上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

ICPC & CCPC Standard Code Library

星尘幻想

Coach	教练
Yong Yu	俞勇
YueYang Feng	冯跃洋

Contestant	队员
JingBo Zhang	张景博
Haoda Hu	胡豪达
Xiaoze Fan	范晓泽

Compiled on October 7, 2023

Contents

1. Geometry

1.1 一些公式

1.1.1 Heron's Formula

$$S = \sqrt{p(p-a)(p-b)(p-c)}$$

$$p = \frac{a+b+c}{2}$$

1.1.2 四面体内接球球心

假设 s_i 是第 i 个顶点相对面的面积，则有

$$\begin{cases} x = \dfrac{s_1x_1 + s_2x_2 + s_3x_3 + s_4x_4}{s_1 + s_2 + s_3 + s_4} \\ y = \dfrac{s_1y_1 + s_2y_2 + s_3y_3 + s_4y_4}{s_1 + s_2 + s_3 + s_4} \\ z = \dfrac{s_1z_1 + s_2z_2 + s_3z_3 + s_4z_4}{s_1 + s_2 + s_3 + s_4} \end{cases}$$

体积可以使用 1/6 混合积求, 内接球半径为

$$r = \frac{3V}{s_1 + s_2 + s_3 + s_4}$$

1.1.3 三角形内心

$$\vec{I} = \frac{a\vec{A} + b\vec{B} + c\vec{C}}{a+b+c}$$

1.1.4 三角形外心

$$\vec{O} = \frac{\vec{A} + \vec{B} - \frac{\overrightarrow{BC}\cdot\overrightarrow{CA}}{\overrightarrow{AB}\times\overrightarrow{BC}}\overrightarrow{AB}^T}{2}$$

1.1.5 三角形垂心

$$\vec{H} = 3\vec{G} - 2\vec{O}$$

1.1.6 三角形偏心

$$\frac{-a\vec{A} + b\vec{B} + c\vec{C}}{-a+b+c}$$

内角的平分线和对边的两个外角平分线交点, 外切圆圆心. 剩余两点的同理.

1.1.7 三角形内接外接圆半径

$$r = \frac{2S}{a+b+c},\ R = \frac{abc}{4S}$$

1.1.8 Pick's Theorem 格点多边形面积

$S = I + \frac{B}{2} - 1$. I 内部点, B 边界点。

1.1.9 Euler's Formula 多面体与平面图的点、边、面

For convex polyhedron: $V - E + F = 2$.
For planar graph: $|F| = |E| - |V| + n + 1$, n : #(connected components).

1.2 三角公式

$$\sin(a \pm b) = \sin a \cos b \pm \cos a \sin b$$

$$\cos(a \pm b) = \cos a \cos b \mp \sin a \sin b$$

$$\tan(a \pm b) = \frac{\tan(a) \pm \tan(b)}{1 \mp \tan(a)\tan(b)}$$

$$\tan(a) \pm \tan(b) = \frac{\sin(a \pm b)}{\cos(a)\cos(b)}$$

$$\sin(a) + \sin(b) = 2\sin(\frac{a+b}{2})\cos(\frac{a-b}{2})$$

$$\sin(a) - \sin(b) = 2\cos(\frac{a+b}{2})\sin(\frac{a-b}{2})$$

$$\cos(a) + \cos(b) = 2\cos(\frac{a+b}{2})\cos(\frac{a-b}{2})$$

$$\cos(a) - \cos(b) = -2\sin(\frac{a+b}{2})\sin(\frac{a-b}{2})$$

$$\sin(na) = n\cos^{n-1} a \sin a - \binom{n}{3}\cos^{n-3} a \sin^3 a + \binom{n}{5}\cos^{n-5} a \sin^5 a - \dots$$

$$\cos(na) = \cos^n a - \binom{n}{2}\cos^{n-2} a \sin^2 a + \binom{n}{4}\cos^{n-4} a \sin^4 a - \dots$$

1.2.1 超球坐标系

$$\begin{aligned} x_1 &= r\cos(\phi_1) \\ x_2 &= r\sin(\phi_1)\cos(\phi_2) \\ &\cdots \\ x_{n-1} &= r\sin(\phi_1)\cdots\sin(\phi_{n-2})\cos(\phi_{n-1}) \\ x_n &= r\sin(\phi_1)\cdots\sin(\phi_{n-2})\sin(\phi_{n-1}) \\ \phi_{n-1} &\in [0, 2\pi] \\ \forall i = 1..n-1\phi_i &\in [0, \pi] \end{aligned}$$

1.2.2 三维旋转公式

绕着 $(0,0,0) - (ux, uy, uz)$ 旋转 θ, (ux, uy, uz) 是单位向量

$$R = \begin{matrix} \cos\theta + u_x^2(1-\cos\theta) & u_xu_y(1-\cos\theta) - u_z\sin\theta & u_xu_z(1-\cos\theta) + u_y\sin\theta \\ u_yu_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_yu_z(1-\cos\theta) - u_x\sin\theta \\ u_zu_x(1-\cos\theta) - u_y\sin\theta & u_zu_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{matrix}.$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

1.2.3 立体角公式

$$\phi: \text{二面角}$$

$$\Omega = (\phi_{ab} + \phi_{bc} + \phi_{ac})\ \text{rad} - \pi\ \text{sr}$$

$$\tan\left(\frac{1}{2}\Omega/\text{rad}\right) = \frac{\left|\vec{a}\,\vec{b}\,\vec{c}\right|}{abc + \left(\vec{a}\cdot\vec{b}\right)c + \left(\vec{a}\cdot\vec{c}\right)b + \left(\vec{b}\cdot\vec{c}\right)a}$$

$$\theta_s = \frac{\theta_a + \theta_b + \theta_c}{2}$$

1.2.4 常用体积公式

- 棱锥 Pyramid $V = \frac{1}{3}Sh$.
- 球 Sphere $V = \frac{4}{3}\pi R^3$.
- 棱台 Frustum $V = \frac{1}{3}h(S_1 + \sqrt{S_1S_2} + S_2)$.
- 椭球 Ellipsoid $V = \frac{4}{3}\pi abc$.
- 球缺 Spherical cap $\frac{\pi}{3}(3R - H)H^2$

1.2.5 扇形与圆弧重心

扇形重心与圆心距离为 $\frac{4r\sin(\theta/2)}{3\theta}$, 圆弧重心与圆心距离为 $\frac{4r\sin^3(\theta/2)}{3(\theta-\sin(\theta))}$.

1.2.6 高维球体积

$$V_2 = \pi R^2,\ S_2 = 2\pi R$$

$$V_3 = \frac{4}{3}\pi R^3,\ S_3 = 4\pi R^2$$

$$V_4 = \frac{1}{2}\pi^2 R^4,\ S_4 = 2\pi^2 R^3$$

$$\text{Generally, } V_n = \frac{2\pi}{n}V_{n-2},\ S_{n-1} = \frac{2\pi}{n-2}S_{n-3}$$

$$\text{Where, } S_0 = 2,\ V_1 = 2,\ S_1 = 2\pi,\ V_2 = \pi$$

1.3 距离

欧式距离

$$\sqrt{\sum_{i=1}^{n}(x_{1,i} - x_{2,i})^2}$$

曼哈顿距离

$$\sum_{i=1}^{n}\left|x_{1,i} - x_{2,i}\right|$$

切比雪夫距离

$$\max_{i=1}^{n}\left\{\left|x_{1,i} - x_{2,i}\right|\right\}$$

曼哈顿距离与切比雪夫距离转换:

- 曼哈顿坐标系是通过切比雪夫坐标系旋转 $45°$ 后，再缩小到原来的一半得到的。
- 将一个点 (x, y) 的坐标变为 $(x + y, x - y)$ 后，原坐标系中的曼哈顿距离等于新坐标系中的切比雪夫距离。
- 将一个点 (x, y) 的坐标变为 $(\frac{x+y}{2}, \frac{x-y}{2})$ 后，原坐标系中的切比雪夫距离等于新坐标系中的曼哈顿距离。

1.4 Pick 定理

给定顶点均为整点的简单多边形，皮克定理说明了其面积 A 和内部格点数目 i、边上格点数目 b 的关系：

$$A = i + \frac{b}{2} - 1$$

推广：

- 取格点的组成图形的面积为一单位。在平行四边形格点，皮克定理依然成立。套用于任意三角形格点，皮克定理则是 $A = 2 \times i + b - 2$。
- 对于非简单的多边形 P，皮克定理 $A = i + \frac{b}{2} - \chi(P)$，其中 $\chi(P)$ 表示 P 的欧拉特征数 $\chi(P) = V - E + F$。
- 皮克定理和欧拉公式（$V - E + F = 2$）等价。

1.5 二维计算几何基础

```cpp
#define cp const vec &
#define cl const line &
struct vec {
    vec rot(db t) const { // 逆时针
        return {x * cos(t) - y * sin(t), x * sin(t) + y * cos(t)}; }
    vec rot90() const { return {-y, x}; }
    db len2() const { return x * x + y * y;}
    db len() const { return sqrt(x * x + y * y);}
    vec unit() const { db d = len(); return {x / d, y / d}; }
};
struct line { vec s, t; }
bool turn_left(cp a, cp b, cp c) {
 | return sgn(crs (b - a, c - a)) >= 0; }
bool point_on_segment(cp a, cl b) { // 点在线段上
    return sgn(crs(a - b.s, b.t - b.s)) == 0  // 在直线上
        && sgn(dot(b.s - a, b.t - a)) <= 0; }
```

```cpp
bool two_side(cp a, cp b, cl c) {
    return sgn(crs(a - c.s, c.t - c.s))
           * sgn(crs(b - c.s, c.t - c.s)) < 0; }
bool intersect_judge(cl a, cl b) { // 线段判非严格交
    if (point_on_segment(b.s, a)
        || point_on_segment(b.t, a)) return true;
    if (point_on_segment(a.s, b)
        || point_on_segment(a.t, b)) return true;
    return two_side(a.s, a.t, b)
           && two_side(b.s, b.t, a); }
vec line_intersect(cl a, cl b) { // 直线交点
    db s1 = crs(a.t - a.s, b.s - a.s);
    db s2 = crs(a.t - a.s, b.t - a.s);
    return (b.s * s2 - b.t * s1) / (s2 - s1); }
bool point_on_ray(cp a, cl b) {  // 点在射线上
    return sgn(crs(a - b.s, b.t - b.s)) == 0
           && sgn(dot(a - b.s, b.t - b.s)) >= 0; }
bool ray_intersect_judge(line a, line b) { // 射线判交
    db s1, s2; // can be LL
    s1 = crs(a.t - a.s, b.s - a.s);
    s2 = crs(a.t - a.s, b.t - a.s);
    if (sgn(s1) == 0 && sgn(s2) == 0) {
        return sgn(dot(a.t - a.s, b.s - a.s)) >= 0
               || sgn(dot(b.t - b.s, a.s - b.s)) >= 0; }
    if (!sgn(s1 - s2) || sgn(s1) == sgn(s2 - s1)) return 0;
    swap(a, b);
    s1 = crs(a.t - a.s, b.s - a.s);
    s2 = crs(a.t - a.s, b.t - a.s);
    return sgn(s1) != sgn(s2 - s1); }
db point_to_line(cp a, cl b) {  // 点到直线距离
    return abs(crs(b.t - b.s, a - b.s)) / dis(b.s, b.t); }
vec project_to_line(cp a, cl b) {  // 点在直线投影
    return b.s + (b.t - b.s)
           * (dot(a - b.s, b.t - b.s) / (b.t - b.s).len2()); }
db point_to_segment(cp a, cl b) {  // 点到线段距离
    if (sgn(dot(b.s - a, b.t - b.s))
        * sgn(dot(b.t - a, b.t - b.s)) <= 0)
        return abs(crs(b.t - b.s, a - b.s)) / dis(b.s, b.t);
    return min(dis(a, b.s), dis(a, b.t)); }
bool in_polygon(cp p, const vector <vec> &po) {
    int n = (int) po.size(); int cnt = 0;
    for (int i = 0; i < n; ++i) {
        vec a = po[i], b = po[(i + 1) % n];
        if (point_on_segment(p, line(a, b))) return true;
        int x = sgn(crs(p - a, b - a)),
            y = sgn(a.y - p.y), z = sgn(b.y - p.y);
        if (x > 0 && y <= 0 && z > 0) ++cnt;
        if (x < 0 && z <= 0 && y > 0) --cnt; }
    return cnt != 0; }
vector <vec> line_circle_intersect(cl a, cc b) {
    if (sgn(point_to_line(b.c, a) - b.r) > 0)
        return vector <vec> ();
    db x = sqrt(sqr(b.r) - sqr(point_to_line(b.c, a)));
    return vector <vec>
           ({project_to_line(b.c, a) + (a.s - a.t).unit() * x,
             project_to_line(b.c, a) - (a.s - a.t).unit() * x}); }
db circle_intersect_area(cc a, cc b) {
    db d = dis(a.c, b.c);
    if (sgn(d - (a.r + b.r)) >= 0) return 0;
    if (sgn(d - abs(a.r - b.r)) <= 0) {
        db r = min(a.r, b.r);
        return r * r * PI; }
    db x = (d * d + a.r * a.r - b.r * b.r) / (2 * d),
           t1 = acos(min(1., max(-1., x / a.r))),
           t2 = acos(min(1., max(-1., (d - x) / b.r)));
    return sqr(a.r) * t1 + sqr(b.r) * t2 - d * a.r * sin(t1); }
vector <vec> circle_intersect(cc a, cc b) {
    if (a.c == b.c
        || sgn(dis(a.c, b.c) - a.r - b.r) > 0
        || sgn(dis(a.c, b.c) - abs(a.r - b.r)) < 0)
        return {};
    vec r = (b.c - a.c).unit();
    db d = dis(a.c, b.c);
    db x = ((sqr(a.r) - sqr(b.r)) / d + d) / 2;
    db h = sqrt(sqr(a.r) - sqr(x));
    if (sgn(h) == 0) return {a.c + r * x};
    return {a.c + r *x + r.rot90() *h,
            a.c + r *x - r.rot90() *h}; }
// 返回按照顺时针方向
vector <vec> tangent(cp a, cc b) {
    circle p = make_circle(a, b.c);
    return circle_intersect(p, b); }
vector <line> extangent(cc a, cc b) {
    vector <line> ret;
    if (sgn(dis(a.c, b.c) - abs(a.r - b.r)) <= 0) return ret;
    if (sgn(a.r - b.r) == 0) {
        vec dir = b.c - a.c;
        dir = (dir * a.r / dir.len()).rot90();
        ret.push_back(line(a.c + dir, b.c + dir));
        ret.push_back(line(a.c - dir, b.c - dir));
    } else {
        vec p = (b.c * a.r - a.c * b.r) / (a.r - b.r);
        vector pp = tangent(p, a), qq = tangent(p, b);
        if (pp.size() == 2 && qq.size() == 2) {
            if (sgn(a.r - b.r) < 0)
                swap(pp[0], pp[1]), swap(qq[0], qq[1]);
            ret.push_back(line(pp[0], qq[0]));
            ret.push_back(line(pp[1], qq[1])); } }
    return ret; }
vector <line> intangent(cc a, cc b) {
    vector <line> ret;
    vec p = (b.c * a.r + a.c * b.r) / (a.r + b.r);
    vector pp = tangent(p, a), qq = tangent(p, b);
    if (pp.size() == 2 && qq.size() == 2) {
        ret.push_back(line(pp[0], qq[0]));
        ret.push_back(line(pp[1], qq[1])); }
    return ret; }
vector <vec> cut(const vector<vec> &c, line p) {
    vector <vec> ret;
    if (c.empty()) return ret;
    for (int i = 0; i < (int) c.size(); ++i) {
        int j = (i + 1) % (int) c.size();
        if (turn_left(p.s, p.t, c[i])) ret.push_back(c[i]);
        if (two_side(c[i], c[j], p))
            ret.push_back(line_intersect(p, line(c[i], c[j]))); }
    return ret; }
```

1.6 三角形

```cpp
vec incenter(cp a, cp b, cp c) { // 内心
    db p = dis(a, b) + dis(b, c) + dis(c, a);
    return (a * dis(b, c) + b * dis(c, a) + c * dis(a, b)) / p; }
vec circumcenter(cp a, cp b, cp c) { // 外心
    vec p = b - a, q = c - a, s(dot(p, p) / 2, dot(q, q) / 2);
    db d = crs(p, q);
    return a + vec(crs(s, vec(p.y, q.y)), crs(vec(p.x, q.x), s)) / d; }
vec orthocenter(cp a, cp b, cp c) { // 垂心
    return a + b + c - circumcenter(a, b, c) * 2.0; }
vec fermat_point(cp a, cp b, cp c) { // 费马点
    if (a == b) return a;
    if (b == c) return b;
    if (c == a) return c;
    db ab = dis(a, b), bc = dis(b, c), ca = dis(c, a);
    db cosa = dot(b - a, c - a) / ab / ca;
    db cosb = dot(a - b, c - b) / ab / bc;
    db cosc = dot(b - c, a - c) / ca / bc;
    db sq3 = PI / 3.0; vec mid;
    if (sgn(cosa + 0.5) < 0) mid = a;
    else if (sgn(cosb + 0.5) < 0) mid = b;
    else if (sgn(cosc + 0.5) < 0) mid = c;
    else if (sgn(crs(b - a, c - a)) < 0)
        mid = line_intersect(line(a, b + (c - b).rot(sq3)), line(b, c + (a - c).rot(sq3)));
    else
        mid = line_intersect(line(a, c + (b - c).rot(sq3)), line(c, b + (a - b).rot(sq3)));
    return mid; } // minimize(|A-x|+|B-x|+|C-x|)
```

1.7 凸包

```cpp
vector<vec> convex_hull(vector<vec> a) {
    int n = (int) a.size(), cnt = 0;
    if (n < 2) return a;
    sort(a.begin(), a.end());  // less<pair>
    vector<vec> ret;
```

```
    for (int i = 0; i < n; ++i) {
        while (cnt > 1
        && turn_left(ret[cnt - 2], a[i], ret[cnt - 1]))
         --cnt, ret.pop_back();
        ++cnt, ret.push_back(a[i]); }
    int fixed = cnt;
    for (int i = n - 2; i >= 0; --i) {
        while (cnt > fixed
        && turn_left(ret[cnt - 2], a[i], ret[cnt - 1]))
         --cnt, ret.pop_back();
        ++cnt, ret.push_back(a[i]); }
    ret.pop_back(); return ret;
} // counter-clockwise
```

1.8 半平面交

```
struct lin {
    vec s, e; db k;
    lin(vec _s, vec _e): s(_s), e(_e), k(atan2((e - s).y, (e - s).x)) {}
    il vec operator()() const {return e - s;}
};
vec cross(const lin &l1, const lin &l2) {return l1.s + l1() * crs(l2.s - l1.s, l2()) / crs(l1(), l2());}
bool cmpl(lin a, lin b) { // 极角排序，极角相同靠左优先
    if (cmp(a.k, b.k) == 0) return sign(crs(b.e-a.s, a())) > 0;
    return cmp(a.k, b.k) < 0;
}
bool Onright(lin a, lin b, lin c) { // a,b 交点在 c 右边
    vec p = cross(a, b);
    return sign(crs(c(), p - c.s)) <= 0;
}

void Halfplane(vector<lin> Ls, vector<vec> &res) { // 半平面交
    res.clear();
    sort(Ls.begin(), Ls.end(), cmpl);
    deque<int> q;
    for (int i = 0; i < (int)Ls.size(); ++i) {
        if (i != 0 && cmp(Ls[i].k, Ls[i - 1].k) == 0) continue;
        while (q.size() >= 2 && Onright(Ls[q[q.size() - 2]], Ls[q.back()], Ls[i])) q.pop_back();
        while (q.size() >= 2 && Onright(Ls[q.front()], Ls[q[1]], Ls[i])) q.pop_front();
        q.push_back(i);
    }
    while (q.size() >= 2 && Onright(Ls[q[q.size() - 2]], Ls[q.back()], Ls[q.front()])) q.pop_back();
    while (q.size() >= 2 && Onright(Ls[q[0]], Ls[q[1]], Ls[q.back()])) q.pop_front();
    if (q.size() >= 2) res.push_back(cross(Ls[q.back()], Ls[q.front()]));
    while (q.size() >= 2) {
        res.push_back(cross(Ls[q[0]], Ls[q[1]]));
        q.pop_front();
    }
}
```

1.9 自适应辛普森

```
db f(db x) { return x * x * x; }
// 辛普森公式  = (r - l) / 6 * (f(l) + f(r) + 4f((l + r) / 2))
db simpson(db l, db r) {
    db mid = (l + r) / 2.0;
    return (r - l) * (f(l) + f(r) + f(mid) * 4.0) / 6.0; }
db simpson(db l, db r, db eps, db ans, int step) {
    db mid = (l + r) / 2.0;
    db fl = simpson(l, mid), fr = simpson(mid, r);
    if (fabs(fl + fr - ans) <= 15.0 * eps && step < 0) return fl + fr + (fl + fr - ans) / 15.0;
    return simpson(l, mid, eps / 2.0, fl, step - 1) + simpson(mid, r, eps / 2.0, fr, step - 1); }
db calc(db l, db r, db eps) { return simpson(l, r, eps, simpson(l, r), 12); }
```

2. Graph

2.1 图论基本知识

2.1.1 树链的交

假设两条链 $(a_1, b_1), (a_2, b_2)$ 的lca分别为 c_1, c_2. 再求出 $(a_1, a_2), (a_1, b_2), (b_1, a_2), (b_1, b_2)$ 的lca, 记为 d_1, d_2, d_3, d_4. 将 (d_1, d_2, d_3, d_4) 按照深度从小到大排序, (c_1, c_2) 也从小到大排序. 两条链有交当且仅当 $dep[c_1] \le dep[d_1]$ 且 $dep[c_2] \le dep[d_3]$, 则 (d_3, d_4) 是链交的两个端点.

2.1.2 带修改MST

维护少量修改的最小生成树，可以缩点缩边使暴力复杂度变低. (银川 21: 求有 16 个 '某两条边中至少选一条' 的限制条件的最小生成树)

找出必须边 将修改边标 $-\infty$, 在MST上的其余边为必须边, 以此缩点.

找出无用边 将修改边标 ∞, 不在MST上的其余边为无用边, 删除之.

假设修改边数为 k, 操作后图中最多剩下 $k+1$ 个点和 $2k$ 条边.

2.1.3 差分约束

$x_r - x_l \le c$:`add(l, r, c)` $x_r - x_l \ge c$:`add(r, l, -c)`

2.1.4 李超线段树

添加若干条线段或直线 $(a_i, b_i) \to (a_j, b_j)$, 每次求 $[l, r]$ 上最上面的那条线段的值. 思想是让线段树中一个节点只对应一条直线, 如果在这个区间加入一条直线, 如果一段比原来的优, 一段比原来的劣, 那么判断一下两条线的交点, 判断哪条直线可以完全覆盖一段一半的区间, 把它保留, 另一条直线下传到另一半区间. 时间复杂度 $O(n \log n)$.

2.1.5 Segment Tree Beats

区间 min, max, 区间求和. 以区间取 min 为例, 额外维护最大值 m, 严格次大值 s 以及最大值个数 t. 现在假设我们要让区间 $[L, R]$ 对 x 取 min, 先在线段树中定位若干个节点, 对于每个节点分三种情况讨论: 1, 当 $m \le x$ 时, 显然这一次修改不会对这个节点产生影响, 直接退出; 2, 当 $se < x < ma$ 时, 显然这一次修改只会影响到所有最大值, 所以把 num 加上 $t*(x-ma)$, 把 ma 更新为 x, 打上标记退出; 3, 当 $se \ge x$ 时, 无法直接更新着一个节点的信息, 对当前节点的左儿子和右儿子递归处理. 单次操作均摊复杂度 $O(\log^2 n)$.

2.1.6 二分图

最小点覆盖=最大匹配数. 独立集与覆盖集互补. 最小点覆盖构造方法: 对二分图流图求割集, 跨过的边指示最小点覆盖. **Hall**定理 $G = (X, Y, E), |M| = |X| \Leftrightarrow \forall S \subseteq X, |S| \le |A(S)|$.

2.1.7 稳定婚姻问题

男士按自己喜欢程度从高到底依次向每位女士求婚, 女士遇到更喜欢的男士时就接受他, 并抛弃以前的配偶. 被抛弃的男士继续按照列表向剩下的女士依次求婚, 直到所有人都有配偶. 算法一定能得到一个匹配, 而且这个匹配一定是稳定的. 时间复杂度 $O(n^2)$.

2.1.8 2-SAT

如果选 A 就必须选 B 就从 A 向 B 连一条边, 如果两个只能选一个的条件在同一个强连通分量中就不合法. 输出可行方案可以比较 X 和 X 的 bl 的大小, 大的选 X. 建图优化一般考虑前后缀的合并.

2.1.9 三元环

对于无向边 (u, v), 如果 $\deg_u < \deg_v$, 那么连有向边 (u, v)(以点标号为第二关键字). 枚举 x 暴力即可. 时间复杂度 $O(m\sqrt{m})$.

2.1.10 图同构

令 $F_t(i) = (F_{t-1}(i) * A + \sum_{i \to j} F_{t-1}(j) * B + \sum_{j \to i} F_{t-1}(j) * C + D * (i - a)) \bmod P$, 枚举点 a, 迭代 K 次后求得的就是 a 点所对应的 $hash$ 值, 其中 K, A, B, C, D, P 为 $hash$ 参数, 可自选.

2.1.11 竞赛图 Landau's Theorem

n 个点竞赛图点按出度按升序排序, 前 i 个点的出度之和不小于 $\frac{i(i-1)}{2}$, 度数总和等于 $\frac{n(n-1)}{2}$. 否则可以用优先队列构造出方案.

2.1.12 Ramsey Theorem R(3,3)=6, R(4,4)=18

6 个人中存在 3 人相互认识或者相互不认识.

2.1.13 树的计数 Prufer序列

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

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

由序列得到树: 先将所有点的度赋初值为 1, 然后加上它的编号在Prufer序列中出现的次数, 得到每个点的度; 执行 $n-2$ 步, 第 i 步选取具有最小标号的度为 1 的点 u 与 $v = p_i$ 相连, 得到树中的一条边, 并将 u 和 v 的度减一. 最后再把剩下的两个度为 1 的点连边, 加入到树中.

相关结论: n 个点完全图, 每个点度数依次为 $d_1, d_2, \ldots, d_n$, 这样生成树的棵树为: $\frac{(n-2)!}{(d_1-1)!(d_2-1)!\ldots(d_n-1)!}$.

左边有 n_1 个点, 右边有 n_2 个点的完全二分图的生成树棵树为 $n_1^{n_2-1} \times n_2^{n_1-1}$.

m 个连通块, 每个连通块有 c_i 个点, 把他们全部连通的生成树方案数: $(\sum c_i)^{m-2} \prod c_i$

2.1.14 有根树的计数

首先, 令 $S_{n,j} = \sum_{1 \le j \le n/j}$; 于是 $n+1$ 个结点的有根树的总数为 $a_{n+1} = \frac{\sum_{j=1}^{n} j a_j S_{n-j}}{n}$. 注: $a_1 = 1, a_2 = 1, a_3 = 2, a_4 = 4, a_5 = 9, a_6 = 20, a_9 = 286, a_{11} = 1842$.

2.1.15 无根树的计数

n 是奇数时, 有 $a_n - \sum_i^{n/2} a_i a_{n-i}$ 种不同的无根树.

n 时偶数时, 有 $a_n - \sum_i^{n/2} a_i a_{n-i} + \frac{1}{2} a_{n/2}(a_{n/2} + 1)$ 种不同的无根树.

2.1.16 生成树计数 Kirchhoff's Matrix-Tree Thoerem

Kirchhoff Matrix $T = Deg - A$, Deg 是度数对角阵, A 是邻接矩阵. 无向图度数矩阵是每个点度数; 有向图度数矩阵是每个点入度.

邻接矩阵 $A[u][v]$ 表示 $u \to v$ 边个数, 重边按照边数计算, 自环不计入度数.

无向图生成树计数: $c = |K$ 的任意1个 $n-1$ 阶主子式 $|$

有向图外向树计数: $c = |$ 去掉根所在的那阶得到的主子式 $|$

2.1.17 有向图欧拉回路计数 BEST Thoerem

$$\mathrm{ec}(G) = t_w(G) \prod_{v \in V} (\deg(v) - 1)!$$

其中 deg 为入度 (欧拉图中等于出度), $t_w(G)$ 为以 w 为根的外向树的个数. 相关计算参考生成树计数.

欧拉连通图中任意两点外向树个数相同: $\mathrm{t_v}(G) = \mathrm{t_w}(G)$.

2.1.18 Tutte Matrix

Tutte matrix A of a graph $G = (V, E)$:

$$A_{ij} = \begin{cases} x_{ij} & \text{if } (i, j) \in E \text{ and } i < j \\ -x_{ij} & \text{if } (i, j) \in E \text{ and } i > j \\ 0 & \text{otherwise} \end{cases}$$

where x_{ij} are indeterminates. The determinant of this skew-symmetric matrix is then a polynomial (in the variables x_{ij}, $i < j$): this coincides with the square of the pfaffian of the matrix A and is non-zero (as a polynomial) if and only if a perfect matching exists.

2.1.19 Edmonds Matrix

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

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

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

2.1.20 有向图无环定向, 色多项式

图的色多项式 $P_G(q)$ 对图 G 的 q-染色计数.

Triangle K_3 : $x(x-1)(x-2)$

Complete graph K_n : $x(x-1)(x-2) \cdots (x-(n-1))$

Tree with n vertices : $x(x-1)^{n-1}$

Cycle C_n : $(x-1)^n + (-1)^n (x-1)$

acyclic orientations of an n-vertex graph G is $(-1)^n P_G(-1)$.

2.2 2 SAT

```cpp
int n, m, h[2000001], p, tot, t;
int ans[2000001], cnt, bel[2000001], dfn[2000001],
    low[2000001], inz[2000001], z[2000001];
char c[2000001];
struct pp {
    int to, ne;
} b[2000001];
void add(int x, int y) {
    b[++p].to = y;
    b[p].ne = h[x];
    h[x] = p;
}
void tarjan(int x) {
    dfn[x] = low[x] = ++cnt;
    z[++t] = x; inz[x] = 1;
    for (int i = h[x]; i; i = b[i].ne) {
        int v = b[i].to;
        if (!dfn[v]) {
            tarjan(v);
            low[x] = min(low[x], low[v]);
        } else if (inz[v]) {
            low[x] = min(low[x], dfn[v]);
        }
    }
    if (dfn[x] == low[x]) {
        tot++;
        do {
            bel[z[t]] = tot;
            inz[z[t]] = 0;
```

```cpp
        } while (z[t--] != x);
    }
}
bool solve() {
    for (int i = 1; i <= 2 * n; i++)
        if (!dfn[i]) tarjan(i);
    for (int i = 1; i <= n; i++)
        if (bel[i] == bel[i + n]) return 0;
    return 1;
}
int main() {
    scanf("%d%d", &n, &m); p = 0;
    for (int i = 1; i <= m; i++) {
        int xx, yy, x, y;
        scanf("%d%d%d%d", &xx, &x, &yy, &y);
        add(xx + (x ^ 1)*n, yy + y * n);
        add(yy + (y ^ 1)*n, xx + x * n);
    }
    if (solve()) {
        printf("POSSIBLE\n");
        for (int i = 1; i <= n; i++) {
            if (bel[i] < bel[i + n]) printf("0 ");
            else printf("1 ");
        }
    } else printf("IMPOSSIBLE\n");
    return 0;
}
```

2.3 极大团

```cpp
const int maxn = 129;
int n, m, S;
int some[maxn][maxn], all[maxn][maxn], none[maxn][maxn], g[maxn][maxn];
void dfs(int d, int an, int sn, int nn) {
    if (!sn && !nn) ++S;
    if (S > 1000) return; //题意表明S超过1000就输出Impossible
    int u = some[d][0];
    for (int i = 0; i < sn; ++i) {
        int v = some[d][i];
        if (g[u][v]) continue;
        int tsn = 0, tnn = 0;
        //for(int j=0; j<an; ++j) all[d+1][j]=all[d][j];
        //all[d+1][an]=v;  //可以不写这两行
        for (int j = 0; j < sn; ++j) if (g[v][some[d][j]])
                some[d + 1][tsn++] = some[d][j];
        for (int j = 0; j < nn; ++j) if (g[v][none[d][j]])
                none[d + 1][tnn++] = none[d][j];
        dfs(d + 1, an + 1, tsn, tnn);
        some[d][i] = 0, none[d][nn++] = v;
    }
}

int main() {
    ios::sync_with_stdio(false);
    while (cin >> n >> m) {
        S = 0;
        memset(g, 0, sizeof(g));
        for (int i = 0; i < m; ++i) {
            int a, b;
            cin >> a >> b;
            g[a][b] = g[b][a] = 1;
        }
        for (int i = 0; i < n; ++i) some[0][i] = i + 1; //some的初始化
        dfs(0, 0, n, 0);
        if (S > 1000) cout << "Too many maximal sets of friends." << endl;
        else cout << S << endl;
    }
}
```

2.4 k短路

```cpp
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <queue>
using namespace std;
const int maxn = 200010;
int n, m, s, t, k, x, y, ww, cnt, fa[maxn];

struct Edge {
  int cur, h[maxn], nxt[maxn], p[maxn], w[maxn];

  void add_edge(int x, int y, int z) {
    cur++;
    nxt[cur] = h[x];
    h[x] = cur;
    p[cur] = y;
    w[cur] = z;
  }
} e1, e2;

int dist[maxn];
bool tf[maxn], vis[maxn], ontree[maxn];

struct node {
  int x, v;

  node* operator=(node a) {
    x = a.x;
    v = a.v;
    return this;
  }

  bool operator<(node a) const { return v > a.v; }
} a;

priority_queue<node> Q;

void dfs(int x) {
  vis[x] = true;
  for (int j = e2.h[x]; j; j = e2.nxt[j])
    if (!vis[e2.p[j]])
      if (dist[e2.p[j]] == dist[x] + e2.w[j])
        fa[e2.p[j]] = x, ontree[j] = true, dfs(e2.p[j]);
}

struct LeftistTree {
  int cnt, rt[maxn], lc[maxn * 20], rc[maxn * 20], dist[maxn * 20];
  node v[maxn * 20];

  LeftistTree() { dist[0] = -1; }

  int newnode(node w) {
    cnt++;
    v[cnt] = w;
    return cnt;
  }

  int merge(int x, int y) {
    if (!x || !y) return x + y;
    if (v[x] < v[y]) swap(x, y);
    int p = ++cnt;
    lc[p] = lc[x];
    v[p] = v[x];
    rc[p] = merge(rc[x], y);
    if (dist[lc[p]] < dist[rc[p]]) swap(lc[p], rc[p]);
    dist[p] = dist[rc[p]] + 1;
    return p;
  }
} st;

void dfs2(int x) {
  vis[x] = true;
  if (fa[x]) st.rt[x] = st.merge(st.rt[x], st.rt[fa[x]]);
  for (int j = e2.h[x]; j; j = e2.nxt[j])
    if (fa[e2.p[j]] == x && !vis[e2.p[j]]) dfs2(e2.p[j]);
}

int main() {
  scanf("%d%d%d%d%d", &n, &m, &s, &t, &k);
  for (int i = 1; i <= m; i++)
    scanf("%d%d%d", &x, &y, &ww), e1.add_edge(x, y, ww), e2.add_edge(y, x, ww);
  Q.push({t, 0});
  while (!Q.empty()) {
    a = Q.top();
    Q.pop();
    if (tf[a.x]) continue;
    tf[a.x] = true;
    dist[a.x] = a.v;
```

```
    for (int j = e2.h[a.x]; j; j = e2.nxt[j]) Q.push({e2.p[j], a.v + e2.w[j]});
  }
  if (k == 1) {
    if (tf[s])
      printf("%d\n", dist[s]);
    else
      printf("-1\n");
    return 0;
  }
  dfs(t);
  for (int i = 1; i <= n; i++)
    if (tf[i])
      for (int j = e1.h[i]; j; j = e1.nxt[j])
        if (!ontree[j])
          if (tf[e1.p[j]])
            st.rt[i] = st.merge(
                st.rt[i],
                st.newnode({e1.p[j], dist[e1.p[j]] + e1.w[j] - dist[i]}));
  for (int i = 1; i <= n; i++) vis[i] = false;
  dfs2(t);
  if (st.rt[s]) Q.push({st.rt[s], dist[s] + st.v[st.rt[s]].v});
  while (!Q.empty()) {
    a = Q.top();
    Q.pop();
    cnt++;
    if (cnt == k - 1) {
      printf("%d\n", a.v);
      return 0;
    }
    if (st.lc[a.x])  // 可并堆删除直接把左右儿子加入优先队列中
      Q.push({st.lc[a.x], a.v - st.v[a.x].v + st.v[st.lc[a.x]].v});
    if (st.rc[a.x])
      Q.push({st.rc[a.x], a.v - st.v[a.x].v + st.v[st.rc[a.x]].v});
    x = st.rt[st.v[a.x].x];
    if (x) Q.push({x, a.v + st.v[x].v});
  }
  printf("-1\n");
  return 0;
}
```

2.5 KM

```
#include<cstdio>
#include<iostream>
#include<cstring>
#include<cmath>
#include<queue>
using namespace std;
int n, N, M, k, d[501][501], match[501], ka[501], kb[501], visb[501], visa[501], p[501];
long long c[501], delta;
void bfs(int x) {
    int a, y = 0, yy = 0;
    for (int i = 1; i <= n; i++)p[i] = 0, c[i] = 1e18;
    match[y] = x;
    do {
        a = match[y], delta = 1e18, visb[y] = 1;
        for (int b = 1; b <= n; b++) {
            if (!visb[b]) {
                if (c[b] > ka[a] + kb[b] - d[a][b])
                    c[b] = ka[a] + kb[b] - d[a][b], p[b] = y;
                if (c[b] < delta)
                    delta = c[b], yy = b;
            }
        }
        for (int b = 0; b <= n; b++) {
            if (visb[b]) {
                ka[match[b]] -= delta, kb[b] += delta;
            } else c[b] -= delta;
        }
        y = yy;
    } while (match[y]);
    while (y)match[y] = match[p[y]], y = p[y];
}
long long KM() {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) visb[j] = 0;
        bfs(i);
    }
    long long ans = 0;
    for (int i = 1; i <= n; i++) ans += d[match[i]][i];
    return ans;
}
int main() {
    scanf("%d%d%d", &N, &M, &k);
    n = max(N, M);
    while (k--) {
        int x, y, z;
        scanf("%d%d%d", &x, &y, &z);
        d[y][x] = z;
    }
    printf("%lld\n", KM());
    for (int i = 1; i <= N; i++)
        printf("%d ", (d[match[i]][i] == 0) ? 0 : match[i]);
    return 0;
}
```

2.6 tarjan

```
// 强连通分量
void tarjan(int x) {
    dfn[x] = low[x] = ++tot;
    s[++len] = x;
    instack[x] = 1;
    for (int i = head[x]; i; i = e[i].next) {
        int y = e[i].to;
        if (!dfn[y]) {
            tarjan(y);
            low[x] = min(low[x], low[y]);
        } else {
            if (instack[y])low[x] = min(low[x], low[y]);
        }
    }
    if (dfn[x] == low[x]) {
        cnt++;
        ans[cnt].push_back(x);
        while (s[len] != x) {
            belong[s[len]] = cnt;
            instack[s[len]] = 0;
            ans[cnt].push_back(s[len]);
            len--;
        }
        len--;
        instack[x] = 0;
        belong[x] = cnt;
    }
}
// 边双
void tarjan(int x, int las) {
    low[x] = dfn[x] = ++cnt;
    st.push(x);
    for (auto i : e[x]) {
        if (i == las) continue;
        if (!dfn[i]) {
            tarjan(i, x);
            low[x] = min(low[x], low[i]);
        } else low[x] = min(low[x], dfn[i]);
    }
    if (dfn[x] == low[x]) {
        vector<int> vec;
        vec.push_back(x);
        while (st.top() != x) {
            vec.push_back(st.top());
            st.pop();
        }
        st.pop();
        ans.push_back(vec);
    }
}
// 点双
void tarjan(int x, int root) { //求割点的改版（其实不需要root）
    dfn[x] = low[x] = ++cnt;
    if (x == root && !head[x]) { //孤立点判定
        dcc[++ans].push_back(x);
    }
    sta.push(x);
```

```
    for (int i = head[x]; i; i = nxt[i]) {
        int g = go[i];
        if (!dfn[g]) {
            tarjan(g, root);
            low[x] = min(low[x], low[g]);
            if (low[g] >= dfn[x]) {
                ans++;
                int p;
                do { //弹栈
                    p = sta.top();
                    sta.pop();
                    dcc[ans].push_back(p);
                } while (p != g); //注意此处，因为要求是不到达出点
                dcc[ans].push_back(x);//别忘了加入源点!
            }
        } else
            low[x] = min(low[x], dfn[g]);
    }
}
```

2.7 最小斯坦纳树

```
//给定一个带边权的无向连通图G，再给定包含k个结点的点集S，选出G的子图G'，使得G' 包含S，G' 为连通图，且G' 边权和最小
#include<bits/stdc++.h>
#define mp make_pair
#define zjx printf("%d",
#define AK dp[c[1]][(1<<k)-1]
#define IOI );
using namespace std;
int n, m, k, p, h[101], dp[101][1024], c[11], vis[101];
struct tree {
    int to, ne, v;
} a[1001];
void add(int x, int y, int z) {
    a[++p].to = y;
    a[p].ne = h[x];
    a[p].v = z;
    h[x] = p;
}
priority_queue<pair<int, int>, vector<pair<int, int> >, greater<pair<int, int> > >q;
void dijkstra(int s) {
    memset(vis, 0, sizeof(vis));
    while (!q.empty()) {
        int x = q.top().second;
        q.pop();
        if (vis[x])continue;
        vis[x] = 1;
        for (int i = h[x]; i; i = a[i].ne) {
            if (dp[a[i].to][s] > dp[x][s] + a[i].v) {
                dp[a[i].to][s] = dp[x][s] + a[i].v;
                q.push(mp(dp[a[i].to][s], a[i].to));
            }
        }
    }
}
int main() {
    scanf("%d%d%d", &n, &m, &k);
    for (int i = 1; i <= m; i++) {
        int x, y, z;
        scanf("%d%d%d", &x, &y, &z);
        add(x, y, z), add(y, x, z);
    }
    memset(dp, 0x3f, sizeof(dp));
    for (int i = 1; i <= k; i++)scanf("%d", &c[i]), dp[c[i]][1 << (i - 1)] = 0;
    for (int s = 1; s < (1 << k); s++) {
        for (int i = 1; i <= n; i++) {
            for (int ss = s & (s - 1); ss; ss = (ss - 1)&s)
                dp[i][s] = min(dp[i][s], dp[i][ss] + dp[i][ss ^ s]);
            if (dp[i][s] != 0x3f3f3f3f)q.push(mp(dp[i][s], i));
        }
        dijkstra(s);
    }
    zjx AK IOI
    return 0;
}
```

2.8 欧拉回路

```
/* comment : directed */
int e, cur[N]/*, deg[N]*/;
vector<int> E[N];
int id[M]; bool vis[M];
stack<int>stk;
void dfs(int u) {
 | for (cur[u]; cur[u] < E[u].size(); cur[u]++) {
 |  | int i = cur[u];
 |  | if (vis[abs(E[u][i])]) continue;
 |  | int v = id[abs(E[u][i])] ^ u;
 |  | vis[abs(E[u][i])] = 1; dfs(v);
 |  | stk.push(E[u][i]); }
}// dfs for all when disconnect
void add(int u, int v) {
 | id[++e] = u ^ v; // s = u
 | E[u].push_back(e); E[v].push_back(-e);
/*  | E[u].push_back(e); deg[v]++; */
} bool valid() {
 | for (int i = 1; i <= n; i++)
 |  | if (E[i].size() & 1) return 0;
/* |  | if (E[i].size() != deg[i]) return 0;*/
 | return 1;}
```

2.9 树哈希

无根树哈希：以重心为根，如果重心有两个，分别判断即可。有根树哈希：

2.9.1 方法一

按照 dfs 序可以将树上结点对应到序列上，在序列上填上对应结点的深度，根据序列可以还原出形态唯一的树，那么树哈希便转换成了序列上的哈希。递归处理：初始 $hash(u) = dep_u$ ，插入 v 子树，就相当于两个序列接起来，$hash(u)' = hash(u) \times base^{siz_v} + hash(v)$ 。如果交换子树的顺序算同构，把 u 所有孩子的哈希值排序后再加入；如果要判断两个不同深度的子树是否同构，把深度换成高度（到最深叶子结点的距离）。

2.9.2 方法二

按照以下公式：

$$f_u = c + \sum_{v \in son(u)} f_v \times prime(size_v)$$

其中 f_u 表以 u 为根子树对应的哈希值。$size_v$ 表 v 子树大小，$son(u)$ 为 u 点孩子的集合，$prime(i)$ 为第 i 个质数，c 随便搞个常数。

3. Data Structure

3.1 LCT 动态树

```
namespace LCT {
    int ch[N][2], f[N], sum[N], val[N], tag[N], dat[N]; // dat 维护的链信息, val 点上信息
    inline void PushUp(int x) {
        dat[x] = dat[ch[x][0]] ^ dat[ch[x][1]] ^ val[x];
    }
    inline void PushRev(int x) {swap(ch[x][0], ch[x][1]); tag[x] ^= 1;}
    inline void PushDown(int x) {
        if (tag[x] == 0) return ;
        PushRev(ch[x][0]); PushRev(ch[x][1]); tag[x] = 0;
    }
    inline bool Get(int x) {return ch[f[x]][1] == x;} // 是父亲的哪个儿子
    inline bool IsRoot(int x) {return (ch[f[x]][1] != x && ch[f[x]][0] != x);} // 是否是当前 Splay 的根
    inline void Rotate(int x) { // Splay 旋转
        int y = f[x], z = f[y], k = Get(x);
        if (!IsRoot(y)) ch[z][Get(y)] = x;
        ch[y][k] = ch[x][k ^ 1]; f[ch[x][k ^ 1]] = y;
        ch[x][k ^ 1] = y; f[y] = x; f[x] = z;
        PushUp(y); PushUp(x);
    }
    void Updata(int x) { // Splay 中从上到下 PushDown
        if (!IsRoot(x)) Updata(f[x]);
        PushDown(x);
    }
    inline void Splay(int x) { // Splay 上把 x 转到根
        Updata(x);
        for (int fa; fa = f[x], !IsRoot(x); Rotate(x)) {
            if (!IsRoot(fa)) Rotate(Get(fa) == Get(x) ? fa : x);
        }
        PushUp(x);
    }
    inline void Access(int x) { // 辅助树上打通 x 到根的路径(即 x 到根变为实链)
        for (int p = 0; x; p = x, x = f[x]) {
            Splay(x); ch[x][1] = p; PushUp(x);
        }
    }
    inline void MakeRoot(int x) { // 钦定 x 为辅助树根
        Access(x); Splay(x); PushRev(x);
    }
    inline int FindRoot(int x) { // 找 x 所在辅助树根
        Access(x); Splay(x);
        while (ch[x][0]) PushDown(x), x = ch[x][0];
        Splay(x); // 不加复杂度会假
        return x;
    }
    inline void Split(int x, int y) { // 把 x 到 y 的路径拢出来, 并以 y 为 Splay 根
        MakeRoot(x); Access(y); Splay(y);
    }
    inline bool Link(int x, int y) { // 连接 x,y 两点
        MakeRoot(x);
        if (FindRoot(y) == x) return false;
        f[x] = y;
        return true;
    }
    inline bool Cut(int x, int y) { // x,y 断边
        MakeRoot(x);
        if (FindRoot(y) == x && f[y] == x && !ch[y][0]) {
            f[y] = ch[x][1] = 0; PushUp(x);
            return true;
        }
        return false;
    }
}
```

3.2 KD Tree

```
// KDTree 二维平面邻域查询 K 远点对 n=1e5 k=100
priority_queue<ll, vector<ll>, greater<ll> >q; // 小根堆
namespace KDTree {
    struct node {
        int X[2];
        int &operator[](const int k) {return X[k];}
    } p[N];
    int nowd;
    bool cmp(node a, node b) {return a.X[nowd] < b.X[nowd];}
    int lc[N], rc[N], L[N][2], R[N][2]; // lc/rc 左右孩子; L/R 对应超矩形各个维度范围
    inline ll sqr(int x) {return 1ll * x * x;}
    void pushup(int x) { // 更新该点所代表空间范围
        L[x][0] = R[x][0] = p[x][0];
        L[x][1] = R[x][1] = p[x][1];
        if (lc[x]) {
            umin(L[x][0], L[lc[x]][0]); umax(R[x][0], R[lc[x]][0]);
            umin(L[x][1], L[lc[x]][1]); umax(R[x][1], R[lc[x]][1]);
        }
        if (rc[x]) {
            umin(L[x][0], L[rc[x]][0]); umax(R[x][0], R[rc[x]][0]);
            umin(L[x][1], L[rc[x]][1]); umax(R[x][1], R[rc[x]][1]);
        }
    }
    int build(int l, int r) {
        if (l > r) return 0;
        int mid = (l + r) >> 1;
        // >>> 方差建树
        db av[2] = {0, 0}, va[2] = {0, 0}; // av 平均数, va 方差
        for (int i = l; i <= r; ++i) av[0] += p[i][0], av[1] += p[i][1];
        av[0] /= (r - l + 1); av[1] /= (r - l + 1);
        for (int i = l; i <= r; ++i) {
            va[0] += sqr(av[0] - p[i][0]);
            va[1] += sqr(av[1] - p[i][1]);
        }
        if (va[0] > va[1]) nowd = 0;
        else nowd = 1; // 找方差大的维度划分
        // >>> 轮换建树 nowd=dep%D
        nth_element(p + l, p + mid, p + r + 1, cmp); // 以该维度中位数分割
        lc[mid] = build(l, mid - 1); rc[mid] = build(mid + 1, r);
        pushup(mid);
        return mid;
    }
    ll dist(int a, int x) { // 估价函数, 点 a 到树上 x 点对应空间最远距离
        return max(sqr(p[a][0] - L[x][0]), sqr(p[a][0] - R[x][0])) +
            max(sqr(p[a][1] - L[x][1]), sqr(p[a][1] - R[x][1]));
    }
    void query(int l, int r, int a) { // 点 a 邻域查询
        if (l > r) return ;
        int mid = (l + r) >> 1;
        ll t = sqr(p[mid][0] - p[a][0]) + sqr(p[mid][1] - p[a][1]);
        if (t > q.top()) q.pop(), q.push(t); // 更新答案
        ll disl = dist(a, lc[mid]), disr = dist(a, rc[mid]);
        if (disl > q.top() && disr > q.top()) // 两边都有机会更新, 优先搜大的
            (disl > disr) ? (query(l, mid - 1, a), query(mid + 1, r, a)) : (query(mid + 1, r, a), query(l, mid - 1, a));
        else
            (disl > q.top()) ? query(l, mid - 1, a) : query(mid + 1, r, a);
    }
}
using namespace KDTree;
int main() {
    red(n); red(k); k *= 2;
    for (int i = 1; i <= k; ++i) q.push(0);
    for (int i = 1; i <= n; ++i) red(p[i][0]), red(p[i][1]);
    build(1, n);
    for (int i = 1; i <= n; ++i) query(1, n, i);
    printf("%lld\n", q.top());
}
```

```
// 动态 KDTree 维护空间权值 (单点修改 & 空间查询)
// 时间复杂度 O(log n) ~ O(n^(1-1/k))
```

```cpp
#define sqr(x) ((x) * (x))
namespace KDT {
    struct dat {
        int X[2];
        int &operator[](const int k) {return X[k];}
    } p[N];
    db alp = 0.725; // 重构常数
    int nowd;
    bool cmp(int a, int b) {return p[a][nowd] < p[b][nowd];}
    //  root: 根 cur: 总点数 d: 当前分割维度 lc/rc: 左右儿子 L/R: 当前空间范围 siz: 子树大小  sum/val 空间的值，单点的值
    int root, cur, d[N], lc[N], rc[N], L[N][2], R[N][2], siz[N], sum[N], val[N];
    int g[N], t; // 用于重构的临时数组
    void pushup(int x) {
        siz[x] = siz[lc[x]] + siz[rc[x]] + 1;
        sum[x] = sum[lc[x]] + sum[rc[x]] + val[x];
        L[x][0] = R[x][0] = p[x][0];
        L[x][1] = R[x][1] = p[x][1];
        if (lc[x]) {
            umin(L[x][0], L[lc[x]][0]); umax(R[x][0], R[lc[x]][0]);
            umin(L[x][1], L[lc[x]][1]); umax(R[x][1], R[lc[x]][1]);
        }
        if (rc[x]) {
            umin(L[x][0], L[rc[x]][0]); umax(R[x][0], R[rc[x]][0]);
            umin(L[x][1], L[rc[x]][1]); umax(R[x][1], R[rc[x]][1]);
        }
    }
    int build(int l, int r) { // 对 g[1...t] 进行建树 , 对应点都是 g[x]。方差建树
        if (l > r) return 0;
        int mid = (l + r) >> 1;
        db av[2] = {0, 0}, va[2] = {0, 0};
        for (int i = l; i <= r; ++i) av[0] += p[g[i]][0], av[1] += p[g[i]][1];
        av[0] /= (r - l + 1); av[1] /= (r - l + 1);
        for (int i = l; i <= r; ++i) va[0] += sqr(av[0] - p[g[i]][0]), va[1] += sqr(av[1] - p[g[i]][1]);
        if (va[0] > va[1]) d[g[mid]] = nowd = 0;
        else d[g[mid]] = nowd = 1;
        nth_element(g + l, g + mid, g + r + 1, cmp);
        lc[g[mid]] = build(l, mid - 1); rc[g[mid]] = build(mid + 1, r);
        pushup(g[mid]);
        return g[mid];
    }
    void expand(int x) { // 将子树展开到临时数组里
        if (!x) return;
        expand(lc[x]);
        g[++t] = x;
        expand(rc[x]);
    }
    void rebuild(int &x) { // x 所在子树重构
        t = 0; expand(x);
        x = build(1, t);
    }
    bool chk(int x) {return alp * siz[x] <= (db)max(siz[lc[x]], siz[rc[x]]);} // 判断失衡
    void insert(int &x, int a) { // 插入点 a , p[a],val[a] 为其信息
        if (!x) { x = a; pushup(x); d[x] = rand() & 1; return; }
        if (p[a][d[x]] <= p[x][d[x]]) insert(lc[x], a);
        else insert(rc[x], a);
        pushup(x);
        if (chk(x)) rebuild(x); // 失衡暴力重构
    }
    dat Lt, Rt; // 询问一块空间的值 (为了减小常数把参数放在外面)
    int query(int x) {
        if (!x || Rt[0] < L[x][0] || Lt[0] > R[x][0] || Rt[1] < L[x][1]
            || Lt[1] > R[x][1]) return 0; // 结点为空或与询问取间无交
        if (Lt[0] <= L[x][0] && R[x][0] <= Rt[0] && Lt[1] <= L[x][1]
            && R[x][1] <= Rt[1]) return sum[x]; // 区间完全覆盖
        int ret = 0;
        if (Lt[0] <= p[x][0] && p[x][0] <= Rt[0] && Lt[1] <= p[x][1]
            && p[x][1] <= Rt[1]) ret += val[x]; // 当前点在区间内
        return query(lc[x]) + query(rc[x]) + ret;
    }
}
using namespace KDT;
int main() {
    int n; read(n);
    for (int op;;) {
        read(op);
        switch (op) {
        case 1:
            ++cur; read(p[cur][0]); read(p[cur][1]); read(val[cur]);
            insert(root, cur);
            break;
        case 2:
            read(Lt[0]); read(Lt[1]); read(Rt[0]); read(Rt[1]);
            printf("%d\n", query(root));
            break;
        case 3: | return 0; break;
        }
    }
    return 0;
}
```

3.3 李超线段树

```cpp
// 李超线段树 对于 (x1,y1) (x2,y2) -> y=0*x+max(y1,y2) [x1,x1]
#define ls (x<<1)
#define rs (x<<1|1)
typedef long long ll;
typedef double db;
const int N = 100010;
const int M = 40000;
struct line {
    db k, b;
} lin[N];
db val(int id, db X) {return lin[id].k * X + lin[id].b;}
int D[N << 2], n, id;
void modify(int L, int R, int id, int l = 1, int r = M - 1, int x = 1) { // 线 lin[id], 范围 [L, R]
    if (L <= l && r <= R) {
        int mid = (l + r) >> 1, lid = D[x];
        db lst = val(D[x], mid), now = val(id, mid);
        if (l == r) {if (now > lst) D[x] = id; return ;}
        if (lin[id].k > lin[D[x]].k) {
            if (now > lst) D[x] = id, modify(L, R, lid, l, mid, ls); // id->lid
            else modify(L, R, id, mid + 1, r, rs);
        } else if (lin[id].k < lin[D[x]].k) {
            if (now > lst) D[x] = id, modify(L, R, lid, mid + 1, r, rs); // id->lid
            else modify(L, R, id, l, mid, ls);
        } else if (lin[id].b > lin[D[x]].k) D[x] = id;
        return ;
    }
    int mid = (l + r) >> 1;
    if (L <= mid) modify(L, R, id, l, mid, x << 1);
    if (R > mid) modify(L, R, id, mid + 1, r, x << 1 | 1);
}
int gmax(int x, int y, int ps) {
    if (val(x, ps) > val(y, ps)) return x;
    if (val(x, ps) < val(y, ps)) return y;
    return (x < y) ? x : y;
}
int query(int ps, int l = 1, int r = M - 1, int x = 1) { // 查 x=ps
    if (l == r) return D[x];
    int mid = (l + r) >> 1, ret = D[x], t = 0;
    if (ps <= mid)
        t = query(ps, l, mid, ls);

    else
        t = query(ps, mid + 1, r, rs);
    return gmax(ret, t, ps);
```

```
}
```

3.4 吉司机线段树

```
/*
 * seg-beats 吉司机线段树
 * 区间最值操作
 * 支持 区间取min, 区间取max, 区间加减, 区间求和, 区间最小/大
 * 复杂度 O(m log n)
 */
#define ls (x << 1)
#define rs (x << 1 | 1)
#define mid ((l + r) >> 1)
typedef long long ll;
const int N = 500010;
const int inf = 0x3f3f3f3f;
struct datmn {
    int fi, se, cnt; // 最小值, 次小值, 最小值个数
    datmn() {fi = se = inf; cnt = 0;}
    void ins(int x, int c) {
        if (x < fi) se = fi, cnt = c, fi = x;
        else if (x == fi) cnt += c;
        else if (x < se) se = x;
    }
    friend datmn operator+(const datmn &a, const datmn &b) {
        datmn r = a; r.ins(b.fi, b.cnt); r.ins(b.se, 0); return r;
    }
};
struct datmx {
    int fi, se, cnt;
    datmx() {fi = se = -inf; cnt = 0;}
    void ins(int x, int c) {
        if (x > fi) se = fi, cnt = c, fi = x;
        else if (x == fi) cnt += c;
        else if (x > se) se = x;
    }
    friend datmx operator+(const datmx &a, const datmx &b) {
        datmx r = a; r.ins(b.fi, b.cnt); r.ins(b.se, 0); return r;
    }
};

struct node {
    datmn mn; datmx mx;
    ll sum; int addmn, addmx, add, len;
} t[N << 2];
int n, m, a[N];
void pushup(int x) {
    t[x].mx = t[ls].mx + t[rs].mx;
    t[x].mn = t[ls].mn + t[rs].mn;
    t[x].sum = t[ls].sum + t[rs].sum;
}
void build(int l = 1, int r = n, int x = 1) {
    t[x].add = t[x].addmn = t[x].addmx = 0;
    t[x].len = r - l + 1;
    if (l == r) {
        t[x].mx = datmx(); t[x].mx.ins(a[l], 1);
        t[x].mn = datmn(); t[x].mn.ins(a[l], 1);
        t[x].sum = a[l];
        return;
    }
    build(l, mid, ls); build(mid + 1, r, rs);
    pushup(x);
}
void update(int x, int vn, int vx, int v) { // vn: addmn, vx: addmx, v: add
    // 所有数相同特判, 此时最大值 tag 和最小值 tag 应该相同且不等于其他值 tag
    if (t[x].mn.fi == t[x].mx.fi) {
        if (vn == v) vn = vx;
        else vx = vn;
        t[x].sum += (ll)vn * t[x].mn.cnt;
    } else t[x].sum += (ll)vn * t[x].mn.cnt + (ll) vx * t[x].mx.cnt + (ll)v * (t[x].len - t[x].mn.cnt - t[x].mx.cnt);
    if (t[x].mn.se == t[x].mx.fi) t[x].mn.se += vx; // 次小值 = 最大值, 应该用最大值 tag 处理
    else if (t[x].mn.se != inf) t[x].mn.se += v;
    if (t[x].mx.se == t[x].mn.fi) t[x].mx.se += vn; // 次大值同理
    else if (t[x].mx.se != -inf) t[x].mx.se += v;
    t[x].mn.fi += vn; t[x].mx.fi += vx;
    t[x].addmn += vn; t[x].addmx += vx; t[x].add += v;
}
void pushdown(int x) {
    int mn = min(t[ls].mn.fi, t[rs].mn.fi);
    int mx = max(t[ls].mx.fi, t[rs].mx.fi);
    update(ls, (mn == t[ls].mn.fi) ? t[x].addmn : t[x].add, (mx == t[ls].mx.fi) ? t[x].addmx : t[x].add, t[x].add);
    update(rs, (mn == t[rs].mn.fi) ? t[x].addmn : t[x].add, (mx == t[rs].mx.fi) ? t[x].addmx : t[x].add, t[x].add);
    t[x].add = t[x].addmn = t[x].addmx = 0;
}
void modifyadd(int L, int R, int v, int l = 1, int r = n, int x = 1) {
    if (r < L || R < l) return ;
    if (L <= l && r <= R) return update(x, v, v, v);
    pushdown(x);
    modifyadd(L, R, v, l, mid, ls);
    modifyadd(L, R, v, mid + 1, r, rs);
    pushup(x);
}
void modifymin(int L, int R, int v, int l = 1, int r = n, int x = 1) {
    if (r < L || R < l) return ;
    if (L <= l && r <= R && v > t[x].mx.se) {
        if (v >= t[x].mx.fi) return ;
        update(x, 0, v - t[x].mx.fi, 0);
        return;
    }
    pushdown(x);
    modifymin(L, R, v, l, mid, ls);
    modifymin(L, R, v, mid + 1, r, rs);
    pushup(x);
}
void modifymax(int L, int R, int v, int l = 1, int r = n, int x = 1) {
    if (r < L || R < l) return ;
    if (L <= l && r <= R && v < t[x].mn.se) {
        if (v <= t[x].mn.fi) return;
        update(x, v - t[x].mn.fi, 0, 0);
        return;
    }
    pushdown(x);
    modifymax(L, R, v, l, mid, ls);
    modifymax(L, R, v, mid + 1, r, rs);
    pushup(x);
}
int querymax(int L, int R, int l = 1, int r = n, int x = 1) {
    if (r < L || R < l) return -inf;
    if (L <= l && r <= R) return t[x].mx.fi;
    pushdown(x);
    return max(querymax(L, R, l, mid, ls), querymax(L, R, mid + 1, r, rs));
}
int querymin(int L, int R, int l = 1, int r = n, int x = 1) {
    if (r < L || R < l) return inf;
    if (L <= l && r <= R) return t[x].mn.fi;
    pushdown(x);
    return min(querymin(L, R, l, mid, ls), querymin(L, R, mid + 1, r, rs));
}
ll querysum(int L, int R, int l = 1, int r = n, int x = 1) {
    if (r < L || R < l) return 0;
    if (L <= l && r <= R) return t[x].sum;
    pushdown(x);
    return querysum(L, R, l, mid, ls) + querysum(L, R, mid + 1, r, rs);
}
```

3.5 FHQ Treep

```
// fhq - treap 简易模板
#define ls(p) t[p].l
#define rs(p) t[p].r
#define mid ((l+r)>>1)
using namespace std;
```

```cpp
const int N = 100010;
mt19937 rd(random_device{}());
struct node {
    int l, r, siz, rnd, val, tag;
} t[N]; int tot, root;
/* 节点回收
int cyc[N],cyccnt;
inline void delnode(int p) {cyc[++cyccnt]=p;}
inline void newnode(int val) {
    int id=(cyccnt>0)?cyc[cyccnt--]:++tot;
    t[id]={0,0,1,(int)(rd()),val}; return id;
}
*/
inline int newnode(int val) { t[++tot] = {0, 0, 1, (int)(rd()), val}; return tot;}
inline void updata(int p) {
    t[p].siz = t[ls(p)].siz + t[rs(p)].siz + 1;
    /* maintain */
}
inline void pushtag(int p, int vl) { /* tag to push */ }
inline void pushdown(int p) {
    if (t[p].tag != std_tag) {
        if (ls(p)) pushtag(ls(p), t[p].tag);
        if (rs(p)) pushtag(rs(p), t[p].tag);
        t[p].tag = std_tag;
    }
}
int merge(int p, int q) {
    if (!p || !q) return p + q;
    if (t[p].rnd < t[q].rnd) {
        pushdown(p);
        rs(p) = merge(rs(p), q);
        updata(p); return p;
    } else {
        pushdown(q);
        ls(q) = merge(p, ls(q));
        updata(q); return q;
    }
}
void split(int p, int k, int &x, int &y) {
    if (!p) x = 0, y = 0;
    else {
        pushdown(p);
        if (t[ls(p)].siz >= k) y = p, split(ls(p), k, x, ls(p));
        else x = p, split(rs(p), k - t[ls(p)].siz - 1, rs(p), y);
        updata(p);
    }
}
int build(int l, int r) { // build tree on a[l..r], return the root
    if (l > r) return 0;
    return merge(build(l, mid - 1), merge(newnode(a[mid]), build(mid + 1, r)));
}
```

3.6 哈希表

```cpp
typedef long long ll;
const int M = 19260817;
const int MAX_SIZE = 2000000;
struct Hash_map {
    struct data {
        int nxt;
        ll key, value; // (key,value)
    } e[MAX_SIZE];
    int head[M], size;
    inline int f(ll key) { return key % M; }
    ll &operator[](const ll &key) {
        int ky = f(key);
        for (int i = head[ky]; i != -1; i = e[i].nxt)
            if (e[i].key == key) return e[i].value;
        return e[++size] = data{head[ky], key, 0}, head[ky] = size, e[size].value;
    }
    void clear() {
        memset(head, -1, sizeof(head));
        size = 0;
    }
    Hash_map() {clear();}
};
```

4. String

4.1 最小表示法

```cpp
//n为串长, a下标从0开始
int Min_show(int *a, int n) {
    int i = 0, j = 1, k = 0;
    while (i < n && j < n && k < n) {
        auto u = a[(i + k) % n];
        auto v = a[(j + k) % n];
        if (u == v) ++k;
        else {
            if (u > v) i += k + 1;
            else j += k + 1;
            if (i == j) ++j;
            k = 0;
        }
    }
    return min(i, j);
}
```

4.2 AC 自动机

```cpp
int son[M][26], fail[M], cnt = 0;
void ins(const char *s) {
    int p = 0, n = strlen(s + 1);
    for (int i = 1; i <= n; ++i) {
        int c = s[i] - 'a';
        if (!son[p][c]) son[p][c] = ++cnt;
        p = son[p][c];
    }
}
queue<int> q;
void get_fail() {
    for (int c = 0; c < 26; ++c)
        if (son[0][c]) q.push(son[0][c]);
    while (!q.empty()) {
        int x = q.front(); q.pop();
        for (int c = 0; c < 26; ++c) {
            if (son[x][c]) {
                fail[son[x][c]] = son[fail[x]][c];
                q.push(son[x][c]);
            } else son[x][c] = son[fail[x]][c];
        }
    }
}

```

4.3 回文树

```cpp
int len[M], fa[M], son[M][26], lst, cnt, f[M];
char s[M];
int extend(int n) {
    int p = lst, c = s[n] - 'a';
    while (s[n - len[p] - 1] != s[n]) p = fa[p];
    if (!son[p][c]) {
        int now = p;
        len[++cnt] = len[p] + 2;//回文串长度
        p = fa[p];
        while (s[n - len[p] - 1] != s[n]) p = fa[p];
        fa[cnt] = son[p][c];
        lst = son[now][c] = cnt;
        f[cnt] = f[fa[cnt]] + 1;//回文串数量
    } else lst = son[p][c];
    return f[lst];
}
int main() {
    fa[0] = cnt = 1;
    val[1] = -1;
}
```

4.4 Manacher

```cpp
char s[M << 1];
int p[M];
//n为串长, a下标从1开始, p为回文串半径 (0~2n+1)
void Manacher(const char *a, int n) {
    int r = 0, mid;
    for (int i = 1; i <= n; ++i) s[i << 1] = a[i];
    for (int i = 0; i <= n; ++i) s[i * 2 + 1] = '#';
    s[0] = '#'; n = n << 1 | 1;
    for (int i = 1; i <= n; ++i) {
```

```
        p[i] = (i <= r ? min(p[mid * 2 - i], p[mid] + mid - i) : 1);
        while (s[i - p[i] - 1] == s[i + p[i] + 1]) ++p[i];
        if (i + p[i] > r) r = i + p[i], mid = i;
    }
}
```

4.5 字符串哈希

```
const int HA = 2;
const int PP[] = {318255569, 66604919, 19260817}, QQ[] = {1010451419, 1011111133, 1033111117};

int pw[HA][N];
void HashInit() {
    for (int h = 0; h < HA; h++) {
        pw[h][0] = 1;
        for (int i = 1; i < N; i++)
            pw[h][i] = (LL)pw[h][i - 1] * PP[h] % QQ[h];
    }
}
struct Hash {
    int hs[HA], len;
    Hash() {
        memset(hs, 0, sizeof hs);
        len = 0;
    }
    Hash(int x) {
        for (int h = 0; h < HA; h++) hs[h] = x;
        len = 1;
    }
    Hash operator + (const int &x)const {
        Hash res;
        res.len = len + 1;
        for (int h = 0; h < HA; h++)
            res.hs[h] = ((LL)hs[h] * PP[h] + x) % QQ[h];
        return res;
    }
    Hash operator - (const Hash &x)const {
        Hash res;
        res.len = len - x.len;
        for (int h = 0; h < HA; h++) {
            res.hs[h] = (hs[h] - (LL)pw[h][res.len] * x.hs[h]) % QQ[h];
            if (res.hs[h] < 0) res.hs[h] += QQ[h];
        }
        return res;
    }
    bool operator == (const Hash &x)const {
        for (int h = 0; h < HA; h++)
            if (hs[h] != x.hs[h]) return false;
        return len == x.len;
    }
    // below : not that frequently used
    Hash operator + (const Hash &x)const {
        Hash res;
        res.len = len + x.len;
        for (int h = 0; h < HA; h++)
            res.hs[h] = ((LL)hs[h] * pw[h][x.len] + x.hs[h]) % QQ[h];
        return res;
    }
} H;
Hash operator + (const int &a, const Hash &b) {
    Hash res;
    res.len = b.len + 1;
    for (int h = 0; h < HA; h++)
        res.hs[h] = ((LL)a * pw[h][b.len] + b.hs[h]) % QQ[h];
    return res;
}
```

4.6 SA

```
// rnk: 排名, sa: 位置
// height[i] = lcp(sa[i], sa[i - 1])
// M开两倍
void get_sa(char *s, int n, int *sa, int *rnk, int *height) { // 1-based
    static int c[M], p[M], t[M];
    int m = 300;
    for (int i = 1; i <= n; ++i) ++c[p[i] = s[i]];
    for (int i = 2; i <= m; ++i) c[i] += c[i - 1];
    for (int i = n; i; --i) sa[c[p[i]]--] = i;
    for (int k = 1; k < n ; k <<= 1) {
        int cnt = 0;
        for (int i = n - k + 1; i <= n; ++i) t[++cnt] = i;
        for (int i = 1; i <= n; ++i) if (sa[i] > k) t[++cnt] = sa[i] - k;
        for (int i = 1; i <= m; ++i) c[i] = 0;
        for (int i = 1; i <= n; ++i) ++c[p[i]];
        for (int i = 2; i <= m; ++i) c[i] += c[i - 1];
        for (int i = n; i; --i) sa[c[p[t[i]]]--] = t[i], t[i] = 0;
        swap(p, t);
        p[sa[1]] = cnt = 1;
        for (int i = 2; i <= n; ++i) {
            if (t[sa[i]] != t[sa[i - 1]] || t[sa[i] + k] != t[sa[i - 1] + k]) ++cnt;
            p[sa[i]] = cnt;
        }
        if (cnt == n) break;
        m = cnt;
    }
    for (int i = 1; i <= n; i++) rnk[sa[i]] = i;
    for (int i = 1, k = 0; i <= n; i++) {
        if (k) k--;
        while (s[i + k] == s[sa[rnk[i] - 1] + k]) k++;
        height[rnk[i]] = k;
    }
}
char s[M];
int sa[M], rnk[M], height[M];
int main() {
    cin >> (s + 1);
    int n = strlen(s + 1);
    get_sa(s, n, sa, rnk, height);
    for (int i = 1; i <= n; i++)
        cout << sa[i] << (i < n ? ' ' : '\n');
    for (int i = 2; i <= n; i++)
        cout << height[i] << (i < n ? ' ' : '\n');
    return 0;
}
```

4.7 SAM

```
int lst = 1, cnt = 1, len[M], fa[M], son[M][26];
void Extend(int c) { // 结点数要开成串长的两倍
    int p = lst, np = lst = ++cnt;
    len[np] = len[p] + 1;
    for (; p && !son[p][c]; p = fa[p]) son[p][c] = np;
    if (!p) return fa[lst = np] = 1, void();
    int q = son[p][c];
    if (len[q] == len[p] + 1)
        return fa[lst = np] = q, void();
    int nq = ++cnt;
    len[nq] = len[p] + 1;
    fa[nq] = fa[q];
    fa[np] = fa[q] = nq;
    memcpy(son[nq], son[q], sizeof(son[q]));
    for (; p && son[p][c] == q; p = fa[p]) son[p][c] = nq;
    lst = np;
}
int c[M], q[M];
int main() {
    for (int i = 1; i <= n ; ++i) Extend(s[i] - 'a');
    for (int i = 1; i <= cnt; i++) ++c[len[i]];
    for (int i = 1; i <= cnt; i++) c[i] += c[i - 1];
    for (int i = 1; i <= cnt; i++) q[c[len[i]]--] = i;
    return 0;
}
```

4.8 KMP and EXKMP

```
// 1-based
int fail[M];
void KMP(const char *s, int n) {
    fail[0] = fail[1] = 0;
    for (int i = 2, j = 0; i <= n; i++) {
        fail[i] = 0;
        while (j && s[i] != s[j + 1]) j = fail[j];
        if (s[i] == s[j + 1]) fail[i] = ++j;
    }
}
// match
```

```cpp
for (int i = 1, j = 0; i <= la; ++i) {
    while (j && b[j + 1] != a[i]) j = fail[j];
    if (b[j + 1] == a[i]) ++j;
    if (j == lb) {
        printf("%d\n", i - lb + 1);
        j = fail[j];
    }
}
// 0-based
// s 和 s 的每一个后缀的最长公共前缀 (LCP) 长度数组
void exKMP(const char *s, int *z, int n) {// get z
    int l = 0, r = 0;
    z[0] = n;
    for (int i = 1; i <= n; ++i) {
        z[i] = i > r ? 0 : 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) r = i + z[l = i] - 1;
    }
}
// t 与 s 的每一个后缀的 LCP 长度数组
void exKMP(const char *s, const char *t, int *z, int *p,
    int sn) {// get p
    int l = -1, r = -1;
    for (int i = 0; i <= sn; ++i) {
        p[i] = i > r ? 0 : min(r - i + 1, z[i - l]);
        while (i + p[i] < sn && t[p[i]] == s[i + p[i]]) +
            +p[i];
        if (i + p[i] - 1 > r) r = i + p[l = i] - 1;
    }
}
```

4.9 Lydon

```cpp
/*
满足s的最小后缀等于s本身的串s称为Lyndon串.
等价于: s是它自己的所有循环移位中唯一最小的一个.
任意字符串s可以分解为 s = s1s2...sk , 其中 si 是Lyndon串,
si ≥si+1. 且这种分解方法是唯一的.
*/
void mnsuf(char *s, int *mn, int n) { // 每个前缀的最小后缀
    for (int i = 0; i < n; ) {
        int j = i, k = i + 1;
        mn[i] = i;
        for (; k < n && s[j] <= s[k]; ++ k)
            if (s[j] == s[k]) mn[k] = mn[j] + k - j, ++j;
            else mn[k] = j = i;
        while(i <= j)i += k - j;
    }
} // lyn+=s[i..i+kj1]
void mxsuf(char *s, int *mx, int n) { // 每个前缀的最大后缀
    fill(mx, mx + n, -1);
    for (int i = 0; i < n; ) {
        int j = i, k = i + 1;
        if (mx[i] == -1) mx[i] = i;
        for (; k < n && s[j] >= s[k]; ++k) {
            j = s[j] == s[k] ? j + 1 : i;
            if (mx[k] == -1) mx[k] = i;
        }
        while(i <= j)i += k - j;
    }
}
```

4.10 SASAM后缀树

```cpp
const int M=1e5;
bool vis[M << 1];
char s[M];
int id[M << 1], ch[M << 1][26], height[M], tim = 0;
void dfs(int x) {
    if (id[x]) {
        height[tim++] = val[lst];
        sa[tim] = id[x];
        lst = x;
    }
    for (int c = 0; c < 26; ++c)
        if (son[x][c]) dfs(son[x][c]);
    lst = fa[x];
}
int main() {
    lst = ++cnt;
    scanf("%s", s + 1);
    int n = strlen(s + 1);
    for (int i = n; i; --i) {
        expand(s[i] - 'a');
        id[lst] = i;
    }
    vis[1] = 1;
    for (int i = 1; i <= cnt; ++i) if (id[i])
            for (int x = i,pos = n; x && !vis[x]; x = fa[x]) {
                vis[x] = 1;
                pos -= val[x] - val[fa[x]];
                son[fa[x]][s[pos + 1] - 'a'] = x;
            }
    dfs(1);
    for (int i = 1; i <= n; ++i) printf("%d",sa[i]);
        puts("");
    for (int i = 1; i < n; ++i) printf("%d",height[i]);
        puts("");
    return 0;
}
```

4.11 后缀平衡树

```cpp
#include <bits/stdc++.h>
using namespace std;
const int M = 3e6 + 5;
const double INF = 1e18;

void decode(char *s, int len, int mask) {
    for (int i = 0; i < len; ++i) {
        mask = (mask * 131 + i) % len;
        swap(s[i], s[mask]);
    }
}
int q, n, len;
char s[M], t[M];
// 顺序加入, 查询时将询问串翻转
// 以i结束的前缀, 对应节点的编号为i
// 注意: 不能写懒惰删除, 否则可能会破坏树的结构
const double alpha = 0.75;
int rt, sz[M], ls[M], rs[M];
double tag[M];
int buffer_size, buffer[M];

bool cmp(int x, int y) {
    if (t[x] != t[y]) return t[x] < t[y];
    return tag[x - 1] < tag[y - 1];
}
void nw(int &rt, int p, double lv, double rv) {
    sz[rt = p] = 1;
    tag[rt] = (lv + rv) / 2;
    ls[rt] = rs[rt] = 0;
}
void Up(int x) {
    if (!x) return;
    sz[x] = sz[ls[x]] + sz[rs[x]] + 1;
}
bool balance(int rt) {
    return alpha * sz[rt] > max(sz[ls[rt]], sz[rs[rt]]);
}
void flatten(int rt) {
    if (!rt) return;
    flatten(ls[rt]);
    buffer[++buffer_size] = rt;
    flatten(rs[rt]);
}
void build(int &rt, int l, int r, double lv, double rv) {
    if (l > r) return rt = 0, void();
    int mid = (l + r) >> 1;
    double mv = (lv + rv) / 2;
    rt = buffer[mid];
    tag[rt] = mv;
    build(ls[rt], l, mid - 1, lv, mv);
    build(rs[rt], mid + 1, r, mv, rv);
    Up(rt);
}
void rebuild(int &rt, double lv, double rv) {
    buffer_size = 0;
    flatten(rt);
    build(rt, 1, buffer_size, lv, rv);
}
void ins(int& rt, int p, double lv = 0, double rv = INF) {
    if (!rt) return nw(rt, p, lv, rv), void();
    if (cmp(p, rt)) ins(ls[rt], p, lv, tag[rt]);
```

```cpp
    else ins(rs[rt], p, tag[rt], rv);
    Up(rt);
    if (!balance(rt)) rebuild(rt, lv, rv);
}
void remove(int &rt, int p, double lv = 0, double rv = INF) {
    if (!rt) return;
    if (rt == p) {
        if (!ls[rt] || !rs[rt]) {
            rt = (ls[rt] | rs[rt]);
            rebuild(rt, lv, rv);
        }
        else {
            int nrt = ls[rt];
            while (rs[nrt]) nrt = rs[nrt];
            remove(ls[rt], nrt, lv, tag[rt]);
            ls[nrt] = ls[rt];
            rs[nrt] = rs[rt];
            rt = nrt;
            tag[rt] = (lv + rv) / 2;
        }
    }
    else {
        double mv = (lv + rv) / 2;
        if (cmp(p, rt)) remove(ls[rt], p, lv, mv);
        else remove(rs[rt], p, mv, rv);
    }
    Up(rt);
    if (!balance(rt)) rebuild(rt, lv, rv);
}
bool qry_cmp(char *s, int len, int p) {
    for (int i = 1; i <= len; ++i, --p)
        if (s[i] != t[p]) return s[i] < t[p];
    return 0;
}
int qry(int rt, char *s, int len) {
    if (!rt) return 0;
    if (qry_cmp(s, len, rt)) return qry(ls[rt], s, len);
    else return sz[ls[rt]] + 1 + qry(rs[rt], s, len);
}
int main() {
    scanf("%d", &q);
    scanf("%s", s + 1); len = strlen(s + 1);
    for (int i = 1; i <= len; ++i)
        t[++n] = s[i], ins(rt, n);
    int mask = 0;
    char op[10];
    for (int i = 1; i <= q; ++i) {
        scanf("%s", op);
        if (op[0] == 'A') {
            scanf("%s", s + 1); len = strlen(s + 1);
            decode(s + 1, len, mask);
            for (int i = 1; i <= len; ++i)
                t[++n] = s[i], ins(rt, n);
        }
        if (op[0] == 'D') {
            int x; scanf("%d", &x);
            while (x--) remove(rt, n), --n;
        }
        if (op[0] == 'Q') {
            scanf("%s", s + 1); len = strlen(s + 1);
            decode(s + 1, len, mask);
            reverse(s + 1, s + len + 1);
            s[len + 1] = 'Z' + 1;
            s[len + 2] = 0;
            int ans = qry(rt, s, len + 1);
            --s[len];
            ans -= qry(rt, s, len + 1);
            printf("%d\n", ans);
            mask ^= ans;
        }
    }
    return 0;
}
```

5. Polynomial

5.1 FFT

```cpp
struct com {
    db a, b;
    com operator + (const com &x)const {
        return (com) {a + x.a, b + x.b};
    }
    com operator - (const com &x)const {
        return (com) {a - x.a, b - x.b};
    }
    com operator * (const com &x)const {
        return (com) {a *x.a - b *x.b, a *x.b + b *x.a};
    }
} a[T], b[T];
void FFT(com *a, int p) {
    for (int i = 0; i < lmt; ++i)
        if (i < rev[i]) swap(a[i], a[rev[i]]);
    for (int mid = 1; mid < lmt; mid <<= 1) {
        com Xn;
        Xn = (com) {cos(pi / mid), sin(pi * p / mid)};
        for (int l = 0; l < lmt; l += mid << 1) {
            com x; x = (com) {1, 0};
            for (int i = 0; i < mid; ++i) {
                com u = a[l + i], v = x * a[l + mid + i];
                a[l + i] = u + v;
                a[l + mid + i] = u - v;
                x = x * Xn;
            }
        }
    }
}
int main() {
    scanf("%d%d", &n, &m);
    for (int i = 0; i <= n; ++i) scanf("%lf", &a[i].a);
    for (int i = 0; i <= m; ++i) scanf("%lf", &b[i].a);
    lmt = 1;
    while (lmt <= n + m)lmt <<= 1, ++t;
    for (int i = 0; i < lmt; ++i) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (t - 1));
    FFT(a, 1);
    FFT(b, 1);
    for (int i = 0; i < lmt; ++i) a[i] = a[i] * b[i];
    FFT(a, -1);
    for (int i = 0; i <= n + m; ++i) printf("%d ", (int)(a[i].a / lmt + 0.5));
    return 0;
}
```

5.2 FMT & FWT

```cpp
void OR(int *a, int len, int x) {
    for (int mid = 1; mid < len; mid <<= 1)
        for (int l = 0; l < len; l += mid << 1)
            for (int i = l; i <= l + mid - 1; ++i)
                ADD(a[i + mid], 1ll * a[i] * x % P);
}
void AND(int *a,int len,int x) {
    for (int mid = 1; mid < len; mid <<= 1)
        for (int l = 0; l < len; l += mid << 1)
            for (int i = l; i <= l + mid - 1; ++i)
                ADD(a[i], 1ll * a[i + mid] * x % P);
}
void XOR(int *a,int len,int x) {
    for (int mid = 1; mid < len; mid <<= 1)
        for (int l = 0; l < len; l += mid << 1)
            for (int i = l; i <= l + mid - 1; ++i) {
                int u = a[i], v = a[i + mid];
                a[i] = 1ll * MOD(u + v) * x % P;
                a[i + mid] = 1ll * MOD(u - v) * x % P;
            }
}
int main() {
    for (int i = 0; i < len; ++i) a[i] = A[i], b[i] = B[i];
    OP(a, len, 1);
    OP(b, len, 1);
    for (int i = 0; i < len; ++i) a[i] = 1ll * a[i] * b[i] % P;
    OP(a, len, -1);
}
```

5.3 任意模数NTT

```cpp
const int P1 = 469762049, P2 = 998244353, P3 = 1004535809;
int n, m, P, rev[M], a[M], b[M], c[M], d[M], ans[3][M], lmt=1, t;
int PW(int x, int y, int P) {
    int res = 1;
```

```
  for (; y; y >>= 1) {
    if(y & 1)res = 1ll * res * x % P;
    x = 1ll * x * x % P;
  }
  return res;
}
LL MUL(LL a, LL b, LL P) {
  a %= P; b %= P;
  return ((a * b - (LL)((LL)((db)a / P * b + 1e-3) * P)) % P + P) % P;
}
void NTT(int *a, int op, int P) {
  for (int i = 0; i < lmt; ++i)
    if (i < rev[i]) swap(a[i], a[rev[i]]);
  for (int mid = 1; mid < lmt; mid <<= 1) {
    int wn = PW(3, (P - 1) / (mid << 1), P);
    for (int l = 0; l < lmt; l += mid << 1) {
      int w = 1;
      for (int i = 0; i < mid; ++i) {
        int u = a[l + i];
        int v = 1ll * w * a[l + mid + i]% P;
        a[l + i] = (u + v) % P;
        a[l + mid + i] = (u - v + P) % P;
        w = 1ll * w * wn % P;
      }
    }
  }
  if(!op) {
    int inv = PW(lmt, P - 2, P);
    a[0] = 1ll * a[0] * inv % P;
    for (int i = 1; i <= lmt>>1; ++i) {
      a[i] = 1ll * a[i] * inv % P;
      if (i != lmt - i) {
        a[lmt - i] = 1ll * a[lmt - i] * inv % P;
        swap(a[i], a[lmt - i]);
      }
    }
  }
}
int main() {
  n = rd(); m = rd(); P = rd();
  for (int i = 0; i <= n; ++i) a[i] = rd();
  for (int i = 0; i <= m; ++i) b[i] = rd();
  while (lmt <= n + m) lmt <<= 1, ++t;
  for (int i = 0; i < lmt; ++i)
    rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (t - 1));
  copy(a, a + n + 1, c);
  copy(b, b + m + 1, d);
  NTT(c, 1, P1);
  NTT(d, 1, P1);
  for (int i = 0; i < lmt; ++i)
    ans[0][i] = 1ll * c[i] * d[i] % P1;
  NTT(ans[0], 0, P1);
  memset(c, 0, sizeof(c));
  memset(d, 0, sizeof(d));
  copy(a, a + n + 1, c);
  copy(b, b + m + 1, d);
  NTT(c, 1, P2);
  NTT(d, 1, P2);
  for (int i = 0; i < lmt; ++i)
    ans[1][i] = 1ll * c[i] * d[i] % P2;
  NTT(ans[1], 0, P2);
  memset(c, 0, sizeof(c));
  memset(d, 0, sizeof(d));
  copy(a, a + n + 1, c);
  copy(b, b + m + 1, d);
  NTT(c, 1, P3);
  NTT(d, 1, P3);
  for (int i = 0; i < lmt; ++i)
    ans[2][i] = 1ll * c[i] * d[i] % P3;
  NTT(ans[2], 0, P3);
  LL f1 = 1ll * P1 * P2;
  int inv1 = PW(P2 % P1, P1 - 2, P1);
  int inv2 = PW(P1 % P2, P2 - 2, P2);
  int inv3 = PW(f1 % P3, P3 - 2, P3);
  for (int i = 0; i <= n + m; ++i) {
    LL A = (MUL(1ll * ans[0][i] * P2 % f1, inv1, f1)+
        MUL(1ll * ans[1][i] * P1 % f1, inv2, f1))%f1;
    LL k = ((ans[2][i] - A) % P3 + P3) % P3 * inv3 % P3;
    printf("%lld ",((k % P) * (f1 % P) % P + A % P) % P);
  }
  return 0;
}
```

5.4 多项式全家桶

```
/*
  NTT 多项式全家桶
  p_mul 乘法; p_inv 求逆; p_div 带余数除法; p_sqrt 开方;
  p_ln Ln;  p_exp EXP; p_int 积分; p_der 求导; p_pow 快速幂;
  DCFFT 分治 FFT 板子;

  to be continue ...
  多项式三角函数，多项式反三角函数，多项式多点求值，多项式快速差值.....
*/

#include <algorithm>
#include <iostream>
#include <cstring>
#include <cstdio>
#include <cmath>
#define clr(f,n) memset(f,0,sizeof(long long)*(n))
#define cpy(f,g,n) memcpy(f,g,sizeof(long long)*(n))
#define Outarr(x,n) cerr<<#x<<" : "; for(int i=0;i<n;++i) cerr<<x[i]<<" ";cout<<endl;
#define outarr(x,n) for(int i=0;i<n;++i) printf("%lld%c",x[i],(i==n-1)?'\n':' ');
#define MOD(x) ((x)<mod?(x):((x)%mod))
using namespace std;

typedef long long ll;

namespace poly {
const int mod = 998244353;
const int N = (1 << 19);
const int _G = 3;
const int _iG = 332748118;
const int inv2 = 499122177;

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

int rev[N], rev_n;
void prerev(int n) {
    if (n == rev_n) return;
    rev_n = n;
    for (int i = 0; i < n; ++i) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) ? (n >> 1) : 0);
}
// NTT : fg=1 DFT fg=-1 IDFT
void NTT(ll f[], int n, int fg) {
    prerev(n);
    for (int i = 0; i < n; ++i) if (i < rev[i]) swap(f[i], f[rev[i]]);
    for (int h = 2; h <= n; h <<= 1) {
        ll Dt = fpow((fg == 1) ? _G : _iG, (mod - 1) / h, mod), w;
        int len = h >> 1;
        for (int j = 0; j < n; j += h) {
            w = 1;
            for (int k = j; k < j + len; ++k) {
                ll tmp = MOD(f[k + len] * w);
                f[k + len] = f[k] - tmp; (f[k + len] < 0) &&(f[k + len] += mod);
                f[k] = f[k] + tmp; (f[k] >= mod) &&(f[k] -= mod);
                w = MOD(w * Dt);
            }
        }
    }
    if (fg == -1) {
        ll invn = fpow(n, mod - 2, mod);
        for (int i = 0; i < n; ++i) f[i] = MOD(f[i] * invn);
    }
}
// f(x) = f*g(x) n = def f ; m = def g ; len = 最终长度 (保留几位)
void p_mul(ll f[], ll g[], int n, int m, int len) {
```

```cpp
    static ll a[N], b[N];
    int nn = 1 << (int)ceil(log2(n + m - 1));
    clr(a, nn); clr(b, nn); cpy(a, f, n); cpy(b, g, m);
    NTT(a, nn, 1); NTT(b, nn, 1);
    for (int i = 0; i < nn; ++i) a[i] = MOD(a[i] * b[i]);
    NTT(a, nn, -1);
    for (int i = 0; i < len; ++i) f[i] = a[i];
}
// f(x) = g^-1(x)  f(x) 为 g(x) 模 x^n 意义下的逆
void p_inv(ll g[], int n, ll f[]) {
    static ll sav[N];
    int nn = 1 << (int)ceil(log2(n));
    clr(f, n * 2);
    f[0] = fpow(g[0], mod - 2, mod);
    for (int h = 2; h <= nn; h <<= 1) {
        cpy(sav, g, h); clr(sav + h, h);
        NTT(sav, h << 1, 1); NTT(f, h << 1, 1);
        for (int i = 0; i < (h << 1); ++i)
            f[i] = f[i] * (2ll - f[i] * sav[i] % mod + mod) % mod;
        NTT(f, h << 1, -1); clr(f + h, h);
    }
    clr(f + n, nn * 2 - n);
}
// f^2(x) = g(x)  f(x) 为 g(x) 模 x^n 意义下的开方
void p_sqrt(ll g[], int n, ll f[]) {
    static ll sav[N], r[N];
    int nn = 1 << (int)ceil(log2(n));
    clr(f, n * 2); f[0] = 1; // g[0] should be 1 otherwise
    for (int h = 2; h <= nn; h <<= 1) {
        cpy(sav, g, h); clr(sav + h, h); p_inv(f, h, r);
        NTT(sav, h << 1, 1); NTT(r, h << 1, 1);
        for (int i = 0; i < (h << 1); ++i) sav[i] = MOD(sav[i] * r[i]);
        NTT(sav, h << 1, -1);
        for (int i = 0; i < h; ++i) f[i] = MOD((f[i] + sav[i]) * inv2);
        clr(f + h, h);
    }
    clr(f + n, nn * 2 - n);
}
// f(x) = g(x) * q(x) + r(x) : q(x) 为商 r(x) 为余数
void p_div(ll f[], ll g[], int n, int m, ll q[], ll r[]) {
    static ll sav1[N], sav2[N];
    int nn;
    for (nn = 1; nn < n - m + 1; nn <<= 1);
    clr(sav1, nn); clr(sav2, nn); cpy(sav1, f, n); cpy(sav2, g, m);
    reverse(sav1, sav1 + n); reverse(sav2, sav2 + m);
    p_inv(sav2, n - m + 1, q); p_mul(q, sav1, n - m + 1, n, n - m + 1);
    reverse(q, q + n - m + 1); | cpy(r, g, m);
    p_mul(r, q, m, n - m + 1, m - 1);
    for (int i = 0; i < m - 1; ++i) r[i] = MOD(f[i] - r[i] + mod);
}
// 预处理乘法逆元
ll inv[N];
void Initinv(int n) {
    inv[0] = inv[1] = 1;
    for (int i = 2; i <= n; ++i) inv[i] = (mod - mod / i) * inv[mod % i] % mod;
}
// 对 f(x) 进行积分  Initinv() first
void p_int(ll f[], int n) {
    for (int i = n - 1; i; --i) f[i] = MOD(f[i - 1] * inv[i]);
    f[0] = 0;
}
// 对 f(x) 进行求导
void p_der(ll f[], int n) {
    for (int i = 1; i < n; ++i) f[i - 1] = MOD(f[i] * i);
    f[n - 1] = 0;
}
//  f(x) <- ln f(x)  f[0] should be 1
void p_ln(ll f[], int n) {
    static ll g[N];
    p_inv(f, n, g); p_der(f, n);
    p_mul(f, g, n, n, n + n);
    p_int(f, n);
}

// f(x) <- exp f(x) (倍增版) f[0] should be 0
void p_exp(ll f[], int n) {
    static ll g[N], sav[N];
    clr(g, n * 2); clr(sav, n * 2); g[0] = 1;
    for (int h = 2; h <= n; h <<= 1) {
        cpy(sav, g, h); p_ln(sav, h);
        for (int i = 0; i < h; ++i) sav[i] = MOD(f[i] - sav[i] + mod);
        sav[0] = MOD(sav[0] + 1);
        p_mul(g, sav, h, h, h);
    }
    cpy(f, g, n);
}


void _p_exp(ll f[],ll g[],int l,int r) {
    static ll A[N],B[N];
    if(r-l==1) {if(l>0) f[l]=MOD(f[l]*fpow(l,mod-2,mod));return ;}
    int mid=(l+r)>>1,len=mid-l;
    _p_exp(f,g,l,mid);
    cpy(A,f+l,len); clr(A+len,len); cpy(B,g,len<<1);
    p_mul(A,B,len<<1,len<<1,len<<1);
    for(int i=mid;i<r;++i) f[i]=MOD(f[i]+A[i-l]);
    _p_exp(f,g,mid,r);
}
// f(x) <- exp f(x) (分治 FFT 版) f[0] should be 0
void p_exp(ll f[],int n) {
    static ll g[N];
    cpy(g,f,n); clr(f,n); f[0]=1;
    for(int i=0;i<n;++i) g[i]=MOD(g[i]*i);
    _p_exp(f,g,0,n);
}


// f(x) <- f^k(x)  f(x) 模 x^n 意义下的 k 次
void p_pow(ll f[], int n, ll k) {
    p_ln(f, n);
    for (int i = 0; i < n; ++i) f[i] = MOD(f[i] * k);
    p_exp(f, n);
}

// 分治FFT [l,r)   F[n] = sum(0<i<=n) F[n-i]G[i]
void DCFFT(ll f[], ll g[], int l, int r) {
    static ll A[N], B[N];
    if (r - l == 1) return ;
    int mid = (l + r) >> 1, len = mid - l;
    DCFFT(f, g, l, mid);
    cpy(A, f + l, len); clr(A + len, len); cpy(B, g, len << 1);
    p_mul(A, B, len << 1, len << 1, len << 1);
    for (int i = mid; i < r; ++i) f[i] = MOD(f[i] + A[i - l]);
    DCFFT(f, g, mid, r);
}
}
```

6. Combinatorics

6.1 lucas & exlucas

```cpp
// lucas
ll Lucas(ll n, ll m, ll p) {
    if (m == 0) return 1;
    return (C(n % p, m % p, p) * Lucas(n / p, m / p, p)) % p;
}
// exlucas
int l, a[33], p[33], P[33];
// 求 n! mod pk^tk, 返回值 U{ 不包含 pk 的值 ,pk 出现的次数 }
U fac(int k, LL n) {
    if (!n)return U{1, 0}; LL x = n / p[k], y = n / P[k], ans = 1; int i;
    if (y) { // 求出循环节的答案
        for (i = 2; i < P[k]; i++)if (i % p[k])ans = ans * i % P[k];
        ans = Pw(ans, y, P[k]);
    } for (i = y * P[k]; i <= n; i++) if (i % p[k])ans = ans * i % M; // 求零散部分
    U z = fac(k, x); return U{ans *z.x % M, x + z.z};
} LL get(int k, LL n, LL m) { // 求 C(n,m) mod pk^tk
    U a = fac(k, n), b = fac(k, m), c = fac(k, n - m); // 分三部分求解
```

```cpp
    return Pw(p[k], a.z - b.z - c.z, P[k]) * a.x % P[k] * inv(b.x, P[k]) % P[k] * inv(c.x,
        P[k]) % P[k];
} LL CRT() { // CRT 合并答案
    LL d, w, y, x, ans = 0;
    fr(i, 1, l)w = M / P[i], exgcd(w, P[i], x, y), ans = (ans + w * x % M * a[i]) % M;
    return (ans + M) % M;
} LL C(LL n, LL m) { // 求 C(n,m)
    fr(i, 1, l)a[i] = get(i, n, m);
    return CRT();
} LL exLucas(LL n, LL m, int M) {
    int jj = M, i; // 求 C(n,m)mod M,M=prod(pi^ki), 时间 O(pi^kilg^2n)
    for (i = 2; i * i <= jj; i++)if (jj % i == 0) for (p[++l] = i, P[l] = 1; jj % i == 0;
        P[l] *= p[l])jj /= i;
    if (jj > 1)l++, p[l] = P[l] = jj;
    return C(n, m);
}
```

6.2 gauss

```cpp
// 系数零场数非零：无解；系数零常数零：无穷解。
void gauss() {
    for (int j = 1; j <= n; j++) {
        int mxp = j;
        for (int i = j + 1; i <= n; i++)
            if (fabs(a[i][j]) > fabs(a[mxp][j])) mxp = i;
        for (int i = 1; i <= n + 1; i++)
            swap(a[j][i], a[mxp][i]);
        if (a[j][j] == 0) continue;
        for (int i = 1; i <= n; i++) {
            if (i == j) continue;
            double tmp = a[i][j] / a[j][j];
            for (int k = 1; k <= n + 1; k++)
                a[i][k] -= a[j][k] * tmp;
        }
    }
    for (int i = 1; i <= n; i++)
        printf("%.6lf\n", a[i][n + 1] / a[i][i]);
}
```

6.3 exgcd

```cpp
ll exgcd(ll a, ll b, ll &x, ll &y) {
    if (!b) {x = 1, y = 0; return a;}
    ll d = exgcd(b, a % b, x, y);
    swap(x, y); y -= a / b * x;
    return d;
}
ll inv(ll a, ll p) {
    ll iv, k;
    exgcd(a, p, iv, k);
    return (iv % p + p) % p;
}
```

6.4 CRT & EXCRT

```cpp
// x=ai (mod mi)   gcd(m1,m2....mn)=1
ll CRT(int n, ll a[], ll m[]) {
    ll M = 1, x = 0;
    for (int i = 1; i <= n; i++) M *= m[i];
    for (int i = 1; i <= n; i++)
        x = (x + (M / m[i]) * inv(M / m[i], m[i]) % M * a[i] % M) % M;
    return x;
}
// x=ai (mod mi)
ll EXCRT(int n, ll a[], ll m[]) {
    for (int i = 2; i <= n; i++) {
        ll d = gcd(m[i - 1], m[i]), x, y;
        if ((a[i] - a[i - 1]) % d != 0) return -1; // 无解
        exgcd(m[i - 1] / d, m[i] / d, x, y);
        m[i] = m[i] / gcd(m[i], m[i - 1]) * m[i - 1];
        a[i] = (a[i - 1] + mul(mul((a[i] - a[i - 1]) / d, x, m[i]), m[i - 1], m[i])) % m[i];
        a[i] = (a[i] + m[i]) % m[i];
    }
    return a[n];
}
```

7. Appendix

7.1 Formulas 公式表

7.1.1 Mobius Inversion

$$F(n)=\sum_{d|n}f(d)\Rightarrow f(n)=\sum_{d|n}\mu(d)F\left(\frac{n}{d}\right)$$

$$F(n)=\sum_{n|d}f(d)\Rightarrow f(n)=\sum_{n|d}\mu\left(\frac{d}{n}\right)F(d)\text{ 引理}$$

$$[x=1]=\sum_{d|x}\mu(d),\ \ x=\sum_{d|x}\mu(d)$$

7.1.2 降幂公式

$$a^k\equiv a^{k\bmod\varphi(p)+\varphi(p)},\ k\ge\varphi(p)$$

7.1.3 其他常用公式

$$\sum_{i=1}^{n}[(i,n)=1]\,i=n\frac{\varphi(n)+e(n)}{2}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{i}[(i,j)=d]=S_\varphi\left(\left\lfloor\frac{n}{d}\right\rfloor\right)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m}[(i,j)=d]=\sum_{d|k}\mu\left(\frac{k}{d}\right)\left\lfloor\frac{n}{k}\right\rfloor\left\lfloor\frac{m}{k}\right\rfloor$$

$$\sum_{i=1}^{n}f(i)\sum_{j=1}^{\lfloor\frac{n}{i}\rfloor}g(j)=\sum_{i=1}^{n}g(i)\sum_{j=1}^{\lfloor\frac{n}{i}\rfloor}f(j)$$

7.1.4 单位根反演

$$\sum_{i=0}^{\lfloor\frac{n}{k}\rfloor}C_n^{ik}$$

$$\frac{1}{k}\sum_{i=0}^{k-1}\omega_k^{in}=[k\mid n]$$

反演

$$Ans=\sum_{i=0}^{n}C_n^i[k\mid i]$$

$$=\sum_{i=0}^{n}C_n^i(\frac{1}{k}\sum_{j=0}^{k-1}\omega_k^{ij})$$

$$=\frac{1}{k}\sum_{i=0}^{n}C_n^i\sum_{j=0}^{k-1}\omega_k^{ij}$$

$$=\frac{1}{k}\sum_{j=0}^{k-1}(\sum_{i=0}^{n}C_n^i(\omega_k^j)^i)$$

$$=\frac{1}{k}\sum_{j=0}^{k-1}(1+\omega_k^j)^n$$

另, 如果要求的是 $[n\%k=t]$, 其实就是 $[k\mid(n-t)]$. 同理推式子即可.

7.1.5 Arithmetic Function

$$(p-1)!\equiv-1(mod\ p)$$

$$a>1,m,n>0,\text{ then }\gcd(a^m-1,a^n-1)=a^{\gcd(n,m)}-1$$

$$\mu^2(n)=\sum_{d^2|n}\mu(d)$$

$$a>b,\gcd(a,b)=1,\text{ then }\gcd(a^m-b^m,a^n-b^n)=a^{\gcd(m,n)}-b^{\gcd(m,n)}$$

$$\prod_{k=1,gcd(k,m)=1}^{m}k\equiv\begin{cases}-1 & \bmod\ m,m=4,p^q,2p^q\\ 1 & \bmod\ m,\text{otherwise}\end{cases}$$

$$\sigma_k(n)=\sum_{d|n}d^k=\prod_{i=1}^{\omega(n)}\frac{p_i^{(a_i+1)k}-1}{p_i^k-1}$$

$$J_k(n)=n^k\prod_{p|n}(1-\frac{1}{p^k})$$

$J_k(n)$ is the number of k-tuples of positive integers all less than or equal to n that form a coprime $(k+1)$-tuple together with n.

$$\sum_{\delta|n}J_k(\delta)=n^k$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}[gcd(i,j)=1]ij=\sum_{i=1}^{n}i^2\varphi(i)$$

$$\sum_{\delta|n}\delta^sJ_r(\delta)J_s(\frac{n}{\delta})=J_{r+s}(n)$$

$$\sum_{\delta|n}\varphi(\delta)d\left(\frac{n}{\delta}\right)=\sigma(n),\ \sum_{\delta|n}|\mu(\delta)|=2^{\omega(n)}$$

$$\sum_{\delta|n}2^{\omega(\delta)}=d(n^2),\ \sum_{\delta|n}d(\delta^2)=d^2(n)$$

$$\sum_{\delta|n}d\left(\frac{n}{\delta}\right)2^{\omega(\delta)}=d^2(n),\ \sum_{\delta|n}\frac{\mu(\delta)}{\delta}=\frac{\varphi(n)}{n}$$

$$\sum_{\delta|n}\frac{\mu(\delta)}{\varphi(\delta)}=d(n),\ \sum_{\delta|n}\frac{\mu^2(\delta)}{\varphi(\delta)}=\frac{n}{\varphi(n)}$$

$$n|\varphi(a^n-1)$$

$$\sum_{\substack{1\le k\le n\\ \gcd(k,n)=1}}f(\gcd(k-1,n))=\varphi(n)\sum_{d|n}\frac{(\mu*f)(d)}{\varphi(d)}$$

$$\varphi(\text{lcm}(m,n))\varphi(\gcd(m,n))=\varphi(m)\varphi(n)$$

$$\sum_{\delta|n}d^3(\delta)=(\sum_{\delta|n}d(\delta))^2$$

$$d(uv)=\sum_{\delta|\gcd(u,v)}\mu(\delta)d(\frac{u}{\delta})d(\frac{v}{\delta})$$

$$\sigma_k(u)\sigma_k(v)=\sum_{\delta|\gcd(u,v)}\delta^k\sigma_k(\frac{uv}{\delta^2})$$

$$\mu(n)=\sum_{k=1}^{n}[\gcd(k,n)=1]\cos 2\pi\frac{k}{n}$$

$$\varphi(n)=\sum_{k=1}^{n}[\gcd(k,n)=1]=\sum_{k=1}^{n}\gcd(k,n)\cos 2\pi\frac{k}{n}$$

$$\begin{cases} S(n)=\sum_{k=1}^{n}(f*g)(k) \\ \sum_{k=1}^{n}S(\lfloor\frac{n}{k}\rfloor)=\sum_{i=1}^{n}f(i)\sum_{j=1}^{\lfloor\frac{n}{i}\rfloor}(g*1)(j) \end{cases}$$

$$\begin{cases} S(n)=\sum_{k=1}^{n}(f\cdot g)(k), g \text{ completely multiplicative} \\ \sum_{k=1}^{n}S\left(\left\lfloor\frac{n}{k}\right\rfloor\right)g(k)=\sum_{k=1}^{n}(f*1)(k)g(k) \end{cases}$$

7.1.6 Binomial Coefficients

C	0	1	2	3	4	5	6	7	8	9	10
0	1										
1	1	1									
2	1	2	1								
3	1	3	3	1							
4	1	4	6	4	1						
5	1	5	10	10	5	1					
6	1	6	15	20	15	6	1				
7	1	7	21	35	35	21	7	1			
8	1	8	28	56	70	56	28	8	1		
9	1	9	36	84	126	126	84	36	9	1	
10	1	10	45	120	210	252	210	120	45	10	1

$$\binom{n}{k}\equiv[n\&k=k]\pmod 2$$

$$\sum_{k=0}^{n}\binom{k}{m}=\binom{n+1}{m+1}$$

$$\sqrt{1+z}=1+\sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{k\times 2^{2k-1}}\binom{2k-2}{k-1}z^k$$

$$\sum_{k=0}^{r}\binom{r-k}{m}\binom{s+k}{n}=\binom{r+s+1}{m+n+1}$$

$$C_{n,m}=\binom{n+m}{m}-\binom{n+m}{m-1}, n\ge m$$

$$\binom{n}{k}=(-1)^k\binom{k-n-1}{k}, \quad \sum_{k\le n}\binom{r+k}{k}=\binom{r+n+1}{n}$$

$$\binom{n_1+\cdots+n_p}{m}=\sum_{k_1+\cdots+k_p=m}\binom{n_1}{k_1}\cdots\binom{n_p}{k_p}$$

7.1.7 Fibonacci Numbers, Lucas Numbers

$$F(z)=\frac{z}{1-z-z^2}$$

$$\hat\phi=\frac{1-\sqrt5}{2}$$

$$\sum_{k=1}^{n}f_k=f_{n+2}-1, \quad \sum_{k=1}^{n}f_k^2=f_nf_{n+1}$$

$$\sum_{k=0}^{n}f_kf_{n-k}=\frac{1}{5}(n-1)f_n+\frac{2}{5}nf_{n-1}$$

$$\frac{f_{2n}}{f_n}=f_{n-1}+f_{n+1}$$

$$f_1+2f_2+3f_3+\cdots+nf_n=nf_{n+2}-f_{n+3}+2$$

$$\gcd(f_m,f_n)=f_{\gcd(m,n)}$$

$$f_n^2+(-1)^n=f_{n+1}f_{n-1}$$

$$f_{n+k}=f_nf_{k+1}+f_{n-1}f_k$$

$$f_{2n+1}=f_n^2+f_{n+1}^2$$

$$(-1)^kf_{n-k}=f_nf_{k-1}-f_{n-1}f_k$$

$$\text{Modulo } f_n, f_{mn+r}\equiv\begin{cases} f_r, & m \bmod 4=0; \\ (-1)^{r+1}f_{n-r}, & m \bmod 4=1; \\ (-1)^nf_r, & m \bmod 4=2; \\ (-1)^{r+1+n}f_{n-r}, & m \bmod 4=3. \end{cases}$$

```python
def fib(n): # 返回 F(n) 和 F(n + 1)
    if not n: return (0, 1)
    a, b = fib(n >> 1)
    c, d = a * (2 * b - a), a * a + b * b
    return (d, c + d) if n & 1 else (c, d)
```

$$L_0=2, L_1=1, L_n=L_{n-1}+L_{n-2}=(\frac{1+\sqrt5}{2})^n+(\frac{1-\sqrt5}{2})^n$$

$$L(x)=\frac{2-x}{1-x-x^2}$$

除了 $n=0,4,8,16$,L_n 是素数, 则 n 是素数.

$$\phi=\frac{1+\sqrt5}{2}, \quad \hat\phi=\frac{1-\sqrt5}{2}$$

$$F_n=\frac{\phi^n-\hat\phi^n}{\sqrt5}, \quad L_n=\phi^n+\hat\phi^n$$

$$\frac{L_n+F_n\sqrt5}{2}=\left(\frac{1+\sqrt5}{2}\right)^n$$

7.1.8 Sum of Powers

$$\sum_{i=1}^{n}i^2=\frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^{n}i^3=\left(\frac{n(n+1)}{2}\right)^2$$

$$\sum_{i=1}^{n}i^4=\frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$$

$$\sum_{i=1}^{n}i^5=\frac{n^2(n+1)^2(2n^2+2n-1)}{12}$$

7.1.9 Catalan Numbers 1, 1, 2, 5, 14, 42, 132, 429, 1430...

$$c_0=1, c_n=\sum_{i=0}^{n-1}c_ic_{n-1-i}=c_{n-1}\frac{4n-2}{n+1}=\frac{\binom{2n}{n}}{n+1}=\binom{2n}{n}-\binom{2n}{n-1}$$

$$c(x)=\frac{1-\sqrt{1-4x}}{2x}$$

Usage: n 对括号序列; n 个点满二叉树; $n\times n$ 的方格左下到右上不过对角线方案数; 凸 $n+2$ 边形三角形分割数; n 个数的出栈方案数; $2n$ 个顶点连接, 线段两两不交的方案数.

类卡特兰数　从 $(1,1)$ 出发走到 (n,m), 只能向右或者向上走, 不能越过 $y=x$ 这条线 (即保证 $x\ge y$), 合法方案数是 $C_{n+m-2}^{n}-C_{n+m-2}^{n-1}$.

7.1.10 Motzkin Numbers 1, 1, 2, 4, 9, 21, 51, 127, 323, 835...

圆上 n 点间画不相交弦的方案数. 选 n 个数 $k_1,k_2,...,k_n\in\{-1,0,1\}$, 保证 $\sum_i^a k_i(1\le a\le n)$ 非负且所有数总和为 0 的方案数.

$$M_{n+1}=M_n+\sum_{i}^{n-1}M_iM_{n-1-i}=\frac{(2n+3)M_n+3nM_{n-1}}{n+3}$$

$$M_n=\sum_{i=0}^{\lfloor\frac{n}{2}\rfloor}\binom{n}{2k}\mathrm{Catlan}(k)$$

$$M(X)=\frac{1-x-\sqrt{1-2x-3x^2}}{2x^2}$$

7.1.11 Derangement 错排数 0, 1, 2, 9, 44, 265, 1854, 14833...

$$D_1=0, D_2=1, D_n=n!(\frac{1}{0!}-\frac{1}{1!}+\frac{1}{2!}-\frac{1}{3!}+...+\frac{(-1)^n}{n!})$$

$$D_n=(n-1)(D_{n-1}+D_{n-2})$$

7.1.12 Bell Numbers 1, 1, 2, 5, 15, 52, 203, 877, 4140 ...

n 个元素集合划分的方案数.

$$B_n=\sum_{k=1}^{n}{n \brace k}, \quad B_{n+1}=\sum_{k=0}^{n}\binom{n}{k}B_k$$

$$B_{p^m+n}\equiv mB_n+B_{n+1}\pmod p$$

$$B(x)=\sum_{n=0}^{\infty}\frac{B_n}{n!}x^n=e^{e^x-1}$$

7.1.13 Stirling Numbers

第一类　n 个元素集合分作 k 个非空轮换方案数.

$$\begin{bmatrix}n+1\\k\end{bmatrix}=n\begin{bmatrix}n\\k\end{bmatrix}+\begin{bmatrix}n\\k-1\end{bmatrix}$$

$$s(n,k)=(-1)^{n-k}\begin{bmatrix}n\\k\end{bmatrix}$$

n\k	0	1	2	3	4	5	6
0	1						
1	0	1					
2	0	1	1				
3	0	2	3	1			
4	0	6	11	6	1		
5	0	24	50	35	10	1	
6	0	120	274	225	85	15	1
7	0	720	1764	1624	735	175	21

$$\begin{bmatrix}n+1\\2\end{bmatrix}=n!H_n \text{ (see 7.1.15)}$$

$$x^{\underline{n}}=\sum_k\begin{bmatrix}n\\k\end{bmatrix}(-1)^{n-k}x^k$$

$$x^{\overline{n}}=\sum_k\begin{bmatrix}n\\k\end{bmatrix}x^k$$

For fixed k, EGF:

$$\sum_{n=0}^{\infty}\begin{bmatrix}n\\k\end{bmatrix}\frac{x^n}{n!}=\frac{x^k}{k!}\left(\frac{\ln(1-x)}{x}\right)^k$$

第二类　把 n 个元素集合分作 k 个非空子集方案数.

$$\left\{ {n+1 \atop k} \right\} = k\left\{ {n \atop k} \right\} + \left\{ {n \atop k-1} \right\}$$

$$m!\left\{ {n \atop m} \right\} = \sum_k \binom{m}{k} k^n (-1)^{m-k}$$

n\k	0	1	2	3	4	5	6
0	1						
1	0	1					
2	0	1	1				
3	0	1	3	1			
4	0	1	7	6	1		
5	0	1	15	25	10	1	
6	0	1	31	90	65	15	1
7	0	1	63	301	350	140	21

$$x^n = \sum_k \left\{ {n \atop k} \right\} x^{\underline{k}} = \sum_k \left\{ {n \atop k} \right\} (-1)^{n-k} x^{\overline{k}}$$

For fixed k, EGF and OGF:

$$\sum_{n=0}^{\infty} \left\{ {n \atop k} \right\} \frac{x^n}{n!} = \frac{x^k}{k!} \left(\frac{e^x-1}{x} \right)^k$$

$$\sum_{n=0}^{\infty} \left\{ {n \atop k} \right\} x^n = x^k \prod_{i=1}^{k} (1-ix)^{-1}$$

7.1.14 Eulerian Numbers

n\k	0	1	2	3	4	5	6
1	1						
2	1	1					
3	1	4	1				
4	1	11	11	1			
5	1	26	66	26	1		
6	1	57	302	302	57	1	
7	1	120	1191	2416	1191	120	1

$$\left\langle {n \atop k} \right\rangle = (k+1)\left\langle {n-1 \atop k} \right\rangle + (n-k)\left\langle {n-1 \atop k-1} \right\rangle$$

$$x^n = \sum_k \left\langle {n \atop k} \right\rangle \binom{x+k}{n}$$

$$\left\langle {n \atop m} \right\rangle = \sum_{k=0}^{m} \binom{n+1}{k} (m+1-k)^n (-1)^k$$

7.1.15 Harmonic Numbers, 1, 3/2, 11/6, 25/12, 137/60 ...

$$H_n = \sum_{k=1}^{n} \frac{1}{k}, \sum_{k=1}^{n} H_k = (n+1)H_n - n$$

$$\sum_{k=1}^{n} kH_k = \frac{n(n+1)}{2} H_n - \frac{n(n-1)}{4}$$

$$\sum_{k=1}^{n} \binom{k}{m} H_k = \binom{n+1}{m+1} (H_{n+1} - \frac{1}{m+1})$$

7.1.16 卡迈克尔函数

卡迈克尔函数表示模 m 剩余系下最大的阶, 即 $\lambda(m) = \max_{a\perp m} \delta_m(a)$. 容易看出, 若 $\lambda(m) = \varphi(m)$, 则 m 存在原根. 该函数可由下述方法计算: 分解质因数 $m = p_1^{\alpha_1} p_2^{\alpha_2} ... p_t^{\alpha_t}$. 则 $\lambda(m) = \mathrm{lcm}(\lambda(p_1^{\alpha_1}), p_2^{\alpha_2}, ..., p_t^{\alpha_t})$. 其中对奇质数 p, $\lambda(p^\alpha) = (p-1)p^{\alpha-1}$. 对2的幂, $\lambda(2^k) = 2^{k-2}, s.t. k \geq 3$. $\lambda(4) = \lambda(2) = 2$.

7.1.17 五边形数定理求拆分数

$$\Phi(x) = \prod_{n=1}^{\infty} (1-x^n) = \sum_{n=-\infty}^{\infty} (-1)^k x^{k(3k-1)/2}$$

记 $p(n)$ 表示 n 的拆分数,$f(n,k)$ 表示将 n 拆分且每种数字使用次数必须小于 k 的拆分数. 则

$$P(x)\Phi(x) = 1, F(x^k)\Phi(x) = 1$$

暴力拆开卷积, 可以得到将1,-1,2,-2... 带入五边形数 $(-1)^k x^{k(3k-1)/2}$ 中, 由于小于n的五边形数只有 $\sqrt{n}$ 个, 可以 $O(n\sqrt{n})$ 计算答案:

$$p(n) = p(n-1) + p(n-2) - p(n-5) - p(n-7) + \cdots$$

$$f(n,k) = p(n) - p(n-k) - p(n-2k) + p(n-5k) + p(n-7k) - \cdots$$

7.1.18 Bernoulli Numbers 1, 1/2, 1/6, 0, -1/30, 0, 1/42 ...

$$B(x) = \sum_{i\geq 0} \frac{B_i x^i}{i!} = \frac{x}{e^x - 1}$$

$$B_n = [n=0] - \sum_{i=0}^{n-1} \binom{n}{i} \frac{B_i}{n-k+1}, \sum_{i=0}^{n} \binom{n+1}{i} B_i = 0$$

$$S_n(m) = \sum_{i=0}^{m-1} i^n = \sum_{i=0}^{n} \binom{n}{i} B_{n-i} \frac{m^{i+1}}{i+1}$$

$B_0 = 1,\ B_1 = -\frac{1}{2},\ B_4 = -\frac{1}{30},\ B_6 = \frac{1}{42},\ B_8 = -\frac{1}{30},\ \ldots$.

(除了 $B_1 = -\frac{1}{2}$ 以外, 伯努利数的奇数项都是 0.)

自然数幂次和关于次数的EGF:

$$F(x) = \sum_{k=0}^{\infty} \frac{\sum_{i=0}^{n} i^k}{k!} x^k = \sum_{i=0}^{n} e^{ix} = \frac{e^{(n+1)x-1}}{e^x - 1}$$

7.1.19 kMAX-MIN反演

$$k\mathrm{MAX}(S) = \sum_{T\subset S, T\neq\emptyset} (-1)^{|T|-k} C_{|T|-1}^{k-1} \mathrm{MIN}(T)$$

代入 $k=1$ 即为MAX-MIN反演:

$$\mathrm{MAX}(S) = \sum_{T\subset S, T\neq\emptyset} (-1)^{|T|-1} \mathrm{MIN}(T)$$

7.1.20 伍德伯里矩阵不等式

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

该等式可以动态维护矩阵的逆, 令 $C = [1], U, V$ 分别为 $1\times n$ 和 $n\times 1$ 的向量, 这样可以构造出 UCV 为只有某行或者某列不为0的矩阵, 一次修改复杂度为 $O(n^2)$.

7.1.21 Sum of Squares

$r_k(n)$ 表示用 k 个平方数组成 n 的方案数. 假设:

$$n = 2^{a_0} p_1^{2a_1} \cdots p_r^{2a_r} q_1^{b_1} \cdots q_s^{b_s}$$

其中 $p_i \equiv 3 \mod 4, q_i \equiv 1 \mod 4$, 那么

$$r_2(n) = \begin{cases} 0 & \text{if any } a_i \text{ is a half-integer} \\ 4\prod_1^r (b_i+1) & \text{if all } a_i \text{ are integers} \end{cases}$$

$r_3(n) > 0$ 当且仅当 n 不满足 $4^a(8b+7)$ 的形式 (a, b 为整数).

7.1.22 枚举勾股数 Pythagorean Triple

枚举 $x^2 + y^2 = z^2$ 的三元组: 可令 $x = m^2 - n^2, y = 2mn, z = m^2 + n^2$, 枚举 m 和 n 即可 $O(n)$ 枚举勾股数. 判断素勾股数方法: m, n 至少一个为偶数并且 m, n 互质, 那么 x, y, z 就是素勾股数.

7.1.23 四面体体积 Tetrahedron Volume

If U, V, W, u, v, w are lengths of edges of the tetrahedron (first three form a triangle; u opposite to U and so on)

$$V = \frac{\sqrt{4u^2v^2w^2 - \sum_{cyc} u^2(v^2+w^2-U^2)^2 + \prod_{cyc}(v^2+w^2-U^2)}}{12}$$

7.1.24 杨氏矩阵与钩子公式

满足: 格子 (i,j) 没有元素, 则它右边和上边相邻格子也没有元素; 格子 (i,j) 有元素 $a[i][j]$, 则它右边和上边相邻格子要么没有元素, 要么有元素且比 $a[i][j]$ 大.

计数: $F_1 = 1, F_2 = 2, F_n = F_{n-1} + (n-1)F_{n-2}, F(x) = e^{x+\frac{x^2}{2}}$

钩子公式: 对于给定形状 λ, 不同杨氏矩阵的个数为:

$$d_\lambda = \frac{n!}{\prod h_\lambda(i,j)}$$

$h_\lambda(i,j)$ 表示该格子右边和上边的格子数量加1.

7.1.25 常见博弈游戏

Nim-K游戏 n 堆石子轮流拿, 每次最多可以拿k堆石子, 谁走最后一步输。结论: 把每一堆石子的sg值 (即石子数量) 二进制分解, 先手必败当且仅当每一位二进制位上1的个数是 (k+1) 的倍数.

Anti-Nim游戏 n 堆石子轮流拿, 谁走最后一步输。结论: 先手胜当且仅当1. 所有堆石子数都为1且游戏的SG值为0 (即有偶数个孤单堆-每堆只有1个石子数) 2. 存在某堆石子数大于1且游戏的SG值不为0.

斐波那契博弈 有一堆物品, 两人轮流取物品, 先手最少取一个, 至多无上限, 但不能把物品取完, 之后每次取的物品数不能超过上一次取的物品数的二倍且至少为一件, 取走最后一件物品的人获胜. 结论: 先手胜当且仅当物品数 n 不是斐波那契数.

威佐夫博弈 有两堆石子, 博弈双方每次可以取一堆石子中的任意个, 不能不取, 或者取两堆石子中的相同个. 先取完者赢. 结论: 求出两堆石子 A 和 B 的差值 C, 如果 $\left\lfloor C * \frac{\sqrt{5}+1}{2} \right\rfloor = min(A,B)$ 那么后手赢, 否则先手赢.

约瑟夫环 n 个人 $0, \ldots, n-1$, 令 $f_{i,m}$ 为 i 个人报 m 的胜利者, 则 $f_{1,m} = 0, f_{i,m} = (f_{i-1,m} + m) \bmod i$.

阶梯Nim 在一个阶梯上, 每次选一个台阶上任意个式子移到下一个台阶上, 不可移动者输. 结论: SG值等于奇数层台阶上石子数的异或和. 对于树形结构也适用, 奇数层节点上所有石子数异或起来即可.

图上博弈 给定无向图, 先手从某点开始走, 只能走相邻且未走过的点, 无法移动者输. 对该图求最大匹配, 若某个点不一定在最大匹配中则先手必败, 否则先手必胜.

最大最小定理求纳什均衡点 在二人零和博弈中, 可以用以下方式求出一个纳什均衡点: 在博弈双方中任选一方, 求混合策略 $\mathbf{p}$ 使得对方选择任意一个纯策略时, 己方的最小收益最大 (等价于对方的最大收益最小). 据此可以求出双方在此局面下的最优期望得分, 分别等于己方最大的最小收益和对方最小的最大收益. 一般而言, 可以得到形如

$$\max_{\mathbf{p}} \min_i \sum_{p_j\in\mathbf{p}} p_j w_{i,j}, \text{s.t. } p_j \geq 0, \sum p_j = 1$$

的形式. 当 $\sum p_j w_{i,j}$ 可以表示成只与 i 有关的函数 $f(i)$ 时, 可以令初始时 $p_i = 0$, 不断调整 $\sum p_j w_{i,j}$ 最小的那个i的概率 p_i, 直至无法调整或者 $\sum p_j = 1$ 为止.

7.1.26 概率相关

$D(X) = E(X-E(X))^2 = E(X^2) - (E(X))^2, D(X+Y) = D(X) + D(Y) D(aX) = a^2 D(X)$

$$E[x] = \sum_{i=1}^{\infty} P(X \geq i)$$

m 个数的方差：

$$s^2 = \frac{\sum_{i=1}^{m} x_i^2}{m} - \overline{x}^2$$

7.1.27 邻接矩阵行列式的意义

在无向图中取若干个环，一种取法权值就是边权的乘积，对行列式的贡献是 $(-1)^{\text{even}}$，其中 even 是偶环的个数.

7.1.28 Others (某些近似数值公式在这里)

$$S_j=\sum_{k=1}^{n}x_k^j$$

$$h_m=\sum_{1\le j_1<\cdots<j_m\le n}x_{j_1}\cdots x_{j_m},\quad H_m=\sum_{1\le j_1\le\cdots\le j_m\le n}x_{j_1}\cdots x_{j_m}$$

$$h_n=\frac{1}{n}\sum_{k=1}^{n}(-1)^{k+1}S_kh_{n-k}$$

$$H_n=\frac{1}{n}\sum_{k=1}^{n}S_kH_{n-k}$$

$$\sum_{k=0}^{n}kc^k=\frac{nc^{n+2}-(n+1)c^{n+1}+c}{(c-1)^2}$$

$$\sum_{i=1}^{n}\frac{1}{n}\approx\ln\left(n+\frac{1}{2}\right)+\frac{1}{24(n+0.5)^2}+\Gamma,(\Gamma\approx 0.577215664901532860606 5)$$

$$n!=\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1+\frac{1}{12n}+\frac{1}{288n^2}+O\left(\frac{1}{n^3}\right)\right)$$

$$\max\{x_a-x_b,y_a-y_b,z_a-z_b\}-\min\{x_a-x_b,y_a-y_b,z_a-z_b\}$$
$$=\frac{1}{2}\sum_{cyc}|(x_a-y_a)-(x_b-y_b)|$$

$$(a+b)(b+c)(c+a)=\frac{(a+b+c)^3-a^3-b^3-c^3}{3}$$

$$a^3+b^3=(a+b)(a^2-ab+b^2),a^3-b^3=(a-b)(a^2+ab+b^2)$$

$n \bmod 2=1$:

$$a^n+b^n=(a+b)(a^{n-1}-a^{n-2}b+a^{n-3}b^2-\ldots-ab^{n-2}+b^{n-1})$$

划分问题: n 个 $k-1$ 维向量最多把 k 维空间分为 $\sum_{i=0}^{k}C_n^i$ 份.

7.2 Calculus Table 导数表

$\left(\frac{u}{v}\right)'=\frac{u'v-uv'}{v^2}$
$(a^x)'=(\ln a)a^x$
$(\tan x)'=\sec^2 x$
$(\cot x)'=\csc^2 x$
$(\sec x)'=\tan x\sec x$
$(\csc x)'=-\cot x\csc x$
$(\arcsin x)'=\frac{1}{\sqrt{1-x^2}}$
$(\arccos x)'=-\frac{1}{\sqrt{1-x^2}}$
$(\arctan x)'=\frac{1}{1+x^2}$
$(\operatorname{arccot} x)'=-\frac{1}{1+x^2}$
$(\operatorname{arccsc} x)'=-\frac{1}{x\sqrt{1-x^2}}$
$(\operatorname{arcsec} x)'=\frac{1}{x\sqrt{1-x^2}}$
$(\tanh x)'=\operatorname{sech}^2 x$
$(\coth x)'=-\operatorname{csch}^2 x$
$(\operatorname{sech} x)'=-\operatorname{sech} x\tanh x$
$(\operatorname{csch} x)'=-\operatorname{csch} x\coth x$
$(\operatorname{arcsinh} x)'=\frac{1}{\sqrt{1+x^2}}$
$(\operatorname{arccosh} x)'=\frac{1}{\sqrt{x^2-1}}$
$(\operatorname{arctanh} x)'=\frac{1}{1-x^2}$
$(\operatorname{arccoth} x)'=\frac{1}{x^2-1}$
$(\operatorname{arccsch} x)'=-\frac{1}{|x|\sqrt{1+x^2}}$
$(\operatorname{arcsech} x)'=-\frac{1}{x\sqrt{1-x^2}}$

7.3 Integration Table 积分表

$ax^2+bx+c(a>0)$

1. $\int\frac{dx}{ax^2+bx+c}=\begin{cases}\frac{2}{\sqrt{4ac-b^2}}\arctan\frac{2ax+b}{\sqrt{4ac-b^2}}+C & (b^2<4ac)\\ \frac{1}{\sqrt{b^2-4ac}}\ln\left|\frac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}}\right|+C & (b^2>4ac)\end{cases}$
2. $\int\frac{x}{ax^2+bx+c}dx=\frac{1}{2a}\ln|ax^2+bx+c|-\frac{b}{2a}\int\frac{dx}{ax^2+bx+c}$

$\sqrt{\pm ax^2+bx+c}(a>0)$

1. $\int\frac{dx}{\sqrt{ax^2+bx+c}}=\frac{1}{\sqrt{a}}\ln|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}|+C$
2. $\int\sqrt{ax^2+bx+c}dx=\frac{2ax+b}{4a}\sqrt{ax^2+bx+c}+\frac{4ac-b^2}{8\sqrt{a^3}}\ln|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}|+C$
3. $\int\frac{x}{\sqrt{ax^2+bx+c}}dx=\frac{1}{a}\sqrt{ax^2+bx+c}-\frac{b}{2\sqrt{a^3}}\ln|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}|+C$
4. $\int\frac{dx}{\sqrt{c+bx-ax^2}}=-\frac{1}{\sqrt{a}}\arcsin\frac{2ax-b}{\sqrt{b^2+4ac}}+C$
5. $\int\sqrt{c+bx-ax^2}dx=\frac{2ax-b}{4a}\sqrt{c+bx-ax^2}+\frac{b^2+4ac}{8\sqrt{a^3}}\arcsin\frac{2ax-b}{\sqrt{b^2+4ac}}+C$
6. $\int\frac{x}{\sqrt{c+bx-ax^2}}dx=-\frac{1}{a}\sqrt{c+bx-ax^2}+\frac{b}{2\sqrt{a^3}}\arcsin\frac{2ax-b}{\sqrt{b^2+4ac}}+C$

$\sqrt{\pm\frac{x-a}{x-b}}$ 或 $\sqrt{(x-a)(x-b)}$

1. $\int\frac{dx}{\sqrt{(x-a)(b-x)}}=2\arcsin\sqrt{\frac{x-a}{b-x}}+C\ (a<b)$
2. $\int\sqrt{(x-a)(b-x)}dx=\frac{2x-a-b}{4}\sqrt{(x-a)(b-x)}+\frac{(b-a)^2}{4}\arcsin\sqrt{\frac{x-a}{b-x}}+C,(a<b)$

三角函数的积分

1. $\int\tan x dx=-\ln|\cos x|+C$
2. $\int\cot x dx=\ln|\sin x|+C$
3. $\int\sec x dx=\ln\left|\tan\left(\frac{\pi}{4}+\frac{x}{2}\right)\right|+C=\ln|\sec x+\tan x|+C$
4. $\int\csc x dx=\ln\left|\tan\frac{x}{2}\right|+C=\ln|\csc x-\cot x|+C$
5. $\int\sec^2 x dx=\tan x+C$
6. $\int\csc^2 x dx=-\cot x+C$
7. $\int\sec x\tan x dx=\sec x+C$
8. $\int\csc x\cot x dx=-\csc x+C$
9. $\int\sin^2 x dx=\frac{x}{2}-\frac{1}{4}\sin 2x+C$
10. $\int\cos^2 x dx=\frac{x}{2}+\frac{1}{4}\sin 2x+C$
11. $\int\sin^n x dx=-\frac{1}{n}\sin^{n-1}x\cos x+\frac{n-1}{n}\int\sin^{n-2}x dx$
12. $\int\cos^n x dx=\frac{1}{n}\cos^{n-1}x\sin x+\frac{n-1}{n}\int\cos^{n-2}x dx$
13. $\int\frac{dx}{\sin^n x}=-\frac{1}{n-1}\frac{\cos x}{\sin^{n-1}x}+\frac{n-2}{n-1}\int\frac{dx}{\sin^{n-2}x}$
14. $\int\frac{dx}{\cos^n x}=\frac{1}{n-1}\frac{\sin x}{\cos^{n-1}x}+\frac{n-2}{n-1}\int\frac{dx}{\cos^{n-2}x}$
15. $$\int\cos^m x\sin^n x dx$$
$$=\frac{1}{m+n}\cos^{m-1}x\sin^{n+1}x+\frac{m-1}{m+n}\int\cos^{m-2}x\sin^n x dx$$
$$=-\frac{1}{m+n}\cos^{m+1}x\sin^{n-1}x+\frac{n-1}{m+1}\int\cos^m x\sin^{n-2}x dx$$
16. $\int\frac{dx}{a+b\sin x}=\begin{cases}\frac{2}{\sqrt{a^2-b^2}}\arctan\frac{a\tan\frac{x}{2}+b}{\sqrt{a^2-b^2}}+C & (a^2>b^2)\\ \frac{1}{\sqrt{b^2-a^2}}\ln\left|\frac{a\tan\frac{x}{2}+b-\sqrt{b^2-a^2}}{a\tan\frac{x}{2}+b+\sqrt{b^2-a^2}}\right|+C & (a^2<b^2)\end{cases}$
17. $\int\frac{dx}{a+b\cos x}=\begin{cases}\frac{2}{a+b}\sqrt{\frac{a+b}{a-b}}\arctan\left(\sqrt{\frac{a-b}{a+b}}\tan\frac{x}{2}\right)+C & (a^2>b^2)\\ \frac{1}{a+b}\sqrt{\frac{a+b}{a-b}}\ln\left|\frac{\tan\frac{x}{2}+\sqrt{\frac{a+b}{b-a}}}{\tan\frac{x}{2}-\sqrt{\frac{a+b}{b-a}}}\right|+C & (a^2<b^2)\end{cases}$
18. $\int\frac{dx}{a^2\cos^2 x+b^2\sin^2 x}=\frac{1}{ab}\arctan\left(\frac{b}{a}\tan x\right)+C$
19. $\int\frac{dx}{a^2\cos^2 x-b^2\sin^2 x}=\frac{1}{2ab}\ln\left|\frac{b\tan x+a}{b\tan x-a}\right|+C$
20. $\int x\sin ax dx=\frac{1}{a^2}\sin ax-\frac{1}{a}x\cos ax+C$
21. $\int x^2\sin ax dx=-\frac{1}{a}x^2\cos ax+\frac{2}{a^2}x\sin ax+\frac{2}{a^3}\cos ax+C$
22. $\int x\cos ax dx=\frac{1}{a^2}\cos ax+\frac{1}{a}x\sin ax+C$
23. $\int x^2\cos ax dx=\frac{1}{a}x^2\sin ax+\frac{2}{a^2}x\cos ax-\frac{2}{a^3}\sin ax+C$

反三角函数的积分 (其中 $a>0$)

1. $\int\arcsin\frac{x}{a}dx=x\arcsin\frac{x}{a}+\sqrt{a^2-x^2}+C$
2. $\int x\arcsin\frac{x}{a}dx=\left(\frac{x^2}{2}-\frac{a^2}{4}\right)\arcsin\frac{x}{a}+\frac{x}{4}\sqrt{x^2-x^2}+C$
3. $\int x^2\arcsin\frac{x}{a}dx=\frac{x^3}{3}\arcsin\frac{x}{a}+\frac{1}{9}(x^2+2a^2)\sqrt{a^2-x^2}+C$
4. $\int\arccos\frac{x}{a}dx=x\ arccos\frac{x}{a}-\sqrt{a^2-x^2}+C$
5. $\int x\arccos\frac{x}{a}dx=\left(\frac{x^2}{2}-\frac{a^2}{4}\right)\arccos\frac{x}{a}-\frac{x}{4}\sqrt{a^2-x^2}+C$
6. $\int x^2\arccos\frac{x}{a}dx=\frac{x^3}{3}\arccos\frac{x}{a}-\frac{1}{9}(x^2+2a^2)\sqrt{a^2-x^2}+C$
7. $\int\arctan\frac{x}{a}dx=x\arctan\frac{x}{a}-\frac{a}{2}\ln(a^2+x^2)+C$
8. $\int x\arctan\frac{x}{a}dx=\frac{1}{2}(a^2+x^2)\arctan\frac{x}{a}-\frac{a}{2}x+C$
9. $\int x^2\arctan\frac{x}{a}dx=\frac{x^3}{3}\arctan\frac{x}{a}-\frac{a}{6}x^2+\frac{a^3}{6}\ln(a^2+x^2)+C$

指数函数的积分

1. $\int a^x dx=\frac{1}{\ln a}a^x+C$
2. $\int e^{ax}dx=\frac{1}{a}a^{ax}+C$
3. $\int xe^{ax}dx=\frac{1}{a^2}(ax-1)a^{ax}+C$
4. $\int x^n e^{ax}dx=\frac{1}{a}x^n e^{ax}-\frac{n}{a}\int x^{n-1}e^{ax}dx$
5. $\int xa^x dx=\frac{x}{\ln a}a^x-\frac{1}{(\ln a)^2}a^x+C$
6. $\int x^n a^x dx=\frac{1}{\ln a}x^n a^x-\frac{n}{\ln a}\int x^{n-1}a^x dx$
7. $\int e^{ax}\sin bx dx=\frac{1}{a^2+b^2}e^{ax}(a\sin bx-b\cos bx)+C$
8. $\int e^{ax}\cos bx dx=\frac{1}{a^2+b^2}e^{ax}(b\sin bx+a\cos bx)+C$
9. $\int e^{ax}\sin^n bx dx=\frac{1}{a^2+b^2n^2}e^{ax}\sin^{n-1}bx(a\sin bx-nb\cos bx)+\frac{n(n-1)b^2}{a^2+b^2n^2}\int e^{ax}\sin^{n-2}bx dx$
10. $\int e^{ax}\cos^n bx dx=\frac{1}{a^2+b^2n^2}e^{ax}\cos^{n-1}bx(a\cos bx+nb\sin bx)+\frac{n(n-1)b^2}{a^2+b^2n^2}\int e^{ax}\cos^{n-2}bx dx$

对数函数的积分

1. $\int \ln x \mathrm{d}x = x \ln x - x + C$
2. $\int \frac{\mathrm{d}x}{x \ln x} = \ln|\ln x| + C$
3. $\int x^n \ln x \mathrm{d}x = \frac{1}{n+1} x^{n+1} (\ln x - \frac{1}{n+1}) + C$
4. $\int (\ln x)^n \mathrm{d}x = x(\ln x)^n - n \int (\ln x)^{n-1} \mathrm{d}x$
5. $\int x^m (\ln x)^n \mathrm{d}x = \frac{1}{m+1} x^{m+1} (\ln x)^n - \frac{n}{m+1} \int x^m (\ln x)^{n-1} \mathrm{d}x$

8. Miscellany

8.1 Zeller 日期公式

```cpp
// weekday=(id+1)%7;{Sun=0,Mon=1,...}   getId(1, 1, 1) = 0
int getId(int y, int m, int d) {
 | if (m < 3) { y --; m += 12; }
 | return 365 * y + y / 4 - y / 100 + y / 400 + (153 * (m -
   ↪ 3) + 2) / 5 + d - 307; }
//  y<0: 统一加400的倍数年
auto date(int id) {
 | int x=id+1789995, n, i, j, y, m, d;
 | 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;
 | return make_tuple(y, m, d); }
```

8.2 基数排序

```cpp
const int SZ = 1 << 8; // almost always fit in L1 cache
void SORT(int a[], int c[], int n, int w) {
 | for(int i=0; i<SZ; i++) b[i] = 0;
 | for(int i=1; i<=n; i++) b[(a[i]>>w) & (SZ-1)]++;
 | for(int i=1; i<SZ; i++) b[i] += b[i - 1];
 | for(int i=n; i; i--) c[b[(a[i]>>w) & (SZ-1)]--] = a[i];}
void Sort(int *a, int n){
 | SORT(a, c, n, 0); SORT(c, a, n, 8);
 | SORT(a, c, n, 16); SORT(c, a, n, 24); }
```

8.3 O3 与读入优化

```cpp
//fast = O3 + ffast-math + fallow-store-data-races
#pragma GCC optimize("Ofast")
#pragma GCC target("lzcnt,popcnt")
const int SZ = 1 << 16;
int getc() {
 | static char buf[SZ], *ptr = buf, *top = buf;
 | if (ptr == top) {
 |  | ptr = buf, top = buf + fread(buf, 1, SZ, stdin);
 |  | if (top == buf) return -1; }
 | return *ptr++; }
bitset._Find_first();bitset._Find_next(idx);
struct HashFunc{size_t operator()(const KEY &key)const{}};
```

8.4 试机赛与纪律文件

- 检查所需身份证件：护照、学生证、胸牌以及现场所需通行证。
- 确认什么东西能带进场，什么不能。特别注意：智能手表、金属（钥匙）等等。
- 测试鼠标、键盘、显示器和座椅。如果有问题，立刻联系工作人员。
- 确认比赛前能动什么，不能动什么，能存储什么配置文件。
- 测试本地栈大小：如果 `ulimit -a` 是 unlimited，那么在 bashrc 里加上 `ulimit -s 65536; ulimit -m 1048576`，否则死递归会死机。
- 测试比赛提交方式。如果有 `submit` 命令，确认如何使用。讨论是否应该不用以避免 `submit > a.cpp`。
- 如果可以的话，设置定期备份文件。
- 测试 OJ 栈大小。如果不合常理，发 clar 问一下。
- 测试提交编译器版本。如 C++17 `auto [x, y]`: a; C++14 `[](auto x, auto y)`; C++11 auto; bits/stdc++.h; pb_ds。

```cpp
#include <ext/rope>
using namespace __gnu_cxx;
rope <int> R;
R.insert(y, x);
R[x];
R.erase(x, 1);
```

- 测试 `__int128, __float128, sizeof (long double)`
- 测试代码长度限制；测试 output limit；测试 stderr limit。
- 测试内存限制：MLE 还是报 RE？栈溢出呢？
- 测试浮点数性能：FFT 能跑多快？测试内存性能：线段树、树状数组、素数筛能跑多快？测试 CPU 性能：阶乘、快速幂能跑多快？记得开 O2。
- 测试 clar：如果问不同类型的愚蠢的问题，得到的回复是否不一样？
- 测试 clock() 是否能够正常工作；测试本地性能与提交性能。
- 测试本地是否有 fsan, gdb。

```
address, undefined, return, shift, integer-divide-by-zero,
bounds-strict, float-cast-overflow, builtin
```

- 测试 Python, Java 本地环境与提交环境。Python 快吗？$A \times B$ 能跑多快？输入输出呢？
- 测试 `time` 命令是否能显示内存占用。

```
/usr/bin/time -v ./a.out
```

8.5 Constant Table 常数表

```
Random primes generated at Sat Oct 7 09:25:28 2023
1e3 953 967 983 991 997 1009 1013 1019 1021 1031 1039 1063 1069
3e4 28607 29147 29587 29683 30091 30119 30307 30817 30859 31981
1e5 94121 97931 99089 99839 102523 102769 103183 105107 105341
5e5 474073 480533 488791 491591 505313 514289 514621 532199
1e6 951343 971713 1017391 1031717 1043593 1055731 1058179 1058983
2e6 1901891 2001347 2023783 2057441 2067001 2067337 2070137
1e7 9817861 9861253 9928907 10284497 10522349 10549681 10665727
2e7 19387169 19982899 20182259 20672803 21097091 21204241
1e9 951676081 981422737 1047728293 1056646771 1068183911
```

n	$\log_{10} n$	$n!$	$C(n, n/2)$	$\mathrm{LCM}(1 \ldots n)$
2	0.30102999	2	2	2
3	0.47712125	6	3	6
4	0.60205999	24	6	12
5	0.69897000	120	10	60
6	0.77815125	720	20	60
7	0.84509804	5040	35	420
8	0.90308998	40320	70	840
9	0.95424251	362880	126	2520
10	1	3628800	252	2520
11	1.04139269	39916800	462	27720
12	1.07918125	479001600	924	27720
15	1.17609126	1.31e12	6435	360360
20	1.30103000	2.43e18	184756	232792560
25	1.39794001	1.55e25	5200300	26771144400
30	1.47712125	2.65e32	155117520	1.444e14

$n \le$	10	100	1e3	1e4	1e5	1e6
$\max\{\omega(n)\}$	2	3	4	5	6	7
$\max\{d(n)\}$	4	12	32	64	128	240
$\pi(n)$	4	25	168	1229	9592	78498
$n \le$	1e7	1e8	1e9	1e10	1e11	1e12
$\max\{\omega(n)\}$	8	8	9	10	10	11
$\max\{d(n)\}$	448	768	1344	2304	4032	6720
$\pi(n)$	664579	5761455	5.08e7	4.55e8	4.12e9	3.7e10
$n \le$	1e13	1e14	1e15	1e16	1e17	1e18
$\max\{\omega(n)\}$	12	12	13	13	14	15
$\max\{d(n)\}$	10752	17280	26880	41472	64512	103680
$\pi(n)$	Prime number theorem: $\pi(x) \sim x/\log(x)$					

Vimrc, Bashrc

```vim
source $VIMRUNTIME/mswin.vim
behave mswin
set mouse=a ci ai si nu ts=4 sw=4 is hls backup undofile
color slate
map <F7> : ! make %<<CR>
map <F8> : ! time ./%< <CR>
```

```bash
export CXXFLAGS='-g -Wall -Wextra -Wconversion -Wshadow
  ↪ -std=c++17'
```

Good Luck && Have Fun!