Contents

输入 & 输出

特殊格式

```
long double %Lf
unsigned int %u
unsigned long long %llu

cout << fixed << setprecision(15);
```

文件和流同步

```
freopen("in.txt", "r", stdin);

ios::sync_with_stdio(false);
cin.tie(0);
```

程序计时

```
(double)clock() / CLOCKS_PER_SEC
```

整行读入

```
scanf("%[^\n]", s)  // 需测试是否可用
getline(cin, s)
```

读到文件尾

```
while (cin) {}
while (~scanf) {}
```

int128

```
// 需测试是否可用
inline __int128 get128() {
    __int128 x = 0, sgn = 1;
    char c = getchar();
    for (; c < '0' || c > '9'; c = getchar()) if (c == '-') sgn = -1;
    for (; c >= '0' && c <= '9'; c = getchar()) x = x * 10 + c - '0';
    return sgn * x;
}

inline void print128(__int128 x) {
    if (x < 0) {
        putchar('-');
        x = -x;
    }
    if (x >= 10) print128(x / 10);
    putchar(x % 10 + '0');
}
```

读入挂

```
class Scanner {
#ifdef qdd
    static constexpr int BUF_SIZE = 1;
#else
    static constexpr int BUF_SIZE = 1048576; // 1MB
#endif

    char buf[BUF_SIZE], *p1 = buf, *p2 = buf;

    char nc() {
        if (p1 == p2) {
            p1 = buf; p2 = buf + fread(buf, 1, BUF_SIZE, stdin);
            // assert(p1 != p2);
        }
        return *p1++;
    }

public:
    int nextInt() {
        int x = 0, sgn = 1;
        char c = nc();
        for (; c < '0' || c > '9'; c = nc()) if (c == '-') sgn = -1;
        for (; c >= '0' && c <= '9'; c = nc()) x = x * 10 + (c - '0');
        return sgn * x;
    }

    double nextDouble() {
        double x = 0, base = 0.1;
        int sgn = 1;
        char c = nc();
```

```
        for (; c < '0' || c > '9'; c = nc()) if (c == '-') sgn = -1;
        for (; c >= '0' && c <= '9'; c = nc()) x = x * 10 + (c - '0');
        for (; c < '0' || c > '9'; c = nc()) if (c != '.') return sgn * x;
        for (; c >= '0' && c <= '9'; c = nc()) x += base * (c - '0'), base *= 0.1;
        return sgn * x;
    }
} in;
```

数据结构

并查集

```
int find(int x) { return (x == pa[x]) ? x : pa[x] = find(pa[x]); }
void merge(int a, int b) { pa[find(a)] = find(b); }
```

- 动态开点并查集

```
// pa 为负数表示集合大小
unordered_map<int, int> pa;

void _set(int x) { if (!pa.count(x)) pa[x] = -1; }
int find(int x) { return (pa[x] < 0) ? x : pa[x] = find(pa[x]); }

void merge(int a, int b) {
    int x = find(a), y = find(b);
    if (x == y) return;
    if (pa[x] > pa[y]) swap(x, y);
    pa[x] += pa[y];
    pa[y] = x;
}
```

RMQ

- 一维

```
// 下标从 0 开始
struct RMQ {
    int st[MAXN][22]; // 22 = ((int)log2(MAXN) + 1)

    int xlog(int x) { return 31 - __builtin_clz(x); }

    void init(int *a, int n) {
        for (int i = 0; i < n; i++) {
            st[i][0] = a[i];
        }
        for (int j = 1; (1 << j) <= n; j++) {
            for (int i = 0; i + (1 << j) - 1 < n; i++) {
                st[i][j] = max(st[i][j - 1], st[i + (1 << (j - 1))][j - 1]);
            }
        }
    }

    int query(int l, int r) {
        int x = xlog(r - l + 1);
        return max(st[l][x], st[r - (1 << x) + 1][x]);
    }
};
```

- 二维

```
struct RMQ {
    int st[11][11][MAXN][MAXN]; // 11 = ((int)log2(MAXN) + 1)

    int xlog(int x) { return 31 - __builtin_clz(x); }
```

```
    void init(int n, int m) {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                st[0][0][i][j] = a[i][j];
            }
        }
        for (int i = 0; (1 << i) <= n; i++) {
            for (int j = 0; (1 << j) <= m; j++) {
                if (i == 0 && j == 0) continue;
                for (int r = 0; r + (1 << i) - 1 < n; r++) {
                    for (int c = 0; c + (1 << j) - 1 < m; c++) {
                        if (i == 0) {
                            st[i][j][r][c] = max(st[i][j - 1][r][c], st[i][j - 1][r][c + (1 << (j - 1))]);
                        } else {
                            st[i][j][r][c] = max(st[i - 1][j][r][c], st[i - 1][j][r + (1 << (i - 1))][c]);
                        }
                    }
                }
            }
        }
    }

    int query(int r1, int c1, int r2, int c2) {
        int x = xlog(r2 - r1 + 1);
        int y = xlog(c2 - c1 + 1);
        int m1 = st[x][y][r1][c1];
        int m2 = st[x][y][r1][c2 - (1 << y) + 1];
        int m3 = st[x][y][r2 - (1 << x) + 1][c1];
        int m4 = st[x][y][r2 - (1 << x) + 1][c2 - (1 << y) + 1];
        return max({m1, m2, m3, m4});
    }
};
```

- 滑动窗口 RMQ

```
// k 为滑动窗口的大小
deque<int> q;
for (int i = 0, j = 0; i + k <= n; i++) {
    while (j < i + k) {
        while (!q.empty() && a[q.back()] < a[j]) q.pop_back(); // 最小值取'>'号
        q.push_back(j++);
    }
    while (q.front() < i) q.pop_front();
    rmq.push_back(a[q.front()]);
}
```

树状数组

- 单点修改，区间和

```
// 支持第 k 大的 BIT
// 下标从 1 开始
struct Tbit {
    int size;
    ll t[MAXN];

    int lowbit(int x) { return x & (-x); }

    void init(int sz) {
        size = sz + 1;
        memset(t, 0, (sz + 2) * sizeof(ll));
    }

    void add(int p, ll x) {
        if (p <= 0) return;
```

```
        for (; p <= size; p += lowbit(p)) t[p] += x;
    }

    ll get(int p) {
        ll sum = 0;
        for (; p > 0; p -= lowbit(p)) sum += t[p];
        return sum;
    }

    void update(int p, ll x) { add(p, x - query(p, p)); }
    ll query(int l, int r) { return get(r) - get(l - 1); }

    int kth(ll k) {
        int p = 0;
        for (int i = 20; i >= 0; i--) {
            int p_ = p + (1 << i);
            if (p_ <= size && t[p_] < k) {
                k -= t[p_];
                p = p_;
            }
        }
        return p + 1;
    }
};
```

- 区间加，单点查询

```
void range_add(int l, int r, ll x) {
    add(l, x);
    add(r + 1, -x);
}
```

- 区间加，区间和

```
Tbit t1, t2;

void range_add(int l, int r, ll x) {
    t1.add(l, x);
    t2.add(l, l * x);
    t1.add(r + 1, -x);
    t2.add(r + 1, (r + 1) * -x);
}

ll range_sum(int l, int r) {
    return (r + 1) * t1.get(r) - t2.get(r) - l * t1.get(l - 1) + t2.get(l - 1);
}
```

- 二维

```
struct Tbit {
    ll t[MAXN][MAXN];

    int lowbit(int x) { return x & (-x); }

    void add(int x, int y, int d) {
        for (int i = x; i <= n; i += lowbit(i))
            for (int j = y; j <= m; j += lowbit(j)) t[i][j] += d;
    }

    ll get(int x, int y) {
        ll sum = 0;
        for (int i = x; i > 0; i -= lowbit(i))
            for (int j = y; j > 0; j -= lowbit(j)) sum += t[i][j];
        return sum;
    }
```

```
    ll query(int x, int y, int xx, int yy) {
        return get(xx, yy) - get(x - 1, yy) - get(xx, y - 1) + get(x - 1, y - 1);
    }
};
```

- 二维区间加，区间和

```
Tbit t0, t1, t2, t3;

void add4(int x, int y, ll d) {
    t0.add(x, y, d);
    t1.add(x, y, d * x);
    t2.add(x, y, d * y);
    t3.add(x, y, d * x * y);
}

void range_add(int x, int y, int xx, int yy, ll d) {
    add4(x, y, d);
    add4(x, yy + 1, -d);
    add4(xx + 1, y, -d);
    add4(xx + 1, yy + 1, d);
}

ll get4(int x, int y) {
    return (x + 1) * (y + 1) * t0.get(x, y)
    - (y + 1) * t1.get(x, y)
    - (x + 1) * t2.get(x, y)
    + t3.get(x, y);
}

ll range_sum(int x, int y, int xx, int yy) {
    return get4(xx, yy) - get4(x - 1, yy) - get4(xx, y - 1) + get4(x - 1, y - 1);
}
```

线段树

- 单点修改，RMQ

```
// 下标从 1 开始
struct Node {
    int val;
    Node(int val = -INF) : val(val) {}
};

Node merge(const Node& a, const Node& b) {
    return Node(max(a.val, b.val));
}

struct SegT {
#define lc (p << 1)
#define rc (p << 1 | 1)
#define mid ((pl + pr) >> 1)

    int size;
    vector<Node> t;

    SegT(int sz) {
        size = 1;
        while (size < sz) size <<= 1;
        t.resize(2 * size);
    }

    Node ask(int p, int l, int r, int pl, int pr) {
        if (l > pr || r < pl) return Node();
        if (l <= pl && r >= pr) return t[p];
        return merge(ask(lc, l, r, pl, mid), ask(rc, l, r, mid + 1, pr));
```

```
    }

    void update(int k, int val) {
        int p = size + k - 1;
        t[p] = Node(val);
        for (p >>= 1; p > 0; p >>= 1) {
            t[p] = merge(t[lc], t[rc]);
        }
    }

    Node query(int l, int r) { return ask(1, l, r, 1, size); }

#undef lc
#undef rc
#undef mid
};
```

- 权值线段树：单点修改，第 k 大

```
void add(int x, ll val) {
    int p = size + x - 1;
    t[p].val += val;
    for (p >>= 1; p > 0; p >>= 1) {
        t[p].val += val;
    }
}

int ask(int p, ll k, int pl, int pr) {
    if (pl == pr) return pl;
    if (k <= t[lc].val) return ask(lc, k, pl, mid);
    return ask(rc, k - t[lc].val, mid + 1, pr);
}

int query(ll k) { return ask(1, k, 1, size); }
```

- 区间加，区间和

```
struct Node {
    ll val, lazy;
};

void pushdown(int p, int pl, int pr) {
    if (!t[p].lazy) return;  // 如果是区间赋值，选取一个数据范围外的值
    t[lc].val += t[p].lazy * (mid - pl + 1);
    t[rc].val += t[p].lazy * (pr - mid);
    t[lc].lazy += t[p].lazy;
    t[rc].lazy += t[p].lazy;
    t[p].lazy = 0;
}

ll ask(int p, int l, int r, int pl, int pr) {
    if (l > pr || r < pl) return 0;
    if (l <= pl && r >= pr) return t[p].val;
    pushdown(p, pl, pr);
    ll vl = ask(lc, l, r, pl, mid);
    ll vr = ask(rc, l, r, mid + 1, pr);
    return vl + vr;
}

void modify(int p, int l, int r, int val, int pl, int pr) {
    if (l > pr || r < pl) return;
    if (l <= pl && r >= pr) {
        t[p].val += 1LL * val * (pr - pl + 1);
        t[p].lazy += val;
        return;
    }
```

```
    pushdown(p, pl, pr);
    modify(lc, l, r, val, pl, mid);
    modify(rc, l, r, val, mid + 1, pr);
    t[p].val = t[lc].val + t[rc].val;
}

void update(int l, int r, int val) { modify(1, l, r, val, 1, size); }
ll query(int l, int r) { return ask(1, l, r, 1, size); }
```

- 区间乘混加，区间和取模

```
struct Node {
    ll val, mul, add;
    Node() : val(0), add(0), mul(1) {}
};

void pushdown(int p, int pl, int pr) {
    if (t[p].mul == 1 && t[p].add == 0) return;
    t[lc].val = (t[lc].val * t[p].mul % MOD + (mid - pl + 1) * t[p].add % MOD) % MOD;
    t[rc].val = (t[rc].val * t[p].mul % MOD + (pr - mid) * t[p].add % MOD) % MOD;
    t[lc].mul = t[p].mul * t[lc].mul % MOD;
    t[rc].mul = t[p].mul * t[rc].mul % MOD;
    t[lc].add = (t[lc].add * t[p].mul % MOD + t[p].add) % MOD;
    t[rc].add = (t[rc].add * t[p].mul % MOD + t[p].add) % MOD;
    t[p].mul = 1;
    t[p].add = 0;
}

ll ask(int p, int l, int r, int pl, int pr) {
    if (l > pr || r < pl) return 0;
    if (l <= pl && r >= pr) return t[p].val;
    pushdown(p, pl, pr);
    ll vl = ask(lc, l, r, pl, mid);
    ll vr = ask(rc, l, r, mid + 1, pr);
    return (vl + vr) % MOD;
}

// x' = ax + b
void modify(int p, int l, int r, int a, int b, int pl, int pr) {
    if (l > pr || r < pl) return;
    if (l <= pl && r >= pr) {
        t[p].val = (t[p].val * a % MOD + 1LL * (pr - pl + 1) * b % MOD) % MOD;
        t[p].mul = t[p].mul * a % MOD;
        t[p].add = (t[p].add * a % MOD + b) % MOD;
        return;
    }
    pushdown(p, pl, pr);
    modify(lc, l, r, a, b, pl, mid);
    modify(rc, l, r, a, b, mid + 1, pr);
    t[p].val = (t[lc].val + t[rc].val) % MOD;
}

void update(int l, int r, int a, int b) { modify(1, l, r, a, b, 1, size); }
ll query(int l, int r) { return ask(1, l, r, 1, size); }
```

动态开点线段树

```
struct Node {
    int lc, rc, val;
    Node(int lc = 0, int rc = 0, int val = 0) : lc(lc), rc(rc), val(val) {}
} t[20 * MAXN];

int cnt;

struct SegT {
```

```
#define mid ((pl + pr) >> 1)

    int rt, size;

    SegT(int sz) : rt(0) {
        size = 1;
        while (size < sz) size <<= 1;
    }

    int modify(int p, int k, int val, int pl, int pr) {
        if (pl > k || pr < k) return p;
        if (!p) p = ++cnt;
        if (pl == pr) t[p].val = val;
        else {
            t[p].lc = modify(t[p].lc, k, val, pl, mid);
            t[p].rc = modify(t[p].rc, k, val, mid + 1, pr);
            t[p].val = max(t[t[p].lc].val, t[t[p].rc].val);
        }
        return p;
    }

    int ask(int p, int l, int r, int pl, int pr) {
        if (l > pr || r < pl) return -INF;
        if (l <= pl && r >= pr) return t[p].val;
        int vl = ask(t[p].lc, l, r, pl, mid);
        int vr = ask(t[p].rc, l, r, mid + 1, pr);
        return max(vl, vr);
    }

    void update(int k, int val) { rt = modify(rt, k, val, 1, size); }
    int query(int l, int r) { return ask(rt, l, r, 1, size); }

#undef mid
};
```

主席树

```
struct Node {
    int lc, rc, val;
    Node(int lc = 0, int rc = 0, int val = 0) : lc(lc), rc(rc), val(val) {}
} t[40 * MAXN]; // (4 + log(size)) * MAXN 小心 MLE

int cnt;

struct FST {
#define mid ((pl + pr) >> 1)

    int size;
    vector<int> root;

    FST(int sz) {
        size = 1;
        while (size < sz) size <<= 1;
        root.push_back(N(0, 0, 0));
    }

    int N(int lc, int rc, int val) {
        t[cnt] = Node(lc, rc, val);
        return cnt++;
    }

    int ins(int p, int x, int pl, int pr) {
        if (pl > x || pr < x) return p;
        if (pl == pr) return N(0, 0, t[p].val + 1);
        return N(ins(t[p].lc, x, pl, mid), ins(t[p].rc, x, mid + 1, pr), t[p].val + 1);
```

```
    }

    int ask(int p1, int p2, int k, int pl, int pr) {
        if (pl == pr) return pl;
        ll vl = t[t[p2].lc].val - t[t[p1].lc].val;
        if (k <= vl) return ask(t[p1].lc, t[p2].lc, k, pl, mid);
        return ask(t[p1].rc, t[p2].rc, k - vl, mid + 1, pr);
    }

    void add(int x) {
        root.push_back(ins(root.back(), x, 1, size));
    }

    int query(int l, int r, int k) {
        return ask(root[l - 1], root[r], k, 1, size);
    }

#undef mid
};
```

Splay

```
// 正常 Splay
struct Node {
    int val, size;
    Node *pa, *lc, *rc;
    Node(int val = 0, Node *pa = nullptr) : val(val), size(1), pa(pa), lc(nullptr), rc(nullptr) {}
    Node*& c(bool x) { return x ? lc : rc; }
    bool d() { return pa ? this == pa->lc : 0; }
} pool[MAXN << 2], *tail = pool;

struct Splay {
    Node *root;

    Splay() : root(nullptr) {}

    Node* N(int val, Node *pa) {
        return new (tail++) Node(val, pa);
    }

    void upd(Node *o) {
        o->size = (o->lc ? o->lc->size : 0) + (o->rc ? o->rc->size : 0) + 1;
    }

    void link(Node *x, Node *y, bool d) {
        if (x) x->pa = y;
        if (y) y->c(d) = x;
    }

    void rotate(Node *o) {
        bool dd = o->d();
        Node *x = o->pa, *xx = x->pa, *y = o->c(!dd);
        link(o, xx, x->d());
        link(y, x, dd);
        link(x, o, !dd);
        upd(x);
        upd(o);
    }

    void splay(Node *o) {
        for (Node *x = o->pa; x = o->pa, x; rotate(o)) {
            if (x->pa) rotate(o->d() == x->d() ? x : o);
        }
        root = o;
    }
```

```
};
```

Treap

```
// split_x 左侧元素 < x
// split_k 左侧分割出 k 个元素
namespace tr {
    using uint = unsigned int;

    uint rnd() {
        static uint A = 1 << 16 | 3, B = 33333331, C = 1091;
        return C = A * C + B;
    }

    struct Node {
        uint key;
        int val, size;
        Node *lc, *rc;
        Node(int val = 0) : key(rnd()), val(val), size(1), lc(nullptr), rc(nullptr) {}
    } pool[MAXN << 2], *tail = pool;

    Node* N(int val) {
        return new (tail++) Node(val);
    }

    void upd(Node *o) {
        o->size = (o->lc ? o->lc->size : 0) + (o->rc ? o->rc->size : 0) + 1;
    }

    Node* merge(Node *l, Node *r) {
        if (!l) return r;
        if (!r) return l;
        if (l->key > r->key) {
            l->rc = merge(l->rc, r);
            upd(l);
            return l;
        } else {
            r->lc = merge(l, r->lc);
            upd(r);
            return r;
        }
    }

    void split_x(Node *o, int x, Node*& l, Node*& r) {
        if (!o) { l = r = nullptr; return; }
        if (o->val < x) {
            l = o;
            split_x(o->rc, x, l->rc, r);
            upd(l);
        } else {
            r = o;
            split_x(o->lc, x, l, r->lc);
            upd(r);
        }
    }

    void split_k(Node *o, int k, Node*& l, Node*& r) {
        if (!o) { l = r = nullptr; return; }
        int lsize = o->lc ? o->lc->size : 0;
        if (lsize < k) {
            l = o;
            split_k(o->rc, k - lsize - 1, o->rc, r);
            upd(l);
        } else {
            r = o;
```

```
            split_k(o->lc, k, l, o->lc);
            upd(r);
        }
    }
}
```

CDQ 分治

- 三维偏序（不严格）

```
struct Node {
    int x, y, z, sum, ans;
} p[MAXN], q[MAXN];

void CDQ(int l, int r) {
    if (l == r) return;
    int mid = (l + r) >> 1;
    CDQ(l, mid);
    CDQ(mid + 1, r);
    int i = l, j = mid + 1;
    for (int t = l; t <= r; t++) {
        if (j > r || (i <= mid && p[i].y <= p[j].y)) {
            q[t] = p[i++];
            bit.add(q[t].z, q[t].sum);
        } else {
            q[t] = p[j++];
            q[t].ans += bit.get(q[t].z);
        }
    }
    for (i = l; i <= r; i++) {
        p[i] = q[i];
        bit.update(p[i].z, 0);
    }
}

void go() {
    sort(p + 1, p + n + 1, [](const Node &a, const Node &b) {
        if (a.x != b.x) return a.x < b.x;
        if (a.y != b.y) return a.y < b.y;
        return a.z < b.z;
    });
    auto eq = [](const Node& a, const Node& b) {
        return a.x == b.x && a.y == b.y && a.z == b.z;
    };
    int k = n;
    for (int i = 1, j = 1; i <= n; i++, j++) {
        if (eq(p[i], p[j - 1])) j--, k--;
        else if (i != j) p[j] = p[i];
        p[j].sum++;
    }
    bit.init(m);
    CDQ(1, k);
}
```

图论

链式前向星

```
int ecnt, mp[MAXN];

struct Edge {
    int to, nxt;
    Edge(int to = 0, int nxt = 0) : to(to), nxt(nxt) {}
} es[MAXM];
```

```
void mp_init() {
    memset(mp, -1, (n + 2) * sizeof(int));
    ecnt = 0;
}

void mp_link(int u, int v) {
    es[ecnt] = Edge(v, mp[u]);
    mp[u] = ecnt++;
}

for (int i = mp[u]; i != -1; i = es[i].nxt)
```

最短路

- Dijkstra

```
struct Edge {
    int to, val;
    Edge(int to = 0, int val = 0) : to(to), val(val) {}
};

vector<Edge> G[MAXN];
ll dis[MAXN];

void dijkstra(int s) {
    using pii = pair<ll, int>;
    memset(dis, 0x3f, sizeof(dis));
    priority_queue<pii, vector<pii>, greater<pii> > q;
    dis[s] = 0;
    q.emplace(0, s);
    while (!q.empty()) {
        pii p = q.top();
        q.pop();
        int u = p.second;
        if (dis[u] < p.first) continue;
        for (Edge& e : G[u]) {
            int v = e.to;
            if (updmin(dis[v], dis[u] + e.val)) {
                q.emplace(dis[v], v);
            }
        }
    }
}
```

- SPFA

```
void spfa(int s) {
    queue<int> q;
    q.push(s);
    memset(dis, 0x3f, sizeof(dis));
    memset(in, 0, sizeof(in));
    in[s] = 1;
    dis[s] = 0;
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        for (Edge& e : G[u]) {
            int v = e.to;
            if (dis[v] > dis[u] + e.val) {
                dis[v] = dis[u] + e.val;
                if (!in[v]) {
                    in[v] = 1;
                    q.push(v);
                }
            }
```

```
        }
        in[u] = 0;
    }
}
```

- Floyd 最小环

```
// 注意 INF 不能超过 1/3 LLONG_MAX
for (int k = 0; k < n; k++) {
    for (int i = 0; i < k; i++) {
        for (int j = 0; j < i; j++) {
            ans = min(ans, G[i][k] + G[k][j] + dis[i][j]);
        }
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            dis[i][j] = min(dis[i][j], dis[i][k] + dis[k][j]);
        }
    }
}
```

拓扑排序

```
int n, deg[MAXN], dis[MAXN];
vector<int> G[MAXN];

bool topo(vector<int>& ans) {
    queue<int> q;
    for (int i = 1; i <= n; i++) {
        if (deg[i] == 0) {
            q.push(i);
            dis[i] = 1;
        }
    }
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        ans.push_back(u);
        for (int v : G[u]) {
            deg[v]--;
            dis[v] = max(dis[v], dis[u] + 1);
            if (deg[v] == 0) q.push(v);
        }
    }
    return ans.size() == n;
}
```

最小生成树

```
// 前置：并查集
struct Edge {
    int from, to, val;
    Edge(int from = 0, int to = 0, int val = 0) : from(from), to(to), val(val) {}
};

vector<Edge> es;

ll kruskal() {
    sort(es.begin(), es.end(), [](Edge& x, Edge& y) { return x.val < y.val; });
    iota(pa, pa + n + 1, 0);
    ll ans = 0;
    for (Edge& e : es) {
        if (find(e.from) != find(e.to)) {
```

```
            merge(e.from, e.to);
            ans += e.val;
        }
    }
    return ans;
}
```

LCA

```
int dep[MAXN], up[MAXN][22]; // 22 = ((int)log2(MAXN) + 1)

void dfs(int u, int pa) {
    dep[u] = dep[pa] + 1;
    up[u][0] = pa;
    for (int i = 1; i < 22; i++) {
        up[u][i] = up[up[u][i - 1]][i - 1];
    }
    for (int i = 0; i < G[u].size(); i++) {
        if (G[u][i] != pa) {
            dfs(G[u][i], u);
        }
    }
}

int lca(int u, int v) {
    if (dep[u] > dep[v]) swap(u, v);
    int t = dep[v] - dep[u];
    for (int i = 0; i < 22; i++) {
        if ((t >> i) & 1) v = up[v][i];
    }
    if (u == v) return u;
    for (int i = 21; i >= 0; i--) {
        if (up[u][i] != up[v][i]) {
            u = up[u][i];
            v = up[v][i];
        }
    }
    return up[u][0];
}
```

网络流

- 最大流

```
const int INF = 0x7fffffff;

struct DEdge {
    int to, cap;
    DEdge(int to, int cap) : to(to), cap(cap) {}
};

struct Dinic {
    int n, s, t;
    vector<DEdge> es;
    vector<vector<int> > G;
    vector<int> dis, cur;

    Dinic(int n, int s, int t) : n(n), s(s), t(t), G(n + 1), dis(n + 1), cur(n + 1) {}

    void add_edge(int u, int v, int cap) {
        G[u].push_back(es.size());
        es.emplace_back(v, cap);
        G[v].push_back(es.size());
        es.emplace_back(u, 0);
```

```
    }

    bool bfs() {
        dis.assign(n + 1, 0);
        queue<int> q;
        q.push(s);
        dis[s] = 1;
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            for (int i : G[u]) {
                DEdge& e = es[i];
                if (!dis[e.to] && e.cap > 0) {
                    dis[e.to] = dis[u] + 1;
                    q.push(e.to);
                }
            }
        }
        return dis[t];
    }

    int dfs(int u, int cap) {
        if (u == t || cap == 0) return cap;
        int tmp = cap, f;
        for (int& i = cur[u]; i < G[u].size(); i++) {
            DEdge& e = es[G[u][i]];
            if (dis[e.to] == dis[u] + 1) {
                f = dfs(e.to, min(cap, e.cap));
                e.cap -= f;
                es[G[u][i] ^ 1].cap += f;
                cap -= f;
                if (cap == 0) break;
            }
        }
        return tmp - cap;
    }

    ll solve() {
        ll flow = 0;
        while (bfs()) {
            cur.assign(n + 1, 0);
            flow += dfs(s, INF);
        }
        return flow;
    }
};
```

- 最小费用流

```
const int INF = 0x7fffffff;

struct MEdge {
    int from, to, cap, cost;
    MEdge(int from, int to, int cap, int cost) : from(from), to(to), cap(cap), cost(cost) {}
};

struct MCMF {
    int n, s, t, flow, cost;
    vector<MEdge> es;
    vector<vector<int> > G;
    vector<int> d, p, a, in;  // dis, prev, add

    MCMF(int n, int s, int t) : n(n), s(s), t(t), flow(0), cost(0), G(n + 1), p(n + 1), a(n + 1) {}

    void add_edge(int u, int v, int cap, int cost) {
        G[u].push_back(es.size());
```

```
        es.emplace_back(u, v, cap, cost);
        G[v].push_back(es.size());
        es.emplace_back(v, u, 0, -cost);
    }

    bool spfa() {
        d.assign(n + 1, INF);
        in.assign(n + 1, 0);
        d[s] = 0;
        in[s] = 1;
        a[s] = INF;
        queue<int> q;
        q.push(s);
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            in[u] = 0;
            for (int& i : G[u]) {
                MEdge& e = es[i];
                if (e.cap && d[e.to] > d[u] + e.cost) {
                    d[e.to] = d[u] + e.cost;
                    p[e.to] = i;
                    a[e.to] = min(a[u], e.cap);
                    if (!in[e.to]) {
                        q.push(e.to);
                        in[e.to] = 1;
                    }
                }
            }
        }
        return d[t] != INF;
    }

    void solve() {
        while (spfa()) {
            flow += a[t];
            cost += a[t] * d[t];
            int u = t;
            while (u != s) {
                es[p[u]].cap -= a[t];
                es[p[u] ^ 1].cap += a[t];
                u = es[p[u]].from;
            }
        }
    }
};
```

无向图最小割

```
namespace stoer_wagner {
    bool vis[MAXN], in[MAXN];
    int G[MAXN][MAXN], w[MAXN];

    void init() {
        memset(G, 0, sizeof(G));
        memset(in, 0, sizeof(in));
    }

    void add_edge(int u, int v, int w) {
        G[u][v] += w;
        G[v][u] += w;
    }

    int search(int& s, int& t) {
        memset(vis, 0, sizeof(vis));
```

```
        memset(w, 0, sizeof(w));
        int maxw, tt = n + 1;
        for (int i = 0; i < n; i++) {
            maxw = -INF;
            for (int j = 0; j < n; j++) {
                if (!in[j] && !vis[j] && w[j] > maxw) {
                    maxw = w[j];
                    tt = j;
                }
            }
            if (t == tt) return w[t];
            s = t; t = tt;
            vis[tt] = true;
            for (int j = 0; j < n; j++) {
                if (!in[j] && !vis[j]) {
                    w[j] += G[tt][j];
                }
            }
        }
        return w[t];
    }

    int go() {
        int s, t, ans = INF;
        for (int i = 0; i < n - 1; i++) {
            s = t = -1;
            ans = min(ans, search(s, t));
            if (ans == 0) return 0;
            in[t] = true;
            for (int j = 0; j < n; j++) {
                if (!in[j]) {
                    G[s][j] += G[t][j];
                    G[j][s] += G[j][t];
                }
            }
        }
        return ans;
    }
}
```

树链剖分

```
// 点权
vector<int> G[MAXN];
int pa[MAXN], sz[MAXN], dep[MAXN], dfn[MAXN], maxc[MAXN], top[MAXN];

void dfs1(int u) {
    sz[u] = 1;
    maxc[u] = -1;
    int maxs = 0;
    for (int& v : G[u]) {
        if (v != pa[u]) {
            pa[v] = u;
            dep[v] = dep[u] + 1;
            dfs1(v);
            sz[u] += sz[v];
            if (updmax(maxs, sz[v])) maxc[u] = v;
        }
    }
}

void dfs2(int u, int tp) {
    static int cnt = 0;
    top[u] = tp;
    dfn[u] = ++cnt;
```

```
    if (maxc[u] != -1) dfs2(maxc[u], tp);
    for (int& v : G[u]) {
        if (v != pa[u] && v != maxc[u]) {
            dfs2(v, v);
        }
    }
}

void init() {
    dep[1] = 1;
    dfs1(1);
    dfs2(1, 1);
}

ll go(int u, int v) {
    int uu = top[u], vv = top[v];
    ll res = 0;
    while (uu != vv) {
        if (dep[uu] < dep[vv]) {
            swap(u, v);
            swap(uu, vv);
        }
        res += segt.query(dfn[uu], dfn[u]);
        u = pa[uu];
        uu = top[u];
    }
    if (dep[u] > dep[v]) swap(u, v);
    res += segt.query(dfn[u], dfn[v]);
    return res;
}
```

Tarjan

- 割点

```
int dfn[MAXN], low[MAXN], clk;

void init() { clk = 0; memset(dfn, 0, sizeof(dfn)); }

void tarjan(int u, int pa) {
    low[u] = dfn[u] = ++clk;
    int cc = (pa != 0);
    for (int v : G[u]) {
        if (v == pa) continue;
        if (!dfn[v]) {
            tarjan(v, u);
            low[u] = min(low[u], low[v]);
            cc += low[v] >= dfn[u];
        } else low[u] = min(low[u], dfn[v]);
    }
    if (cc > 1) // ...
}
```

- 桥

```
int dfn[MAXN], low[MAXN], clk;

void init() { clk = 0; memset(dfn, 0, sizeof(dfn)); }

void tarjan(int u, int pa) {
    low[u] = dfn[u] = ++clk;
    int f = 0;
    for (int v : G[u]) {
        if (v == pa && ++f == 1) continue;
        if (!dfn[v]) {
            tarjan(v, u);
```

```
            if (low[v] > dfn[u]) // ...
            low[u] = min(low[u], low[v]);
        } else low[u] = min(low[u], dfn[v]);
    }
}
```

- 强连通分量缩点

```
int dfn[MAXN], low[MAXN], clk, tot, color[MAXN];
vector<int> scc[MAXN];

void init() { tot = clk = 0; memset(dfn, 0, sizeof dfn); }

void tarjan(int u) {
    static int st[MAXN], p;
    static bool in[MAXN];
    dfn[u] = low[u] = ++clk;
    st[p++] = u;
    in[u] = true;
    for (int v : G[u]) {
        if (!dfn[v]) {
            tarjan(v);
            low[u] = min(low[u], low[v]);
        } else if (in[v]) {
            low[u] = min(low[u], dfn[v]);
        }
    }
    if (dfn[u] == low[u]) {
        ++tot;
        for (;;) {
            int x = st[--p];
            in[x] = false;
            color[x] = tot;
            scc[tot].push_back(x);
            if (x == u) break;
        }
    }
}
```

- 2-SAT

```
// MAXN 开两倍
void two_sat() {
    for (int i = 1; i <= n * 2; i++) {
        if (!dfn[i]) tarjan(i);
    }
    for (int i = 1; i <= n; i++) {
        if (color[i] == color[i + n]) {
            // impossible
        }
    }
    for (int i = 1; i <= n; i++) {
        if (color[i] < color[i + n]) {
            // select
        }
    }
}
```

支配树

- 有向无环图

```
// rt 是 G 中入度为 0 的点（可能需要建超级源点）
int n, deg[MAXN], dep[MAXN], up[MAXN][22];
vector<int> G[MAXN], rG[MAXN], dt[MAXN];
```

```cpp
bool topo(vector<int>& ans, int rt) {
    queue<int> q;
    q.push(rt);
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        ans.push_back(u);
        for (int v : G[u]) {
            deg[v]--;
            if (deg[v] == 0) q.push(v);
        }
    }
    return ans.size() == n;
}

int lca(int u, int v) {
    if (dep[u] > dep[v]) swap(u, v);
    int t = dep[v] - dep[u];
    for (int i = 0; i < 22; i++) {
        if ((t >> i) & 1) v = up[v][i];
    }
    if (u == v) return u;
    for (int i = 21; i >= 0; i--) {
        if (up[u][i] != up[v][i]) {
            u = up[u][i];
            v = up[v][i];
        }
    }
    return up[u][0];
}

void go(int rt) {
    vector<int> a;
    topo(a, rt);
    dep[rt] = 1;
    for (int i = 1; i < a.size(); i++) {
        int u = a[i], pa = -1;
        for (int v : rG[u]) {
            pa = (pa == -1) ? v : lca(pa, v);
        }
        dt[pa].push_back(u);
        dep[u] = dep[pa] + 1;
        up[u][0] = pa;
        for (int i = 1; i < 22; i++) {
            up[u][i] = up[up[u][i - 1]][i - 1];
        }
    }
}
```

- 一般有向图

```cpp
vector<int> G[MAXN], rG[MAXN];
vector<int> dt[MAXN];

namespace tl {
    int pa[MAXN], dfn[MAXN], clk, rdfn[MAXN];
    int c[MAXN], best[MAXN], sdom[MAXN], idom[MAXN];

    void init(int n) {
        clk = 0;
        fill(c, c + n + 1, -1);
        fill(dfn, dfn + n + 1, 0);
        for (int i = 1; i <= n; i++) {
            dt[i].clear();
            sdom[i] = best[i] = i;
        }
```

```
    }

    void dfs(int u) {
        dfn[u] = ++clk;
        rdfn[clk] = u;
        for (int& v: G[u]) {
            if (!dfn[v]) {
                pa[v] = u;
                dfs(v);
            }
        }
    }

    int fix(int x) {
        if (c[x] == -1) return x;
        int& f = c[x], rt = fix(f);
        if (dfn[sdom[best[x]]] > dfn[sdom[best[f]]]) best[x] = best[f];
        return f = rt;
    }

    void go(int rt) {
        dfs(rt);
        for (int i = clk; i > 1; i--) {
            int x = rdfn[i], mn = clk + 1;
            for (int& u: rG[x]) {
                if (!dfn[u]) continue; // 可能不能到达所有点
                fix(u);
                mn = min(mn, dfn[sdom[best[u]]]);
            }
            c[x] = pa[x];
            dt[sdom[x] = rdfn[mn]].push_back(x);
            x = rdfn[i - 1];
            for (int& u: dt[x]) {
                fix(u);
                idom[u] = (sdom[best[u]] == x) ? x : best[u];
            }
            dt[x].clear();
        }
        for (int i = 2; i <= clk; i++) {
            int u = rdfn[i];
            if (idom[u] != sdom[u]) idom[u] = idom[idom[u]];
            dt[idom[u]].push_back(u);
        }
    }
}
```

字符串

哈希

```
// open hack 不要用哈希
using ull = unsigned long long;

const int x = 135, p1 = 1e9 + 7, p2 = 1e9 + 9;
const ull mask32 = ~(0u);

ull xp1[MAXN], xp2[MAXN];

void init_xp() {
    xp1[0] = xp2[0] = 1;
    for (int i = 1; i < MAXN; i++) {
        xp1[i] = xp1[i - 1] * x % p1;
        xp2[i] = xp2[i - 1] * x % p2;
    }
}
```

```
struct Hash {
    vector<ull> h;

    Hash() : h(1) {}

    void add(const string& s) {
        ull res1 = h.back() >> 32;
        ull res2 = h.back() & mask32;
        for (char c : s) {
            res1 = (res1 * x + c) % p1;
            res2 = (res2 * x + c) % p2;
            h.push_back((res1 << 32) | res2);
        }
    }

    ull get(int l, int r) {
        r++;
        int len = r - l;
        ull l1 = h[l] >> 32, r1 = h[r] >> 32;
        ull l2 = h[l] & mask32, r2 = h[r] & mask32;
        ull res1 = (r1 - l1 * xp1[len] % p1 + p1) % p1;
        ull res2 = (r2 - l2 * xp2[len] % p2 + p2) % p2;
        return (res1 << 32) | res2;
    }
};
```

- 二维哈希

```
const ll basex = 239, basey = 241, p = 998244353;

ll pwx[MAXN], pwy[MAXN];

void init_xp() {
    pwx[0] = pwy[0] = 1;
    for (int i = 1; i < MAXN; i++) {
        pwx[i] = pwx[i - 1] * basex % p;
        pwy[i] = pwy[i - 1] * basey % p;
    }
}

struct Hash2D {
    vector<vector<ll> > h;

    Hash2D(const vector<vector<int> >& a, int n, int m) : h(n + 1, vector<ll>(m + 1)) {
        for (int i = 0; i < n; i++) {
            ll s = 0;
            for (int j = 0; j < m; j++) {
                s = (s * basey + a[i][j] + 1) % p;
                h[i + 1][j + 1] = (h[i][j + 1] * basex + s) % p;
            }
        }
    }

    ll get(int x, int y, int xx, int yy) {
        ++xx; ++yy;
        int dx = xx - x, dy = yy - y;
        ll res = h[xx][yy]
            - h[x][yy] * pwx[dx]
            - h[xx][y] * pwy[dy]
            + h[x][y] * pwx[dx] % p * pwy[dy];
        return (res % p + p) % p;
    }
};
```

Manacher

```
// "aba" => "#a#b#a#"
string make(string& s) {
    string t = "#";
    for (int i = 0; i < s.size(); i++) {
        t.push_back(s[i]);
        t.push_back('#');
    }
    return t;
}

void manacher(string& s, vector<int>& d) {
    int n = s.size();
    d.resize(n);
    for (int i = 0, l = 0, r = -1; i < n; i++) {
        int k = (i > r) ? 1 : min(d[l + r - i], r - i);
        while (i - k >= 0 && i + k < n && s[i - k] == s[i + k]) k++;
        d[i] = --k;
        if (i + k > r) {
            l = i - k;
            r = i + k;
        }
    }
}
```

KMP

```
// 前缀函数（每一个前缀的最长公共前后缀）
void get_pi(const string& s, vector<int>& a) {
    int n = s.size(), j = 0;
    a.resize(n);
    for (int i = 1; i < n; i++) {
        while (j && s[j] != s[i]) j = a[j - 1];
        if (s[j] == s[i]) j++;
        a[i] = j;
    }
}

void kmp(const string& s, vector<int>& a, const string& t) {
    int j = 0;
    for (int i = 0; i < t.size(); i++) {
        while (j && s[j] != t[i]) j = a[j - 1];
        if (s[j] == t[i]) j++;
        if (j == s.size()) {
            // ...
            j = a[j - 1]; // 允许重叠匹配 j = 0 不允许
        }
    }
}

// Z 函数（每一个后缀和该字符串的最长公共前缀）
void get_z(const string& s, vector<int>& z) {
    int n = s.size(), l = 0, r = 0;
    z.resize(n);
    for (int i = 1; i < n; i++) {
        if (i <= r) z[i] = min(r - i + 1, z[i - l]);
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) z[i]++;
        if (i + z[i] - 1 > r) {
            l = i;
            r = i + z[i] - 1;
        }
    }
}
```

最小表示法

```
int get(const string& s) {
    int k = 0, i = 0, j = 1, n = s.size();
    while (k < n && i < n && j < n) {
        if (s[(i + k) % n] == s[(j + k) % n]) {
            k++;
        } else {
            s[(i + k) % n] > s[(j + k) % n] ? i = i + k + 1 : j = j + k + 1;
            if (i == j) i++;
            k = 0;
        }
    }
    return min(i, j);
}
```

Trie

```
// 01 Trie
struct Trie {
    int t[31 * MAXN][2], sz;

    void init() {
        memset(t, 0, 2 * (sz + 2) * sizeof(int));
        sz = 1;
    }

    void insert(int x) {
        int p = 0;
        for (int i = 30; i >= 0; i--) {
            bool d = (x >> i) & 1;
            if (!t[p][d]) t[p][d] = sz++;
            p = t[p][d];
        }
    }
};

// 正常 Trie
struct Trie {
    int t[MAXN][26], sz, cnt[MAXN];

    void init() {
        memset(t, 0, 26 * (sz + 2) * sizeof(int));
        memset(cnt, 0, (sz + 2) * sizeof(int));
        sz = 1;
    }

    void insert(const string& s) {
        int p = 0;
        for (char c : s) {
            int d = c - 'a';
            if (!t[p][d]) t[p][d] = sz++;
            p = t[p][d];
        }
        cnt[p]++;
    }
};
```

AC 自动机

```
struct ACA {
    int t[MAXN][26], sz, fail[MAXN], nxt[MAXN], cnt[MAXN];

    void init() {
```

```
        memset(t, 0, 26 * (sz + 2) * sizeof(int));
        memset(fail, 0, (sz + 2) * sizeof(int));
        memset(nxt, 0, (sz + 2) * sizeof(int));
        memset(cnt, 0, (sz + 2) * sizeof(int));
        sz = 1;
    }

    void insert(const string& s) {
        int p = 0;
        for (char c : s) {
            int d = c - 'a';
            if (!t[p][d]) t[p][d] = sz++;
            p = t[p][d];
        }
        cnt[p]++;
    }

    void build() {
        queue<int> q;
        for (int i = 0; i < 26; i++) {
            if (t[0][i]) q.push(t[0][i]);
        }
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            for (int i = 0; i < 26; i++) {
                int& v = t[u][i];
                if (v) {
                    fail[v] = t[fail[u]][i];
                    nxt[v] = cnt[fail[v]] ? fail[v] : nxt[fail[v]];
                    q.push(v);
                } else {
                    v = t[fail[u]][i];
                }
            }
        }
    }
};
```

回文自动机

```
// WindJ0Y
struct Palindromic_Tree {
    static constexpr int MAXN = 300005;
    static constexpr int N = 26;

    int next[MAXN][N]; // next 指针，next 指针和字典树类似，指向的串为当前串两端加上同一个字符构成
    int fail[MAXN]; // fail 指针，失配后跳转到 fail 指针指向的节点
    int cnt[MAXN]; // 表示节点 i 表示的本质不同的串的个数 aftre count()
    int num[MAXN]; // 表示以节点 i 表示的最长回文串的最右端点为回文串结尾的回文串个数。
    int len[MAXN]; // len[i] 表示节点 i 表示的回文串的长度
    int lcnt[MAXN];
    int S[MAXN]; // 存放添加的字符
    int last; // 指向上一个字符所在的节点，方便下一次 add
    int n; // 字符数组指针
    int p; // 节点指针

    int newnode(int l, int vc) { // 新建节点
        for (int i = 0; i < N; ++i) next[p][i] = 0;
        cnt[p] = 0;
        num[p] = 0;
        len[p] = l;
        lcnt[p] = vc;
        return p++;
    }
```

```
    void init() { // 初始化
        p = 0;
        newnode(0, 0);
        newnode(-1, 0);
        last = 0;
        n = 0;
        S[n] = -1; // 开头放一个字符集中没有的字符，减少特判
        fail[0] = 1;
    }

    int get_fail(int x) { // 和 KMP 一样，失配后找一个尽量最长的
        while (S[n - len[x] - 1] != S[n]) x = fail[x];
        return x;
    }

    void add(int c) {
        S[++n] = c;
        int cur = get_fail(last); // 通过上一个回文串找这个回文串的匹配位置
        if (!next[cur][c]) { // 如果这个回文串没有出现过，说明出现了一个新的本质不同的回文串
            int now = newnode(len[cur] + 2, lcnt[cur] | (1 << c)); // 新建节点
            fail[now] = next[get_fail(fail[cur])][c]; // 和 AC 自动机一样建立 fail 指针，以便失配后跳转
            next[cur][c] = now;
            num[now] = num[fail[now]] + 1;
        }
        last = next[cur][c];
        cnt[last]++;
    }

    void count() {
        for (int i = p - 1; i >= 0; --i) cnt[fail[i]] += cnt[i];
        // 父亲累加儿子的 cnt，因为如果 fail[v]=u，则 u 一定是 v 的子回文串
    }
} pt;
```

后缀数组

```
// 下标从 1 开始
// sa[i]: 排名为 i 的后缀位置
// rk[i]: 第 i 个后缀的排名
// ht[i]: LCP(sa[i], sa[i - 1])
struct SA {
    int n, m;
    vector<int> a, d, sa, rk, ht;

    void rsort() {
        vector<int> c(m + 1);
        for (int i = 1; i <= n; i++) c[rk[d[i]]]++;
        for (int i = 1; i <= m; i++) c[i] += c[i - 1];
        for (int i = n; i; i--) sa[c[rk[d[i]]]--] = d[i];
    }

    SA(const string& s) : n(s.size()), m(128), a(n + 1), d(n + 1), sa(n + 1), rk(n + 1), ht(n + 1) {
        for (int i = 1; i <= n; i++) { rk[i] = a[i] = s[i - 1]; d[i] = i; }
        rsort();
        for (int j = 1, i, k; k < n; m = k, j <<= 1) {
            for (i = n - j + 1, k = 0; i <= n; i++) d[++k] = i;
            for (i = 1; i <= n; i++) if (sa[i] > j) d[++k] = sa[i] - j;
            rsort(); swap(rk, d); rk[sa[1]] = k = 1;
            for (i = 2; i <= n; i++) {
                rk[sa[i]] = (d[sa[i]] == d[sa[i - 1]] && d[sa[i] + j] == d[sa[i - 1] + j]) ? k : ++k;
            }
        }
        int j, k = 0;
        for (int i = 1; i <= n; ht[rk[i++]] = k) {
```

```
            for (k ? k-- : k, j = sa[rk[i] - 1]; a[i + k] == a[j + k]; ++k);
        }
    }
};
```

数学

GCD & LCM

```
ll gcd(ll a, ll b) { return b ? gcd(b, a % b) : a; }
ll lcm(ll a, ll b) { return a / gcd(a, b) * b; }
```

快速乘 & 快速幂

```
// 模数爆 int 时使用
ll mul(ll a, ll b, ll p) {
    ll ans = 0;
    for (a %= p; b; b >>= 1) {
        if (b & 1) ans = (ans + a) % p;
        a = (a << 1) % p;
    }
    return ans;
}

// O(1)
ll mul(ll a, ll b, ll p) {
    return (ll)(__int128(a) * b % p);
}

ll qk(ll a, ll b, ll p) {
    ll ans = 1 % p;
    for (a %= p; b; b >>= 1) {
        if (b & 1) ans = ans * a % p;
        a = a * a % p;
    }
    return ans;
}

// 十进制快速幂
ll qk(ll a, const string& b, ll p) {
    ll ans = 1;
    for (int i = b.size() - 1; i >= 0; i--) {
        ans = ans * qk(a, b[i] - '0', p) % p;
        a = qk(a, 10, p);
    }
    return ans;
}
```

矩阵快速幂

```
const int M_SZ = 3;

using Mat = array<array<ll, M_SZ>, M_SZ>;

#define rep2 for (int i = 0; i < M_SZ; i++) for (int j = 0; j < M_SZ; j++)

void zero(Mat& a) { rep2 a[i][j] = 0; }
void one(Mat& a) { rep2 a[i][j] = (i == j); }

Mat mul(const Mat& a, const Mat& b, ll p) {
    Mat ans; zero(ans);
    rep2 if (a[i][j]) for (int k = 0; k < M_SZ; k++) {
```

```
        (ans[i][k] += a[i][j] * b[j][k]) %= p;
    }
    return ans;
}

Mat qk(Mat a, ll b, ll p) {
    Mat ans; one(ans);
    for (; b; b >>= 1) {
        if (b & 1) ans = mul(a, ans, p);
        a = mul(a, a, p);
    }
    return ans;
}

// 十进制快速幂
Mat qk(Mat a, const string& b, ll p) {
    Mat ans; one(ans);
    for (int i = b.size() - 1; i >= 0; i--) {
        ans = mul(qk(a, b[i] - '0', p), ans, p);
        a = qk(a, 10, p);
    }
    return ans;
}

#undef rep2
```

素数判断

```
bool isprime(int x) {
    if (x < 2) return false;
    for (int i = 2; i * i <= x; i++) {
        if (x % i == 0) return false;
    }
    return true;
}

// O(logn)
// 前置：快速乘、快速幂
// int 范围只需检查 2, 7, 61
bool Rabin_Miller(ll a, ll n) {
    if (n == 2 || a >= n) return 1;
    if (n == 1 || !(n & 1)) return 0;
    ll d = n - 1;
    while (!(d & 1)) d >>= 1;
    ll t = qk(a, d, n);
    while (d != n - 1 && t != 1 && t != n - 1) {
        t = mul(t, t, n);
        d <<= 1;
    }
    return t == n - 1 || d & 1;
}

bool isprime(ll n) {
    static vector<ll> t = {2, 325, 9375, 28178, 450775, 9780504, 1795265022};
    if (n <= 1) return false;
    for (ll k : t) if (!Rabin_Miller(k, n)) return false;
    return true;
}
```

线性筛

```
// 注意 0 和 1 不是素数
bool vis[MAXN];
int prime[MAXN];
```

```cpp
void get_prime() {
    int tot = 0;
    for (int i = 2; i < MAXN; i++) {
        if (!vis[i]) prime[tot++] = i;
        for (int j = 0; j < tot; j++) {
            int d = i * prime[j];
            if (d >= MAXN) break;
            vis[d] = true;
            if (i % prime[j] == 0) break;
        }
    }
}

// 最小素因子
bool vis[MAXN];
int spf[MAXN], prime[MAXN];

void get_spf() {
    int tot = 0;
    for (int i = 2; i < MAXN; i++) {
        if (!vis[i]) {
            prime[tot++] = i;
            spf[i] = i;
        }
        for (int j = 0; j < tot; j++) {
            int d = i * prime[j];
            if (d >= MAXN) break;
            vis[d] = true;
            spf[d] = prime[j];
            if (i % prime[j] == 0) break;
        }
    }
}

// 欧拉函数
bool vis[MAXN];
int phi[MAXN], prime[MAXN];

void get_phi() {
    int tot = 0;
    phi[1] = 1;
    for (int i = 2; i < MAXN; i++) {
        if (!vis[i]) {
            prime[tot++] = i;
            phi[i] = i - 1;
        }
        for (int j = 0; j < tot; j++) {
            int d = i * prime[j];
            if (d >= MAXN) break;
            vis[d] = true;
            if (i % prime[j] == 0) {
                phi[d] = phi[i] * prime[j];
                break;
            }
            else phi[d] = phi[i] * (prime[j] - 1);
        }
    }
}

// 莫比乌斯函数
bool vis[MAXN];
int mu[MAXN], prime[MAXN];

void get_mu() {
    int tot = 0;
```

```
    mu[1] = 1;
    for (int i = 2; i < MAXN; i++) {
        if (!vis[i]) {
            prime[tot++] = i;
            mu[i] = -1;
        }
        for (int j = 0; j < tot; j++) {
            int d = i * prime[j];
            if (d >= MAXN) break;
            vis[d] = true;
            if (i % prime[j] == 0) {
                mu[d] = 0;
                break;
            }
            else mu[d] = -mu[i];
        }
    }
}
```

区间筛

```
// a, b <= 1e13, b - a <= 1e6
bool vis_small[MAXN], vis_big[MAXN];
ll prime[MAXN];
int tot = 0;

void get_prime(ll a, ll b) {
    ll c = ceil(sqrt(b));
    for (ll i = 2; i <= c; i++) {
        if (!vis_small[i]) {
            for (ll j = i * i; j <= c; j += i) {
                vis_small[j] = 1;
            }
            for (ll j = max(i, (a + i - 1) / i) * i; j <= b; j += i) {
                vis_big[j - a] = 1;
            }
        }
    }
    for (int i = max(0LL, 2 - a); i <= b - a; i++) {
        if (!vis_big[i]) prime[tot++] = i + a;
    }
}
```

找因数

```
// O(sqrt(n))
vector<int> getf(int x) {
    vector<int> v;
    for (int i = 1; i * i <= x; i++) {
        if (x % i == 0) {
            v.push_back(i);
            if (x / i != i) v.push_back(x / i);
        }
    }
    sort(v.begin(), v.end());
    return v;
}
```

找质因数

```
// O(sqrt(n)), 无重复
vector<int> getf(int x) {
```

```
    vector<int> v;
    for (int i = 2; i * i <= x; i++) {
        if (x % i == 0) {
            v.push_back(i);
            while (x % i == 0) x /= i;
        }
    }
    if (x != 1) v.push_back(x);
    return v;
}

// O(sqrt(n))，有重复
vector<int> getf(int x) {
    vector<int> v;
    for (int i = 2; i * i <= x; i++) {
        while (x % i == 0) {
            v.push_back(i);
            x /= i;
        }
    }
    if (x != 1) v.push_back(x);
    return v;
}

// 前置：线性筛
// O(logn)，无重复
vector<int> getf(int x) {
    vector<int> v;
    while (x > 1) {
        int p = spf[x];
        v.push_back(p);
        while (x % p == 0) x /= p;
    }
    return v;
}

// O(logn)，有重复
vector<int> getf(int x) {
    vector<int> v;
    while (x > 1) {
        int p = spf[x];
        while (x % p == 0) {
            v.push_back(p);
            x /= p;
        }
    }
    return v;
}
```

Pollard-Rho

```
mt19937_64 mt_rand(time(0));

ll pollard_rho(ll n, ll c) {
    ll x = mt_rand() % (n - 1) + 1, y = x;
    auto f = [&](ll v) {
        ll t = mul(v, v, n) + c;
        return t < n ? t : t - n;
    };
    for (;;) {
        x = f(x); y = f(f(y));
        if (x == y) return n;
        ll d = gcd(abs(x - y), n);
        if (d != 1) return d;
    }
```

```
}

vector<ll> getf(ll x) {
    vector<ll> v;
    if (x <= 1) return v;
    function<void(ll)> f = [&](ll n) {
        if (n == 4) { v.push_back(2); v.push_back(2); return; }
        if (isprime(n)) { v.push_back(n); return; }
        ll p = n, c = 19260817;
        while (p == n) p = pollard_rho(n, --c);
        f(p); f(n / p);
    };
    f(x);
    return v;
}
```

欧拉函数

```
// 前置：找质因数（无重复）
int phi(int x) {
    int ret = x;
    vector<int> v = getf(x);
    for (int f : v) ret = ret / f * (f - 1);
    return ret;
}

// O(nloglogn)
int phi[MAXN];

void get_phi() {
    phi[1] = 1;
    for (int i = 2; i < MAXN; i++) {
        if (!phi[i]) {
            for (int j = i; j < MAXN; j += i) {
                if (!phi[j]) phi[j] = j;
                phi[j] = phi[j] / i * (i - 1);
            }
        }
    }
}
```

EXGCD

```
// ax + by = gcd(a, b)
ll exgcd(ll a, ll b, ll &x, ll &y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    ll d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}
```

类欧几里得

```
// f(a,b,c,n) = ∑(i=[0,n]) (ai+b)/c
// g(a,b,c,n) = ∑(i=[0,n]) i*((ai+b)/c)
// h(a,b,c,n) = ∑(i=[0,n]) ((ai+b)/c)^2
ll f(ll a, ll b, ll c, ll n);
ll g(ll a, ll b, ll c, ll n);
```

```
ll h(ll a, ll b, ll c, ll n);

ll f(ll a, ll b, ll c, ll n) {
    if (n < 0) return 0;
    ll m = (a * n + b) / c;
    if (a >= c || b >= c) {
        return (a / c) * n * (n + 1) / 2
        + (b / c) * (n + 1)
        + f(a % c, b % c, c, n);
    } else {
        return n * m - f(c, c - b - 1, a, m - 1);
    }
}

ll g(ll a, ll b, ll c, ll n) {
    if (n < 0) return 0;
    ll m = (a * n + b) / c;
    if (a >= c || b >= c) {
        return (a / c) * n * (n + 1) * (2 * n + 1) / 6
        + (b / c) * n * (n + 1) / 2
        + g(a % c, b % c, c, n);
    } else {
        return (n * (n + 1) * m
        - f(c, c - b - 1, a, m - 1)
        - h(c, c - b - 1, a, m - 1)) / 2;
    }
}

ll h(ll a, ll b, ll c, ll n) {
    if (n < 0) return 0;
    ll m = (a * n + b) / c;
    if (a >= c || b >= c) {
        return (a / c) * (a / c) * n * (n + 1) * (2 * n + 1) / 6
        + (b / c) * (b / c) * (n + 1)
        + (a / c) * (b / c) * n * (n + 1)
        + h(a % c, b % c, c, n)
        + 2 * (a / c) * g(a % c, b % c, c, n)
        + 2 * (b / c) * f(a % c, b % c, c, n);
    } else {
        return n * m * (m + 1)
        - 2 * g(c, c - b - 1, a, m - 1)
        - 2 * f(c, c - b - 1, a, m - 1)
        - f(a, b, c, n);
    }
}
```

逆元

```
ll inv(ll x) { return qk(x, MOD - 2, MOD); }

// EXGCD
// gcd(a, p) = 1 时有逆元
ll inv(ll a, ll p) {
    ll x, y;
    ll d = exgcd(a, p, x, y);
    if (d == 1) return (x % p + p) % p;
    return -1;
}

// 逆元打表
ll inv[MAXN];

void init_inv() {
    inv[1] = 1;
    for (int i = 2; i < MAXN; i++) {
```

```
        inv[i] = 1LL * (MOD - MOD / i) * inv[MOD % i] % MOD;
    }
}
```

组合数

```
// 组合数打表
ll C[MAXN][MAXN];

void initC() {
    C[0][0] = 1;
    for (int i = 1; i < MAXN; i++) {
        C[i][0] = 1;
        for (int j = 1; j <= i; j++) {
            C[i][j] = (C[i - 1][j] + C[i - 1][j - 1]) % MOD;
        }
    }
}

// 快速组合数取模
// MAXN 开 2 倍上限
ll fac[MAXN], ifac[MAXN];

void init_inv() {
    fac[0] = 1;
    for (int i = 1; i < MAXN; i++) {
        fac[i] = fac[i - 1] * i % MOD;
    }
    ifac[MAXN - 1] = qk(fac[MAXN - 1], MOD - 2, MOD);
    for (int i = MAXN - 2; i >= 0; i--) {
        ifac[i] = ifac[i + 1] * (i + 1);
        ifac[i] %= MOD;
    }
}

ll C(int n, int m) {
    if (n < m || m < 0) return 0;
    return fac[n] * ifac[m] % MOD * ifac[n - m] % MOD;
}

// Lucas
ll C(ll n, ll m) {
    if (n < m || m < 0) return 0;
    if (n < MOD && m < MOD) return fac[n] * ifac[m] % MOD * ifac[n - m] % MOD;
    return C(n / MOD, m / MOD) * C(n % MOD, m % MOD) % MOD;
}

// 可重复组合数
ll H(int n, int m) { return C(n + m - 1, m); }
```

康托展开

```
// 需要预处理阶乘
int cantor(vector<int>& s) {
    int n = s.size(), ans = 0;
    for (int i = 0; i < n - 1; i++) {
        int cnt = 0;
        for (int j = i + 1; j < n; j++) {
            if (s[j] < s[i]) cnt++;
        }
        ans += cnt * fac[n - i - 1];
    }
    return ans + 1;
```

```
}

vector<int> inv_cantor(int x, int n) {
    x--;
    vector<int> ans(n), rk(n);
    iota(rk.begin(), rk.end(), 1);
    for (int i = 0; i < n; i++) {
        int t = x / fac[n - i - 1];
        x %= fac[n - i - 1];
        ans[i] = rk[t];
        for (int j = t; rk[j] < n; j++) {
            rk[j] = rk[j + 1];
        }
    }
    return ans;
}
```

高斯消元

```
// n 方程个数，m 变量个数，a 是 n*(m+1) 的增广矩阵，free 是否为自由变量
// 返回自由变量个数，-1 无解
const double EPS = 1e-8;
const int MAXN = 2000 + 7;

double x[MAXN];
bool free_x[MAXN];

int sgn(double x) { return x < -EPS ? -1 : x > EPS; }

int gauss(vector<vector<double> >& a, int n, int m) {
    fill(x, x + m + 1, 0);
    fill(free_x, free_x + m + 1, true);

    // 求上三角矩阵
    int r = 0, c = 0;
    while (r < n && c < m) {
        int mr = r;
        for (int i = r + 1; i < n; i++) {
            if (abs(a[i][c]) > abs(a[mr][c])) mr = i;
        }
        if (mr != r) swap(a[r], a[mr]);
        if (!sgn(a[r][c])) {
            a[r][c] = 0;
            ++c;
            continue;
        }
        for (int i = r + 1; i < n; i++) {
            if (a[i][c]) {
                double t = a[i][c] / a[r][c];
                for (int j = c; j <= m; j++) a[i][j] -= a[r][j] * t;
            }
        }
        ++r, ++c;
    }
    for (int i = r; i < n; i++) {
        if (sgn(a[i][m])) return -1;
    }

    // 求解 x0, x1, ..., xm-1
    if (r < m) {
        for (int i = r - 1; i >= 0; i--) {
            int fcnt = 0, k = -1;
            for (int j = 0; j < m; j++) {
                if (sgn(a[i][j]) && free_x[j]) {
                    ++fcnt;
```

```
                k = j;
            }
        }
        if (fcnt > 0) continue;
        double s = a[i][m];
        for (int j = 0; j < m; j++) {
            if (j != k) s -= a[i][j] * x[j];
        }
        x[k] = s / a[i][k];
        free_x[k] = 0;
    }
    return m - r;
}
for (int i = m - 1; i >= 0; i--) {
    double s = a[i][m];
    for (int j = i + 1; j < m; j++) s -= a[i][j] * x[j];
    x[i] = s / a[i][i];
}
return 0;
}
```

线性基

```
ll a[65];

void insert(ll x) {
    for (int i = 60; i >= 0; i--) {
        if ((x >> i) & 1) {
            if (a[i]) x ^= a[i];
            else { a[i] = x; break; }
        }
    }
}
```

中国剩余定理

```
// 前置：exgcd
ll excrt(vector<ll>& m, vector<ll>& r) {
    ll M = m[0], R = r[0], x, y, d;
    for (int i = 1; i < m.size(); i++) {
        d = exgcd(M, m[i], x, y);
        if ((r[i] - R) % d) return -1;
        x = mul(x, (r[i] - R) / d, m[i] / d);
        R += x * M;
        M = M / d * m[i];
        R %= M;
    }
    return R >= 0 ? R : R + M;
}
```

原根

```
// 前置：找质因数（无重复）
ll primitive_root(ll p) {
    vector<ll> facs = getf(p - 1);
    for (ll i = 2; i < p; i++) {
        bool flag = true;
        for (ll x : facs) {
            if (qk(i, (p - 1) / x, p) == 1) {
                flag = false;
                break;
            }
```

```
        }
        if (flag) return i;
    }
    return -1;
}
```

离散对数

```
// a ^ x = b (mod p), 要求模数为素数
ll BSGS(ll a, ll b, ll p) {
    a %= p;
    if (!a && !b) return 1;
    if (!a) return -1;
    map<ll, ll> mp;
    ll m = ceil(sqrt(p)), v = 1;
    for (int i = 1; i <= m; i++) {
        (v *= a) %= p;
        mp[v * b % p] = i;
    }
    ll vv = v;
    for (int i = 1; i <= m; i++) {
        auto it = mp.find(vv);
        if (it != mp.end()) return i * m - it->second;
        (vv *= v) %= p;
    }
    return -1;
}

// 模数可以非素数
ll exBSGS(ll a, ll b, ll p) {
    a %= p; b %= p;
    if (a == 0) return b > 1 ? -1 : (b == 0 && p != 1);
    ll c = 0, q = 1;
    for (;;) {
        ll g = gcd(a, p);
        if (g == 1) break;
        if (b == 1) return c;
        if (b % g) return -1;
        ++c; b /= g; p /= g; q = a / g * q % p;
    }
    map<ll, ll> mp;
    ll m = ceil(sqrt(p)), v = 1;
    for (int i = 1; i <= m; i++) {
        (v *= a) %= p;
        mp[v * b % p] = i;
    }
    for (int i = 1; i <= m; i++) {
        (q *= v) %= p;
        auto it = mp.find(q);
        if (it != mp.end()) return i * m - it->second + c;
    }
    return -1;
}

// 已知 x, b, p, 求 a
ll SGSB(ll x, ll b, ll p) {
    ll g = primitive_root(p);
    return qk(g, BSGS(qk(g, x, p), b, p), p);
}
```

二次剩余

```
ll Quadratic_residue(ll a) {
    if (a == 0) return 0;
    ll b;
    do b = mt_rand() % MOD;
    while (qk(b, (MOD - 1) >> 1, MOD) != MOD - 1);
    ll s = MOD - 1, t = 0, f = 1;
    while (!(s & 1)) s >>= 1, t++, f <<= 1;
    t--, f >>= 1;
    ll x = qk(a, (s + 1) >> 1, MOD), inv_a = qk(a, MOD - 2, MOD);
    while (t) {
        f >>= 1;
        if (qk(inv_a * x % MOD * x % MOD, f, MOD) != 1) {
            (x *= qk(b, s, MOD)) %= MOD;
        }
        t--, s <<= 1;
    }
    if (x * x % MOD != a) return -1;
    return min(x, MOD - x);
}
```

FFT & NTT & FWT

- FFT

```
const double PI = acos(-1);
using cp = complex<double>;

int n1, n2, n, k, rev[MAXN];

void fft(vector<cp>& a, int p) {
    for (int i = 0; i < n; i++) if (i < rev[i]) swap(a[i], a[rev[i]]);
    for (int h = 1; h < n; h <<= 1) {
        cp wn(cos(PI / h), p * sin(PI / h));
        for (int i = 0; i < n; i += (h << 1)) {
            cp w(1, 0);
            for (int j = 0; j < h; j++, w *= wn) {
                cp x = a[i + j], y = w * a[i + j + h];
                a[i + j] = x + y, a[i + j + h] = x - y;
            }
        }
    }
    if (p == -1) for (int i = 0; i < n; i++) a[i] /= n;
}

void go(vector<cp>& a, vector<cp>& b) {
    n = 1, k = 0;
    while (n <= n1 + n2) n <<= 1, k++;
    a.resize(n); b.resize(n);
    for (int i = 0; i < n; i++) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (k - 1));
    fft(a, 1); fft(b, 1);
    for (int i = 0; i < n; i++) a[i] *= b[i];
    fft(a, -1);
}
```

- NTT

```
const int MOD = 998244353, G = 3, IG = 332748118;

int n1, n2, n, k, rev[MAXN];

void ntt(vector<ll>& a, int p) {
    for (int i = 0; i < n; i++) if (i < rev[i]) swap(a[i], a[rev[i]]);
    for (int h = 1; h < n; h <<= 1) {
        ll wn = qk(p == 1 ? G : IG, (MOD - 1) / (h << 1), MOD);
        for (int i = 0; i < n; i += (h << 1)) {
            ll w = 1;
```

```
            for (int j = 0; j < h; j++, (w *= wn) %= MOD) {
                ll x = a[i + j], y = w * a[i + j + h] % MOD;
                a[i + j] = (x + y) % MOD, a[i + j + h] = (x - y + MOD) % MOD;
            }
        }
    }
    if (p == -1) {
        ll ninv = qk(n, MOD - 2, MOD);
        for (int i = 0; i < n; i++) (a[i] *= ninv) %= MOD;
    }
}

void go(vector<ll>& a, vector<ll>& b) {
    n = 1, k = 0;
    while (n <= n1 + n2) n <<= 1, k++;
    a.resize(n); b.resize(n);
    for (int i = 0; i < n; i++) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (k - 1));
    ntt(a, 1); ntt(b, 1);
    for (int i = 0; i < n; i++) (a[i] *= b[i]) %= MOD;
    ntt(a, -1);
}
```

- FWT

```
void AND(ll& a, ll& b) { a += b; }
void rAND(ll& a, ll& b) { a -= b; }

void OR(ll& a, ll& b) { b += a; }
void rOR(ll& a, ll& b) { b -= a; }

void XOR(ll& a, ll& b) {
    ll x = a, y = b;
    a = (x + y) % MOD;
    b = (x - y + MOD) % MOD;
}
void rXOR(ll& a, ll& b) {
    static ll inv2 = (MOD + 1) / 2;
    ll x = a, y = b;
    a = (x + y) * inv2 % MOD;
    b = (x - y + MOD) * inv2 % MOD;
}

template<class T>
void fwt(vector<ll>& a, int n, T f) {
    for (int d = 1; d < n; d <<= 1) {
        for (int i = 0; i < n; i += (d << 1)) {
            for (int j = 0; j < d; j++) {
                f(a[i + j], a[i + j + d]);
            }
        }
    }
}
```

自适应 Simpson 积分

```
double simpson(double l, double r) {
    double c = (l + r) / 2;
    return (f(l) + 4 * f(c) + f(r)) * (r - l) / 6;
}

double asr(double l, double r, double eps, double S) {
    double mid = (l + r) / 2;
    double L = simpson(l, mid), R = simpson(mid, r);
    if (fabs(L + R - S) < 15 * eps) return L + R + (L + R - S) / 15;
    return asr(l, mid, eps / 2, L) + asr(mid, r, eps / 2, R);
```

```
}

double asr(double l, double r) { return asr(l, r, EPS, simpson(l, r)); }
```

BM 线性递推

```
namespace BerlekampMassey {
    using V = vector<ll>;

    void up(ll & a, ll b) { (a += b) %= MOD; }

    V mul(const V& a, const V& b, const V& m, int k) {
        V r(2 * k - 1);
        for (int i = 0; i < k; i++)
            for (int j = 0; j < k; j++)
                up(r[i + j], a[i] * b[j]);
        for (int i = k - 2; i >= 0; i--) {
            for (int j = 0; j < k; j++)
                up(r[i + j], r[i + k] * m[j]);
            r.pop_back();
        }
        return r;
    }

    V pow(ll n, const V& m) {
        int k = (int)m.size() - 1;
        assert(m[k] == -1 || m[k] == MOD - 1);
        V r(k), x(k);
        r[0] = x[1] = 1;
        for (; n; n >>= 1, x = mul(x, x, m, k))
            if (n & 1) r = mul(x, r, m, k);
        return r;
    }

    ll go(const V& a, const V& x, ll n) {
        // a: (-1, a1, a2, ..., ak).reverse
        // x: x1, x2, ..., xk
        // x[n] = sum[a[i]*x[n-i],{i,1,k}]
        int k = (int)a.size() - 1;
        if (n <= k) return x[n - 1];
        if (a.size() == 2) return x[0] * qk(a[0], n - 1, MOD) % MOD;
        V r = pow(n - 1, a);
        ll ans = 0;
        for (int i = 0; i < k; i++) up(ans, r[i] * x[i]);
        return (ans + MOD) % MOD;
    }

    V BM(const V& x) {
        V C{-1}, B{-1};
        ll L = 0, m = 1, b = 1;
        for (int n = 0; n < (int)x.size(); n++) {
            ll d = 0;
            for (int i = 0; i <= L; i++) up(d, C[i] * x[n - i]);
            if (d == 0) { ++m; continue; }
            V T = C;
            ll c = MOD - d * inv(b, MOD) % MOD;
            C.resize(max(C.size(), size_t(B.size() + m)));
            for (int i = 0; i < (int)B.size(); i++) up(C[i + m], c * B[i]);
            if (2 * L > n) { ++m; continue; }
            L = n + 1 - L; B.swap(T); b = d; m = 1;
        }
        reverse(C.begin(), C.end());
        return C;
    }
}
```

拉格朗日插值

```
// 求 f(k) 的值，O(n^2)
ll La(const vector<pair<ll, ll> >& v, ll k) {
    ll ret = 0;
    for (int i = 0; i < v.size(); i++) {
        ll up = v[i].second % MOD, down = 1;
        for (int j = 0; j < v.size(); j++) {
            if (i != j) {
                (up *= (k - v[j].first) % MOD) %= MOD;
                (down *= (v[i].first - v[j].first) % MOD) %= MOD;
            }
        }
        if (up < 0) up += MOD;
        if (down < 0) down += MOD;
        (ret += up * inv(down) % MOD) %= MOD;
    }
    return ret;
}

// 求 f(x) 的系数表达式，O(n * 2^n)（适合打表）
vector<double> La(vector<pair<double, double> > v) {
    int n = v.size(), t;
    vector<double> ret(n);
    double p, q;
    for (int i = 0; i < n; i++) {
        p = v[i].second;
        for (int j = 0; j < n; j++) {
            p /= (i == j) ? 1 : (v[i].first - v[j].first);
        }
        for (int j = 0; j < (1 << n); j++) {
            q = 1, t = 0;
            for (int k = 0; k < n; k++) {
                if (i == k) continue;
                if ((j >> k) & 1) q *= -v[k].first;
                else t++;
            }
            ret[t] += p * q / 2;
        }
    }
    return ret;
}
```

计算几何

二维几何基础

```
#define y1 qwq

using ld = double;

const ld PI = acos(-1);
const ld EPS = 1e-8;

int sgn(ld x) { return x < -EPS ? -1 : x > EPS; }

// 不要直接使用 sgn
bool eq(ld x, ld y) { return sgn(x - y) == 0; }
bool lt(ld x, ld y) { return sgn(x - y) < 0; }
bool gt(ld x, ld y) { return sgn(x - y) > 0; }
bool leq(ld x, ld y) { return sgn(x - y) <= 0; }
bool geq(ld x, ld y) { return sgn(x - y) >= 0; }

struct V {
```

```
    ld x, y;
    V(ld x = 0, ld y = 0) : x(x), y(y) {}
    V(const V& a, const V& b) : x(b.x - a.x), y(b.y - a.y) {}
    V operator + (const V& b) const { return V(x + b.x, y + b.y); }
    V operator - (const V& b) const { return V(x - b.x, y - b.y); }
    V operator * (ld k) const { return V(x * k, y * k); }
    V operator / (ld k) const { return V(x / k, y / k); }
    ld len() const { return hypot(x, y); }
    ld len2() const { return x * x + y * y; }
};

ostream& operator << (ostream& os, const V& p) { return os << "(" << p.x << ", " << p.y << ")"; }
istream& operator >> (istream& is, V& p) { return is >> p.x >> p.y; }

ld dist(const V& a, const V& b) { return (b - a).len(); }
ld dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
ld det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
ld cross(const V& s, const V& t, const V& o) { return det(V(o, s), V(o, t)); }

ld to_rad(ld deg) { return deg / 180 * PI; }

// 象限
int quad(const V& p) {
    int x = sgn(p.x), y = sgn(p.y);
    if (x > 0 && y >= 0) return 1;
    if (x <= 0 && y > 0) return 2;
    if (x < 0 && y <= 0) return 3;
    if (x >= 0 && y < 0) return 4;
    assert(0);
}

// 极角排序
struct cmp_angle {
    V p;
    cmp_angle(const V& p = V()) : p(p) {}
    bool operator () (const V& a, const V& b) const {
        int qa = quad(a - p), qb = quad(b - p);
        if (qa != qb) return qa < qb;
        int d = sgn(cross(a, b, p));
        if (d) return d > 0;
        return dist(a, p) < dist(b, p);
    }
};

// 单位向量
V unit(const V& p) { return eq(p.len(), 0) ? V(1, 0) : p / p.len(); }

// 逆时针旋转 r 弧度
V rot(const V& p, ld r) {
    return V(p.x * cos(r) - p.y * sin(r), p.x * sin(r) + p.y * cos(r));
}
V rot_ccw90(const V& p) { return V(-p.y, p.x); }
V rot_cw90(const V& p) { return V(p.y, -p.x); }

// 点在线段上 leq(dot(...), 0) 包含端点 lt(dot(...), 0) 则不包含
bool p_on_seg(const V& p, const V& a, const V& b) {
    return eq(det(p - a, b - a), 0) && leq(dot(p - a, p - b), 0);
}

// 点在射线上 geq(dot(...), 0) 包含端点 gt(dot(...), 0) 则不包含
bool p_on_ray(const V& p, const V& a, const V& b) {
    return eq(det(p - a, b - a), 0) && geq(dot(p - a, b - a), 0);
}

// 点到直线距离
ld dist_to_line(const V& p, const V& a, const V& b) {
```

```
    return abs(cross(a, b, p) / dist(a, b));
}

// 点到线段距离
ld dist_to_seg(const V& p, const V& a, const V& b) {
    if (lt(dot(b - a, p - a), 0)) return dist(p, a);
    if (lt(dot(a - b, p - b), 0)) return dist(p, b);
    return dist_to_line(p, a, b);
}

// 求直线交点
V intersect(const V& a, const V& b, const V& c, const V& d) {
    ld s1 = cross(c, d, a), s2 = cross(c, d, b);
    return (a * s2 - b * s1) / (s2 - s1);
}

// 三角形重心
V centroid(const V& a, const V& b, const V& c) {
    return (a + b + c) / 3;
}

// 内心
V incenter(const V& a, const V& b, const V& c) {
    ld AB = dist(a, b), AC = dist(a, c), BC = dist(b, c);
    // ld r = abs(cross(b, c, a)) / (AB + AC + BC);
    return (a * BC + b * AC + c * AB) / (AB + BC + AC);
}

// 外心
V circumcenter(const V& a, const V& b, const V& c) {
    V mid1 = (a + b) / 2, mid2 = (a + c) / 2;
    // ld r = dist(a, b) * dist(b, c) * dist(c, a) / 2 / abs(cross(b, c, a));
    return intersect(mid1, mid1 + rot_ccw90(b - a), mid2, mid2 + rot_ccw90(c - a));
}

// 垂心
V orthocenter(const V& a, const V& b, const V& c) {
    return centroid(a, b, c) * 3 - circumcenter(a, b, c) * 2;
}

// 旁心（三个）
vector<V> escenter(const V& a, const V& b, const V& c) {
    ld AB = dist(a, b), AC = dist(a, c), BC = dist(b, c);
    V p1 = (a * (-BC) + b * AC + c * AB) / (AB + AC - BC);
    V p2 = (a * BC + b * (-AC) + c * AB) / (AB - AC + BC);
    V p3 = (a * BC + b * AC + c * (-AB)) / (-AB + AC + BC);
    return {p1, p2, p3};
}
```

多边形

```
// 多边形面积
ld area(const vector<V>& s) {
    ld ret = 0;
    for (int i = 0; i < s.size(); i++) {
        ret += det(s[i], s[(i + 1) % s.size()]);
    }
    return ret / 2;
}

// 多边形重心
V centroid(const vector<V>& s) {
    V c;
    for (int i = 0; i < s.size(); i++) {
        c = c + (s[i] + s[(i + 1) % s.size()]) * det(s[i], s[(i + 1) % s.size()]);
```

```
    }
    return c / 6.0 / area(s);
}

// 点是否在多边形中
// 1 inside 0 on border -1 outside
int inside(const vector<V>& s, const V& p) {
    int cnt = 0;
    for (int i = 0; i < s.size(); i++) {
        V a = s[i], b = s[(i + 1) % s.size()];
        if (p_on_seg(p, a, b)) return 0;
        if (leq(a.y, b.y)) swap(a, b);
        if (gt(p.y, a.y)) continue;
        if (leq(p.y, b.y)) continue;
        cnt += gt(cross(b, a, p), 0);
    }
    return (cnt & 1) ? 1 : -1;
}

// 构建凸包 点不可以重复
// lt(cross(...), 0) 边上可以有点 leq(cross(...), 0) 则不能
// 会改变输入点的顺序
vector<V> convex_hull(vector<V>& s) {
    // assert(s.size() >= 3);
    sort(s.begin(), s.end(), [](V &a, V &b) { return eq(a.x, b.x) ? lt(a.y, b.y) : lt(a.x, b.x); });
    vector<V> ret(2 * s.size());
    int sz = 0;
    for (int i = 0; i < s.size(); i++) {
        while (sz > 1 && leq(cross(ret[sz - 1], s[i], ret[sz - 2]), 0)) sz--;
        ret[sz++] = s[i];
    }
    int k = sz;
    for (int i = s.size() - 2; i >= 0; i--) {
        while (sz > k && leq(cross(ret[sz - 1], s[i], ret[sz - 2]), 0)) sz--;
        ret[sz++] = s[i];
    }
    ret.resize(sz - (s.size() > 1));
    return ret;
}

// 多边形是否为凸包
bool is_convex(const vector<V>& s) {
    for (int i = 0; i < s.size(); i++) {
        if (lt(cross(s[(i + 1) % s.size()], s[(i + 2) % s.size()], s[i]), 0)) return false;
    }
    return true;
}

// 点是否在凸包中
// 1 inside 0 on border -1 outside
int inside(const vector<V>& s, const V& p) {
    for (int i = 0; i < s.size(); i++) {
        if (lt(cross(s[i], s[(i + 1) % s.size()], p), 0)) return -1;
        if (p_on_seg(p, s[i], s[(i + 1) % s.size()])) return 0;
    }
    return 1;
}
```

圆

```
struct C {
    V o;
    ld r;
    C(const V& o, ld r) : o(o), r(r) {}
};
```

```cpp
// 过一点求圆的切线，返回切点
vector<V> tangent_point(const C& c, const V& p) {
    ld k = c.r / dist(c.o, p);
    if (gt(k, 1)) return vector<V>();
    if (eq(k, 1)) return {p};
    V a = V(c.o, p) * k;
    return {c.o + rot(a, acos(k)), c.o + rot(a, -acos(k))};
}

// 最小圆覆盖
C min_circle_cover(vector<V> a) {
    shuffle(a.begin(), a.end(), mt_rand);
    V o = a[0];
    ld r = 0;
    int n = a.size();
    for (int i = 1; i < n; i++) if (gt(dist(a[i], o), r)) {
        o = a[i]; r = 0;
        for (int j = 0; j < i; j++) if (gt(dist(a[j], o), r)) {
            o = (a[i] + a[j]) / 2;
            r = dist(a[j], o);
            for (int k = 0; k < j; k++) if (gt(dist(a[k], o), r)) {
                o = circumcenter(a[i], a[j], a[k]);
                r = dist(a[k], o);
            }
        }
    }
    return C(o, r);
}
```

杂项

防爆 vector

```cpp
template<class T>
class vector_s : public vector<T> {
public:
    vector_s(size_t n = 0, const T& x = T()) : vector<T>(n, x) {}
    T& operator [] (size_t n) { return this->at(n); }
    const T& operator [] (size_t n) const { return this->at(n); }
};

#define vector vector_s
```

pair_hash

```cpp
template<class T1, class T2>
struct pair_hash {
    size_t operator () (const pair<T1, T2>& p) const {
        return hash<T1>()(p.first) * 19260817 + hash<T2>()(p.second);
    }
};

unordered_set<pair<int, int>, pair_hash<int, int> > st;
unordered_map<pair<int, int>, int, pair_hash<int, int> > mp;
```

updmax/min

```cpp
template<class T> inline bool updmax(T &a, T b) { return a < b ? a = b, 1 : 0; }
template<class T> inline bool updmin(T &a, T b) { return a > b ? a = b, 1 : 0; }
```

离散化

```cpp
// 重复元素 id 不同
template<class T>
vector<int> dc(const vector<T>& a, int start_id) {
    int n = a.size();
    vector<pair<T, int> > t(n);
    for (int i = 0; i < n; i++) {
        t[i] = make_pair(a[i], i);
    }
    sort(t.begin(), t.end());
    vector<int> id(n);
    for (int i = 0; i < n; i++) {
        id[t[i].second] = start_id + i;
    }
    return id;
}

// 重复元素 id 相同
template<class T>
vector<int> unique_dc(const vector<T>& a, int start_id) {
    int n = a.size();
    vector<T> t(a);
    sort(t.begin(), t.end());
    t.resize(unique(t.begin(), t.end()) - t.begin());
    vector<int> id(n);
    for (int i = 0; i < n; i++) {
        id[i] = start_id + lower_bound(t.begin(), t.end(), a[i]) - t.begin();
    }
    return id;
}
```

加强版优先队列

```cpp
struct heap {
    priority_queue<int> q1, q2;
    void push(int x) { q1.push(x); }
    void erase(int x) { q2.push(x); }
    int top() {
        while (q2.size() && q1.top() == q2.top()) q1.pop(), q2.pop();
        return q1.top();
    }
    void pop() {
        while (q2.size() && q1.top() == q2.top()) q1.pop(), q2.pop();
        q1.pop();
    }
    int size() { return q1.size() - q2.size(); }
};
```

分数

```cpp
struct Frac {
    ll x, y;

    Frac(ll p = 0, ll q = 1) {
        ll d = __gcd(p, q);
        x = p / d, y = q / d;
        if (y < 0) x = -x, y = -y;
    }

    Frac operator + (const Frac& b) { return Frac(x * b.y + y * b.x, y * b.y); }
    Frac operator - (const Frac& b) { return Frac(x * b.y - y * b.x, y * b.y); }
    Frac operator * (const Frac& b) { return Frac(x * b.x, y * b.y); }
```

```
    Frac operator / (const Frac& b) { return Frac(x * b.y, y * b.x); }
};

ostream& operator << (ostream& os, const Frac& f) {
    if (f.y == 1) return os << f.x;
    else return os << f.x << '/' << f.y;
}
```

二分答案

```
// 二分闭区间 [l, r]
// 可行下界
while (l < r) {
    mid = (l + r) / 2;
    if (check(mid)) r = mid;
    else l = mid + 1;
}

// 可行上界
while (l < r) {
    mid = (l + r + 1) / 2;
    if (check(mid)) l = mid;
    else r = mid - 1;
}
```

三分

```
// 实数范围
double l, r, mid1, mid2;
for (int i = 0; i < 75; i++) {
    mid1 = (l * 5 + r * 4) / 9;
    mid2 = (l * 4 + r * 5) / 9;
    if (f(mid1) > f(mid2)) r = mid2; // 单峰函数取'>'号，单谷函数取'<'号
    else l = mid1;
}

// 整数范围
int l, r, mid1, mid2;
while (l < r - 2) {
    mid1 = (l + r) / 2;
    mid2 = mid1 + 1;
    if (f(mid1) > f(mid2)) r = mid2; // 单峰函数取'>'号，单谷函数取'<'号
    else l = mid1;
}
int maxval = f(l), ans = l;
for (int i = l + 1; i <= r; i++) {
    if (updmax(maxval, f(i))) ans = i;
}
```

日期

```
// 0 ~ 6 对应 周一 ~ 周日
int zeller(int y, int m, int d) {
    if (m <= 2) m += 12, y--;
    return (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 - y / 100 + y / 400) % 7;
}

// date_to_int(1, 1, 1) = 1721426
// date_to_int(2019, 10, 27) = 2458784
int date_to_int(int y, int m, int d) {
    return
    1461 * (y + 4800 + (m - 14) / 12) / 4 +
```

```
    367 * (m - 2 - (m - 14) / 12 * 12) / 12 -
    3 * ((y + 4900 + (m - 14) / 12) / 100) / 4 +
    d - 32075;
}

void int_to_date(int jd, int &y, int &m, int &d) {
    int x, n, i, j;

    x = jd + 68569;
    n = 4 * x / 146097;
    x -= (146097 * n + 3) / 4;
    i = (4000 * (x + 1)) / 1461001;
    x -= 1461 * i / 4 - 31;
    j = 80 * x / 2447;
    d = x - 2447 * j / 80;
    x = j / 11;
    m = j + 2 - 12 * x;
    y = 100 * (n - 49) + i + x;
}
```

子集枚举

```
// 枚举真子集
for (int t = (x - 1) & x; t; t = (t - 1) & x)

// 枚举大小为 k 的子集
// 注意 k 不能为 0
void subset(int k, int n) {
    int t = (1 << k) - 1;
    while (t < (1 << n)) {
        // do something
        int x = t & -t, y = t + x;
        t = ((t & ~y) / x >> 1) | y;
    }
}
```

最长上升子序列

```
vector<int> dp(n, INF);
for (int i = 0; i < n; i++) {
    // 最长不下降 upper_bound
    *lower_bound(dp.begin(), dp.end(), a[i]) = a[i];
}
```

数位 dp

```
// 小于等于 x 的 base 进制下回文数个数
ll dp[20][20][20][2], tmp[20], a[20];

ll dfs(ll base, ll pos, ll len, ll s, bool limit) {
    if (pos == -1) return s;
    if (!limit && dp[base][pos][len][s] != -1) return dp[base][pos][len][s];
    ll ret = 0;
    ll ed = limit ? a[pos] : base - 1;
    for (int i = 0; i <= ed; i++) {
        tmp[pos] = i;
        if (len == pos)
            ret += dfs(base, pos - 1, len - (i == 0), s, limit && i == a[pos]);
        else if (s && pos < (len + 1) / 2)
            ret += dfs(base, pos - 1, len, tmp[len - pos] == i, limit && i == a[pos]);
        else
            ret += dfs(base, pos - 1, len, s, limit && i == a[pos]);
```

```
    }
    if (!limit) dp[base][pos][len][s] = ret;
    return ret;
}

ll solve(ll x, ll base) {
    memset(dp, -1, sizeof(dp));
    ll sz = 0;
    while (x) {
        a[sz++] = x % base;
        x /= base;
    }
    return dfs(base, sz - 1, sz - 1, 1, true);
}
```

表达式求值

```
print(input()) # Python2
print(eval(input())) # Python3
```

对拍

- *unix

```
#!/bin/bash
cd "$(dirname "${BASH_SOURCE[0]}")"

g++ gen.cpp -o gen -O2 -std=c++11
g++ my.cpp -o my -O2 -std=c++11
g++ std.cpp -o std -O2 -std=c++11

while true
do
    ./gen > in.txt
    ./std < in.txt > stdout.txt
    ./my < in.txt > myout.txt

    if test $? -ne 0
    then
        printf "RE\n"
        exit 0
    fi

    if diff stdout.txt myout.txt
    then
        printf "AC\n"
    else
        printf "WA\n"
        exit 0
    fi
done
```

- Windows

```
@echo off

g++ gen.cpp -o gen.exe -O2 -std=c++11
g++ my.cpp -o my.exe -O2 -std=c++11
g++ std.cpp -o std.exe -O2 -std=c++11

:loop
    gen.exe > in.txt
    std.exe < in.txt > stdout.txt
    my.exe < in.txt > myout.txt
    if errorlevel 1 (
```

```
        echo RE
        pause
        exit
    )
    fc stdout.txt myout.txt
    if errorlevel 1 (
        echo WA
        pause
        exit
    )
goto loop
```

Java

- Main

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

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        PrintStream out = System.out;

    }
}
```

- 皮特老师读入挂

```
public class Main {
    public static void main(String[] args) {
        InputStream inputStream = System.in;
        OutputStream outputStream = System.out;
        InputReader in = new InputReader(inputStream);
        PrintWriter out = new PrintWriter(outputStream);

        out.close();
    }

    static class InputReader {
        public BufferedReader reader;
        public StringTokenizer tokenizer;

        public InputReader(InputStream stream) {
            reader = new BufferedReader(new InputStreamReader(stream), 32768);
            tokenizer = null;
        }

        public String next() {
            while (tokenizer == null || !tokenizer.hasMoreTokens()) {
                try {
                    tokenizer = new StringTokenizer(reader.readLine());
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
            return tokenizer.nextToken();
        }

        public int nextInt() {
            return Integer.parseInt(next());
        }
    }
}
```

- 大整数

```java
import java.math.BigInteger;

BigInteger.ZERO
BigInteger.ONE
BigInteger.TWO // since Java 9
BigInteger.TEN
BigInteger.valueOf(2)

BigInteger abs()
BigInteger negate() // -this

BigInteger add(BigInteger x)
BigInteger subtract(BigInteger x)
BigInteger multiply(BigInteger x)
BigInteger divide(BigInteger x)

BigInteger pow(int exp)
BigInteger sqrt() // since Java 9

BigInteger mod(BigInteger m)
BigInteger modPow(BigInteger exp, BigInteger m)
BigInteger modInverse(BigInteger m)

boolean isProbablePrime(int certainty) // probability: 1 - (1/2) ^ (certainty)

BigInteger gcd(BigInteger x)

BigInteger not() // ~this
BigInteger and(BigInteger x)
BigInteger or(BigInteger x)
BigInteger xor(BigInteger x)
BigInteger shiftLeft(int n)
BigInteger shiftRight(int n)

int compareTo(BigInteger x) // -1, 0, 1
BigInteger max(BigInteger x)
BigInteger min(BigInteger x)

int intValue()
long longValue()
String toString()

public static BigInteger getsqrt(BigInteger n) {
    if (n.compareTo(BigInteger.ZERO) <= 0) return n;
    BigInteger x, xx, txx;
    xx = x = BigInteger.ZERO;
    for (int t = n.bitLength() / 2; t >= 0; t--) {
        txx = xx.add(x.shiftLeft(t + 1)).add(BigInteger.ONE.shiftLeft(t + t));
        if (txx.compareTo(n) <= 0) {
            x = x.add(BigInteger.ONE.shiftLeft(t));
            xx = txx;
        }
    }
    return x;
}
```

- 浮点数格式

```java
import java.text.DecimalFormat;

DecimalFormat fmt;

// String s = fmt.format(...)

// round to at most 2 digits, leave of digits if not needed
fmt = new DecimalFormat("#.##");
```

```
// 12345.6789 -> "12345.68"
// 12345.0 -> "12345"
// 0.0 -> "0"
// 0.01 -> ".1"

// round to precisely 2 digits
fmt = new DecimalFormat("#.00");
// 12345.6789 -> "12345.68"
// 12345.0 -> "12345.00"
// 0.0 -> ".00"

// round to precisely 2 digits, force leading zero
fmt = new DecimalFormat("0.00");
// 12345.6789 -> "12345.68"
// 12345.0 -> "12345.00"
// 0.0 -> "0.00"

// round to precisely 2 digits, force leading zeros
fmt = new DecimalFormat("000000000.00");
// 12345.6789 -> "000012345.68"
// 12345.0 -> "000012345.00"
// 0.0 -> "000000000.00"
```

pb_ds

```
// 平衡树
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
template<class T>
using rank_set = tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>;
template<class Key, class T>
using rank_map = tree<Key, T, less<Key>, rb_tree_tag, tree_order_statistics_node_update>;

// 优先队列
#include <ext/pb_ds/priority_queue.hpp>
using namespace __gnu_pbds;
template<class T, class Cmp = less<T> >
using pair_heap = __gnu_pbds::priority_queue<T, Cmp>;
```

待验证

版权归原作者所有 部分代码有风格调整 不保证内容的正确性

约瑟夫问题

```
// n 个人，1 至 m 报数，问最后留下来的人的编号
// 公式：f(n,m)=(f(n-1,m)+m)%n, f(0,m)=0;
// O(n)
ll calc(int n, ll m) {
    ll p = 0;
    for (int i = 2; i <= n; i++) {
        p = (p + m) % i;
    }
    return p + 1;
}

// n 个人，1 至 m 报数，问第 k 个出局的人的编号
// 公式：f(n,k)=(f(n-1,k-1)+m-1)%n+1
// f(n-k+1,1)=m%(n-k+1)
// if (f==0) f=n-k+1
// O(k)
ll cal1(ll n, ll m, ll k) {  // (k == n) equal(calc)
    ll p = m % (n - k + 1);
```

```
    if (p == 0) p = n - k + 1;
    for (ll i = 2; i <= k; i++) {
        p = (p + m - 1) % (n - k + i) + 1;
    }
    return p;
}

// n 个人，1 至 m 报数，问第 k 个出局的人的编号
// O(m*log(m))
ll cal2(ll n, ll m, ll k) {
    if (m == 1)
        return k;
    else {
        ll a = n - k + 1, b = 1;
        ll c = m % a, x = 0;
        if (c == 0) c = a;
        while (b + x <= k) {
            a += x, b += x, c += m * x;
            c %= a;
            if (c == 0) c = a;
            x = (a - c) / (m - 1) + 1;
        }
        c += (k - b) * m;
        c %= n;
        if (c == 0) c = n;
        return c;
    }
}

// n 个人，1 至 m 报数，问编号为 k 的人是第几个出局的
// O(n)
ll n, k;  //可做 n<=4e7，询问个数<=100，下标范围 [0,n-1]
ll dieInXturn(int n, int k, int x) {  // n 个人，报数 k，下标为 X 的人第几个死亡
    ll tmp = 0;
    while (n) {
        x = (x + n) % n;
        if (k > n) x += (k - x - 1 + n - 1) / n * n;
        if ((x + 1) % k == 0) {
            tmp += (x + 1) / k;
            break;
        } else {
            if (k > n) {
                tmp += x / k;
                ll ttmp = x;
                x = x - (x / n + 1) * (x / k) + (x + n) / n * n - k;
                n -= ttmp / k;
            } else {
                tmp += n / k;
                x = x - x / k;
                x += n - n / k * k;
                n -= n / k;
            }
        }
    }
    return tmp;
}
```

二分图最大权匹配 KM

```
// ECNU
namespace R {
    int n;
    int w[MAXN][MAXN], kx[MAXN], ky[MAXN], py[MAXN], vy[MAXN], slk[MAXN], pre[MAXN];

    ll go() {
```

```
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                kx[i] = max(kx[i], w[i][j]);
        for (int i = 1; i <= n; i++) {
            fill(vy, vy + n + 1, 0);
            fill(slk, slk + n + 1, INF);
            fill(pre, pre + n + 1, 0);
            int k = 0, p = -1;
            for (py[k = 0] = i; py[k]; k = p) {
                int d = INF;
                vy[k] = 1;
                int x = py[k];
                for (int j = 1; j <= n; j++) {
                    if (!vy[j]) {
                        int t = kx[x] + ky[j] - w[x][j];
                        if (t < slk[j]) { slk[j] = t; pre[j] = k; }
                        if (slk[j] < d) { d = slk[j]; p = j; }
                    }
                }
                for (int j = 0; j <= n; j++) {
                    if (vy[j]) { kx[py[j]] -= d; ky[j] += d; }
                    else slk[j] -= d;
                }
            }
            for (; k; k = pre[k]) py[k] = py[pre[k]];
        }
        ll ans = 0;
        for (int i = 1; i <= n; i++) ans += kx[i] + ky[i];
        return ans;
    }
}
```

上下界网络流

```
// wxh
const int INF = 0x3f3f3f3f;

struct edge {
    int to, cap, rev;
};

const int MAXN = 60003;
const int MAXM = 400003;

struct graph {
    int n, m;
    edge w[MAXM];
    int fr[MAXM];
    int num[MAXN], cur[MAXN], first[MAXN];
    edge e[MAXM];

    void init(int n) {
        this->n = n;
        m = 0;
    }

    void add_edge(int from, int to, int cap) {
        w[++m] = (edge){to, cap};
        num[from]++, fr[m] = from;
        w[++m] = (edge){from, 0};
        num[to]++, fr[m] = to;
    }

    void prepare() {
        first[1] = 1;
```

```
        for (int i = 2; i <= n; i++) first[i] = first[i - 1] + num[i - 1];
        for (int i = 1; i < n; i++) num[i] = first[i + 1] - 1;
        num[n] = m;
        for (int i = 1; i <= m; i++) {
            e[first[fr[i]] + (cur[fr[i]]++)] = w[i];

            if (!(i % 2)) {
                e[first[fr[i]] + cur[fr[i]] - 1].rev =
                    first[w[i].to] + cur[w[i].to] - 1;
                e[first[w[i].to] + cur[w[i].to] - 1].rev =
                    first[fr[i]] + cur[fr[i]] - 1;
            }
        }
    }

    int q[MAXN];
    int dist[MAXN];
    int t;

    bool bfs(int s) {
        int l = 1, r = 1;
        q[1] = s;
        memset(dist, -1, (n + 1) * 4);
        dist[s] = 0;
        while (l <= r) {
            int u = q[l++];
            for (int i = first[u]; i <= num[u]; i++) {
                int v = e[i].to;
                if ((dist[v] != -1) || (!e[i].cap)) continue;
                dist[v] = dist[u] + 1;
                if (v == t) return true;
                q[++r] = v;
            }
        }
        return dist[t] != -1;
    }

    int dfs(int u, int flow) {
        if (u == t) return flow;
        for (int& i = cur[u]; i <= num[u]; i++) {
            int v = e[i].to;
            if (!e[i].cap || dist[v] != dist[u] + 1) continue;
            int t = dfs(v, min(flow, e[i].cap));
            if (t) {
                e[i].cap -= t;
                e[e[i].rev].cap += t;
                return t;
            }
        }
        return 0;
    }

    ll dinic(int s, int t) {
        ll ans = 0;
        this->t = t;
        while (bfs(s)) {
            int flow;
            for (int i = 1; i <= n; i++) cur[i] = first[i];
            while (flow = dfs(s, INF)) ans += (ll)flow;
        }
        return ans;
    }
};

struct graph_bounds {
    int in[MAXN];
```

```
    int S, T, sum, cur;
    graph g;
    int n;

    void init(int n) {
        this->n = n;
        S = n + 1;
        T = n + 2;
        sum = 0;
        g.init(n + 2);
    }

    void add_edge(int from, int to, int low, int up) {
        g.add_edge(from, to, up - low);
        in[to] += low;
        in[from] -= low;
    }

    void build() {
        for (int i = 1; i <= n; i++)
            if (in[i] > 0)
                g.add_edge(S, i, in[i]), sum += in[i];
            else if (in[i])
                g.add_edge(i, T, -in[i]);
        g.prepare();
    }

    bool canflow() {
        build();
        int flow = g.dinic(S, T);
        return flow >= sum;
    }

    bool canflow(int s, int t) {
        g.add_edge(t, s, INF);
        build();
        for (int i = 1; i <= g.m; i++) {
            edge& e = g.e[i];
            if (e.to == s && e.cap == INF) {
                cur = i;
                break;
            }
        }
        int flow = g.dinic(S, T);
        return flow >= sum;
    }

    int maxflow(int s, int t) {
        if (!canflow(s, t)) return -1;
        return g.dinic(s, t);
    }

    int minflow(int s, int t) {
        if (!canflow(s, t)) return -1;
        edge& e = g.e[cur];
        int flow = INF - e.cap;
        e.cap = g.e[e.rev].cap = 0;
        return flow - g.dinic(t, s);
    }
} g;

void solve() {
    int n = read(), m = read(), s = read(), t = read();
    g.init(n);
    while (m--) {
        int u = read(), v = read(), low = read(), up = read();
```

```
        g.add_edge(u, v, low, up);
    }
}
```

Link-Cut Tree

```
// Chestnut
const int MAXN = 50005;

#define lc son[x][0]
#define rc son[x][1]

struct Splay {
    int fa[MAXN], son[MAXN][2];
    int st[MAXN];
    bool rev[MAXN];
    inline int which(int x) {
        for (int i = 0; i < 2; i++)
            if (son[fa[x]][i] == x) return i;
        return -1;
    }

    inline void pushdown(int x) {
        if (rev[x]) {
            rev[x] ^= 1;
            rev[lc] ^= 1;
            rev[rc] ^= 1;
            swap(lc, rc);
        }
    }

    inline void rotate(int x) {
        int f = fa[x], w = which(x) ^ 1, c = son[x][w];
        fa[x] = fa[f];
        if (which(f) != -1) son[fa[f]][which(f)] = x;
        fa[c] = f;
        son[f][w ^ 1] = c;
        fa[f] = x;
        son[x][w] = f;
    }

    inline void splay(int x) {
        int top = 0;
        st[++top] = x;
        for (int i = x; which(i) != -1; i = fa[i]) {
            st[++top] = fa[i];
        }
        for (int i = top; i; i--) pushdown(st[i]);
        while (which(x) != -1) {
            int f = fa[x];
            if (which(f) != -1) {
                if (which(x) ^ which(f)) rotate(x);
                else rotate(f);
            }
            rotate(x);
        }
    }

    void access(int x) {
        int t = 0;
        while (x) {
            splay(x);
            rc = t;
            t = x;
            x = fa[x];
```

```
        }
    }

    void rever(int x) {
        access(x);
        splay(x);
        rev[x] ^= 1;
    }

    void link(int x, int y) {
        rever(x);
        fa[x] = y;
        splay(x);
    }

    void cut(int x, int y) {
        rever(x);
        access(y);
        splay(y);
        son[y][0] = fa[x] = 0;
    }

    int find(int x) {
        access(x);
        splay(x);
        int y = x;
        while (son[y][0]) y = son[y][0];
        return y;
    }
} T;

int n, m;

int main() {
    char ch[10];
    int x, y;
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++) {
        scanf("%s", ch);
        scanf("%d%d", &x, &y);
        if (ch[0] == 'C') T.link(x, y);
        else if (ch[0] == 'D') T.cut(x, y);
        else {
            if (T.find(x) == T.find(y)) printf("Yes\n");
            else printf("No\n");
        }
    }
}
```

后缀自动机

```
// Chestnut
char s[50100];

struct samnode {
    samnode *par, *ch[26];
    int val;
    samnode() {
        par = 0;
        memset(ch, 0, sizeof(ch));
        val = 0;
    }
} node[100100], *root, *last;

int size = 0;
```

```
inline void init() { last = root = &node[0]; }

inline void add(int c) {
    samnode *p = last;
    samnode *np = &node[++size];
    np->val = p->val + 1;
    while (p && !p->ch[c])
        p->ch[c] = np, p = p->par;
    if (!p) np->par = root;
    else {
        samnode *q = p->ch[c];
        if (q->val == p->val + 1)
            np->par = q;
        else {
            samnode *nq = &node[++size];
            nq->val = p->val + 1;
            memcpy(nq->ch, q->ch, sizeof(q->ch));
            nq->par = q->par;
            q->par = np->par = nq;
            while (p && p->ch[c] == q)
                p->ch[c] = nq, p = p->par;
        }
    }
    last = np;
}

int main() {
    init();
    scanf("%s", s);
    int n = strlen(s), ans = 0;
    for (int i = 0; i < n; i++) add(s[i] - 'A');
    for (int i = 1; i <= size; i++) ans += node[i].val - node[i].par->val;
    printf("%d\n", ans);
    return 0;
}
```

- 广义后缀自动机

```
// Chestnut
int v[100005], head[100005], tot, d[100005];

struct node {
    node *fa, *go[11];
    int max;
} *root, pool[4000005], *cnt;

struct edge {
    int go, next;
} e[100005];

void add(int x, int y) {
    e[++tot] = (edge){y, head[x]}; head[x] = tot;
    e[++tot] = (edge){x, head[y]}; head[y] = tot;
}

void init() { cnt = root = pool + 1; }

node *newnode(int _val) {
    (++cnt)->max = _val;
    return cnt;
}

ostream& operator , (ostream& os, int a) {}

node *extend(node *p, int c) {
    node *np = newnode(p->max + 1);
```

```
    while (p && !p->go[c]) p->go[c] = np, p = p->fa;
    if (!p) np->fa = root;
    else {
        node *q = p->go[c];
        if (p->max + 1 == q->max) np->fa = q;
        else {
            node *nq = newnode(p->max + 1);
            memcpy(nq->go, q->go, sizeof q->go);
            nq->fa = q->fa;
            np->fa = q->fa = nq;
            while (p && p->go[c] == q) p->go[c] = nq, p = p->fa;
        }
    }
    return np;
}

ll solve() {
    ll ans = 0;
    for (node *i = root + 1; i <= cnt; i++)
        ans += i->max - i->fa->max;
    return ans;
}

void dfs(int x, int fa, node *p) {
    node *t = extend(p, v[x]);
    for (int i = head[x]; i; i = e[i].next)
        if (e[i].go != fa)
            dfs(e[i].go, x, t);
}

int n, c, x, y;

int main() {
    init();
    scanf("%d%d", &n, &c);
    for (int i = 1; i <= n; i++) scanf("%d", &v[i]);
    for (int i = 1; i < n; i++) {
        scanf("%d%d", &x, &y);
        add(x, y);
        d[x]++, d[y]++;
    }
    for (int i = 1; i <= n; i++)
        if (d[i] == 1) dfs(i, 0, pool + 1);
    printf("%lld", solve());
}
```

任意模数 NTT

```
// memset0
const int MAXN = 4e5 + 10, G = 3, P[3] = {469762049, 998244353, 1004535809};
int n1, n2, k, n, p, p1, p2, M2;
int a[MAXN], b[MAXN], f[3][MAXN], g[MAXN], rev[MAXN], ans[MAXN];

void ntt(int *a, int g, int p) {
    for (int i = 0; i < n; i++) if (i < rev[i]) swap(a[i], a[rev[i]]);
    for (int len = 1; len < n; len <<= 1) {
        int wn = qk(g, (p - 1) / (len << 1), p);
        for (int i = 0; i < n; i += (len << 1)) {
            int w = 1;
            for (int j = 0; j < len; j++, w = (ll)w * wn % p) {
                int x = a[i + j], y = (ll)w * a[i + j + len] % p;
                a[i + j] = (x + y) % p, a[i + j + len] = (x - y + p) % p;
            }
        }
    }
```

```
}

int merge(int a1, int a2, int A2) {
    ll M1 = (ll)p1 * p2;
    ll A1 = ((ll)inv(p2, p1) * a1 % p1 * p2 + (ll)inv(p1, p2) * a2 % p2 * p1) % M1;
    ll K = ((A2 - A1) % M2 + M2) % M2 * inv(M1 % M2, M2) % M2;
    int ans = (A1 + M1 % p * K) % p;
    return ans;
}

void go() {
    read(n1), read(n2), read(p);
    p1 = P[0], p2 = P[1], M2 = P[2];
    for (int i = 0; i <= n1; i++) read(a[i]);
    for (int i = 0; i <= n2; i++) read(b[i]);
    n = 1; while (n <= (n1 + n2)) n <<= 1, ++k;
    for (int i = 0; i < n; i++) {
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (k - 1));
    }
    for (int k = 0; k < 3; k++) {
        for (int i = 0; i < n; i++) f[k][i] = a[i] % P[k];
        for (int i = 0; i < n; i++) g[i] = b[i] % P[k];
        ntt(f[k], G, P[k]), ntt(g, G, P[k]);
        for (int i = 0; i < n; i++) f[k][i] = (ll)f[k][i] * g[i] % P[k];
        ntt(f[k], inv(G, P[k]), P[k]);
        for (int i = 0; i < n; i++) f[k][i] = (ll)f[k][i] * inv(n, P[k]) % P[k];
    }
    for (int i = 0; i <= n1 + n2; i++) ans[i] = merge(f[0][i], f[1][i], f[2][i]);
}
```

计算几何

```
// 经纬度球面最短距离
// Voleking
ld Dist(ld la1, ld lo1, ld la2, ld lo2, ld R) {
    la1 *= PI / 180, lo1 *= PI / 180, la2 *= PI / 180, lo2 *= PI / 180;
    ld x1 = cos(la1) * sin(lo1), y1 = cos(la1) * cos(lo1), z1 = sin(la1);
    ld x2 = cos(la2) * sin(lo2), y2 = cos(la2) * cos(lo2), z1 = sin(la2);
    return R * acos(x1 * x2 + y1 * y2 + z1 * z2);
}

// jiry_2
int cmp(ld k1, ld k2) {
    return sgn(k1 - k2);
}
V proj(V k1, V k2, V q) { // q 到直线 k1,k2 的投影
    V k = k2 - k1;
    return k1 + k * (dot(q - k1, k) / k.abs2());
}
V reflect(V k1, V k2, V q) {
    return proj(k1, k2, q) * 2 - q;
}
int clockwise(V k1, V k2, V k3) { // k1 k2 k3 逆时针 1 顺时针 -1 否则 0
    return sgn(det(k2 - k1, k3 - k1));
}
int checkLL(V k1, V k2, V k3, V k4) { // 求直线 (L) 线段 (S) k1,k2 和 k3,k4 的交点
    return cmp(det(k3 - k1, k4 - k1), det(k3 - k2, k4 - k2)) != 0;
}
V getLL(V k1, V k2, V k3, V k4) {
    ld w1 = det(k1 - k3, k4 - k3), w2 = det(k4 - k3, k2 - k3);
    return (k1 * w2 + k2 * w1) / (w1 + w2);
}
vector<line> getHL(vector<line>& L) { // 求半平面交，半平面是逆时针方向，输出按照逆时针
    sort(L.begin(), L.end());
    deque<line> q;
```

```
    for (int i = 0; i < (int) L.size(); i++) {
        if (i && sameDir(L[i], L[i - 1])) continue;
        while (q.size() > 1 && !checkpos(q[q.size() - 2], q[q.size() - 1], L[i])) q.pop_back();
        while (q.size() > 1 && !checkpos(q[1], q[0], L[i])) q.pop_front();
        q.push_back(L[i]);
    }
    while (q.size() > 2 && !checkpos(q[q.size() - 2], q[q.size() - 1], q[0])) q.pop_back();
    while (q.size() > 2 && !checkpos(q[1], q[0], q[q.size() - 1])) q.pop_front();
    vector<line> ans;
    for (int i = 0; i < q.size(); i++) ans.push_back(q[i]);
    return ans;
}
ld closepoint(vector<V>& A, int l, int r) { // 最近点对，先要按照 x 坐标排序
    if (r - l <= 5) {
        ld ans = 1e20;
        for (int i = l; i <= r; i++)
            for (int j = i + 1; j <= r; j++) ans = min(ans, A[i].dis(A[j]));
        return ans;
    }
    int mid = l + r >> 1;
    ld ans = min(closepoint(A, l, mid), closepoint(A, mid + 1, r));
    vector<V> B;
    for (int i = l; i <= r; i++)
        if (abs(A[i].x - A[mid].x) <= ans) B.push_back(A[i]);
    sort(B.begin(), B.end(), [](V k1, V k2) {
        return k1.y < k2.y;
    });
    for (int i = 0; i < B.size(); i++)
        for (int j = i + 1; j < B.size() && B[j].y - B[i].y < ans; j++) ans = min(ans, B[i].dis(B[j]));
    return ans;
}
int checkposCC(circle k1, circle k2) { // 返回两个圆的公切线数量
    if (cmp(k1.r, k2.r) == -1) swap(k1, k2);
    ld dis = k1.o.dis(k2.o);
    int w1 = cmp(dis, k1.r + k2.r), w2 = cmp(dis, k1.r - k2.r);
    if (w1 > 0) return 4;
    else if (w1 == 0) return 3;
    else if (w2 > 0) return 2;
    else if (w2 == 0) return 1;
    else return 0;
}
vector<V> getCL(circle k1, V k2, V k3) { // 沿着 k2->k3 方向给出，相切给出两个
    V k = proj(k2, k3, k1.o);
    ld d = k1.r * k1.r - (k - k1.o).abs2();
    if (sgn(d) == -1) return {};
    V del = (k3 - k2).unit() * sqrt(max((ld) 0.0, d));
    return {k - del, k + del};
}
vector<line> TangentoutCC(circle k1, circle k2) {
    int pd = checkposCC(k1, k2);
    if (pd == 0) return {};
    if (pd == 1) {
        V k = getCC(k1, k2)[0];
        return { (line){k, k} };
    }
    if (cmp(k1.r, k2.r) == 0) {
        V del = (k2.o - k1.o).unit().turn90().getdel();
        return {
            (line){k1.o - del * k1.r, k2.o - del * k2.r},
            (line){k1.o + del * k1.r, k2.o + del * k2.r}
        };
    } else {
        V p = (k2.o * k1.r - k1.o * k2.r) / (k1.r - k2.r);
        vector<V> A = TangentCP(k1, p), B = TangentCP(k2, p);
        vector<line> ans;
        for (int i = 0; i < A.size(); i++) ans.push_back((line){A[i], B[i]});
```

```
        return ans;
    }
}
vector<line> TangentinCC(circle k1, circle k2) {
    int pd = checkposCC(k1, k2);
    if (pd <= 2) return {};
    if (pd == 3) {
        V k = getCC(k1, k2)[0];
        return { (line){k, k} };
    }
    V p = (k2.o * k1.r + k1.o * k2.r) / (k1.r + k2.r);
    vector<V> A = TangentCP(k1, p), B = TangentCP(k2, p);
    vector<line> ans;
    for (int i = 0; i < A.size(); i++) ans.push_back((line){A[i], B[i]});
    return ans;
}
vector<line> TangentCC(circle k1, circle k2) {
    int flag = 0;
    if (k1.r < k2.r) swap(k1, k2), flag = 1;
    vector<line> A = TangentoutCC(k1, k2), B = TangentinCC(k1, k2);
    for (line k: B) A.push_back(k);
    if (flag) for (line& k: A) swap(k[0], k[1]);
    return A;
}
ld convexDiameter(vector<V> A) {
    int now = 0, n = A.size();
    ld ans = 0;
    for (int i = 0; i < A.size(); i++) {
        now = max(now, i);
        while (1) {
            ld k1 = A[i].dis(A[now % n]), k2 = A[i].dis(A[(now + 1) % n]);
            ans = max(ans, max(k1, k2));
            if (k2 > k1) now++;
            else break;
        }
    }
    return ans;
}
vector<V> convexcut(vector<V> A, V k1, V k2) { // 保留 k1,k2,p 逆时针的所有点
    int n = A.size();
    A.push_back(A[0]);
    vector<V> ans;
    for (int i = 0; i < n; i++) {
        int w1 = clockwise(k1, k2, A[i]), w2 = clockwise(k1, k2, A[i + 1]);
        if (w1 >= 0) ans.push_back(A[i]);
        if (w1 * w2 < 0) ans.push_back(getLL(k1, k2, A[i], A[i + 1]));
    }
    return ans;
}
```

本模板未涉及的专题

ECNU

数据结构

- 均摊复杂度线段树
- K-DTree
- 树状数组套主席树
- 左偏树
- Treap-序列
- 可回滚并查集
- 舞蹈链
- 笛卡尔树
- 莫队

数学

- min_25
- 杜教筛
- 伯努利数和等幂求和
- 单纯形
- 数论分块

图论

- zkw 费用流
- 树上点分治
- 二分图匹配
- 虚树
- 欧拉路径
- 一般图匹配
- 点双连通分量/广义圆方树
- 圆方树
- 最小树形图
- 三元环、四元环

计算几何

- 圆与多边形交
- 圆的离散化、面积并
- 圆的反演
- 三维计算几何
- 旋转
- 线、面
- 凸包

kuangbin

数学

- 整数拆分
- 求 A^B 的约数之和对 MOD 取模
- 斐波那契数列取模循环节

图论

- 次小生成树
- 生成树计数
- 曼哈顿最小生成树