx) id = i, mx = p2[i] + i;
    }
}
int main()
{
    scanf("%s", str + 1);
    n = strlen(str + 1);
    str[0] = '#';
    str[n + 1] = '$';
    manacher1();
    manacher2();
    return 0;
}
```

4.7 回文树 (ct)

```cpp
char str[maxn];
int next[maxn][26], fail[maxn], len[maxn], cnt[maxn], last, tot, n;
inline int new_node(int l)
{
    len[++tot] = l;
    return tot;
}
inline void init()
{
    tot = -1;
    new_node(0);
    new_node(-1);
    str[0] = -1;
    fail[0] = 1;
}
inline int get_fail(int x)
{
    while (str[n - len[x] - 1] != str[n]) x = fail[x];
    return x;
}
inline void extend(int c)
{
    ++n;
    int cur = get_fail(last);
    if (!next[cur][c])
    {
        int now = new_node(len[cur] + 2);
        fail[now] = next[get_fail(fail[cur])][c];
        next[cur][c] = now;
    }
    last = next[cur][c];
    ++cnt[last];
}
long long ans;
inline void count()
{
    for (int i = tot; i; --i)
    {
        cnt[fail[i]] += cnt[i];
        cmax(ans, 1ll * len[i] * cnt[i]);
    }
}
int main()
{
```

```
    scanf("%s", str + 1);
    init();
    for (int i = 1; str[i]; ++i)
        extend(str[i] - 'a');
    count();
    printf("%lld\n", ans );
    return 0;
}
```

4.8 最小表示法 (ct)

```
int main()
{
    int i = 0, j = 1, k = 0;
    while (i < n && j < n && k < n)
    {
        int tmp = a[(i + k) % n] - a[(j + k) % n];
        if (!tmp) k++;
        else
        {
            if (tmp > 0) i += k + 1;
            else j += k + 1;
            if (i == j) ++j;
            k = 0;
        }
    }
    j = dmin(i, j);
    for (int i = j; i < n; ++i) printf("%d ", a[i]);
    for (int i = 0; i < j - 1; ++i) printf("%d ",
    ↪ a[i]);
    if (j > 0) printf("%d\n", a[j - 1]);
    return 0;
}
```

4.9 字符串知识 (Nightfall)

双回文串

如果 $s = x_1x_2 = y_1y_2 = z_1z_2$, $|x_1| < |y_1| < |z_1|$, x_2, y_1, y_2, z_1 是回文串，则 x_1 和 z_2 也是回文串。

Border 的结构

字符串 s 的所有不小于 $|s|/2$ 的 border 长度构成一个等差数列。
字符串 s 的所有 border 按长度排序后可分成 $O(\log|s|)$ 段，每段是一个等差数列。
回文串的回文后缀同时也是它的 border。

子串最小后缀

设 $s[p..n]$ 是 $s[i..n], (l \le i \le r)$ 中最小者，则 $\mathrm{minsuf}(l, r)$ 等于 $s[p..r]$ 的最短非空 border。$\mathrm{minsuf}(l, r) = \min\{s[p..r], \mathrm{minsuf}(r - 2^k + 1, r)\}, (2^k < r - l + 1 \le 2^{k+1})$。

子串最大后缀

从左往右，用 set 维护后缀的字典序递减的单调队列，并在对应时刻添加 “小于事件” 点以便以后修改队列；查询直接在 set 里 lower_bound

5. Data Structure

5.1 莫队 (ct)

```
int size;
struct Query {
    int l, r, id;
    inline bool operator < (const Queuy &that) const
    ↪ {return l / size != that.l / size ? l < that.l
    ↪ : ((l / size) & 1 ? r < that.r : r > that.r);}
} q[maxn];
int main()
{
    size = (int) sqrt(n * 1.0);
    std::sort(q + 1, q + m + 1);
    int l = 1, r = 0;
    for (int i = 1; i <= m; ++i)
    {
        for (; r < q[i].r; ) add(++r);
        for (; r > q[i].r; ) del(r--);
        for (; l < q[i].l; ) del(l++);
        for (; l > q[i].l; ) add(--l);
        /*
            write your code here.
        */
    }
    return 0;
}
```

5.2 ST 表 (ct)

```
int a[maxn], f[20][maxn], n;
int Log[maxn];

void build()
{
    for (int i = 1; i <= n; ++i) f[0][i] = a[i];

    int lim = Log[n];
    for (int j = 1; j <= lim; ++j)
    {
        int *fj = f[j], *fj1 = f[j - 1];
        for (int i = 1; i <= n - (1 << j) + 1; ++i)
            fj[i] = dmax(fj1[i], fj1[i + (1 << (j -
            ↪ 1))]);
    }
}
int Query(int l, int r)
{
    int k = Log[r - l + 1];
    return dmax(f[k][l], f[k][r - (1 << k) + 1]);
}
int main()
{
    scanf("%d", &n);
    Log[0] = -1;
    for (int i = 1; i <= n; ++i)
    {
        scanf("%d", &a[i]);
        Log[i] = Log[i >> 1] + 1;
    }
    build();
```

```cpp
    int q;
    scanf("%d", &q);
    for (; q; --q)
    {
        int l, r; scanf("%d%d", &l, &r);
        printf("%d\n", Query(l, r) );
    }
}
```

5.3 带权并查集 (ct)

```cpp
struct edge
{
    int a, b, w;
    inline bool operator < (const edge &that) const {return w > that.w;}
} e[maxm];
int fa[maxn], f1[maxn], f2[maxn], f1cnt, f2cnt, val[maxn], size[maxn];
int main()
{
    int n, m; scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; ++i)
        scanf("%d%d%d", &e[i].a, &e[i].b, &e[i].w);
    for (int i = 1; i <= n; ++i) size[i] = 1;
    std::sort(e + 1, e + m + 1);
    for (int i = 1; i <= m; ++i)
    {
        int x = e[i].a, y = e[i].b;
        for ( ; fa[x]; x = fa[x]) ;
        for ( ; fa[y]; y = fa[y]) ;
        if (x != y)
        {
            if (size[x] < size[y]) std::swap(x, y);
            size[x] += size[y];
            val[y] = e[i].w;
            fa[y] = x;
        }
    }

    int q; scanf("%d", &q);
    for (; q; --q)
    {
        int a, b; scanf("%d%d", &a, &b); f1cnt = f2cnt = 0;
        for (; fa[a]; a = fa[a]) f1[++f1cnt] = a;
        for (; fa[b]; b = fa[b]) f2[++f2cnt] = b;
        if (a != b) {puts("-1"); continue;}
        while (f1cnt && f2cnt && f1[f1cnt] == f2[f2cnt]) --f1cnt, --f2cnt;
        int ret = 0x7fffffff;
        for (; f1cnt; --f1cnt) cmin(ret, val[f1[f1cnt]]);
        for (; f2cnt; --f2cnt) cmin(ret, val[f2[f2cnt]]);
        printf("%d\n", ret);
    }
    return 0;
}
```

5.4 可并堆 (ct)

```cpp
struct Node {
    Node *ch[2];
    ll val; int size;
    inline void update()
    {
        size = ch[0] -> size + ch[1] -> size + 1;
    }
} mem[maxn], *rt[maxn];
Node *merge(Node *a, Node *b)
{
    if (a == mem) return b;
    if (b == mem) return a;
    if (a -> val < b -> val) std::swap(a, b);
    // a -> pushdown();
    std::swap(a -> ch[0], a -> ch[1]);
    a -> ch[1] = merge(a -> ch[1], b);
    a -> update();
    return a;
}
```

5.5 线段树 (ct)

zkw 线段树

0-based

```cpp
inline void build()
{
    for (int i = M - 1; i; --i) tr[i] = dmax(tr[i << 1], tr[i << 1 | 1]);
}
inline void Change(int x, int v)
{
    x += M; tr[x] = v; x >>= 1;
    for (; x; x >>= 1) tr[x] = dmax(tr[x << 1], tr[x << 1 | 1]);
}
inline int Query(int s, int t)
{
    int ret = -0x7fffffff;
    for (s = s + M - 1, t = t + M + 1; s ^ t ^ 1; s >>= 1, t >>= 1)
    {
        if (~s & 1) cmax(ret, tr[s ^ 1]);
        if (t & 1) cmax(ret, tr[t ^ 1]);
    }
    return ret;
}
int main()
{
    int n; scanf("%d", &n);
    for (M = 1; M < n; M <<= 1) ;
    for (int i = 0; i < n; ++i)
        scanf("%d", &tr[i + M]);
    for (int i = n; i < M; ++i) tr[i + M] = -0x7fffffff;
    build();
    int q; scanf("%d", &q);
    for (; q; --q)
    {
        int l, r; scanf("%d%d", &l, &r); --l, --r;
        printf("%d\n", Query(l, r));
    }
    return 0;
}
```

李超线段树

```cpp
int size[maxn], dep[maxn], son[maxn], fa[maxn], top[maxn], dfn[maxn], pos[maxn], timer, rig[maxn];
ll dis[maxn];
bool vis[maxn];
// 树链剖分 begin
```

```cpp
void dfs1(int x);
void dfs2(int x){cmax(rig[top[x]], dfn[x]);}
inline int getlca(int a, int b);
// 树链剖分 end
struct Seg {
    Seg *ls, *rs;
    ll min, k, b, vl, vr;
    // min 表示区间最小值
    // k 表示区间内 直线标记的斜率
    // b 表示区间内 直线标记的截距
    // vl, vr 表示区间内 x 的最小值和最大值
    inline void update()
    {
        min = dmin(ls -> min, rs -> min);
        k > 0 ? cmin(min, k * vl + b) : cmin(min, k * vr + b);
    }
} ssegg[maxn << 2], *scnt = ssegg, *rt[maxn];
void build(int l, int r)
{
    R Seg *o = scnt; o -> k = 0; o -> b = inf;
    o -> vl = dis[pos[l]]; o -> vr = dis[pos[r]]; o -> min = inf;
    if (l == r) return ;
    int mid = l + r >> 1;
    o -> ls = ++scnt; build(l, mid);
    o -> rs = ++scnt; build(mid + 1, r);
    o -> update();
}
int ql, qr, qk;
ll qb;
void modify(R Seg *o, int l, int r, int k, ll b)
{
    int mid = l + r >> 1;

    if (ql <= l && r <= qr)
    {
        if (l == r)
        {
            cmin(o -> min, k * o -> vl + b);
            return ;
        }
        ll
        val = o -> vl * k + b,
        var = o -> vr * k + b,
        vbl = o -> vl * o -> k + o -> b,
        vbr = o -> vr * o -> k + o -> b;
        if (val <= vbl && var <= vbr)
        {
            o -> k = k; o -> b = b;
            o -> update();
            return ;
        }
        if (val >= vbl && var >= vbr) return ;
        ll dam = dis[pos[mid]], vam = dam * k + b, vbm = dam * o -> k + o -> b;
        if (val >= vbl && vam <= vbm)
        {
            modify(o -> ls, l, mid, o -> k, o -> b);
            o -> k = k; o -> b = b;
        }
        else if (val <= vbl && vam >= vbm)
            modify(o -> ls, l, mid, k, b);
        else
        {
            if (vam <= vbm && var >= vbr)
            {
                modify(o -> rs, mid + 1, r, o -> k, o -> b);
                o -> k = k; o -> b = b;
            }
            else
                modify(o -> rs, mid + 1, r, k, b);
        }
        o -> update();
        return ;
    }
    if (ql <= mid) modify(o -> ls, l, mid, k, b);
    if (mid < qr) modify(o -> rs, mid + 1, r, k, b);
    o -> update();
}
ll query(R Seg *o, int l, int r)
{
    if (ql <= l && r <= qr) return o -> min;
    int mid = l + r >> 1; ll ret = inf, tmp;
    cmin(ret, dis[pos[dmax(ql, l)]] * o -> k + o -> b);
    cmin(ret, dis[pos[dmin(qr, r)]] * o -> k + o -> b);
    if (ql <= mid) tmp = query(o -> ls, l, mid), cmin(ret, tmp);
    if (mid < qr) tmp = query(o -> rs, mid + 1, r), cmin(ret, tmp);
    return ret;
}
inline void tr_modify(int x, int f)
{
    while (top[x] != top[f])
    {
        ql = dfn[top[x]]; qr = dfn[x];
        modify(rt[top[x]], ql, rig[top[x]], qk, qb);
        x = fa[top[x]];
    }
    ql = dfn[f]; qr = dfn[x];
    modify(rt[top[x]], dfn[top[x]], rig[top[x]], qk, qb);
}
inline ll tr_query(int s, int t)
{
    ll ret = inf, tmp;
    while (top[s] != top[t])
    {
        if (dep[top[s]] < dep[top[t]])
        {
            ql = dfn[top[t]]; qr = dfn[t];
            tmp = query(rt[top[t]], ql, rig[top[t]]);
            cmin(ret, tmp);
            t = fa[top[t]];
        }
        else
        {
            ql = dfn[top[s]]; qr = dfn[s];
            tmp = query(rt[top[s]], ql, rig[top[s]]);
            cmin(ret, tmp);
            s = fa[top[s]];
        }
    }
    ql = dfn[s]; qr = dfn[t]; ql > qr ? std::swap(ql, qr), 1 : 0;
    tmp = query(rt[top[s]], dfn[top[s]], rig[top[s]]);
    cmin(ret, tmp);
    return ret;
}
int main()
{
    int n, m; scanf("%d%d", &n, &m);
    for (int i = 1; i < n; ++i)
```

```
    {
        int a, b, w; scanf("%d%d%d", &a, &b, &w);
         link(a, b, w);
    }
    dfs1(1); dfs2(1);
    for (int i = 1; i <= n; ++i)
        if (top[i] == i)
        {
            rt[i] = ++scnt;
            build(dfn[i], rig[i]);
        }
    for (; m; --m)
    {
        int opt, s, t, lca; scanf("%d%d%d", &opt, &s,
         &t);
        lca = getlca(s, t);
        if (opt == 1)
        {
            int a; ll b; scanf("%d%lld", &a, &b);
            lca = getlca(s, t);
            qk = -a; qb = a * dis[s] + b;
            tr_modify(s, lca);
            qk = a; qb = a * dis[s] - dis[lca] * 2 * a
             + b;
            tr_modify(t, lca);
        }
        else
        {
            printf("%lld\n", tr_query(s, t));
        }
    }
    return 0;
}
```

吉利线段树

吉利线段树能解决一类区间与某个数取最大或最小，区间求和的问题。以区间取最小值为例，在线段树的每一个节点额外维护区间中的最大值 ma，严格次大值 se 以及最大值个树 t。现在假设我们要让区间 $[L, R]$ 对 x 取最小值，先在线段树中定位若干个节点，对于每个节点分三种情况讨论：

- 当 $ma \le x$ 时，显然这一次修改不会对这个节点产生影响，直接推出。
- 当 $se < x < ma$ 时，显然这一次修改只会影响到所有最大值，所以把 num 加上 $t \times (x - ma)$，把 ma 更新为 x，打上标记推出。
- 当 $x \le se$ 时，无法直接更新这一个节点的信息，对当前节点的左儿子和右儿子递归处理。

单次操作的均摊复杂度为 $O(\log^2 n)$

线段树维护折线

对于线段树每个结点维护两个值：ans 和 max，ans 表示只考虑这个区间的可视区间的答案，max 表示这个区间的最大值。那么问题的关键就在于如何合并两个区间，显然左区间的答案肯定可以作为总区间的答案，那么接下来就是看右区间有多少个在新加入左区间的约束后是可行的。考虑如果右区间最大值都小于等于左区间最大值那么右区间就没有贡献了，相当于是被整个挡住了。

如果大于最大值，就再考虑右区间的两个子区间：左子区间、右子区间，加入左子区间的最大值小于等于左区间最大值，那么就递归处理右子区间；否则就递归处理左子区间，然后加上右子区间原本的答案。考虑这样做的必然性：因为加入左区间最高的比左子区间最高的矮，那么相当于是左区间对于右子区间没有约束，都是左子区间产生的约束。但是右子区间的答案要用右区间答案 − 左子区间答案，不能直接调用右子区间本身答案，因为其本身答案没有考虑左子区间的约束。

线段树维护矩形面积并

线段树上维护两个值：$Cover$ 和 Len
$Cover$ 意为这个区间被覆盖了多少次
Len 意为区间被覆盖的总长度
Maintain 的时候，如果 $Cover > 0$，Len 直接为区间长
否则从左右子树递推 Len
修改的时候直接改 $Cover$ 就好

5.6 二进制分组 (ct)

用线段树维护时间的操作序列，每次操作一个一个往线段树里面插，等到一个线段被插满的时候用归并来维护区间的信息。查询的时候如果一个线段没有被插满就递归下去。定位到一个区间的时候在区间里面归并出来的信息二分。

```
int x[maxn], tnum;
struct Seg {
    int l, r, a, b;
} p[maxn * 200];
int lef[maxm << 2], rig[maxm << 2], pcnt, ta, tb, ql,
 qr, n, m, k, ans;
void update(int o, int l, int r)
{
    lef[o] = pcnt + 1;
    for (int i = lef[o << 1], j = lef[o << 1 | 1], head
     = 1; i <= rig[o << 1] || j <= rig[o << 1 | 1];
     )
        if (p[i].r <= p[j].r)
        {
            p[++pcnt] = (Seg) {head, p[i].r, 1ll *
             p[i].a * p[j].a % m, (1ll * p[j].a *
             p[i].b + p[j].b) % m};
            head = p[i].r + 1;
            p[i].r == p[j].r ? ++j : 0; ++i;
        }
        else
        {
            p[++pcnt] = (Seg) {head, p[j].r, 1ll *
             p[i].a * p[j].a % m, (1ll * p[j].a *
             p[i].b + p[j].b) % m};
            head = p[j].r + 1; ++j;
        }
    rig[o] = pcnt;
}
int find(int o, int t, int &s)
{
    int l = lef[o], r = rig[o];
    while (l < r)
    {
        int mid = l + r >> 1;
        if (t <= p[mid].r) r = mid;
        else l = mid + 1;
    }
//    printf("%d %d t %d s %d %d %d\n", p[l].l, p[l].r,
 t, s, p[l].a, p[l].b);
    s = (1ll * s * p[l].a + p[l].b) % m;
}
void modify(int o, int l, int r, int t)
{
    if (l == r)
    {
        lef[o] = pcnt + 1;
```

```cpp
        ql > 1 ? p[++pcnt] = (Seg) {1, ql - 1, 1, 0}, 1: 0;
        p[++pcnt] = (Seg) {ql, qr, ta, tb};
        qr < n ? p[++pcnt] = (Seg) {qr + 1, n, 1, 0}, 1: 0;
        rig[o] = pcnt;
        return ;
    }
    int mid = l + r >> 1;
    if (t <= mid) modify(o << 1, l, mid, t);
    else modify(o << 1 | 1, mid + 1, r, t);

    if (t == r) update(o, l, r);
}
void query(int o, int l, int r)
{
    if (ql <= l && r <= qr)
    {
        find(o, k, ans);
        return ;
    }
    int mid = l + r >> 1;
    if (ql <= mid) query(o << 1, l, mid);
    if (mid < qr) query(o << 1 | 1, mid + 1, r);
}
int main()
{
    int type; scanf("%d%d%d", &type, &n, &m);
    for (int i = 1; i <= n; ++i) scanf("%d", &x[i]);
    int Q; scanf("%d", &Q);
    for (int QQ = 1; QQ <= Q; ++QQ)
    {
        int opt, l, r; scanf("%d%d%d", &opt, &l, &r);
        type & 1 ? l ^= ans, r ^= ans : 0;
        if (opt == 1)
        {
            scanf("%d%d", &ta, &tb); ++tnum; ql = l; qr = r;
            modify(1, 1, Q, tnum);
        }
        else
        {
            scanf("%d", &k); type & 1 ? k ^= ans : 0; ql = l; qr = r;
            ans = x[k];
            query(1, 1, Q);
            printf("%d\n", ans);
        }
    }
    return 0;
}
```

5.7 Splay (ct)

指针版

```cpp
struct Node *null;
struct Node {
    Node *ch[2], *fa;
    int val; bool rev;
    inline bool type()
    {
        return fa -> ch[1] == this;
    }
    inline void pushup()
    {
    }
    inline void pushdown()
    {
        if (rev)
        {
            ch[0] -> rev ^= 1;
            ch[1] -> rev ^= 1;
            std::swap(ch[0], ch[1]);
            rev ^= 1;
        }
    }
    inline void rotate()
    {
        bool d = type(); Node *f = fa, *gf = f -> fa;
        (fa = gf, f -> fa != null) ? fa -> ch[f -> type()] = this : 0;
        (f -> ch[d] = ch[!d]) != null ? ch[!d] -> fa = f : 0;
        (ch[!d] = f) -> fa = this;
        f -> pushup();
    }
    inline void splay()
    {
        for (; fa != null; rotate())
            if (fa -> fa != null)
                (type() == fa -> type() ? fa : this) -> rotate();
        pushup();
    }
} mem[maxn];
```

维修序列

```cpp
int fa[maxn], ch[maxn][2], a[maxn], size[maxn], cnt;
int sum[maxn], lmx[maxn], rmx[maxn], mx[maxn], v[maxn], id[maxn], root;
bool rev[maxn], tag[maxn];
inline void update(int x)
{
    int ls = ch[x][0], rs = ch[x][1];
    size[x] = size[ls] + size[rs] + 1;
    sum[x] = sum[ls] + sum[rs] + v[x];
    mx[x] = gmax(mx[ls], mx[rs]);
    cmax(mx[x], lmx[rs] + rmx[ls] + v[x]);
    lmx[x] = gmax(lmx[ls], sum[ls] + v[x] + lmx[rs]);
    rmx[x] = gmax(rmx[rs], sum[rs] + v[x] + rmx[ls]);
}
inline void pushdown(int x)
{
    int ls = ch[x][0], rs = ch[x][1];
    if (tag[x])
    {
        rev[x] = tag[x] = 0;
        if (ls) tag[ls] = 1, v[ls] = v[x], sum[ls] = size[ls] * v[x];
        if (rs) tag[rs] = 1, v[rs] = v[x], sum[rs] = size[rs] * v[x];
        if (v[x]>= 0)
        {
            if (ls) lmx[ls] = rmx[ls] = mx[ls] = sum[ls];
            if (rs) lmx[rs] = rmx[rs] = mx[rs] = sum[rs];
        }
        else
        {
            if (ls) lmx[ls] = rmx[ls] = 0, mx[ls] = v[x];
            if (rs) lmx[rs] = rmx[rs] = 0, mx[rs] = v[x];
        }
```

```cpp
    }
    if (rev[x])
    {
        rev[x] ^= 1; rev[ls] ^= 1; rev[rs] ^= 1;
        swap(lmx[ls], rmx[ls]);swap(lmx[rs], rmx[rs]);
        swap(ch[ls][0], ch[ls][1]); swap(ch[rs][0], ch[rs][1]);
    }
}
inline void rotate(int x)
{
    int f = fa[x], gf = fa[f], d = ch[f][1] == x;
    if (f == root) root = x;
    (ch[f][d] = ch[x][d ^ 1]) > 0 ? fa[ch[f][d]] = f : 0;
    (fa[x] = gf) > 0 ? ch[gf][ch[gf][1] == f] = x : 0;
    fa[ch[x][d ^ 1] = f] = x;
    update(f);
}
inline void splay(int x, int rt)
{
    while (fa[x] != rt)
    {
        int f = fa[x], gf = fa[f];
        if (gf != rt) rotate((ch[gf][1] == f) ^ (ch[f][1] == x) ? x : f);
        rotate(x);
    }
    update(x);
}
void build(int l, int r, int rt)
{
    if (l > r) return ;
    int mid = l + r >> 1, now = id[mid], last = id[rt];
    if (l == r)
    {
        sum[now] = a[l];
        size[now] = 1;
        tag[now] = rev[now] = 0;
        if (a[l] >= 0) lmx[now] = rmx[now] = mx[now] = a[l];
        else lmx[now] = rmx[now] = 0, mx[now] = a[l];
    }
    else
    {
        build(l, mid - 1, mid);
        build(mid + 1, r, mid);
    }
    v[now] = a[mid];
    fa[now] = last;
    update(now);
    ch[last][mid >= rt] = now;
}
int find(int x, int rank)
{
    if (tag[x] || rev[x]) pushdown(x);
    int ls = ch[x][0], rs = ch[x][1], lsize = size[ls];
    if (lsize + 1 == rank) return x;
    if (lsize >= rank)
        return find(ls, rank);
    else
        return find(rs, rank - lsize - 1);
}
inline int prepare(int l, int tot)
{
    int x = find(root, l - 1), y = find(root, l + tot);
    splay(x, 0);
    splay(y, x);
    return ch[y][0];
}
std::queue <int> q;
inline void Insert(int left, int tot)
{
    for (int i = 1; i <= tot; ++i ) a[i] = FastIn();
    for (int i = 1; i <= tot; ++i )
        if (!q.empty()) id[i] = q.front(), q.pop();
        else id[i] = ++cnt;
    build(1, tot, 0);
    int z = id[(1 + tot) >> 1];
    int x = find(root, left), y = find(root, left + 1);
    splay(x, 0);
    splay(y, x);
    fa[z] = y;
    ch[y][0] = z;
    update(y);
    update(x);
}
void rec(int x)
{
    if (!x) return ;
    int ls = ch[x][0], rs = ch[x][1];
    rec(ls); rec(rs); q.push(x);
    fa[x] = ch[x][0] = ch[x][1] = 0;
    tag[x] = rev[x] = 0;
}
inline void Delete(int l, int tot)
{
    int x = prepare(l, tot), f = fa[x];
    rec(x); ch[f][0] = 0;
    update(f); update(fa[f]);
}
inline void Makesame(int l, int tot, int val)
{
    int x = prepare(l, tot), y = fa[x];
    v[x] = val; tag[x] = 1; sum[x] = size[x] * val;
    if (val >= 0) lmx[x] = rmx[x] = mx[x] = sum[x];
    else lmx[x] = rmx[x] = 0, mx[x] = val;
    update(y); update(fa[y]);
}
inline void Reverse(int l, int tot)
{
    int x = prepare(l, tot), y = fa[x];
    if (!tag[x])
    {
        rev[x] ^= 1;
        swap(ch[x][0], ch[x][1]);
        swap(lmx[x], rmx[x]);
        update(y); update(fa[y]);
    }
}
inline void Query(int l, int tot)
{
    int x = prepare(l, tot);
    printf("%d\n",sum[x] );
}
#define inf ((1 << 30))
int main()
{
    int n = FastIn(),  m = FastIn(), l, tot, val;
    char op, op2;
    mx[0] = a[1] = a[n + 2] = -inf;
    for (int i = 2; i <= n + 1; i++ )
    {
        a[i] = FastIn();
    }
```

```cpp
    for (int i = 1; i <= n + 2; ++i) id[i] = i;
    n += 2; cnt = n; root = (n + 1) >> 1;
    build(1, n, 0);
    for (int i = 1; i <= m; i++ )
    {
        op = getc();
        while (op < 'A' || op > 'Z') op = getc();
        getc(); op2 = getc();getc();getc();getc();getc();
        if (op == 'M' && op2 == 'X')
        {
            printf("%d\n",mx[root] );
        }
        else
        {
            l = FastIn() + 1; tot = FastIn();
            if (op == 'I') Insert(l, tot);
            if (op == 'D') Delete(l, tot);
            if (op == 'M') val = FastIn(), Makesame(l, tot, val);
            if (op == 'R')
                Reverse(l, tot);
            if (op == 'G')
                Query(l, tot);
        }
    }
    return 0;
}
```

5.8 Treap (ct)

```cpp
struct Treap {
    Treap *ls, *rs;
    int size;
    bool rev;
    inline void update()
    {
        size = ls -> size + rs -> size + 1;
    }
    inline void set_rev()
    {
        rev ^= 1;
        std::swap(ls, rs);
    }
    inline void pushdown()
    {
        if (rev)
        {
            ls -> set_rev();
            rs -> set_rev();
            rev = 0;
        }
    }
} mem[maxn], *root, *null = mem;
struct Pair {
    Treap *fir, *sec;
};
Treap *build(R int l, R int r)
{
    if (l > r) return null;
    R int mid = l + r >> 1;
    R Treap *now = mem + mid;
    now -> rev = 0;
    now -> ls = build(l, mid - 1);
    now -> rs = build(mid + 1, r);
    now -> update();

    return now;
}
inline Treap *Find_kth(R Treap *now, R int k)
{
    if (!k) return mem;
    if (now -> ls -> size >= k) return Find_kth(now -> ls, k);
    else if (now -> ls -> size + 1 == k) return now;
    else return Find_kth(now -> rs, k - now -> ls -> size - 1);
}
Treap *merge(R Treap *a, R Treap *b)
{
    if (a == null) return b;
    if (b == null) return a;
    if (rand() % (a -> size + b -> size) < a -> size)
    {
        a -> pushdown();
        a -> rs = merge(a -> rs, b);
        a -> update();
        return a;
    }
    else
    {
        b -> pushdown();
        b -> ls = merge(a, b -> ls);
        b -> update();
        return b;
    }
}
Pair split(R Treap *now, R int k)
{
    if (now == null) return (Pair) {null, null};
    R Pair t = (Pair) {null, null};
    now -> pushdown();
    if (k <= now -> ls -> size)
    {
        t = split(now -> ls, k);
        now -> ls = t.sec;
        now -> update();
        t.sec = now;
    }
    else
    {
        t = split(now -> rs, k - now -> ls -> size - 1);
        now -> rs = t.fir;
        now -> update();
        t.fir = now;
    }
    return t;
}
inline void set_rev(int l, int r)
{
    R Pair x = split(root, l - 1);
    R Pair y = split(x.sec, r - l + 1);
    y.fir -> set_rev();
    root = merge(x.fir, merge(y.fir, y.sec));
}
```

5.9 可持久化平衡树 (ct)

```cpp
char str[maxn];
struct Treap
{
    Treap *ls, *rs;
    char data; int size;
    inline void update()
    {
```

```
        size = ls -> size + rs -> size + 1;
    }
} *root[maxn], mem[maxcnt], *tot = mem, *last = mem,
 ↪ *null = mem;
inline Treap* new_node(char ch)
{
    *++tot = (Treap) {null, null, ch, 1};
    return tot;
}
struct Pair
{
    Treap *fir, *sec;
};
inline Treap *copy(Treap *x)
{
    if (x == null) return null;
    if(x > last) return x;
    *++tot = *x;
    return tot;
}
Pair Split(Treap *x, int k)
{
    if (x == null) return (Pair) {null, null};
    Pair y;
    Treap *nw = copy(x);
    if (nw -> ls -> size >= k)
    {
        y = Split(nw -> ls, k);
        nw -> ls = y.sec;
        nw -> update();
        y.sec = nw;
    }
    else
    {
        y = Split(nw -> rs, k - nw -> ls -> size - 1);
        nw -> rs = y.fir;
        nw -> update();
        y.fir = nw;
    }
    return y;
}
Treap *Merge(Treap *a, Treap *b)
{
    if (a == null) return b;
    if (b == null) return a;
    Treap *nw;
    if (rand() % (a -> size + b -> size) < a -> size)
    {
        nw = copy(a);
        nw -> rs = Merge(nw -> rs, b);
    }
    else
    {
        nw = copy(b);
        nw -> ls = Merge(a, nw -> ls);
    }
    nw -> update();
    return nw;
}
Treap *Build(int l, int r)
{
    if (l > r) return null;
    int mid = l + r >> 1;
    Treap *nw = new_node(str[mid]);
    nw -> ls = Build(l, mid - 1);
    nw -> rs = Build(mid + 1, r);
    nw -> update();
    return nw;
}
int now;
inline void Insert(int k, char ch)
{
    Pair x = Split(root[now], k);
    Treap *nw = new_node(ch);
    root[++now] = Merge(Merge(x.fir, nw), x.sec);
}
inline void Del(int l, int r)
{
    Pair x = Split(root[now], l - 1);
    Pair y = Split(x.sec, r - l + 1);
    root[++now] = Merge(x.fir, y.sec);
}
inline void Copy(int l, int r, int ll)
{
    Pair x = Split(root[now], l - 1);
    Pair y = Split(x.sec, r - l + 1);
    Pair z = Split(root[now], ll);
    Treap *ans = y.fir;
    root[++now] = Merge(Merge(z.fir, ans), z.sec);
}
void Print(Treap *x, int l, int r)
{
    if (!x) return ;
    if (l > r) return;
    int mid = x -> ls -> size + 1;
    if (r < mid)
    {
        Print(x -> ls, l, r);
        return ;
    }
    if (l > mid)
    {
        Print(x -> rs, l - mid, r - mid);
        return ;
    }
    Print(x -> ls, l, mid - 1);
    printf("%c", x -> data );
    Print(x -> rs, 1, r - mid);
}
void Printtree(Treap *x)
{
    if (!x) return;
    Printtree(x -> ls);
    printf("%c", x -> data );
    Printtree(x -> rs);
}
int main()
{
    srand(time(0) + clock());
    null -> ls = null -> rs = null; null -> size = 0;
     ↪ null -> data = 0;
    int n = F();
    gets(str + 1);
    int len = strlen(str + 1);
    root[0] = Build(1, len);
    while (1)
    {
        last = tot;
        char opt = getc();
        while (opt < 'A' || opt > 'Z')
        {
            if (opt == EOF) return 0;
            opt = getc();
        }
        if (opt == 'I')
        {
            int x = F();
```

```
            char ch = getc();
            Insert(x, ch);
        }
        else if (opt == 'D')
        {
            int l = F(), r = F();
            Del(l, r);
        }
        else if (opt == 'C')
        {
            int x = F(), y = F(), z = F();
            Copy(x, y, z);
        }
        else if (opt == 'P')
        {
            int x = F(), y = F(), z = F();
            Print(root[now - x], y, z);
            puts("");
        }
    }
    return 0;
}
```

5.10 CDQ 分治 (ct)

```
struct event
{
    int x, y, id, opt, ans;
} t[maxn], q[maxn];
void cdq(int left, int right)
{
    if (left == right) return ;
    int mid = left + right >> 1;
    cdq(left, mid);
    cdq(mid + 1, right);
    //分成若干个子问题
    ++now;
    for (int i = left, j = mid + 1; j <= right; ++j)
    {
        for (; i <= mid && q[i].x <= q[j].x; ++i)
            if (!q[i].opt)
                add(q[i].y, q[i].ans);
        //考虑前面的修改操作对后面的询问的影响
        if (q[j].opt)
            q[j].ans += query(q[j].y);
    }
    int i, j, k = 0;
    //以下相当于归并排序
    for (i = left, j = mid + 1; i <= mid && j <= right;
    ↪ )
    {
        if (q[i].x <= q[j].x)
            t[k++] = q[i++];
        else
            t[k++] = q[j++];
    }
    for (; i <= mid; )
        t[k++] = q[i++];
    for (; j <= right; )
        t[k++] = q[j++];
    for (int i = 0; i < k; ++i)
        q[left + i] = t[i];
}
```

5.11 Bitset (ct)

```
namespace Game {
#define maxn 300010
#define maxs 30010
uint b1[32][maxs], b2[32][maxs];
int popcnt[256];
inline void set(R uint *s, R int pos)
{
    s[pos >> 5] |= 1u << (pos & 31);
}
inline int popcount(R uint x)
{
    return popcnt[x >> 24 & 255]
         + popcnt[x >> 16 & 255]
         + popcnt[x >> 8  & 255]
         + popcnt[x       & 255];
}
void main() {
    int n, q;
    scanf("%d%d", &n, &q);

    char *s1 = new char[n + 1];
    char *s2 = new char[n + 1];
    scanf("%s%s", s1, s2);

    uint *anss = new uint[q];

    for (R int i = 1; i < 256; ++i) popcnt[i] =
    ↪   popcnt[i >> 1] + (i & 1);

    #define modify(x, _p)\
    {\
        for (R int j = 0; j < 32 && j <= _p; ++j)\
            set(b##x[j], _p - j);\
    }
    for (R int i = 0; i < n; ++i)
        if (s1[i] == '0') modify(1, 3 * i)
        else if (s1[i] == '1') modify(1, 3 * i + 1)
        else modify(1, 3 * i + 2)

    for (R int i = 0; i < n; ++i)
        if (s2[i] == '1') modify(2, 3 * i)
        else if (s2[i] == '2') modify(2, 3 * i + 1)
        else modify(2, 3 * i + 2)

    for (int Q = 0; Q < q; ++Q) {
        R int x, y, l;
        scanf("%d%d%d", &x, &y, &l); x *= 3; y *= 3; l
        ↪   *= 3;
        uint *f1 = b1[x & 31], *f2 = b2[y & 31], ans =
        ↪   0;
        R int i = x >> 5, j = y >> 5, p, lim;
        for (p = 0, lim = l >> 5; p + 8 < lim; p += 8,
        ↪   i += 8, j += 8)
        {
            ans += popcount(f1[i + 0] & f2[j + 0]);
            ans += popcount(f1[i + 1] & f2[j + 1]);
            ans += popcount(f1[i + 2] & f2[j + 2]);
            ans += popcount(f1[i + 3] & f2[j + 3]);
            ans += popcount(f1[i + 4] & f2[j + 4]);
            ans += popcount(f1[i + 5] & f2[j + 5]);
            ans += popcount(f1[i + 6] & f2[j + 6]);
            ans += popcount(f1[i + 7] & f2[j + 7]);
        }
        for (; p < lim; ++p, ++i, ++j) ans +=
        ↪   popcount(f1[i] & f2[j]);
        R uint S = (1u << (l & 31)) - 1;
        ans += popcount(f1[i] & f2[j] & S);
        anss[Q] = ans;
    }

```

```cpp
    output_arr(anss, q * sizeof(uint));
}
}
```

5.12 斜率优化 (ct)

对于斜截式 $y = kx + b$，如果把 k_i 看成斜率，那 dp 时需要最小化截距，把斜截式转化为 $b_i = -k_i x_j + y_j$，就可以把可以转移到这个状态的点看作是二维平面上的点 $(-x_j, y_j)$，问题转化为了在平面上找一个点使得斜率为 k_i 的直线的截距最小。这样的点一定在凸包上，这样的点在凸包上和前一个点的斜率 $\leq k_i$，和后面一个点的斜率 $\geq k_i$。这样就可以在凸包上二分来加速转移。当点的横坐标 x_i 和斜率 k_i 都是单调的，还可以用单调队列来维护凸包。

单调队列

```cpp
int a[maxn], n, l;
ll sum[maxn], f[maxn];
inline ll sqr(ll x) {return x * x;}
#define y(_i) (f[_i] + sqr(sum[_i] + l))
#define x(_i) (2 * sum[_i])
inline double slope(int i, int j)
{
    return (y(i) - y(j)) / (1.0 * (x(i) - x(j)));
}
int q[maxn];
int main()
{
    n = F(), l = F() + 1;
    for (int i = 1; i <= n; ++i) a[i] = F(), sum[i] = sum[i - 1] + a[i];
    for (int i = 1; i <= n; ++i) sum[i] += i;
    f[0] = 0;
/*
    memset(f, 63, sizeof (f));
    for (int i = 1; i <= n; ++i)
    {
        int pos;
        for (int j = 0; j < i; ++j)
        {
            long long tmp = f[j] + sqr(sum[i] - sum[j] - l);
            f[i] > tmp ? f[i] = tmp, pos = j : 0;
        }
    }
*/
    int h = 1, t = 1;
    q[h] = 0;
    for (int i = 1; i <= n; ++i)
    {
        while (h < t && slope(q[h], q[h + 1]) <= sum[i]) ++h;
        f[i] = f[q[h]] + sqr(sum[i] - sum[q[h]] - l);
        while (h < t && slope(q[t - 1], i) < slope(q[t - 1], q[t])) --t;
        q[++t] = i;
    }
    printf("%lld\n", f[n] );
    return 0;
}
```

线段树

```cpp
// NOI 2014 购票
int dep[maxn], fa[maxn], son[maxn], dfn[maxn], timer, pos[maxn], size[maxn], n, top[maxn];
ll d[maxn], p[maxn], q[maxn], l[maxn], f[maxn];
int stcnt;
void dfs1(int x);
void dfs2(int x);
#define P pair<ll, ll>
#define mkp make_pair
#define x first
#define y second
#define inf ~0ULL >> 2
inline double slope(const P &a, const P &b)
{
    return (b.y - a.y) / (double) (b.x - a.x);
}
struct Seg
{
    vector<P> v;
    inline void add(const P &that)
    {
        int top = v.size();
        P *v = this -> v.data() - 1;
        while (top > 1 && slope(v[top - 1], v[top]) > slope(v[top], that)) --top;
        this -> v.erase(this -> v.begin() + top, this -> v.end());
        this -> v.push_back(that);
    }
    inline ll query(ll k)
    {
        if (v.empty()) return inf;
        int l = 0, r = v.size() - 1;
        while (l < r)
        {
            int mid = l + r >> 1;
            if (slope(v[mid], v[mid + 1]) > k) r = mid;
            else l = mid + 1;
        }
        cmin(l, v.size() - 1);
        return v[l].y - v[l].x * k;
    }
} tr[1 << 19];
void Change(int o, int l, int r, int x, P val)
{
    tr[o].add(val);
    if (l == r) return;
    int mid = l + r >> 1;
    if (x <= mid) Change(o << 1, l, mid, x, val);
    else Change(o << 1 | 1, mid + 1, r, x, val);
}
int ql, qr, now, tmp;
ll len;
inline ll Query(int o, int l, int r)
{
    if (ql <= l && r <= qr && d[tmp] - d[pos[r]] > len) return inf;
    if (ql <= l && r <= qr && d[tmp] - d[pos[l]] <= len)
        return tr[o].query(p[now]);
    ll ret = inf, temp;
    int mid = l + r >> 1;
    if (ql <= mid) temp = Query(o << 1, l, mid), cmin(ret, temp);
    if (mid < qr) temp = Query(o << 1 | 1, mid + 1, r), cmin(ret, temp);
    return ret;
}
inline ll calc()
{
    ll ret = inf;
```

```cpp
    ll lx = l[now];
    tmp = now;
    while (lx >= 0 && tmp)
    {
        len = lx;
        ql = dfn[top[tmp]];
        qr = dfn[tmp];
        ll g = Query(1, 1, n);
        cmin(ret, g);
        lx -= d[tmp] - d[fa[top[tmp]]];
        tmp = fa[top[tmp]];
    }
    return ret;
}
int main()
{
    n = F(); int t = F();
    for (int i = 2; i <= n; ++i)
    {
        fa[i] = F(); ll dis = F(); p[i] = F(), q[i] = F(), l[i] = F();
        link(fa[i], i); d[i] = d[fa[i]] + dis;
    }
    dfs1(1);
    dfs2(1);
    Change(1, 1, n, 1, mkp(0, 0));
    for (now = 2; now <= n; ++now)
    {
        f[now] = calc() + q[now] + d[now] * p[now];
        Change(1, 1, n, dfn[now], mkp(d[now], f[now]));
        printf("%lld\n", f[now] );
    }
    return 0;
}
```

5.13 树分块 (ct)

树分块套分块：给定一棵有点权的树，每次询问链上不同点权个数

```cpp
int col[maxn], hash[maxn], hcnt, n, m;
int near[maxn];
bool vis[maxn];
int mark[maxn], mcnt, tcnt[maxn], tans;
int pre[256][maxn];
struct Block {
    int cnt[256];
} mem[maxn], *tot = mem;
inline Block *nw(Block *last, int v)
{
    Block *ret = ++tot;
    memcpy(ret -> cnt, last -> cnt, sizeof (ret -> cnt));
    ++ret -> cnt[v & 255];
    return ret;
}
struct Arr {
    Block *b[256];
    inline int v(int c) {return b[c >> 8] -> cnt[c & 255];}
} c[maxn];
inline Arr cp(Arr last, int v)
{
    Arr ret;
    memcpy(ret.b, last.b, sizeof (ret.b));
    ret.b[v >> 8] = nw(last.b[v >> 8], v);
    return ret;
}
void bfs()
{
    int head = 0, tail = 1; q[1] = 1;
    while (head < tail)
    {
        int now = q[++head]; size[now] = 1; vis[now] = 1; dep[now] = dep[fa[now]] + 1;
        for (Edge *iter = last[now]; iter; iter = iter -> next)
            if (!vis[iter -> to])
                fa[q[++tail] = iter -> to] = now;
    }
    for (int i = n; i; --i)
    {
        int now = q[i];
        size[fa[now]] += size[now];
        size[son[fa[now]]] < size[now] ? son[fa[now]] = now : 0;
    }
    for (int i = 0; i < 256; ++i) c[0].b[i] = mem;
    for (int i = 1; i <= n; ++i)
    {
        int now = q[i];
        c[now] = cp(c[fa[now]], col[now]);
        top[now] = son[fa[now]] == now ? top[fa[now]] : now;
    }
}
inline int getlca(int a, int b) ;
void dfs_init(int x)
{
    vis[x] = 1; ++tcnt[col[x]] == 1 ? ++tans : 0;
    pre[mcnt][x] = tans;
    for (Edge *iter = last[x]; iter; iter = iter -> next)
        if (!vis[iter -> to]) dfs_init(iter -> to);
    --tcnt[col[x]] == 0 ? --tans : 0;
}
int jp[maxn];
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; ++i) scanf("%d", &col[i]), hash[++hcnt] = col[i];
    std::sort(hash + 1, hash + hcnt + 1);
    hcnt = std::unique(hash + 1, hash + hcnt + 1) - hash - 1;
    for (int i = 1; i <= n; ++i) col[i] = std::lower_bound(hash + 1, hash + hcnt + 1, col[i]) - hash;
    for (int i = 1; i < n; ++i)
    {
        int a, b; scanf("%d%d", &a, &b); link(a, b);
    }
    bfs();
    int D = sqrt(n);
    for (int i = 1; i <= n; ++i)
        if (dep[i] % D == 0 && size[i] >= D)
        {
            memset(vis, 0, n + 1);
            mark[i] = ++mcnt;
            dfs_init(i);
        }
    for (int i = 1; i <= n; ++i) near[q[i]] = mark[q[i]] ? q[i] : near[fa[q[i]]];
    int ans = 0;
    memset(vis, 0, n + 1);
    for (; m; --m)
```

```cpp
    {
        int x, y; scanf("%d%d", &x, &y);
        x ^= ans; ans = 0;
        int lca = getlca(x, y);
        if (dep[near[x]] < dep[lca]) std::swap(x, y);
        if (dep[near[x]] >= dep[lca])
        {
            Arr *_a = c + near[x];
            Arr *_b = c + y;
            Arr *_c = c + lca;
            Arr *_d = c + fa[lca];
            for (; !mark[x]; x = fa[x])
                if (_a -> v(col[x]) + _b -> v(col[x]) == _c -> v(col[x]) + _d -> v(col[x]) && !vis[col[x]])
                    vis[jp[++ans] = col[x]] = 1;
            for (int i = 1; i <= ans; ++i) vis[jp[i]] = 0;
            ans += pre[mark[near[x]]][y];
        }
        else
        {
            for (; x != lca; x = fa[x]) !vis[col[x]] ? vis[jp[++ans] = col[x]] = 1 : 0;
            for (; y != lca; y = fa[y]) !vis[col[y]] ? vis[jp[++ans] = col[y]] = 1 : 0;
            !vis[col[lca]] ? vis[jp[++ans] = col[lca]] = 1 : 0;
            for (int i = 1; i <= ans; ++i) vis[jp[i]] = 0;
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

5.14 KD tree (lhy)

```cpp
inline int cmp(const lhy &a,const lhy &b)
{
    return a.d[D]<b.d[D];
}

inline void updata(int x)
{
    if(p[x].l)
    {
        for(int i=0;i<2;i++)
            p[x].min[i]=min(p[x].min[i],p[p[x].l].min[i]),
            p[x].max[i]=max(p[x].max[i],p[p[x].l].max[i]);
    }
    if(p[x].r)
    {
        for(int i=0;i<2;i++)
            p[x].min[i]=min(p[x].min[i],p[p[x].r].min[i]),
            p[x].max[i]=max(p[x].max[i],p[p[x].r].max[i]);
    }
}

int build(int l,int r,int d)
{
    D=d;
    int mid=(l+r)>>1;
    nth_element(p+l,p+mid,p+r+1,cmp);
    for(int i=0;i<2;i++)
        p[mid].max[i]=p[mid].min[i]=p[mid].d[i];
    if(l<mid)p[mid].l=build(l,mid-1,d^1);
    if(mid<r)p[mid].r=build(mid+1,r,d^1);
    updata(mid);
    return mid;
}

void insert(int now,int D)
{
    if(p[now].d[D]>=p[n].d[D])
    {
        if(p[now].l)insert(p[now].l,D^1);
        else p[now].l=n;
        updata(now);
    }
    else
    {
        if(p[now].r)insert(p[now].r,D^1);
        else p[now].r=n;
        updata(now);
    }
}

int dist(lhy &P,int X,int Y)
{
    int nowans=0;
    if(X>=P.max[0])nowans+=X-P.max[0];
    if(X<=P.min[0])nowans+=P.min[0]-X;
    if(Y>=P.max[1])nowans+=Y-P.max[1];
    if(Y<=P.min[1])nowans+=P.min[1]-Y;
    return nowans;
}

void ask1(int now)
{
    int pl,pr;
    ans=min(ans,abs(x-p[now].d[0])+abs(y-p[now].d[1]));
    if(p[now].l)pl=dist(p[p[now].l],x,y);
    else pl=0x3f3f3f3f;
    if(p[now].r)pr=dist(p[p[now].r],x,y);
    else pr=0x3f3f3f3f;
    if(pl<pr)
    {
        if(pl<ans)ask(p[now].l);
        if(pr<ans)ask(p[now].r);
    }
    else
    {
        if(pr<ans)ask(p[now].r);
        if(pl<ans)ask(p[now].l);
    }
}

void ask2(int now)
{
    if(x1<=p[now].min[0]&&x2>=p[now].max[0]&&y1<=p[now].min[1]&&y2>=p[now].max[1])
    {
        ans+=p[now].sum;
        return;
    }
    if(x1>p[now].max[0]||x2<p[now].min[0]||y1>p[now].max[1]||y2<p[now].min[1])return;
    if(x1<=p[now].d[0]&&x2>=p[now].d[0]&&y1<=p[now].d[1]&&y2>=p[now].d[1])ans+=p[now].val;
```

```
    if(p[now].l)ask(p[now].l);
    if(p[now].r)ask(p[now].r);
}
```

5.15 DLX (Nightfall)

```
struct node{
    node *left,*right,*up,*down,*col; int row,cnt;
}*head,*col[MAXC],Node[MAXNODE],*ans[MAXNODE];
int totNode, ansNode;
void insert(const std::vector<int> &V,int rownum){
    std::vector<node*> N;
    for(int i=0;i<int(V.size());++i){
        node* now=Node+(totNode++); now->row=rownum;
        now->col=now->up=col[V[i]], now->down=col[V[i]]->down;
        now->up->down=now, now->down->up=now;
        now->col->cnt++; N.push_back(now); }
    for(int i=0;i<int(V.size());++i)
        N[i]->right=N[(i+1)%V.size()], N[i]->left=N[(i-1+V.size())%V.size()];
}
void Remove(node *x){
    x->left->right=x->right, x->right->left=x->left;
    for(node *i=x->down;i!=x;i=i->down)
        for(node *j=i->right;j!=i;j=j->right)
            j->up->down=j->down, j->down->up=j->up, --(j->col->cnt);
}
void Resume(node *x){
    for(node *i=x->up;i!=x;i=i->up)
        for(node *j=i->left;j!=i;j=j->left)
            j->up->down=j->down->up=j, ++(j->col->cnt);
    x->left->right=x, x->right->left=x;
}
bool search(int tot){
    if(head->right==head) return ansNode = tot, true;
    node *choose=NULL;
    for(node *i=head->right;i!=head;i=i->right){
        if(choose==NULL||choose->cnt>i->cnt) choose=i;
        if(choose->cnt<2) break; }
    Remove(choose);
    for(node *i=choose->down;i!=choose;i=i->down){
        for(node *j=i->right;j!=i;j=j->right) Remove(j->col);
        ans[tot]=i;
        if(search(tot+1)) return true;
        ans[tot]=NULL;
        for(node *j=i->left;j!=i;j=j->left) Resume(j->col); }
    Resume(choose); return false;
}
void prepare(int totC){
    head=Node+totC;
    for(int i=0;i<totC;++i) col[i]=Node+i;
    totNode=totC+1; ansNode = 0;
    for(int i=0;i<=totC;++i){
        (Node+i)->right=Node+(i+1)%(totC+1);
        (Node+i)->left=Node+(i+totC)%(totC+1);
        (Node+i)->up=(Node+i)->down=Node+i;
        (Node+i)->cnt=0; }
}
prepare(C); for (i (rows)) insert({col_id}, C); search(0);
```

6. Others

6.1 vimrc (gy)

```
se et ts=4 sw=4 sts=4 nu sc sm lbr is hls mouse=a
sy on
ino <tab> <c-n>
ino <s-tab> <tab>
au bufwinenter * winc L

nm <f6> ggVG"+y
nm <f7> :w<cr>:!rm ##<cr>:make<cr>
nm <f8> :!@@<cr>
nm <f9> :!@@ < in<cr>
nm <s-f9> :!(time @@ < in &> out) &>> out<cr>:sp out<cr>

au filetype cpp cm @@ ./a.out | cm ## a.out | se cin fdm=syntax mp=g++\ %\ -std=c++11\ -Wall\ -Wextra\ -Wconversion\ -O2

map <c-p> :ha<cr>
se pheader=%N@%F

au filetype java cm @@ java %< | cm ## %<.class | se cin fdm=syntax mp=javac\ %
au filetype python cm @@ python % | se si fdm=indent
au bufenter *.kt setf kotlin
au filetype kotlin cm @@ kotlin _%<Kt | cm ## _%<Kt.class | se si mp=kotlinc\ %
```

6.2 STL 释放内存 (Durandal)

```
template <typename T>
__inline void clear(T &container) {
    container.clear();
    T(container).swap(container);
}
```

6.3 开栈 (Durandal)

```
register char *_sp __asm__("rsp");
int main() {
    const int size = 400 << 20; // 400 MB
    static char *sys, *mine(new char[size] + size - 4096);
    sys = _sp; _sp = mine;
    _main(); // main method
    _sp = sys;
    return 0;
}
```

6.4 O3 (gy)

```
__attribute__((optimize("-O3"))) int main() { return 0; }
```

6.5 读入优化 (ct)

```
char S[1 << 20], *T = S;
inline int F()
{
    char ch; int cnt = 0;
    while (ch = *T++, ch < '0' || ch > '9') ;
    cnt = ch - '0';
    while (ch = *T++, ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
    return cnt;
}
fread(S, 1, 1 << 20, stdin);
```

6.6 Java Template (gy)

```
import java.io.*;
import java.math.*;
import java.util.*;

public class Template {
    // Input
    private static BufferedReader reader;
    private static StringTokenizer tokenizer;

    private static String next() {
        try {
            while (tokenizer == null || !tokenizer.hasMoreTokens())
                tokenizer = new StringTokenizer(reader.readLine());
        } catch (IOException e) {
            // do nothing
        }
        return tokenizer.nextToken();
    }

    private static int nextInt() {
        return Integer.parseInt(next());
    }

    private static double nextDouble() {
        return Double.parseDouble(next());
    }

    private static BigInteger nextBigInteger() {
        return new BigInteger(next());
    }

    public static void main(String[] args) {
        reader = new BufferedReader(new InputStreamReader(System.in));
        Scanner scanner = new Scanner(System.in);
        while (scanner.hasNext())
            scanner.next();
    }

    // BigInteger & BigDecimal
    private static void bigDecimal() {
        BigDecimal a = BigDecimal.valueOf(1.0);
        BigDecimal b = a.setScale(50, RoundingMode.HALF_EVEN);
        BigDecimal c = b.abs();
        // if scale omitted, b.scale is used
        BigDecimal d = c.divide(b, 50, RoundingMode.HALF_EVEN);
        // since Java 9
        BigDecimal e = d.sqrt(new MathContext(50, RoundingMode.HALF_EVEN));
        BigDecimal x = new BigDecimal(BigInteger.ZERO);
        BigInteger y = BigDecimal.ZERO.toBigInteger(); // RoundingMode.DOWN
        y = BigDecimal.ZERO.setScale(0, RoundingMode.HALF_EVEN).unscaledValue();
    }

    // sqrt for Java 8
    // can solve scale=100 for 10000 times in about 1 second
    private static BigDecimal sqrt(BigDecimal a, int scale) {
        if (a.compareTo(BigDecimal.ZERO) < 0)
            return BigDecimal.ZERO.setScale(scale, RoundingMode.HALF_EVEN);
        int length = a.precision() - a.scale();
        BigDecimal ret = new BigDecimal(BigInteger.ONE, -length / 2);
        for (int i = 1; i <= Integer.highestOneBit(scale) + 10; i++)
            ret = ret.add(a.divide(ret, scale, RoundingMode.HALF_EVEN)).divide(BigDecimal.valueOf(2), scale, RoundingMode.HALF_EVEN);
        return ret;
    }

    // can solve a=2^10000 for 100000 times in about 1 second
    private static BigInteger sqrt(BigInteger a) {
        int length = a.bitLength() - 1;
        BigInteger l = BigInteger.ZERO.setBit(length / 2), r = BigInteger.ZERO.setBit(length / 2);
        while (!l.equals(r)) {
            BigInteger m = l.add(r).shiftRight(1);
            if (m.multiply(m).compareTo(a) < 0)
                l = m.add(BigInteger.ONE);
            else
                r = m;
        }
        return l;
    }

    // Collections
    private static void arrayList() {
        List<Integer> list = new ArrayList<>();
        // Generic array is banned
        List[] lists = new List[100];
        lists[0] = new ArrayList<Integer>();
        // for List<Integer>, remove(Integer) stands for element, while remove(int) stands for index
        list.remove(list.get(1));
        list.remove(list.size() - 1);
        list.clear();
        Queue<Integer> queue = new LinkedList<>();
        // return the value without popping
        queue.peek();
        // pop and return the value
        queue.poll();
        Queue<Integer> priorityQueue = new PriorityQueue<>();
        Deque<Integer> deque = new ArrayDeque<>();
        deque.peekFirst();
        deque.peekLast();
```

```java
        deque.pollFirst();
        TreeSet<Integer> set = new TreeSet<>();
        TreeSet<Integer> anotherSet = new TreeSet<>(Comparator.reverseOrder());
        set.ceiling(1);
        set.floor(1);
        set.lower(1);
        set.higher(1);
        set.contains(1);
        HashSet<Integer> hashSet = new HashSet<>();
        HashMap<String, Integer> map = new HashMap<>();
        map.put("", 1);
        map.get("");
        map.forEach((string, integer) -> System.out.println(string + integer));
        TreeMap<String, Integer> treeMap = new TreeMap<>();
        Arrays.sort(new int[10]);
        Arrays.sort(new Integer[10], (a, b) -> {
            if (a.equals(b)) return 0;
            if (a > b) return -1;
            return 1;
        });
        Arrays.sort(new Integer[10], Comparator.comparingInt((a) -> (int) a).reversed());
        long a = 1_000_000_000_000_000_000L;
        int b = Integer.MAX_VALUE;
        int c = 'a';
    }

    private static class BigFraction {
        private BigInteger a, b;

        BigFraction(BigInteger a, BigInteger b) {
            BigInteger gcd = a.gcd(b);
            this.a = a.divide(gcd);
            this.b = b.divide(gcd);
        }

        BigFraction add(BigFraction o) {
            BigInteger gcd = b.gcd(o.b);
            BigInteger tempProduct = b.divide(gcd).multiply(o.b.divide(gcd));
            BigInteger ansA = a.multiply(o.b.divide(gcd)).add(o.a.multiply(b.divide(gcd)));
            BigInteger gcd2 = ansA.gcd(gcd);
            ansA = ansA.divide(gcd2);
            gcd2 = gcd.divide(gcd2);
            return new BigFraction(ansA, gcd2.multiply(tempProduct));
        }

        BigFraction subtract(BigFraction o) {
            BigInteger gcd = b.gcd(o.b);
            BigInteger tempProduct = b.divide(gcd).multiply(o.b.divide(gcd));
            BigInteger ansA = a.multiply(o.b.divide(gcd)).subtract(o.a.multiply(b.divide(gcd)));
            BigInteger gcd2 = ansA.gcd(gcd);
            ansA = ansA.divide(gcd2);
            gcd = gcd.divide(gcd2);
            return new BigFraction(ansA, gcd2.multiply(tempProduct));
        }

        BigFraction multiply(BigFraction o) {
            BigInteger gcd1 = a.gcd(o.b);
            BigInteger gcd2 = b.gcd(o.a);
            return new BigFraction(a.divide(gcd1).multiply(o.a.divide(gcd2)), b.divide(gcd2).multiply(o.b.divide(gcd1)));
        }

        @Override
        public String toString() {
            return a + "/" + b;
        }
    }
}
```

6.7 模拟退火 (ct)

```cpp
db ans_x, fans;
inline double rand01() {return rand() / 2147483647.0;}
inline double randp() {return (rand() & 1 ? 1 : -1) * rand01();}
inline double f(double x)
{
    /*
        write your function here.
    */
    if (maxx < fans) {fans = maxx; ans_x = x;}
    return maxx;
}
int main()
{
    srand(time(NULL) + clock());
    db x = 0, fnow = f(x);
    fans = 1e30;
    for (db T = 1e4; T > 1e-4; T *= 0.997)
    {
        db nx = x + randp() * T, fnext = f(nx);
        db delta = fnext - fnow;
        if (delta < 1e-9 || exp(-delta / T) > rand01())
        {
            x = nx;
            fnow = fnext;
        }
    }
    return 0;
}
```

6.8 Simpson 积分 (gy)

```cpp
number f(number x) {
    return /* Take circle area as example */ std::sqrt(1 - x * x) * 2;
}
number simpson(number a, number b) {
    number c = (a + b) / 2;
    return (f(a) + f(b) + 4 * f(c)) * (b - a) / 6;
}
number integral(number a, number b, number eps) {
    number c = (a + b) / 2;
    number mid = simpson(a, b), l = simpson(a, c), r = simpson(c, b);
    if (std::abs(l + r - mid) <= 15 * eps)
        return l + r + (l + r - mid) / 15;
    else
```

```
        return integral(a, c, eps / 2) + integral(c, b,
        ↪  eps / 2);
}
```

6.9 Zeller Congruence (gy)

```
int day_in_week(int year, int month, int day) {
    if (month == 1 || month == 2)
        month += 12, year--;
    int c = year / 100, y = year % 100, m = month, d =
    ↪  day;
    int ret = (y + y / 4 + c / 4 + 5 * c + 13 * (m +
    ↪  1) / 5 + d + 6) % 7;
    return ret >= 0 ? ret : ret + 7;
}
```

6.10 博弈论模型 (gy)

- Wythoff's game

给定两堆石子，每次可以从任意一堆中取至少一个石子，或从两堆中取相同的至少一个石子，取走最后石子的胜
先手胜当且仅当石子数满足：
$\lfloor (b-a) \times \phi \rfloor = a, (a \le b, \phi = \frac{\sqrt{5}+1}{2})$
先手胜对应的石子数构成两个序列：
Lower Wythoff sequence: $a_n = \lfloor n \times \phi \rfloor$
Upper Wythoff sequence: $b_n = \lfloor n \times \phi^2 \rfloor$

- Fibonacci nim

给定一堆石子，第一次可以取至少一个、少于石子总数数量的石子，之后每次可以取至少一个、不超过上次取石子数量两倍的石子，取走最后石子的胜
先手胜当且仅当石子数为斐波那契数

- anti-SG

决策集合为空的游戏者胜
先手胜当且仅当满足以下任一条件

– 所有单一游戏的 SG 值都 < 2 且游戏的 SG 值为 0
– 至少有一个单一游戏的 SG 值 ≥ 2 且游戏的 SG 值不为 0

6.11 积分表 (integral-table.com)

$$\int x^n dx = \frac{1}{n+1}x^{n+1}, \ n \ne -1$$

$$\int \frac{1}{x} dx = \ln|x|$$

$$\int u dv = uv - \int v du$$

$$\int \frac{1}{ax+b} dx = \frac{1}{a}\ln|ax+b|$$

$$\int \frac{1}{(x+a)^2} dx = -\frac{1}{x+a}$$

$$\int (x+a)^n dx = \frac{(x+a)^{n+1}}{n+1}, n \ne -1$$

$$\int x(x+a)^n dx = \frac{(x+a)^{n+1}((n+1)x-a)}{(n+1)(n+2)}$$

$$\int \frac{1}{1+x^2} dx = \tan^{-1} x$$

$$\int \frac{1}{a^2+x^2} dx = \frac{1}{a}\tan^{-1}\frac{x}{a}$$

$$\int \frac{x}{a^2+x^2} dx = \frac{1}{2}\ln|a^2+x^2|$$

$$\int \frac{x^2}{a^2+x^2} dx = x - a\tan^{-1}\frac{x}{a}$$

$$\int \frac{x^3}{a^2+x^2} dx = \frac{1}{2}x^2 - \frac{1}{2}a^2\ln|a^2+x^2|$$

$$\int \frac{1}{ax^2+bx+c} dx = \frac{2}{\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}}$$

$$\int \frac{1}{(x+a)(x+b)} dx = \frac{1}{b-a}\ln\frac{a+x}{b+x}, \ a \ne b$$

$$\int \frac{x}{(x+a)^2} dx = \frac{a}{a+x} + \ln|a+x|$$

$$\int \frac{x}{ax^2+bx+c} dx = \frac{1}{2a}\ln|ax^2+bx+c| - \frac{b}{a\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}}$$

$$\int \sqrt{x-a}\ dx = \frac{2}{3}(x-a)^{3/2}$$

$$\int \frac{1}{\sqrt{x \pm a}}\ dx = 2\sqrt{x \pm a}$$

$$\int \frac{1}{\sqrt{a-x}}\ dx = -2\sqrt{a-x}$$

$$\int x\sqrt{x-a}\ dx = \begin{cases} \frac{2a}{3}(x-a)^{3/2} + \frac{2}{5}(x-a)^{5/2}, & \text{or} \\ \frac{2}{3}x(x-a)^{3/2} - \frac{4}{15}(x-a)^{5/2}, & \text{or} \\ \frac{2}{15}(2a+3x)(x-a)^{3/2} \end{cases}$$

$$\int \sqrt{ax+b}\ dx = \left(\frac{2b}{3a} + \frac{2x}{3}\right)\sqrt{ax+b}$$

$$\int (ax+b)^{3/2}\ dx = \frac{2}{5a}(ax+b)^{5/2}$$

$$\int \frac{x}{\sqrt{x \pm a}}\ dx = \frac{2}{3}(x \mp 2a)\sqrt{x \pm a}$$

$$\int \sqrt{\frac{x}{a-x}}\ dx = -\sqrt{x(a-x)} - a\tan^{-1}\frac{\sqrt{x(a-x)}}{x-a}$$

$$\int \sqrt{\frac{x}{a+x}}\ dx = \sqrt{x(a+x)} - a\ln\left(\sqrt{x} + \sqrt{x+a}\right)$$

$$\int x\sqrt{ax+b}\ dx = \frac{2}{15a^2}(-2b^2 + abx + 3a^2x^2)\sqrt{ax+b}$$

$$\int \sqrt{x(ax+b)}\ dx = \frac{1}{4a^{3/2}}\left((2ax+b)\sqrt{ax(ax+b)} - b^2\ln\left|a\sqrt{x} + \sqrt{a(ax+b)}\right|\right)$$

$$\int \sqrt{x^3(ax+b)}\ dx = \left(\frac{b}{12a} - \frac{b^2}{8a^2x} + \frac{x}{3}\right)\sqrt{x^3(ax+b)} + \frac{b^3}{8a^{5/2}}\ln\left|a\sqrt{x} + \sqrt{a(ax+b)}\right|$$

$$\int \sqrt{x^2 \pm a^2}\ dx = \frac{1}{2}x\sqrt{x^2 \pm a^2} \pm \frac{1}{2}a^2\ln\left|x + \sqrt{x^2 \pm a^2}\right|$$

$$\int \sqrt{a^2 - x^2}\ dx = \frac{1}{2}x\sqrt{a^2 - x^2} + \frac{1}{2}a^2\tan^{-1}\frac{x}{\sqrt{a^2 - x^2}}$$

$$\int x\sqrt{x^2 \pm a^2}\ dx = \frac{1}{3}\left(x^2 \pm a^2\right)^{3/2}$$

$$\int \frac{1}{\sqrt{x^2 \pm a^2}}\ dx = \ln\left|x + \sqrt{x^2 \pm a^2}\right|$$

$$\int \frac{1}{\sqrt{a^2 - x^2}}\ dx = \sin^{-1}\frac{x}{a}$$

$$\int \frac{x}{\sqrt{x^2 \pm a^2}}\ dx = \sqrt{x^2 \pm a^2}$$

$$\int \frac{x}{\sqrt{a^2 - x^2}}\ dx = -\sqrt{a^2 - x^2}$$

$$\int \frac{x^2}{\sqrt{x^2 \pm a^2}}\ dx = \frac{1}{2}x\sqrt{x^2 \pm a^2} \mp \frac{1}{2}a^2\ln\left|x + \sqrt{x^2 \pm a^2}\right|$$

$$\int \sqrt{ax^2+bx+c}\ dx = \frac{b+2ax}{4a}\sqrt{ax^2+bx+c} + \frac{4ac-b^2}{8a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right|$$

$$\int x\sqrt{ax^2+bx+c}\ dx = \frac{1}{48a^{5/2}}\left(2\sqrt{a}\sqrt{ax^2+bx+c}\left(-3b^2+2abx+8a(c+ax^2)\right) + 3(b^3-4abc)\ln\left|b+2ax+2\sqrt{a}\sqrt{ax^2+bx+c}\right|\right)$$

$$\int \frac{1}{\sqrt{ax^2+bx+c}}\ dx = \frac{1}{\sqrt{a}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right|$$

$$\int \frac{x}{\sqrt{ax^2+bx+c}}\,dx =$$
$$\frac{1}{a}\sqrt{ax^2+bx+c} - \frac{b}{2a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right|$$

$$\int \frac{dx}{(a^2+x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2+x^2}}$$

$$\int \sin ax\ dx = -\frac{1}{a}\cos ax$$

$$\int \sin^2 ax\ dx = \frac{x}{2} - \frac{\sin 2ax}{4a}$$

$$\int \sin^3 ax\ dx = -\frac{3\cos ax}{4a} + \frac{\cos 3ax}{12a}$$

$$\int \cos ax\ dx = \frac{1}{a}\sin ax$$

$$\int \cos^2 ax\ dx = \frac{x}{2} + \frac{\sin 2ax}{4a}$$

$$\int \cos^3 ax dx = \frac{3\sin ax}{4a} + \frac{\sin 3ax}{12a}$$

$$\int \cos x \sin x\ dx = \frac{1}{2}\sin^2 x + c_1 = -\frac{1}{2}\cos^2 x + c_2 = -\frac{1}{4}\cos 2x + c_3$$

$$\int \cos ax \sin bx\ dx = \frac{\cos[(a-b)x]}{2(a-b)} - \frac{\cos[(a+b)x]}{2(a+b)},\ a \neq b$$

$$\int \sin^2 ax \cos bx\ dx = -\frac{\sin[(2a-b)x]}{4(2a-b)} + \frac{\sin bx}{2b} - \frac{\sin[(2a+b)x]}{4(2a+b)}$$

$$\int \sin^2 x \cos x\ dx = \frac{1}{3}\sin^3 x$$

$$\int \cos^2 ax \sin bx\ dx = \frac{\cos[(2a-b)x]}{4(2a-b)} - \frac{\cos bx}{2b} - \frac{\cos[(2a+b)x]}{4(2a+b)}$$

$$\int \cos^2 ax \sin ax\ dx = -\frac{1}{3a}\cos^3 ax$$

$$\int \sin^2 ax \cos^2 bx dx =$$
$$\frac{x}{4} - \frac{\sin 2ax}{8a} - \frac{\sin[2(a-b)x]}{16(a-b)} + \frac{\sin 2bx}{8b} - \frac{\sin[2(a+b)x]}{16(a+b)}$$

$$\int \sin^2 ax \cos^2 ax\ dx = \frac{x}{8} - \frac{\sin 4ax}{32a}$$

$$\int \tan ax\ dx = -\frac{1}{a}\ln\cos ax$$

$$\int \tan^2 ax\ dx = -x + \frac{1}{a}\tan ax$$

$$\int \tan^3 ax dx = \frac{1}{a}\ln\cos ax + \frac{1}{2a}\sec^2 ax$$

$$\int \sec x\ dx = \ln|\sec x + \tan x| = 2\tanh^{-1}\left(\tan\frac{x}{2}\right)$$

$$\int \sec^2 ax\ dx = \frac{1}{a}\tan ax$$

$$\int \sec^3 x\ dx = \frac{1}{2}\sec x\tan x + \frac{1}{2}\ln|\sec x + \tan x|$$

$$\int \sec x \tan x\ dx = \sec x$$

$$\int \sec^2 x \tan x\ dx = \frac{1}{2}\sec^2 x$$

$$\int \sec^n x \tan x\ dx = \frac{1}{n}\sec^n x,\ n \neq 0$$

$$\int \csc x\ dx = \ln\left|\tan\frac{x}{2}\right| = \ln|\csc x - \cot x| + C$$

$$\int \csc^2 ax\ dx = -\frac{1}{a}\cot ax$$

$$\int \csc^3 x\ dx = -\frac{1}{2}\cot x\csc x + \frac{1}{2}\ln|\csc x - \cot x|$$

$$\int \csc^n x \cot x\ dx = -\frac{1}{n}\csc^n x,\ n \neq 0$$

$$\int \sec x \csc x\ dx = \ln|\tan x|$$

$$\int x\cos x\ dx = \cos x + x\sin x$$

$$\int x\cos ax\ dx = \frac{1}{a^2}\cos ax + \frac{x}{a}\sin ax$$

$$\int x^2\cos x\ dx = 2x\cos x + \left(x^2-2\right)\sin x$$

$$\int x^2\cos ax\ dx = \frac{2x\cos ax}{a^2} + \frac{a^2x^2-2}{a^3}\sin ax$$

$$\int x\sin x\ dx = -x\cos x + \sin x$$

$$\int x\sin ax\ dx = -\frac{x\cos ax}{a} + \frac{\sin ax}{a^2}$$

$$\int x^2\sin x\ dx = \left(2-x^2\right)\cos x + 2x\sin x$$

$$\int x^2\sin ax\ dx = \frac{2-a^2x^2}{a^3}\cos ax + \frac{2x\sin ax}{a^2}$$

$$\int x\cos^2 x\ dx = \frac{x^2}{4} + \frac{1}{8}\cos 2x + \frac{1}{4}x\sin 2x$$

$$\int x\sin^2 x\ dx = \frac{x^2}{4} - \frac{1}{8}\cos 2x - \frac{1}{4}x\sin 2x$$