Templates for ACMICPC 2014 Taichung Regional

ACMICPC 2014 台中区域赛 **参考模板**

円環の理

Shanghai Jiao Tong University : **MahoushojoMiracle**

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Coach	**教　练**
Yong YU	俞　勇

Team Members	**队　员**
Haobin NI	倪昊斌
Siqi CHEN	陈思奇
Yuyang HUANG	黄宇扬

目录

二维几何基础

```
struct Point {
    double x, y;
    Point rotate(const double ang) { //逆时针旋转 ang 弧度
        return Point(cos(ang) * x - sin(ang) * y, cos(ang) * y + sin(ang) * x);
    }
    Point turn90() { //逆时针旋转 90 度
        return Point(-y, x);
    }
};
Point intersect(const Line &l0, const Line &l1) {
    double s1 = det(l0.b - l0.a, l1.a - l0.a),
           s2 = det(l0.b - l0.a, l1.b - l0.a);
    return (l1.a * s2 - l1.b * s1) / (s2 - s1);
}
bool onSeg(const Line &l, const Point &p) { //点在线段上
    return sign(det(p - l.a, l.b - l.a)) == 0 && sign(dot(p - l.a, p - l.b)) <=
0;
}
Point projection(const Line &l, const Point &p) { //点到直线投影
    return l.a + (l.b - l.a) * (dot(p - l.a, l.b - l.a) / (l.b - l.a).len2());
}
double disToLine(const Line &l, const Point &p) {
    return abs(det(p - l.a, l.b - l.a) / (l.b - l.a).len());
}
double disToSeg(const Line &l, const Point &p) { //点到线段距离
    return sign(dot(p - l.a, l.b - l.a)) * sign(dot(p - l.b, l.a - l.b)) != 1 ?
disToLine(l, p) : min((p - l.a).len(), (p - l.b).len());
}
Point symmetryPoint(const Point a, const Point b) { //点 b 关于点 a 的中心对称点
    return a + a - b;
}
Point reflection(const Line &l, const Point &p) { //点关于直线的对称点
    return symmetryPoint(projection(l, p), p);
}
```

点在多边形内

```
//注意点在多边形上时的情况
bool contain(const vector<Point> &p, const Point &q) {
    int cnt = 0;
    for (int i = 0; i < (int)p.size(); ++i) {
        Point a = p[i], b = p[(i + 1) % p.size()];
        if (onSeg(Line(a, b), q)) {
            return true;
        }
        if (sign(a.y - b.y) <= 0) {
            swap(a, b);
        }
        if (sign(q.y - a.y) > 0) {
            continue;
        }
        if (sign(q.y - b.y) <= 0) {
            continue;
        }
        cnt += sign(det(b - q, a - q)) > 0;
    }
    return cnt & 1;
}
```

半平面交（带排序）

```
struct Point {
    Point norm() {
        double l = len();
        return Point(x / l, y / l);
    }
    int quad() const {
        return sign(y) == 1 || sign(y) == 0 && sign(x) >= 0;
    }
};
struct Line {
    bool include(const Point &p) const {
        return sign(det(b - a, p - a)) > 0;
    }
    Line push() { //将半平面向外推 eps
        const double eps = 1e-6;
        Point delta = (b - a).turn90().norm() * eps;
        return Line(a - delta, b - delta);
    }
};
bool sameDir(const Line &l0, const Line &l1) {
    return parallel(l0, l1) && sign(dot(l0.b - l0.a, l1.b - l1.a)) == 1;
}
bool operator < (const Point &a, const Point &b) {
    if (a.quad() != b.quad()) {
        return a.quad() < b.quad();
    } else {
        return sign(det(a, b)) > 0;
    }
}
bool operator < (const Line &l0, const Line &l1) {
    if (sameDir(l0, l1)) {
        return l1.include(l0.a);
    } else {
        return (l0.b - l0.a) < (l1.b - l1.a);
    }
}

bool check(const Line &u, const Line &v, const Line &w) {
    return w.include(intersect(u, v));
}
vector<Point> intersection(vector<Line> &l) {
    sort(l.begin(), l.end());
    deque<Line> q;
    for (int i = 0; i < (int)l.size(); ++i) {
        if (i && sameDir(l[i], l[i - 1])) {
            continue;
        }
        while (q.size() > 1 && !check(q[q.size() - 2], q[q.size() - 1], l[i])) {
            q.pop_back();
        }
        while (q.size() > 1 && !check(q[1], q[0], l[i])) {
            q.pop_front();
        }
        q.push_back(l[i]);
    }
    while (q.size() > 2 && !check(q[q.size() - 2], q[q.size() - 1], q[0])) {
        q.pop_back();
    }
    while (q.size() > 2 && !check(q[1], q[0], q[q.size() - 1])) {
        q.pop_front();
    }
    vector<Point> ret;
    for (int i = 0; i < (int)q.size(); ++i) {
        ret.push_back(intersect(q[i], q[(i + 1) % q.size()]));
    }
    return ret;
}
```

三角形内外心

```
Point inCenter(const Point &A, const Point &B, const Point &C) {
    double  a = (B - C).len(), b = (C - A).len(), c = dis(A - B).len(),
            p = (a + b + c) / 2,
            s = sqrt(p * (p - a) * (p - b) * (p - c)),
            r = s / p;
    return (A * a + B * b + C * c) / (a + b + c);
}
Point exCenter(const Point &a, const Point &b, const Point &c) {
    double  a1 = b.x - a.x, b1 = b.y - a.y, c1 = (sqr(a1) + sqr(b1)) / 2,
            a2 = c.x - a.x, b2 = c.y - a.y, c2 = (sqr(a2) + sqr(b2)) / 2,
            d = a1 * b2 - a2 * b1;
    return a + Point((c1 * b2 - c2 * b1), (a1 * c2 - a2 * c1)) / d;
}
```

圆的面积模板

```
const double PI = acos(-1);
struct Circle {
    Point o;
    double r;
    Circle (Point o = Point(0, 0), double r = 0) : o(o), r(r) {}
};
struct Event {
    Point p;
    double ang;
    int delta;
    Event (Point p = Point(0, 0), double ang = 0, double delta = 0) : p(p), ang(ang),
delta(delta) {}
};
bool operator < (const Event &a, const Event &b) {
    return a.ang < b.ang;
}
void addEvent(const Circle &a, const Circle &b, vector<Event> &evt, int &cnt) {
    double d2 = (a.o - b.o).len2(),
           dRatio = ((a.r - b.r) * (a.r + b.r) / d2 + 1) / 2,
           pRatio = sqrt(-(d2 - sqr(a.r - b.r)) * (d2 - sqr(a.r + b.r)) / (d2 * d2
* 4));
    Point d = b.o - a.o, p = d.rotate(PI / 2),
          q0 = a.o + d * dRatio + p * pRatio,
          q1 = a.o + d * dRatio - p * pRatio;
    double ang0 = (q0 - a.o).ang(),
           ang1 = (q1 - a.o).ang();
    evt.push_back(Event(q1, ang1, 1));
    evt.push_back(Event(q0, ang0, -1));
    cnt += ang1 > ang0;
}
bool issame(const Circle &a, const Circle &b) {
    return sign((a.o - b.o).len()) == 0 && sign(a.r - b.r) == 0;
}
bool overlap(const Circle &a, const Circle &b) {
    return sign(a.r - b.r - (a.o - b.o).len()) >= 0;
}
bool intersect(const Circle &a, const Circle &b) {
    return sign((a.o - b.o).len() - a.r - b.r) < 0;
}
int C;
Circle c[N];
double area[N];
void solve() {
    memset(area, 0, sizeof(double) * (C + 1));
    for (int i = 0; i < C; ++i) {
        int cnt = 1;
        vector<Event> evt;
        for (int j = 0; j < i; ++j) {
            if (issame(c[i], c[j])) {
                ++cnt;
            }
        }
        for (int j = 0; j < C; ++j) {
            if (j != i && !issame(c[i], c[j]) && overlap(c[j], c[i])) {
                ++cnt;
            }
        }
        for (int j = 0; j < C; ++j) {
            if (j != i && !overlap(c[j], c[i]) && !overlap(c[i], c[j]) &&
intersect(c[i], c[j])) {
                addEvent(c[i], c[j], evt, cnt);
            }
        }
        if (evt.size() == 0) {
            area[cnt] += PI * c[i].r * c[i].r;
        } else {
            sort(evt.begin(), evt.end());
            evt.push_back(evt.front());
            for (int j = 0; j + 1 < (int)evt.size(); ++j) {
                cnt += evt[j].delta;
                area[cnt] += det(evt[j].p, evt[j + 1].p) / 2;
                double ang = evt[j + 1].ang - evt[j].ang;
                if (ang < 0) {
                    ang += PI * 2;
                }
                area[cnt] += ang * c[i].r * c[i].r / 2 - sin(ang) * c[i].r * c[i].r / 2;
            }
        }
    }
}
```

圆和多边形面积交

```
double circleCrossSegment(Point pa, Point pb, double r) {
    if (pa.len() < pb.len()) {
        swap(pa, pb);
    }
    if (sign(pb.len()) == 0) {
        return 0;
    }
    double a = pb.len(), b = pa.len(), c = (pb - pa).len();
    double sinB = fabs(det(pb, pb - pa) / a / c),
          cosB = dot(pb, pb - pa) / a / c,
          sinC = fabs(det(pa, pb) / a/ b),
          cosC = dot(pa, pb) / a / b;
    double B = atan2(sinB, cosB), C = atan2(sinC, cosC);
    if (a > r) {
        S = C / 2 * r * r;
        h = a * b * sinC / c;
        if (h < r && B < PI / 2) {
            S -= (acos(h / r) * r * r - h * sqrt(r * r - h * h));
        }
    } else if (b > r) {
        double theta = PI - B - asin(sinB / r * a);
        S = a * r * sin(theta) / 2 + (C - theta) / 2 * r * r;
    } else {
        S = sinC * a * b / 2;
    }
    return S;
}

double circleCrossTriangle(const Circle &c, const Point &a, const Point &b, const Point &c) {
    Point p[4] = {a, b, c};
    double s = 0;
    for (int i = 0; i < 3; ++i) {
        p[i] = p[i] - c.o;
    }
    p[3] = p[0];
    for (int i = 0; i < 3; ++i) {
        S += circleCrossSegment(p[i], p[i + 1]) * sign(det(p[i], p[i + 1]));
    }
    return fabs(S);
}
```

圆的交点和切线

```
vector<Point> circleIntersectLine(const Circle &c, const Line &l) {
    double x = dot(l.a - c.o, l.b - l.a),
           y = (l.b - l.a).len2(),
           d = x * x - y * ((l.a - c.o).len2() - c.r * c.r);
    vector<Point> ret;
    if (d < eps) {
        return ret;
    }
    if (d < 0) {
        d = 0;
    }
    Point p = l.a - ((l.b - l.a) * (x / y)), delta = (l.b - l.a) * (sqrt(d) / y);
    ret.push_back(p + delta);
    ret.push_back(p - delta);
    return ret;
}

vector<Point> circleIntersectCircle(const Circle &c0, const Circle &c1) {
    double x = (c1.o - c2.o).len2(),
           y = ((sqr(c1.r) - sqr(c2.r)) / x + 1) / 2,
           d = sqr(c1.r) / x - sqr(y);
    vector<Point> ret;
    if (d < -eps) {
        return ret;
    }
    if (d < 0) {
        d = 0;
    }
    Point p = c1.o + (c2.o - c1.o) * y, delta = ((c2.o - c1.o) * sqrt(d)).turn90();
    ret.push_back(p - delta);
    ret.push_back(p + delta);
    return ret;
}

vector<Point> circleTangentPoint(const Circle &c, const Point &p0) {
    double x = (p0 - c.o).len2(),
           d = x - r * r;
    vector<Point> ret;
    if (d < -eps) {
        return ret;
    }
    if (d < 0) {
        d = 0;
    }
    Point p = (p0 - c.o) * (r * r / x), delta = ((p0 - c.o) * (-r * sqrt(d) / x)).turn90();
    ret.push_back(c.o + p + delta);
    ret.push_back(c.o + p - delta);
    return ret;
}
vector<Line> circleTangentCircle(const Circle &c0, const Circle &c1) {
    vector<Line> ret;
    if (sign(c0.r - c1.r) == 0) {
        Point dir = c1.o - c0.o;
        dir = (dir * (c0.r / dir.len())).turn90();
        ret.push_back(Line(c0.o + dir, c1.o + dir));
        ret.push_back(Line(c0.o - dir, c1.o - dir));
    } else {
        Point p = (c1.o * -r2 + (c1.o * r1)) / (c0.r - c1.r);
        vector<Point> ret1 = circleTangentPoint(c0, p), ret2 =
circleTangentPoint(c1, p);
        for (int i = 0; i < (int)ret1.size() && i < (int)ret2.size(); ++i) {
            ret.push_back(Line(ret1[i], ret2[i]));
        }
    }
    Point p = (c0.o * c1.r + c1.o * c0.r) / (c1.r + c0.r);
    vector<Point> ret1 = circleTangentPoint(c0, p), ret2 = circleTangentPoint(c1,
p);
    for (int i = 0; i < (int)ret1.size() && i < (int)ret2.size(); ++i) {
        ret.push_back(Line(ret1[i], ret2[i]));
    }
    return ret;
}
double circleCrossCircle(const Circle &c0, const Circle &c1) {
    double d = (c0.o - c1.o).len();
    if (sign(d - c0.r + c1.r) > 0) {
        return 0;
    }
    if (sign(d - fabs(r1 + r2)) < 0) {
        double r = min(c0.r, c1.r);
        return r * r * PI;
    }
    double x = (d * d + c0.r * c0.r - c1.r * c1.r) / (2 * d),
           t1 = acos(x / c0.r), t2 = acos((d - x) / c1.r);
    return c0.r * c0.r * t1 + c1.r * c1.r * t2 - d * c0.r * sin(t1);
}
```

三维几何基础

```
struct Point3D {
    double x, y, z;
};
Point3D det(const Point3D &a, const Point3D &b) {
    return Point3D(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y
* b.x);
}
//平面法向量：平面上两个向量叉积
//共平面：平面上一点与之的向量点积法向量为 0
//点在线段（直线）上：共线且两边点积非正
//点在三角形内（不包含边界需在判断是与某条边共线）
bool pointInTri(const Point3D &a, const Point3D &b, const Point3D &c, const Point3D
&p) {
    return sign(det(a - b, a - c).len() - det(p - a, p - b).len() - det(p - b, p
- c).len() - det(p - c, p - a).len()) == 0;
}
//共平面的两点是否在这平面上一条直线的同侧
bool sameSide(const Point3D &a, const Point3D &b, const Point3D &p0, const Point3D
&p1) {
    return sign(dot(det(a - b, p0 - b), det(a - b, p1 - b))) > 0;
}
//两点在平面同侧：点积法向量符号相同
//两直线平行/垂直：同二维
//平面平行/垂直：判断法向量
//线面垂直：法向量和直线平行
//判断空间线段是否相交：四点共面两线段不平行相互在异侧
//线段和三角形是否相交：线段在三角形平面不同侧 三角形任意两点在线段和第三点组成的平面的不同侧
//求空间直线交点
Point3D intersection(const Point3D &a0, const Point3D &b0, const Point3D &a1, const
Point3D &b1) {
    double t = ((a0.x - a1.x) * (a1.y - b1.y) - (a0.y - a1.y) * (a1.x - b1.x)) /
((a0.x - b0.x) * (a1.y - b1.y) - (a0.y - b0.y) * (a1.x - b1.x));
    return a0 + (b0 - a0) * t;
}
//求平面和直线的交点
Point3D intersection(const Point3D &a, const Point3D &b, const Point3D &c, const
Point3D &l0, const Point3D &l1) {
    Point3D p = pVec(a, b, c);
    double t = (p.x * (a.x - l0.x) + p.y * (a.y - l0.y) + p.z * (a.z - l0.z)) /
(p.x * (l1.x - l0.x) + p.y * (l1.y - l0.y) + p.z * (l1.z - l0.z));
    return l0 + (l1 - l0) * t;
}
//求平面交线：取不平行的一条直线的一个交点，以及法向量叉积得到直线方向
//点到直线距离：叉积得到三角形的面积除以底边
//点到平面距离：点积法向量
//直线间距离：平行时随便取一点求距离，否则叉积方向向量得到方向点积计算长度
//直线夹角：点积 平面夹角：法向量点积
//三维向量旋转操作（绕向量 s 旋转 ang 角度）
//矩阵版：
void rotate(const Point3D &s, double ang) {
    double l = s.len(), x = s.x / l, y = s.y / l, z = s.z / l,
           sinA = sin(ang), cosA = cos(ang);
    double p[4][4]= {CosA + (1 - CosA) * x * x, (1 - CosA) * x * y - SinA * z, (1
- CosA) * x * z + SinA * y, 0,
                    (1 - CosA) * y * x + SinA * z, CosA + (1 - CosA) * y * y, (1 -
CosA) * y * z - SinA * x, 0,
                    (1 - CosA) * z * x - SinA * y, (1 - CosA) * z * y + SinA * x,
CosA + (1 - CosA) * z * z, 0,
                    0, 0, 0, 1
                   };
}
//计算版：把需要旋转的向量按照 s 分解，做二维旋转，再回到三维
```

三维凸包

```
double mix(const Point &a, const Point &b, const Point &c) {
    return a.dot(b.cross(c));
}
double area(int a, int b, int c) {
    return ((info[b] - info[a]).cross(info[c] - info[a])).length();
}
double volume(int a, int b, int c, int d) {
    return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]);
}
struct Face {
    int a, b, c;
};
vector <Face> face;
inline void insert(int a, int b, int c) {
    face.push_back(Face(a, b, c));
}
void add(int v) {
    vector <Face> tmp;
    int a, b, c;
    cnt ++;
    for (int i = 0; i < SIZE(face); i ++) {
        a = face[i][0];
        b = face[i][1];
        c = face[i][2];
        if (Sign(volume(v, a, b, c)) < 0)
            mark[a][b]=mark[b][a]=mark[b][c] = mark[c][b] = mark[c][a] = mark[a][c]
= cnt;
        else  tmp.push_back(face[i]);
    }
    face = tmp;
    for (int i = 0; i < SIZE(tmp); i ++) {
        a = face[i][0];
        b = face[i][1];
        c = face[i][2];
        if (mark[a][b] == cnt) insert(b, a, v);
        if (mark[b][c] == cnt) insert(c, b, v);
        if (mark[c][a] == cnt) insert(a, c, v);
    }
}
int Find() {
    for (int i = 2; i < n; i ++) {
        Point ndir = (info[0] - info[i]).cross(info[1] - info[i]);
        if (ndir == Point()) continue;
        swap(info[i], info[2]);
        for (int j = i + 1; j < n; j ++)
            if (Sign(volume(0, 1, 2, j)) != 0) {
                swap(info[j], info[3]);
                insert(0, 1, 2);
                insert(0, 2, 1);
                return 1;
            }
    }
    return 0;
}
int main() {
    double ans, ret;
    int Case;
    for (scanf("%d", &Case); Case; Case --) {
        scanf("%d", &n);
        for (int i = 0; i < n; i ++) info[i].read();
        sort(info, info + n);
        n = unique(info, info + n) - info;
        face.clear();
        random_shuffle(info, info + n);
        ans = ret = 0;
        if (Find()) {
            memset(mark, 0, sizeof(mark));
            cnt = 0;
            for (int i = 3; i < n; i ++) add(i);
            int first = face[0][0];
            for (int i = 0; i < SIZE(face); i ++) {
                ret += area(face[i][0], face[i][1], face[i][2]);
                ans += fabs(volume(first, face[i][0], face[i][1], face[i][2]));
            }
            ans /= 6;
            ret /= 2;
        }
        printf("%.3f %.3f\n", ret, ans);
    }
    return 0;
}
```

三维凸包求重心

```
double calcDist(const Point &p, int a, int b, int c) {
    return fabs(mix(info[a] - p, info[b] - p, info[c] - p) / area(a, b, c));
}
//compute the minimal distance of center of any faces
double findDist() {
    //compute center of mass
    double totalWeight = 0;
    Point center(.0, .0, .0);
    Point first = info[face[0][0]];
    for (int i = 0; i < SIZE(face); ++i) {
        Point p = (info[face[i][0]] + info[face[i][1]] + info[face[i][2]] + first)
* .25;
        double weight = mix(info[face[i][0]] - first, info[face[i][1]] - first,
info[face[i][2]] - first);
        totalWeight += weight;
        center = center + p * weight;
    }
    center = center / totalWeight;
    //compute distance
    double res = 1e100;
    for (int i = 0; i < SIZE(face); ++i) {
        res = min(res, calcDist(center, face[i][0], face[i][1], face[i][2]));
    }
    return res;
}
```

质因数分解 PollardRho

```
typedef long long ll;
ll mulmod(ll x, ll y, ll n) {
    ll d = (long long)((long double)x * y / n);
    d = x * y - n * d;
    while (d < 0) d += n;
    while (d >= n) d -= n;
    return d;
}
ll powmod(ll a, ll b, ll mo) {
    ll ret = 1;
    while (b) {
        if (b & 1) ret = mulmod(ret, a, mo);
        a = mulmod(a, a, mo), (b >>= 1);
    }
    return ret;
}
int testP(ll n, int base) {
    ll n2 = n - 1, res;
    int s = 0;
    while (n2 % 2 == 0) n2 >>= 1, s++;
    res = powmod(base, n2, n);
    if ((res == 1) || (res == n - 1)) return 1;
    s--;
    while (s >= 0) {
        res = mulmod(res, res, n);
        if (res == n - 1)  return 1;
        s--;
    }
    return 0;
}
int testNumber[19] = {4, 2, 3, 1373653ll, 5, 25326001ll, 7, 25000000000ll, 11,
2152302898747ll, 13, 3474749660383ll, 17, 341550071728321ll, 19, 23, 29, 31, 37};
int testType[19] = {0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1};
int isPrime(long long n) {
    if (n < 2 || n == 3215031751ll) return 0;
    for (int i = 0; i < 19; i++) {
        if (testType[i] == 1) {
            if (testP(n, testNumber[i]) == 0) return 0;
        } else {
            if (n < testNumber[i]) return 1;
        }
    }
    return 1;
}
ll pollardRho(ll n, ll seed) {
    ll x, y;
    x = y = rand() % (n - 1) + 1;
    ll head = 1, tail = 2;
    while (true) {
        x = mulmod(x, x, n);
        x = (x + seed) % n;
        if (x == y) return n;
        ll d = __gcd(abs(x - y), n);
        if (1 < d && d < n) return d;
        head ++;
        if (head == tail) {
            y = x;
            tail <<= 1;
        }
    }
}
vector <ll> divisors;
void factor(ll n) {
    if (n > 1) {
        if (isPrime(n)) {
            divisors.push_back(n);
        } else {
            ll d = n;
            while (d >= n) {
                d = pollardRho(n, rand() % (n - 1) + 1);
            }
            factor(n / d);
            factor(d);
        }
    }
}
```

离散对数

```
void extendedGcd (int a, int b, long long &x, long long y) {
    if (b) {
        extendedGcd(b, a % b, y, x);
        y -= a / b * x;
    } else {
        x = a;
        y = 0;
    }
}
int inverse (int a, int m) {
    long long x, y;
    extendedGcd(a, m, x, y);
    return (x % m + m) % m;
}
// a ^ x = b (mod m)
int solve (int a, int b, int m) {
    int tmp = 1 % m, c;
    map<int, int> s;
    if (tmp == b) {
        return 0;
    }
    for (int i = 1; i <= 50; ++ i) {
        tmp = ((long long)tmp * a) % m;
        if (tmp == b) {
            return i;
        }
    }
    int x_0 = 0, d = 1 % m;
    while (true) {
        tmp = gcd(a, m);
        if (tmp == 1) {
            break;
        }
        x_0 ++;
        d = ((long long)d * (a / tmp)) % m;
        if (b % tmp) {
            return -1;
        }
        b /= tmp;
        m /= tmp;
    }
    b = ((long long)b * inverse(d, m)) % m;
    c = int(ceil(sqrt(m)));
    s.clear();
    tmp = b;
    int tmpInv = intverse(a, m);
    for (int i = 0; i != c; ++ i) {
        if (s.find(tmp) == s.end()) {
            s[tmp] = i;
        }
        tmp = ((long long)tmp * tmpInv) % m;
    }
    tmp = 1;
    for (int i = 0; i != c; ++ i) {
        tmp = ((long long)tmp * a) % m;
    }
    int ans = 1;
    for (int i = 0; i != c; ++ i) {
        if (s.find(ans) != s.end()) {
            return x_0 + i * c + s.find(ans)->second;
        }
        ans = ((long long)ans * tmp) % m;
    }
    return -1;
}
```

二次剩余

```
/*a*x^2+b*x+c==0 (mod P) 求 0..P-1 的根 */
int pDiv2,P,a,b,c,Pb,d;
//calc:快速幂 rev:逆元
inline void Compute() {
    while (1) {
        b=rand()%(P-2)+2;
        if (calc(b,pDiv2)+1==P) return;
    }
}
int main() {
    srand(time(0)^312314);
    int T;
    for (scanf("%d",&T); T; --T) {
        scanf("%d%d%d%d",&a,&b,&c,&P);
        if (P==2)  {
            int cnt=0;
            for (int i=0; i<2; ++i) if ((a*i*i+b*i+c)%P==0) ++cnt;
            printf("%d",cnt);
            for (int i=0; i<2; ++i) if ((a*i*i+b*i+c)%P==0) printf(" %d",i);
            puts("");
        } else  {
            int delta=(long long)b*rev(a)*rev(2)%P;
            a=(long long)c*rev(a)%P-sqr( (long long)delta )%P;
            a=(p-((a%p)+p)%p)%p;
            pDiv2=P/2;
            if (calc(a,pDiv2)+1==P) puts("0");
            else {
                int t=0,h=pDiv2;
                while (!(h%2)) ++t,h/=2;
                int root=calc(a,h/2);
                if (t>0) {
                    Compute();
                    Pb=calc(b,h);
                }
                for (int i=1; i<=t; ++i) {
                    d=(long long)root*root*a%P;
                    for (int j=1; j<=t-i; ++j)  d=(long long)d*d%P;
                    if (d+1==P)  root=(long long)root*Pb%P;
                    Pb=(long long)Pb*Pb%P;
                }
                root=(long long)a*root%P;
                int root1=P-root;
                root-=delta;
                root%=P;
                if (root<0) root+=P;
                root1-=delta;
                root1%=P;
                if (root1<0) root1+=P;
                if (root>root1) {
                    t=root;
                    root=root1;
                    root1=t;
                }
                if (root==root1) printf("1 %d\n",root);
                else printf("2 %d %d\n",root,root1);
            }
        }
    }
    return 0;
}
```

Pell 方程

```
//求 x^2-ny^2=1 的最小正整数根,n 不是完全平方数
p[1]=1;p[0]=0;q[1]=0;q[0]=1;
a[2]=(int)(floor(sqrt(n)+1e-7));
g[1]=0;h[1]=1;
for (int i=2; i; ++i) {
    g[i]=-g[i-1]+a[i]*h[i-1];
    h[i]=(n-sqr(g[i]))/h[i-1];
    a[i+1]=(g[i]+a[2])/h[i];
    p[i]=a[i]*p[i-1]+p[i-2];
    q[i]=a[i]*q[i-1]+q[i-2];
    //检查 p[i],q[i]是否为解，如果是，则退出
}
```

FFT

```
typedef complex<double> Complex;
void FFT(Complex p[], int n, int oper) {
    for (int i = 1, j = 0; i < n - 1; ++i) {
        for (int s = n; j ^= s >>= 1, ~j & s;);
        if (i < j) {
            swap(p[i], p[j]);
        }
    }
    for (int d = 0; (1 << d) < n; ++d) {
        int m = 1 << d, m2 = m * 2;
        double p0 = pi / m * oper;
        Complex unit_p0(cos(p0), sin(p0));
        for (int i = 0; i < n; i += m2) {
            Complex unit(1, 0);
            for (int j = 0; j < m; ++j) {
                Complex &p1 = p[i + j + m], &p2 = p[i + j];
                Complex t = unit * p1;
                p1 = p2 - t;
                p2 = p2 + t;
                unit = unit * unit_p0;
            }
        }
    }
}
```

NFT

```
//R 是 2^n*q+1 形质数 p 的原根
void NFT(int P[], int n, int oper) {
    for (int i = 1, j = 0; i < n - 1; ++i) {
        for (int s = n; j ^= s >>= 1, ~j & s;);
        if (i < j) {
            swap(P[i], P[j]);
        }
    }
    for (int d = 0; (1 << d) < n; ++d) {
        int m = 1 << d, m2 = m * 2;
        int unit_p0 = powmod(R, (MOD - 1) / m2);
        if (oper < 0) {
            unit_p0 = inverse(unit_p0);
        }
        for (int i = 0; i < n; i += m2) {
            int unit = 1;
            for (int j = 0; j < m; ++j) {
                int &P1 = P[i + j + m], &P2 = P[i + j];
                int t = (long long)unit * P1 % MOD;
                P1 = (P2 - t + MOD) % MOD;
                P2 = (P2 + t) % MOD;
                unit = (long long)unit * unit_p0 % MOD;
            }
        }
    }
}
```

最小费用流

```
struct MinCostFlow {
    int e[M], succ[M], last[N], val[M], cost[M], sum;
    int dis[N], visit[N], slack[N];
    int source, target, totFlow, totCost;
    void init(int n) {
        for (int i = 1; i <= n; i++) {
            last[i] = 0;
        }
        sum = 1;
    }
    void add(int a, int b, int c, int d) {
        e[++sum] = b, succ[sum] = last[a], last[a] = sum;
        e[++sum] = a, succ[sum] = last[b], last[b] = sum;
        val[sum - 1] = c, val[sum] = 0;
        cost[sum - 1] = d, cost[sum] = -d;
    }
    int modlable() {
        int delta = INF;
        for (int i = 1; i <= target; i++) {
         if (!visit[i] && slack[i] < delta) {
            delta = slack[i];
         }
         slack[i] = INF;
      }
      if (delta == INF) {
         return 1;
      }
      for (int i = 1; i <= target; i++) {
         if (visit[i]) {
            dis[i] += delta;
         }
      }
      return 0;
   }
   int dfs(int x, int flow) {
      if (x == target) {
         totFlow += flow;
         totCost += flow * (dis[source] - dis[target]);
         return flow;
      }
      visit[x] = 1;
      int left = flow;
      for (int i = last[x]; i; i = succ[i]) {
         if (val[i] > 0 && !visit[e[i]]) {
            int y = e[i];
            if (dis[y] + cost[i] == dis[x]) {
               int delta = dfs(y, min(left, val[i]));
               val[i] -= delta;
               val[i ^ 1] += delta;
               left -= delta;
               if (!left) {
                  visit[x] = 0;
                  return flow;
               }
            } else {
               slack[y] = min(slack[y], dis[y] + cost[i] - dis[x]);
            }
         }
      }
      return flow - left;
   }
   pair <int, int> minCost() {
      totFlow = 0, totCost = 0;
      fill(dis + 1, dis + target + 1, 0);
      do {
         do {
            fill(visit + 1, visit + target + 1, 0);
         } while (dfs(source, INF));
      } while (!modlable());
      return make_pair(totFlow, totCost);
```

```
   }
} mcf;
```

无源汇最小割

```
long long globalMinCut(long long n) {
   bool* A=new bool[n];
   long long* V=new long long[n];
   long long* W=new long long[n];
   initSet(n, V);
   long long best=MAX;
   while (n>1) {
      long long maxj=1;
      A[V[0]] = true;
      for (long long i=1; i<n; ++i) {
         A[V[i]]=false;
         W[i]=graph[V[0]][V[i]];
         if (W[i]>W[maxj])
            maxj=i;
      }
      long long prev=0,buf=n;
      while (--buf) {
         A[V[maxj]]=true;
         if (buf==1) {
            best=min(best,W[maxj]);
            for (long long k=0; k<n; ++k)
               graph[V[k]][V[prev]]=
                             (graph[V[prev]][V[k]]+=graph[V[maxj]][V[k]]);
            V[maxj]=V[--n];
         }
         prev=maxj;
         maxj=-1;
         for (long long j=1; j<n; ++j)
            if (!A[V[j]]) {
               W[j]+=graph[V[prev]][V[j]];
               if (maxj<0 || W[j]>W[maxj])
                  maxj=j;
            }
      }
   }
   delete[] A; delete[] V; delete[] W;
   return best;
}
```

有向图最小生成树

```
const int MAXN = 1100;
int n, m, g[MAXN][MAXN], used[MAXN], pass[MAXN], eg[MAXN], more, queue[MAXN];
void combine(int id, int &sum) {
   int tot = 0, from, i, j, k;
   for (; id!=0 && !pass[id]; id=eg[id] ) {
      queue[tot++] = id;
      pass[id] = 1 ;
   }
   for ( from = 0 ; from<tot && queue[from]!=id; from++);
```

```
    if ( from == tot ) return ;
    more = 1 ;
    for ( i=from ; i<tot ; i++ ) {
        sum += g[eg[queue[i]]][queue[i]];
        if ( i != from ) {
            used[queue[i]] = 1 ;
            for ( j = 1 ; j <= n ; ++j ) if ( !used[j] )
                    if ( g[queue[i]][j] < g[id][j] ) g[id][j] = g[queue[i]][j];
        }
    }
    for ( i = 1 ; i <= n ; ++i ) if ( !used[i] && i != id ) {
            for ( j = from ; j < tot ; ++j ) {
                k = queue[j];
                if ( g[i][id] > g[i][k]-g[eg[k]][k] ) g[i][id]=g[i][k]-g[eg[k]][k];
            }
        }
}
int mdst( int root ) {// return the total length of MDST
    int i, j, k, sum = 0 ;
    memset( used, 0, sizeof(used) ) ;
    for ( more = 1 ; more ; ) {
        more = 0 ;
        memset( eg,0,sizeof(eg) );
        for ( i = 1 ; i <= n ; ++i ) if ( !used[i] && i != root ) {
                for ( j = 1 , k = 0 ; j <= n ; ++j ) if ( !used[j] && i != j )
                        if ( k==0 || g[j][i] < g[k][i] ) k = j;
                eg[i] = k ;
            }
        memset( pass , 0 , sizeof(pass));
        for ( i = 1 ; i <= n ; ++i ) if ( !used[i] && !pass[i] && i != root )
combine(i,sum);
    }
    for ( i = 1 ; i <= n ; ++i ) if ( !used[i] && i != root ) sum += g[eg[i]][i];
    return sum ;
}
int main() {
    //    g[a][b] = inf when a,b disconnect
    return 0;
}
```

最大匹配 Hopcroft

```
#define maxn 50005   #define maxm 150005
inline int Maxmatch() {
    memset(mk,-1,sizeof(mk));
    memset(cx,-1,sizeof(cx));
    memset(cy,-1,sizeof(cy));
    for (int p=1,fl=1,h,tail; fl; ++p) {
        fl=0;
        h=tail=0;
        for (int i=0; i<n; ++i) if (cx[i]==-1)
                q[++tail]=i,pre[i]=-1,src[i]=i;
        for (h=1; h<=tail; ++h) {
            int u=q[h];
            if (cx[src[u]]!=-1) continue;
            for (int pp=head[u],v=vtx[pp]; pp; pp=next[pp],v=vtx[pp])
                if (mk[v]!=p)   {
                    mk[v]=p;
                    q[++tail]=cy[v];
                    if (cy[v]>=0)        {
                        pre[cy[v]]=u;
                        src[cy[v]]=src[u];
                        continue;
                    }
                    int d,e,t;
                    for (--tail,fl=1,d=u,e=v; d!=-1;
t=cx[d],cx[d]=e,cy[e]=d,e=t,d=pre[d]);
                    break;
                }
        }
    }
    int res=0;
    for (int i=0; i<n; ++i)   res+=(cx[i]!=-1);
    return res;
}
```

最优匹配 KM

```
const int MAXN = 300 ;
const int oo = 0x7fffffff;
int
w[MAXN][MAXN],x[MAXN],y[MAXN],px[MAXN],py[MAXN],sy[MAXN],slack[MAXN],par[MAXN
];
int n,pa[MAXN][2],pb[MAXN][2],n0,m0,na,nb;
void adjust( int v ) {
    sy[v] = py[v];
    if ( px[sy[v]]!=-2 ) adjust( px[sy[v]] );
}
bool find( int v ) {
    int i;
    for ( i = 0 ; i < n ; ++ i )
        if ( py[i]==-1 ) {
            if ( slack[i] > x[v]+y[i]-w[v][i] ) {
                slack[i] = x[v]+y[i]-w[v][i];
                par[i] = v ;
            }
            if ( x[v]+y[i] == w[v][i] ) {
                py[i] = v;
                if ( sy[i] == -1 ) {
                    adjust(i);
                    return 1;
                }
                if ( px[sy[i]] != -1 ) continue ;
                px[sy[i]] = i ;
                if ( find(sy[i]) ) return 1;
            }
        }
    return 0;
}
int km() {
    int i,j,m;
    for ( i = 0 ; i < n ; ++i ) sy[i] = -1 , y[i] = 0 ;
    for ( i = 0 ; i < n ; ++i ) {
        x[i] = 0 ;
        for ( j = 0 ; j < n ; ++j ) x[i] = max( x[i],w[i][j] );
    }
    bool flag ;
    for ( i = 0 ; i < n ; ++i ) {
        for ( j = 0 ; j < n ; ++j ) px[j] = py[j] = -1 , slack[j] = oo ;
        px[i] = -2 ;
        if ( find(i) ) continue ;
        flag = false ;
        for ( ; !flag; ) {
            m = oo;
            for ( j = 0 ; j < n ; ++j ) if ( py[j] == -1 ) m = min( m , slack[j] ) ;
            for ( j = 0 ; j < n ; ++j ) {
                if ( px[j] != -1 ) x[j] -= m ;
                if ( py[j] != -1 ) y[j] += m ;
                else slack[j] -= m ;
            }

            for ( j = 0 ; j < n ; ++j ) {
                if ( py[j] == -1 && !slack[j] ) {
                    py[j] = par[j] ;
                    if ( sy[j] == -1 ) {
                        adjust(j);
                        flag = true ;
                        break;
                    }
                    px[sy[j]] = j ;
                    if ( find(sy[j])) {
                        flag = true ;
                        break;
                    }
                }
            }
        }
    }
    int ans = 0 ;
    for ( i = 0 ; i < n ; ++i ) ans += w[sy[i]][i];
    return ans ;
}
```

一般图最大匹配

```
const int N = 200;
int n, next[N], f[N], mark[N], visited[N], link[N], q[N], head, tail;
vector <int> e[N];
int getf(int x) {
    return f[x] == x ? x : f[x] = getf(f[x]);
}
void merge(int x, int y) {
    x = getf(x), y = getf(y);
    if (x != y) f[x] = y;
}
int lca(int x, int y) {
    static int flag = 0;
    flag++;
    for (; ; swap(x, y)) {
        if (x != -1) {
            x = getf(x);
            if (visited[x] == flag) return x;
            visited[x] = flag;
            if (link[x] != -1) {
                x = next[link[x]];
            } else {
                x = -1;
            }
        }
    }
}
void go(int a, int p) {
    while (a != p) {
        int b = link[a], c = next[b];
        if (getf(c) != p) next[c] = b;
        if (mark[b] == 2) mark[q[tail++] = b] = 1;
        if (mark[c] == 2) mark[q[tail++] = c] = 1;
        merge(a, b), merge(b, c), a = c;
    }
}
void find(int s) {
    for (int i = 0; i < n; i++) {
        next[i] = -1, f[i] = i;
        mark[i] = 0, visited[i] = -1;
    }
    head = tail = 0, q[tail++] = s, mark[s] = -1;
    for (; head < tail && link[s] == -1; ) {
        for (int i = 0, x = q[head++]; i < (int)e[x].size(); i++) {
            if (link[x] != e[x][i] && getf(x) != getf(e[x][i]) && mark[e[x][i]] !=
2) {
                int y = e[x][i];
                if (mark[y] == 1) {
                    int p = lca(x, y);
                    if (getf(x) != p) next[x] = y;
                    if (getf(y) != p) next[y] = x;
                    go(x, p), go(y, p);
                } else if (link[y] == -1) {
                    next[y] = x;
                    for (int j = y; j != -1; ) {
                        int k = next[j], temp = link[k];
                        link[j] = k, link[k] = j, j = temp;
                    }
                    break;
                } else {
                    next[y] = x;
                    mark[q[tail++] = link[y]] = 1;
                    mark[y] = 2;
                }
            }
        }
    }
}
int getMaxMatch(const int &n) {
    for (int i = 0; i < n; i++) link[i] = -1;
    for (int i = 0; i < n; i++) {
        if (link[i] == -1) find(i);
    }
    int ans = 0;
    for (int i = 0; i < n; i++) ans += (link[i] != -1);
    return ans;
}
```

弦图/完美消除序列

从 n 到 1 的顺序依次给点标号（标号为 i 的点出现在完美消除序列的第 i 个）设 lable[i]表示第 i 个点与多少已标号的点相邻，每次选择 label[i]最大的未标号点进行标号。任取一个已标号的与当前新标号的点相邻的点，如果与其他的已标号的且与当前点相邻的点之间没有边，则无解。

1.团数 ≤ 色数

2.最大独立集数 ≤ 最小团覆盖数

3.任何一个弦图都至少有一个单纯点，不是完全图的弦图至少有两个不相邻的单纯点。

4.设第 i 个点在弦图的完美消除序列第 p(i)个。令 N(v) = {w | w 与 v 相邻且 p(w) > p(v)}弦图的极大团一定是 v∪N(v)的形式。

5.弦图最多有 n 个极大团。

6.设 next(v) 表示 N(v)中最前的点。令 w*表示所有满足 A∈B 的 w 中最后的一个点。判断 v∪N(v)是否为极大团,只需判断是否存在一个 w，满足 Next(w) = v 且|N(v)| + 1 ≤ |N(w)|即可。

7.最小染色：完美消除序列从后往前依次给每个点染色，给每个点染上可以染的最小的颜色。//团数=色数

8.最大独立集：完美消除序列从前往后能选就选。

9.最小团覆盖：设最大独立集为{p1 , p2 , …, pt}，则{p1∪N(p1), …, pt∪N(pt)}为最小团覆盖。//最大独立集数 = 最小团覆盖数!!!

```
const int MAXN = 1001 ;
int label[MAXN] , used[MAXN] , number[MAXN] , g[MAXN][MAXN] , n , m ;
priority_queue< pair<int,int> > heaps;
vector<int> link[MAXN] ;
void push( int x , int label ) {
    heaps.push( make_pair(label,x) );
}
int top() {
    pair<int,int> r ;
    do {
        r = heaps.top();
        heaps.pop();
    } while ( r.first != label[r.second] || used[ r.second ] == 1 );
    return r.second;
}
void work() {//每个点 i，在所有 number 标号比 i 小的点构成的诱导子图中是单纯点 O( M*log(N) )
    for ( int i = 1 ; i <= n ; ++i ) {
        label[i] = 0 , used[i] = 0 ;
        push( i , label[i] ) ;
    }
    for ( int time = 0 ; time < n ; ++time ) {
        int x = top();
        number[x] = time ;
        used[x] = 1 ;
        for ( int i = 0 ; i < link[x].size() ; ++i ) {
            int y = link[x][i] ;
            if ( used[y] == 0 ) {
                label[y] ++ ;
                push( y , label[y] );
            }
        }
    }
}
bool judge() {// 判断是否为完美消除序列
    for ( int i = 1 ; i <= n ; ++i ) {
        int c = -1 ;
        for ( int t = 0 ; t < link[i].size() ; ++t ) {
            int j = link[i][t] ;
            if ( number[j] < number[i] ) {
                if ( c == -1 || number[j] > number[c] )
                    c = j ;
            }
        }
        if ( c != -1 )
            for ( int t = 0 ; t < link[i].size() ; ++t ) {
                int j = link[i][t] ;
                if ( number[j] < number[i] ) {
                    if ( c != j )
                        if ( g[j][c] != 1 ) return false ;
                }
            }
    }
    return true ;
}
```

DominatorTree

```
// idom 即为 immediate dominator
const int oo=1073741819;
int tail[4][200000];
int next[4][2000000],sora[4][2000000];
int ss[4],top,w_time,n,m;
```

```
int
rel[200000],semi[200000],b[200000],idom[200000],best[200000],st[200000],pre[2
00000];
int ans[200000];
void origin() {
    for (int e=0; e<=3; e++) ss[e]=n;
    for (int i=1; i<=n; i++) {
        for (int e=0; e<=3; e++)
            tail[e][i]=i,next[e][i]=0;
        rel[i]=0;
        semi[i]=idom[i]=pre[i]=0,best[i]=i;
        b[i]=i;
        ans[i]=0;
    }
    rel[0]=oo;
}
void link(int e,int x,int y) {

++ss[e],next[e][tail[e][x]]=ss[e],tail[e][x]=ss[e],sora[e][ss[e]]=y,next[e][s
s[e]]=0;
}
void dfs(int x,int y) {
    ++w_time,rel[x]=w_time;
    st[++top]=x,pre[x]=y;
    for (int i=x,ne; next[0][i];) {
        i=next[0][i],ne=sora[0][i];
        if (!rel[ne]) dfs(ne,x);
    }
}
int find(int x) {
    int y=b[x];
    if (b[x]!=x) b[x]=find(b[x]);
    if (rel[semi[best[y]]]<rel[semi[best[x]]])
        best[x]=best[y];
    return b[x];
}
void getans(int x,int sum) {
    ans[x]=sum+x;
    for (int i=x,ne; next[3][i];) {
        i=next[3][i],ne=sora[3][i];
        getans(ne,sum+x);
    }
}
int main() {
    for (; scanf("%d%d",&n,&m)==2;) {
        origin();
        for (int i=1; i<=m; i++) {
            int x,y;
            scanf("%d%d",&x,&y);
            link(0,x,y);
            link(1,y,x);
        }
        w_time=0;
        top=0;
        dfs(n,0);
        for (int i=top; i>=1; i--) {
            int ne=st[i];
            for (int j=ne,na; next[1][j];) {
                j=next[1][j],na=sora[1][j];
                if (!rel[na]) continue;
                if (rel[na]>rel[ne]) {
                    find(na);
                    int y=semi[best[na]];
                    if (rel[y]<rel[semi[ne]]) semi[ne]=y;
                } else {
                    int y=na;
                    if (rel[y]<rel[semi[ne]]) semi[ne]=y;
                }
            }
            if (ne!=n) link(2,semi[ne],ne);
            for (int j=ne,na; next[2][j];) {
                j=next[2][j],na=sora[2][j];
                find(na);
                int y=best[na];
                if (semi[y]==semi[na]) idom[na]=semi[na];
                else idom[na]=y;
            }
            for (int j=ne,na; next[0][j];) {
                j=next[0][j],na=sora[0][j];
                if (pre[na]==ne) {
                    na=find(na);
                    b[na]=ne;
                }
            }
        }
        for (int i=2; i<=top; i++) {
            int ne=st[i];
            if (idom[ne]!=semi[ne]) idom[ne]=idom[idom[ne]];
            link(3,idom[ne],ne);
        }
        getans(n,0);
        for (int i=1; i<=n-1; i++) printf("%d ",ans[i]);
        printf("%d\n",ans[n]);
        //for (int i=1;i<=n;i++) cout<<semi[i]<<' ';cout<<endl;
    }
    return 0;
}
```

AC 自动机

```
const int LEN = 200005;
const int LIMIT = 500;
struct trie {
    int ch;
    int go[3], father, suffix;
    int danger, isEnd;
};
struct ACZDJ {
    int m, qq[LEN];
    trie a[LEN];
    void clear(int x) {
        a[x].danger = a[x].isEnd = 0;
        for (int i = 1; i <= 2; i++) {
            a[x].go[i] = 0;
        }
    }
    void init() {
        clear(m = 1);
    }
    void insert(char *s) {
        int now = 1;
        for (int i = 1; s[i]; i++) {
            int x = s[i] - '0' + 1;
            if (!a[now].go[x]) {
                a[now].go[x] = ++m;
                clear(m);
                a[m].ch = x;
                a[m].father = now;
            }
            now = a[now].go[x];
        }
        a[now].isEnd = 1;
    }
    int find(char *s) {
        int now = 1;
        for (int i = 1; s[i]; i++) {
            int x = s[i] - '0' + 1;
            if (!a[now].go[x]) {
                return 0;
            }
            now = a[now].go[x];
        }
        return a[now].isEnd;
    }
    int child(int x, int ch) {
        if (a[x].go[ch]) {
            return a[x].go[ch];
        } else if (x == 1) {
            return 1;
        } else {
            return child(a[x].suffix, ch);
        }
    }
    void build() {
        int l, r;
        l = r = 1;
        qq[1] = 1;
        while (l <= r) {
            int x = qq[l++];
            for (int i = 1; i <= 2; i++) {
                if (a[x].go[i]) {
                    qq[++r] = a[x].go[i];
                }
            }
        }
        a[1].suffix = 1;
        for (int i = 2; i <= r; i++) {
            int x = qq[i];
            a[x].danger = a[x].isEnd;
            if (a[x].father == 1) {
                a[x].suffix = 1;
                continue;
            }
            a[x].suffix = child(a[a[x].father].suffix, a[x].ch);
            a[x].danger += a[a[x].suffix].danger;
        }
    }
    long long getAns(char *s, int pos, int len) {
        int now = 1;
        long long ans = 0;
        for (int i = 1; i <= len; i++) {
            now = child(now, s[pos] - '0' + 1);
            ans += a[now].danger;
            pos++;
            if (pos > len) {
                pos = 1;
            }
        }
        return ans;
    }
};
```

扩展 KMP

```
void ExtendedKMP(char *a, char *b, int M, int N, int *Next, int *ret) {// a -> 模式串 b -> 匹配串
    int i, j, k;
    for (j = 0; 1 + j < M && a[j] == a[1 + j]; j++);
    Next[1] = j;
    k = 1;
    for (i = 2; i < M; i++) {
        int Len = k + Next[k], L = Next[i - k];
        if (L < Len - i) {
            Next[i] = L;
        } else {
            for (j = max(0, Len - i); i + j < M && a[j] == a[i + j]; j++);
            Next[i] = j;
            k = i;
        }
    }
    for (j = 0; j < N && j < M && a[j] == b[j]; j++);
    ret[0] = j;
    k = 0;
    for (i = 1; i < N; i++) {
        int Len = k + ret[k], L = Next[i - k];
        if (L < Len - i) {
            ret[i] = L;
        } else {
            for (j = max(0, Len - i); j < M && i + j < N && a[j] == b[i + j]; j++);
            ret[i] = j;
            k = i;
        }
    }
}
```

Manacher

```
void manacher(char text[], int n, int palindrome[]) {
    palindrome[0] = 1;
    for (int i = 1, j = 0, i < (n << 1) - 1; ++ i) {
        int p = i >> 1;
        int q = i - p;
        int r = (j + 1 >> 1) + palindrome[j] - 1;
        palindrome[i] = r < q? 0: min(r - q + 1, palindrome[(j << 1) - i]);
        while (0 <= p - palindrome[i] && q + palindrome[i] < n
                && text[p - palindrome[i]] == text[q + palindrome[i]]) {
            palindrome[i] ++;
        }
        if (q + palindrome[i] - 1 > r) {
            j = i;
        }
    }
}
```

后缀数组 DC3

```
//DC3 待排序的字符串放在 r 数组中，从 r[0]到 r[n-1]，长度为 n，且最大值小于 m
//约定除 r[n-1]外所有的 r[i]都大于 0, r[n-1]=0
//函数结束后，结果放在 sa 数组中，从 sa[0]到 sa[n-1]
//r 必须开长度的 3 倍
#define maxn 10000
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int wa[maxn],wb[maxn],wv[maxn],wss[maxn];
int s[maxn*3],sa[maxn*3];
int c0(int *r,int a,int b) {
    return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];
}
int c12(int k,int *r,int a,int b) {
    if (k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
    else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
}
void sort(int *r,int *a,int *b,int n,int m) {
    int i;
    for (i=0; i<n; i++) wv[i]=r[a[i]];
    for (i=0; i<m; i++) wss[i]=0;
    for (i=0; i<n; i++) wss[wv[i]]++;
    for (i=1; i<m; i++) wss[i]+=wss[i-1];
    for (i=n-1; i>=0; i--) b[--wss[wv[i]]]=a[i];
}
void dc3(int *r,int *sa,int n,int m) {
    int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
    r[n]=r[n+1]=0;
    for (i=0; i<n; i++)
        if (i%3!=0) wa[tbc++]=i;
    sort(r+2,wa,wb,tbc,m);
    sort(r+1,wb,wa,tbc,m);
    sort(r,wa,wb,tbc,m);
    for (p=1,rn[F(wb[0])]=0,i=1; i<tbc; i++)
        rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
    if (p<tbc) dc3(rn,san,tbc,p);
    else for (i=0; i<tbc; i++) san[rn[i]]=i;
    for (i=0; i<tbc; i++)
        if (san[i]<tb) wb[ta++]=san[i]*3;
    if (n%3==1) wb[ta++]=n-1;
    sort(r,wb,wa,ta,m);
    for (i=0; i<tbc; i++)
        wv[wb[i]=G(san[i])]=i;
    for (i=0,j=0,p=0; i<ta && j<tbc; p++)
        sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
    for (; i<ta; p++) sa[p]=wa[i++];
    for (; j<tbc; p++) sa[p]=wb[j++];
}
```

后缀自动机

```
struct State {
    int length;
    State *parent;
    State* go[C];
    State(int length) : length(length), parent(NULL) {
        memset(go, NULL, sizeof(go));
        states.push_back(this);
    }
    State* extend(State* start, int token) {
        State *p = this;
        State *np = new State(length + 1);
        while (p && !p->go[token]) {
            p->go[token] = np;
            p = p->parent;
        }
        if (!p) {
            np->parent = start;
        } else {
            State *q = p->go[token];
            if (p->length + 1 == q->length) {
                np->parent = q;
            } else {
                State *nq = new State(p->length + 1);
                memcpy(nq->go, q->go, sizeof(q->go));
                nq->parent = q->parent;
                np->parent = q->parent = nq;
                while (p && p->go[token] == q) {
                    p->go[token] = nq;
                    p = p->parent;
                }
            }
        }
        return np;
    }
};
```

字符串环状同构/最小表示

```
int main() {
    int n;
    scanf("%d", &n);
    scanf("%s%s", s1, s2);
    for (int i = 0; i < n; ++i) {
        s1[i + n] = s1[i], s2[i + n] = s2[i];
    }
    int i = 0, j = 0, k = 0;
    while (k < n && i + k < n * 2 && j + k < n * 2) {
        if (s1[i + k] == s2[j + k]) {
            ++k;
        } else if (s1[i + k] > s2[j + k]) {
            i += k + 1;
            k = 0;
        } else {
            j += k + 1;
            k = 0;
        }
    }
    if (k == n) {
        printf("%d\n", (j - i + n) % n);
    } else {
        printf("-1\n");
    }
    return 0;
}

int main() {
    scanf("%s", s);
    n = strlen(s);
    for (int i = 0; i < n; ++i) {
        s[i + n] = s[i];
    }
    int i = 0, j = 1, k = 0;
    while (k < n && i + k < n * 2 && j + k < n * 2) {
        if (s[i + k] == s[j + k]) {
            ++k;
        } else if (s[i + k] > s[j + k]) {
            i = max(i + k + 1, j + 1);
            k = 0;
        } else {
            j = max(j + k + 1, i + 1);
            k = 0;
        }
    }
    printf("%d\n", min(i, j));
    return 0;
}
```

环状最长公共子串

```
int n, a[N << 1], b[N << 1];
bool has(int i, int j) {
    return a[(i - 1) % n] == b[(j - 1) % n];
}
const int DELTA[3][2] = {{0, -1}, {-1, -1}, {-1, 0}};
int from[N][N];
int solve() {
    memset(from, 0, sizeof(from));
    int ret = 0;
    for (int i = 1; i <= 2 * n; ++ i) {
        from[i][0] = 2;
        int left = 0, up = 0;
        for (int j = 1; j <= n; ++ j) {
            int upleft = up + 1 + !!from[i - 1][j];
            if (!has(i, j)) {
                upleft = INT_MIN;
            }
            int max = std::max(left, std::max(upleft, up));
            if (left == max) {
                from[i][j] = 0;
            } else if (upleft == max) {
                from[i][j] = 1;
            } else {
                from[i][j] = 2;
            }
            left = max;
        }
        if (i >= n) {
            int count = 0;
            for (int x = i, y = n; y;) {
                int t = from[x][y];
                count += t == 1;
                x += DELTA[t][0];
                y += DELTA[t][1];
            }
            ret = std::max(ret, count);
            int x = i - n + 1;
            from[x][0] = 0;
            int y = 0;
            while (y <= n && from[x][y] == 0) {
                y++;
            }
            for (; x <= i; ++ x) {
                from[x][y] = 0;
                if (x == i) {
                    break;
                }
                for (; y <= n; ++ y) {
                    if (from[x + 1][y] == 2) {
                        break;
                    }
                    if (y + 1 <= n && from[x + 1][y + 1] == 1) {
                        y ++;
                        break;
                    }
                }
            }
        }
    }
    return ret;
}
```

动态树

```
const ui N = 100005; //最大总点数
const ui mo = 51061;
//维护的数据
struct Node {
    ui size, sum;
    ui addTag, mulTag;
    void init() {
        size = mulTag = sum = 1;
        addTag = 0;
    }
    void add(const ui &x) {
        (addTag += x) %= mo;
        (sum += size * x % mo) %= mo;
    }
    void mul(const ui &x) {
        addTag = addTag * x % mo;
        mulTag = mulTag * x % mo;
        sum = sum * x % mo;
    }
};
Node merge(const Node &a, const Node &b) {
    Node ret;
    ret.size = a.size + b.size;
    ret.sum = (a.sum + b.sum) % mo;
    ret.addTag = 0, ret.mulTag = 1;
    return ret;
}
vector <int> vForLCT; //用来下传标记
struct LinkCutTree {
    ui father[N], ch[N][2], isRoot[N], reverse[N]; //reverse 是翻转标记
    Node info[N], val[N]; //info 为节点本身信息, val 为子树信息
    void init(const ui &n) {
        for (ui i = 1; i <= n; i++) {
            isRoot[i] = 1, father[i] = ch[i][0] = ch[i][1] = reverse[i] = 0;
            info[i].init(), val[i] = info[i];
        }
    }
    void up(ui i) {
        val[i] = ch[i][0] ? merge(val[ch[i][0]], info[i]) : info[i];
        val[i] = ch[i][1] ? merge(val[i], val[ch[i][1]]) : val[i];
    }
    void down(ui i) {
        if (!i) {
            return;
        }
        ui lc = ch[i][0], rc = ch[i][1];
        if (reverse[i]) {
            lc ? (swap(ch[lc][0], ch[lc][1]), reverse[lc] ^= 1) : lc;
            rc ? (swap(ch[rc][0], ch[rc][1]), reverse[rc] ^= 1) : rc;
            reverse[i] = 0;
        }
        if (val[i].mulTag == 1 && val[i].addTag == 0) {
            return;
        }
        if (lc) {
            val[lc].mul(val[i].mulTag);
            val[lc].add(val[i].addTag);
            info[lc].mul(val[i].mulTag);
            info[lc].add(val[i].addTag);
        }
        if (rc) {
            val[rc].mul(val[i].mulTag);
            val[rc].add(val[i].addTag);
            info[rc].mul(val[i].mulTag);
            info[rc].add(val[i].addTag);
        }
        val[i].addTag = 0, val[i].mulTag = 1;
    }
    void rotate(ui i) {
        ui fa = father[i], fa2 = father[fa];
        ui child = (ch[fa][0] == i) ? ch[i][1] : ch[i][0];
        isRoot[i] = isRoot[fa], isRoot[fa] = 0;
        father[i] = fa2, father[fa] = i;
        father[child] = child ? fa : 0;
        if (fa2 && !isRoot[i]) {
            (ch[fa2][0] == fa ? ch[fa2][0] : ch[fa2][1]) = i;
        }
        ui t = (i == ch[fa][0]);
        ch[i][t] = fa, ch[fa][t ^ 1] = child;
        up(fa);
    }
    void splay(ui x) {
        ui now = x;
        vForLCT.clear();
        while (!isRoot[now]) {
            vForLCT.push_back(now), now = father[now];
        }
        down(now);
        for (ui i = vForLCT.size() - 1; i + 1; i--) {
            down(vForLCT[i]);
        }
        while (!isRoot[x]) {
            if (!isRoot[father[x]]) {
                rotate((ch[father[x]][0] == x) ==
                      (ch[father[father[x]]][0] == father[x]) ? father[x] : x);
            }
            rotate(x);
        }
        up(x);
    }
    void expose(ui now, const ui &type = 0, const ui &value = 0) {
        ui now2 = 0;
        while (now) {
            splay(now);
            if (!father[now] && type) {
                if (type == 1) {
                    info[now].add(value);
                    if (now2) {
                        val[now2].add(value), info[now2].add(value);
                    }
                    if (ch[now][1]) {
                        val[ch[now][1]].add(value), info[ch[now][1]].add(value);
                    }
                } else if (type == 2) {
                    info[now].mul(value);
                    if (now2) {
                        val[now2].mul(value), info[now2].mul(value);
                    }
                    if (ch[now][1]) {
                        val[ch[now][1]].mul(value), info[ch[now][1]].mul(value);
                    }
                } else {
                    printf("%d\n", (info[now].sum + val[now2].sum +
val[ch[now][1]].sum) % mo);
                }
            }
            isRoot[now2] = 0;
            isRoot[ch[now][1]] = (ch[now][1] > 0);
            ch[now][1] = now2, up(now);
            now2 = now, now = father[now];
        }
    }
    void link(ui a, ui b) {
        expose(a), splay(a);
        father[a] = b, reverse[a] ^= 1, swap(ch[a][0], ch[a][1]);
    }
    void cut(ui a, ui b) {
        expose(a), splay(b);
        if (father[b] == a) {
            father[b] = 0;
        } else {
            expose(b), splay(a), father[a] = 0;
        }
    }
    void add(const ui &a, const ui &b, const ui &c) {
        expose(a), expose(b, 1, c);
    }
    void mul(const ui &a, const ui &b, const ui &c) {
        expose(a), expose(b, 2, c);
    }
    void askSum(const ui &a, const ui &b) {
        expose(a), expose(b, 3);
    }
} lct;
```

虚树

```
for (int j = 1; j <= k; j++) {
    scanf("%d %d", &h[j].first, &h[j].second);
    t[++tot] = h[j].first;
    v[h[j].first] = h[j].second;
}
sort(h + 1, h + k + 1, byLeft);
for (int j = 1; j <= k; j++) {
    if (!top) {
        father[st[++top] = h[j].first] = 0;
    } else {
        int p = h[j].first, c = lca(p, st[top]);
        for (; dis[st[top]] > dis[c]; --top) {
            if (dis[st[top - 1]] <= dis[c]) {
                father[st[top]] = c;
            }
        }
        if (st[top] != c) {
            t[++tot] = c;
            v[c] = -100;
            father[c] = st[top];
            st[++top] = c;
        }
        father[p] = c;
        st[++top] = p;
    }
}
for (int j = 1; j <= tot; j++) {
    int now = t[j];
    if (father[now] == 0) {
        root = now;
    } else {
        int f = father[now];
        child[f].pb(now);
    }
}
```

DancingLinks

```
struct Node {
    int x, y, u, d, l, r;
    Node (int x = 0, int y = 0, int u = 0, int d = 0, int l = 0, int r = 0) : x(x),
y(y), u(u), d(d), l(l), r(r) {}
};
void insert(int x, int y) {
    int id = a.size();
    a.push_back(Node(x, y, id, id, id, id));
    ++sum[y];
    for (int i = id - 1; i >= 0; --i) {
        if (a[id].u == id && a[i].y == y) {
            a[id].d = a[i].d;
            a[a[i].d].u = id;
            a[i].d = id;
            a[id].u = i;
        }
        if (a[id].l == id && a[i].x == x) {
            a[id].r = a[i].r;
            a[a[i].r].l = id;
            a[id].l = i;
            a[i].r = id;
        }
    }
}
void del(int x) {
    a[a[x].l].r = a[x].r;
    a[a[x].r].l = a[x].l;
    for (int i = a[x].d; i != x; i = a[i].d) {
        for (int j = a[i].r; j != i; j = a[j].r) {
            sum[a[j].y]--;
            a[a[j].u].d = a[j].d;
            a[a[j].d].u = a[j].u;
        }
    }
}
void renew(int x) {
    a[a[x].l].r = x;
    a[a[x].r].l = x;
    for (int i = a[x].u; i != x; i = a[i].u) {
        for (int j = a[i].l; j != i; j = a[j].l) {
            sum[a[j].y]++;
            a[a[j].u].d = j;
            a[a[j].d].u = j;
        }
    }
}
bool DLX(int dep) {
    if (a[0].r == 0) {
        return true;
    }
    int k = -1, mi = 1 << 30;
    for (int i = a[0].r; i != 0; i = a[i].r) {
        if (sum[i] < mi) {
            mi = sum[k = i];
        }
    }
    del(k);
    for (int i = a[k].d; i != k; i = a[i].d) {
        for (int j = a[i].r; j != i; j = a[j].r) {
            del(a[j].y);
        }
        use[a[i].x] = true;
        if (DLX(dep + 1)) {
            return true;
        }
        use[a[i].x] = false;
        for (int j = a[i].l; j != i; j = a[j].l) {
            renew(a[j].y);
        }
    }
    renew(k);
    return false;
}
```

最大团

```
/*Int g[][]为图的邻接矩阵。 MC(V)表示点集 V 的最大团
令 Si= {vi, vi+1, ..., vn}, mc[i]表示 MC(Si). 倒着算 mc[i], 那么显然 MC(V)=mc[1]
        此外有 mc[i]=mc[i+1] or mc[i]=mc[i+1]+1*/
void init() {
    int i, j;
    for (i=1; i<=n; ++i) for (j=1; j<=n; ++j) scanf("%d", &g[i][j]);
}
void dfs(int size) {
    int i, j, k;
    if (len[size]==0) {
        if (size>ans) {
            ans=size;
            found=true;
        }
        return;
    }
    for (k=0; k<len[size] && !found; ++k) {
        if (size+len[size]-k<=ans) break;
        i=list[size][k];
        if (size+mc[i]<=ans) break;
        for (j=k+1, len[size+1]=0; j<len[size]; ++j)
            if (g[i][list[size][j]]) list[size+1][len[size+1]++]=list[size][j];
        dfs(size+1);
    }
}
void work() {
    int i, j;
    mc[n]=ans=1;
    for (i=n-1; i; --i) {
        found=false;
        len[1]=0;
        for (j=i+1; j<=n; ++j) if (g[i][j]) list[1][len[1]++]=j;
        dfs(1);
        mc[i]=ans;
    }
}
```

极大团计数

```
//Bool g[][] 为图的邻接矩阵，图点的标号由 1 至 n。
void dfs(int size) {
    int i, j, k, t, cnt, best = 0;
    bool bb;
    if (ne[size]==ce[size]) {
        if (ce[size]==0) ++ans;
        return;
    }
    for (t=0, i=1; i<=ne[size]; ++i) {
        for (cnt=0, j=ne[size]+1; j<=ce[size]; ++j)
            if (!g[list[size][i]][list[size][j]]) ++cnt;
        if (t==0 || cnt<best) t=i, best=cnt;
    }
    if (t && best<=0) return;
    for (k=ne[size]+1; k<=ce[size]; ++k) {
        if (t>0) {
            for (i=k; i<=ce[size]; ++i)
                if (!g[list[size][t]][list[size][i]]) break;
            swap(list[size][k], list[size][i]);
        }
        i=list[size][k];
        ne[size+1]=ce[size+1]=0;
        for (j=1; j<k; ++j)if (g[i][list[size][j]])
                list[size+1][++ne[size+1]]=list[size][j];
        for (ce[size+1]=ne[size+1], j=k+1; j<=ce[size]; ++j)
            if (g[i][list[size][j]]) list[size+1][++ce[size+1]]=list[size][j];
        dfs(size+1);
        ++ne[size];
        --best;
        for (j=k+1, cnt=0; j<=ce[size]; ++j) if (!g[i][list[size][j]]) ++cnt;
        if (t==0 || cnt<best) t=k, best=cnt;
        if (t && best<=0) break;
    }
}
void work() {
    int i;
    ne[0]=0; ce[0]=0;
    for (i=1; i<=n; ++i) list[0][++ce[0]]=i;
    ans=0;
    dfs(0);
}
```

线性规划

```
//max{cx | Ax <= b, x >= 0}
//无解或无唯一解;返回空 vector
vector<double> simplex(vector<vector<double> > A, vector<double> b, vector<double>
c) {
    int n = A.size(), m = A[0].size() + 1, r = n, s = m - 1;
    vector<vector<double> > D(n + 2, vector<double>(m + 1, 0));
    vector<int> ix(n + m);
    for (int i = 0; i < n + m; ++ i) ix[i] = i;
    for (int i = 0; i < n; ++ i) {
        for (int j = 0; j < m - 1; ++ j) D[i][j] = -A[i][j];
        D[i][m - 1] = 1;
        D[i][m] = b[i];
        if (D[r][m] > D[i][m]) r = i;
    }
    for (int j = 0; j < m - 1; ++ j) D[n][j] = c[j];
    D[n + 1][m - 1] = -1;
    for (double d; ; ) {
        if (r < n) {
            int t = ix[s];
            ix[s] = ix[r + m];
            ix[r + m] = t;
            D[r][s] = 1.0 / D[r][s];
            vector<int> speedUp;
            for (int j = 0; j <= m; ++ j) if (j != s) {
                    D[r][j] *= -D[r][s];
                    if (D[r][j]) {
                        speedUp.push_back(j);
                    }
                }
            for (int i = 0; i <= n + 1; ++ i) if (i != r) {
                    for (int j = 0; j < speedUp.size(); ++ j)
                        D[i][speedUp[j]] += D[r][speedUp[j]] * D[i][s];
                    D[i][s] *= D[r][s];
                }
        }
        r = -1; s = -1;
        for (int j = 0; j < m; ++ j) if (s < 0 || ix[s] > ix[j]) {
                if (D[n + 1][j] > EPS || (D[n + 1][j] > -EPS && D[n][j] > EPS)) s
= j;
            }
        if (s < 0) break;
        for (int i = 0; i < n; ++ i) if (D[i][s] < -EPS) {
                if (r < 0 || (d = D[r][m] / D[r][s] - D[i][m] / D[i][s]) < -EPS ||
(d < EPS && ix[r + m] > ix[i + m]))
                    r = i;
            }
        if (r < 0) return vector<double>(); // 非有界
    }
    if (D[n + 1][m] < -EPS) return vector<double>(); // 无解
    vector<double> x(m - 1);
    for (int i = m; i < n + m; ++ i) if (ix[i] < m - 1) x[ix[i]] = D[i - m][m];
    return x; // 答案存在 D[n][m]
}
```

Romberg

```
template<class T>
double romberg( const T &f , double a , double b , double eps=1e-8 ) {
    std::vector<double> t;
    double h=(b-a),last,curr;
    int k=1,i=1;
    t.push_back(h*(f(a)+f(b))/2);
    do {
        last=t.back();
        curr=0;
        double x=a+h/2;
        for (int j=0; j<k; ++j) {
            curr+=f(x);
            x+=h;
        }
        curr=(t[0]+h*curr)/2;
        double k1=4.0/3.0,k2=1.0/3.0;
        for (int j=0; j<i; ++j) {
            double temp=k1*curr-k2*t[j];
            t[j]=curr;
            curr=temp;
            k2/=4*k1-k2;
            k1=k2+1;
        }
        t.push_back(curr);
        k*=2;
        h/=2;
        ++i;
    } while (std::fabs(last-curr)>eps);
    return t.back();
}
template<class T>
double simpson( const T &f , double a, double b , int n ) {
    double h=(b-a)/n;
    double ans=f(a)+f(b);
    for (int i=1; i<n; i+=2) ans+=4*f(a+i*h);
    for (int i=2; i<n; i+=2) ans+=2*f(a+i*h);
    return ans*h/3;
}
```

球面距离公式

```
//longitude 经度 latitude 纬度 东北为正
double sphereDis(double lon1, double lat1, double lon2, double lat2, double R)
{
    return R * acos(cos(lat1) * cos(lat2) * cos(lon1 - lon2) + sin(lat1) * sin(lat2));
}
```

五边形数定理

```
int partition (int n) {
    p[0] = 1;
    for (int i = 1; i <= n; ++i) {
        for (int j = 1, r = 1; i - (3 * j * j - j) / 2 >= 0; ++j, r *= -1) {
            dp[i] += dp[i - (3 * j * j - j) / 2] * r;
            if (i - (3 * j * j + j) / 2 >= 0) {
                dp[i] += dp[i - (3 * j * j + j) / 2] * r;
            }
        }
    }
    return p[n];
}
```

直线下格点个数

```
//Peake's theorem: S=a+b/2+1 , S 是面积，a 是多边形内部点数，b 是边界上点数
LL solve( LL n , LL a , LL b , LL m ) {
    //count :for(i=0;i<n;++i) res+=floor((a+b*i)/m)
    //n,m,a,b>0
    if (b==0) return n*(a/m);
    if (a>=m) return n*(a/m)+solve(n,a%m,b,m);
    if (b>=m) return (n-1)*n/2*(b/m)+solve(n,a,b%m,m);
    LL q=(a+b*n)/m;
    return solve(q,(a+b*n)%m,m,b);
}
```

日期公式

```
//0 ~ 6 : SUN ~ SAT
int weekday (int y, int m, int d) {
    if (m < 3) {
        --y; m += 12;
    }
    int c = y / 100,
        w = ((y + y / 4 + c / 4 - 2 * c + 26 * (m + 1) / 10 + d - 1) % 7 + 7) % 7;
    return w;
}
int days (int y, int m, int d) {
    if (m < 3) {
        --y; m += 12;
    }
    return 365 * y + y / 4 - y / 100 + y / 400 + (153 * m + 2) / 5 + d;
}
```

读入优化(JAVA)

```
import java.io.*;
import java.util.*;
import java.math.*;
public class javaIO {
    public static void main(String[] args) {
        InputStream inputStream = System.in;
        OutputStream outputStream = System.out;
        InputReader in = new InputReader(inputStream);
        OutputWriter out = new OutputWriter(outputStream);
        TaskC solver = new TaskC();
        solver.solve(1, in, out);
        out.close();
    }
}
class InputReader {
    private InputStream stream;
    private byte[] buf = new byte[1024];
    private int curChar;
    private int numChars;
    private SpaceCharFilter filter;
    public InputReader(InputStream stream) {
        this.stream = stream;
    }
    public int read() {
        if (numChars == -1)
            throw new InputMismatchException();
        if (curChar >= numChars) {
            curChar = 0;
            try {
                numChars = stream.read(buf);
            } catch (IOException e) {
                throw new InputMismatchException();
            }
            if (numChars <= 0)
                return -1;
        }
        return buf[curChar++];
    }
    public int readInt() {
        int c = read();
        while (isSpaceChar(c))
            c = read();
        int sgn = 1;
        if (c == '-') {
            sgn = -1;
            c = read();
        }
        int res = 0;
        do {
            if (c < '0' || c > '9')
                throw new InputMismatchException();
            res *= 10;
            res += c - '0';
            c = read();
        } while (!isSpaceChar(c));
        return res * sgn;
    }
    public boolean isSpaceChar(int c) {
        if (filter != null)
            return filter.isSpaceChar(c);
        return isWhitespace(c);
    }
    public static boolean isWhitespace(int c) {
        return c == ' ' || c == '\n' || c == '\r' || c == '\t' || c == -1;
    }
    public interface SpaceCharFilter {
        public boolean isSpaceChar(int ch);
    }
}
class OutputWriter {
    private final PrintWriter writer;
    public OutputWriter(OutputStream outputStream) {
        writer = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(outputStream)));
    }
    public OutputWriter(Writer writer) {
        this.writer = new PrintWriter(writer);
    }
    public void print(Object...objects) {
        for (int i = 0; i < objects.length; i++) {
            if (i != 0)
                writer.print(' ');
            writer.print(objects[i]);
        }
    }

    public void printLine(Object...objects) {
        print(objects);
        writer.println();
    }
    public void close() {
        writer.close();
    }
}
```

数学公式和结论

椭圆：

椭圆$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$，其中离心率$e=\frac{c}{a}$， $c=\sqrt{a^2-b^2}$;焦点参数$p=\frac{b^2}{a}$

椭圆上(x,y)点处的曲率半径为 $R=a^2b^2\left(\frac{x^2}{a^4}+\frac{y^2}{b^4}\right)^{\frac{3}{2}}=\frac{(r_1r_2)^{\frac{3}{2}}}{ab}$,其中r_1和r_2分别为(x,y)与两焦点F_1和F_2的距

离。设点 A 和点 M 的坐标分别为(a, 0)和(x, y)，则 AM 的弧长为

$$\mathrm{L_{AM}} = \mathrm{a}\int_0^{\arccos\frac{x}{a}} \sqrt{1-\mathrm{e}^2\cos^2 t}\,\mathrm{dt} = \mathrm{a}\int_{\arccos\frac{x}{a}}^{\frac{\pi}{2}} \sqrt{1-\mathrm{e}^2\sin^2 t}\,\mathrm{dt}$$

椭圆的周长为 $\mathrm{L} = 4\mathrm{a}\int_0^{\frac{\pi}{2}}\sqrt{1-\mathrm{e}^2\sin^2 t}\,\mathrm{dt} = 4\mathrm{aE}(\mathrm{e},\frac{\pi}{2})$ ，其中

$$\mathrm{E}\left(\mathrm{e},\frac{\pi}{2}\right) = \frac{\pi}{2}[1-\left(\frac{1}{2}\right)^2\mathrm{e}^2-\left(\frac{1*3}{2*4}\right)^2\frac{\mathrm{e}^4}{3}-\left(\frac{1*3*5}{2*4*6}\right)^2\frac{\mathrm{e}^6}{5}-\cdots$$

设椭圆上点 M(x, y), N(x, −y), x, y>0, A(a, 0), 原点 O(0, 0)。

扇形 OAM 的面积 $\mathrm{S_{OAM}} = \frac{1}{2}ab\mathrm{arccos}\frac{x}{a}$ 弓形 MAN 的面积$\mathrm{S_{MAN}} = ab\mathrm{arccos}\frac{x}{a} - \mathrm{xy}$

方程，5 个点确定一个圆锥曲线。

θ 为(x, y)点关于椭圆中心的极角，r 为(x, y)到椭圆中心的距离，椭圆极坐标方程:

$$\mathrm{x} = \mathrm{r}\cos\theta\,, \mathrm{y} = \mathrm{r}\sin\theta\text{，其中}\mathrm{r}^2 = \frac{\mathrm{b}^2\mathrm{a}^2}{b^2\cos^2\theta + a^2\sin^2\theta}$$

抛物线

标准方程 $y^2 = 2px$ 曲率半径 $\mathrm{R} = ((\mathrm{p}+2\mathrm{x})\hat{}(3/2))/\mathrm{sqrt}(\mathrm{p})$

弧长：设 M(x, y)是抛物线上一点，则 $L_{OM} = \frac{p}{2}[\sqrt{\frac{2x}{p}\left(1+\frac{2x}{p}\right)} + ln(\sqrt{\frac{2x}{p}} + \sqrt{1+\frac{2x}{p}})]$

弓形面积：设M，D是抛物线上两点，且分居一、四象限。作一条平行于MD且与抛物线相切的直线L。若M到L的距离为h。则有 $S_{MOD} = \frac{2}{3}MD\cdot h$

重心

半径为 r、圆心角为θ的扇形的重心与圆心的距离为(4rsin (θ/2))/3θ

半径为 r、圆心角为θ的圆弧的重心与圆心的距离为(4rsin^3 (θ/2))/(3(θ − sinθ))

椭圆上半部分的重心与圆心的距离为 (4/3π) b

抛物线中弓形 MOD 的重心满足 CQ = (2/5) PQ，P 是直线 L 与抛物线的切点，Q 在 MD 上且 PQ 平行 x 轴。C 是重心。

内心 r = 三角形面积/(p = 1/2(a + b + c)) I = (aA + bB + cC)/(a + b + c)

三重积公式 $a\times(b\times c) = b(a\cdot c) - c(a\cdot b)$

额外的公式

四边形: D1,D2 为对角线,M 对角线中点连线,A 为对角线夹角

1.a^2+b^2+c^2+d^2=D1^2+D2^2+4M^2 2. S=D1D2sin(A)/2

(以下对圆的内接四边形)

3. ac+bd=D1D2 4.S=sqrt((P-a)(P-b)(P-c)(P-d)),P 为半周长

*正 n 边形:*R 为外接圆半径,r 为内切圆半径

1. 中心角 A=2PI/n 2. 内角 C=(n-2)PI/n

3. 边长 a=2sqrt(R^2-r^2)=2Rsin(A/2)=2rtan(A/2)

4. 面积 S=nar/2=nr^2tan(A/2)=nR^2sin(A)/2=na^2/(4tan(A/2))

圆: 1. 弧长 l=rA 2. 弦长 a=2sqrt(2hr-h^2)=2rsin(A/2)

3. 弓形高 h=r-sqrt(r^2-a^2/4)=r(1-cos(A/2))=atan(A/4)/2

4.扇形面积 S1=rl/2=r^2A/2

5.弓形面积 S2=(rl-a(r-h))/2=r^2(A-sin(A))/2

棱柱: 1. 体积 V=Ah,A 为底面积,h 为高

2. 侧面积 S=lp,l 为棱长,p 为直截面周长 3. 全面积 T=S+2A

棱锥: 1.体积 V=Ah/3,A 为底面积,h 为高 (以下对正棱锥)

2. 侧面积 S=lp/2,l 为斜高,p 为底面周长 3. 全面积 T=S+A

*棱台:*1. 体积 V=(A1+A2+sqrt(A1A2))h/3,A1.A2 为上下底面积,h 为高

(以下为正棱台)

2. 侧面积 S=(p1+p2)l/2,p1.p2 为上下底面周长,l 为斜高

3. 全面积 T=S+A1+A2

有根树的计数

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

于是，n+1 个结点的有根树的总数为 $\mathrm{a_{n+1}} = \frac{\sum_{1\le j\le n} ja_jS_{n,j}}{n}$

附：$\mathrm{a_1} = 1, a_2 = 1, a_3 = 2, a_4 = 4, a_5 = 9\,, a_6 = 20, a_9 = 286\,, a_{11} = 1842$

无根树的计数

当 n 是奇数时，则有 $a_n - \sum_{1\le i\le n/2} a_ia_{n-i}$ 种不同的无根树。

当 n 是偶数时，则有这么多种不同的无根树。

$$a_n - \sum_{1\le i\le\frac{n}{2}} a_ia_{n-i} + \frac{1}{2}a_{n/2}(a_{n/2}+1)$$

代数

Burnside引理 $\mathrm{ans} = \frac{(\sum\text{每种置换下的不变的元素个数})}{\text{置换群中置换的个数}}$

三次方程求根公式 $\mathrm{x}^3 + px + q = 0$

$$\mathrm{x_j} = \omega^j\sqrt[3]{-\frac{q}{2}+\sqrt{\left(\frac{q}{2}\right)^2+\left(\frac{p}{3}\right)^3}} + \omega^{2j}\sqrt[3]{-\frac{q}{2}-\sqrt{\left(\frac{q}{2}\right)^2+\left(\frac{p}{3}\right)^3}}$$

其中 j=0, 1, 2, $\omega = (-1+i\sqrt{3})/2$

当求解$\mathrm{ax}^3 + bx^2 + cx + d = 0$ 时， 令 x = y − b/3a 再求解y，即转化成 $\mathrm{x}^3 + px + q = 0$的形式

组合公式

$\sum_{\mathrm{k}=1}^n(2k-1)^2 = \frac{n(4n^2-1)}{3}$ $\sum_{\mathrm{k}=1}^n k^3 = \left(\frac{n(n+1)}{2}\right)^2$

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

$\sum_{\mathrm{k}=1}^n k^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$ $\sum_{\mathrm{k}=1}^n k(k+1) = \frac{n(n+1)(n+2)}{3}$

$\sum_{\mathrm{k}=1}^n k(k+1)(k+2) = \frac{n(n+1)(n+2)(n+3)}{4}$

$\sum_{\mathrm{k}=1}^n k(k+1)(k+2)(k+3) = \frac{n(n+1)(n+2)(n+3)(n+4)}{5}$

错排：$\mathrm{D_n} = n!\,(1-\frac{1}{1!}+\frac{1}{2!}-\frac{1}{3!}+\cdots+\frac{(-1)^n}{n!} = (n-1)(D_{n-2}-D_{n-1})$

三角公式

$\sin(\alpha\pm\beta) = sin\alpha\, cos\beta \pm \cos\alpha\, sin\beta$ $\cos(\alpha\pm\beta) = cos\alpha\, cos\beta \mp sin\alpha\, sin\beta$

$\tan(\alpha\pm\beta) = \frac{\tan(\alpha)\pm\tan(\beta)}{1\mp\tan(\alpha)\tan(\beta)}$ $\tan(\alpha)\pm\tan(\beta) = \frac{\sin(\alpha\pm\beta)}{\cos(\alpha)\cos(\beta)}$

$\sin(\alpha)+\sin(\beta) = 2\sin\frac{(\alpha+\beta)}{2}\ cos\frac{(\alpha-\beta)}{2}$ $\sin(\alpha)-\sin(\beta) = 2\cos\frac{(\alpha+\beta)}{2}\ sin\frac{(\alpha-\beta)}{2}$

$\cos(\alpha)+cos(\beta) = 2\cos\frac{(\alpha+\beta)}{2}\ cos\frac{(\alpha-\beta)}{2}$ $\cos(\alpha)-cos(\beta) = -2\sin\frac{(\alpha+\beta)}{2}\ sin\frac{(\alpha-\beta)}{2}$

$$\sin(\mathrm{n}\alpha) = ncos^{n-1}\alpha\, sin\alpha - \binom{n}{3}\cos^{\mathrm{n}-3}\alpha\sin^3\alpha + \binom{n}{5}\cos^{\mathrm{n}-5}\alpha\sin^5\alpha - \cdots$$

$$\cos(\mathrm{n}\alpha) = cos^n\alpha - \binom{n}{2}\cos^{\mathrm{n}-2}\alpha\sin^2\alpha + \binom{n}{4}\cos^{\mathrm{n}-4}\alpha\sin^4\alpha - \cdots$$

积分表

$(\arcsin x)' = \frac{1}{\sqrt{1-x^2}}$	$(\arccos x)' = -\frac{1}{\sqrt{1-x^2}}$	$(\arctan x)' = \frac{1}{1+x^2}$
$a\textasciicircum x \to a\textasciicircum x/lna$	$sinx \to -cosx$	$\cos x \to \sin x$
$tanx \to -lncosx$	$\sec x \to \ln\tan(x/2+\pi/4)$	$\tan^2 x \to tanx - x$
$cscx \to lntan\frac{x}{2}$	$\sin^2 x \to \frac{x}{2} - \frac{1}{2}sinxcosx$	$\cos^2 x \to \frac{x}{2} + \frac{1}{2}sinxcosx$
$\sec^2 x \to tanx$	$\frac{1}{\sqrt{a^2-x^2}} \to \arcsin\left(\frac{x}{a}\right)$	$csc^2x \to -cotx$

$\frac{1}{a^2-x^2}(\lvert x\rvert < \lvert a\rvert) \to \frac{1}{2a}\ln\frac{(a+x)}{a-x}$	$\frac{1}{x^2-a^2}(\lvert x\rvert > \lvert a\rvert) \to \frac{1}{2a}\ln\frac{(x-a)}{x+a}$
$\sqrt{a^2-x^2} \to \frac{x}{2}\sqrt{a^2-x^2} + \frac{a^2}{2}\arcsin\frac{x}{a}$	$\frac{1}{\sqrt{x^2+a^2}} \to \ln\left(x+\sqrt{a^2+x^2}\right)$
$\sqrt{a^2+x^2} \to \frac{x}{2}\sqrt{a^2+x^2} + \frac{a^2}{2}\ln\left(x+\sqrt{a^2+x^2}\right)$	$\frac{1}{\sqrt{x^2-a^2}} \to \ln\left(x+\sqrt{x^2-a^2}\right)$
$\sqrt{x^2-a^2} \to \frac{x}{2}\sqrt{x^2-a^2} - \frac{a^2}{2}\ln\left(x+\sqrt{x^2-a^2}\right)$	$\frac{1}{x\sqrt{a^2-x^2}} \to -\frac{1}{a}ln\frac{a+\sqrt{a^2-x^2}}{x}$
$\frac{1}{x\sqrt{x^2-a^2}} \to \frac{1}{a}\arccos\frac{a}{x}$	$\frac{1}{x\sqrt{a^2+x^2}} \to -\frac{1}{a}ln\frac{a+\sqrt{a^2+x^2}}{x}$
$\frac{1}{\sqrt{2ax-x^2}} \to \arccos(1-\frac{x}{a})$	$\frac{x}{ax+b} \to \frac{x}{a} - \frac{b}{a^2}\ln(ax+b)$
$\sqrt{2ax-x^2} \to \frac{x-a}{2}\sqrt{2ax-x^2} + \frac{a^2}{2}\arcsin(\frac{x}{a}-1)$	
$\frac{1}{x\sqrt{ax+b}}(b<0) \to \frac{2}{\sqrt{-b}}\arctan\sqrt{\frac{ax+b}{-b}}$	$x\sqrt{ax+b} \to \frac{2(3ax-2b)}{15a^2}(ax+b)^{\frac{3}{2}}$
$\frac{1}{x\sqrt{ax+b}}(b>0) \to \frac{1}{\sqrt{-b}}ln\frac{\sqrt{ax+b}-\sqrt{b}}{\sqrt{ax+b}+\sqrt{b}}$	$\frac{x}{\sqrt{ax+b}} \to \frac{2(ax-2b)}{3a^2}\sqrt{ax+b}$
$\frac{1}{x^2\sqrt{ax+b}} \to -\frac{\sqrt{ax+b}}{bx} - \frac{a}{2b}\int\frac{dx}{x\sqrt{ax+b}}$	$\frac{\sqrt{ax+b}}{x} \to 2\sqrt{ax+b} + \mathrm{b}\int\frac{dx}{x\sqrt{ax+b}}$
$\frac{1}{\sqrt{(ax+b)^n}}(n>2) \to \frac{-2}{a(n-2)}\cdot\frac{1}{\sqrt{(ax+b)^{n-2}}}$	
$\frac{1}{ax^2+c}(a>0,c>0) \to \frac{1}{\sqrt{ac}}\arctan(x\sqrt{\frac{a}{c}})$	$\frac{x}{ax^2+c} \to \frac{1}{2a}\ln(ax^2+c)$
$\frac{1}{ax^2+c}(a+,c-) \to \frac{1}{2\sqrt{-ac}}ln\frac{x\sqrt{a}-\sqrt{-c}}{x\sqrt{a}+\sqrt{-c}}$	$\frac{1}{x(ax^2+c)} \to \frac{1}{2c}\ln\frac{x^2}{ax^2+c}$
$\frac{1}{ax^2+c}(a-,c+) \to \frac{1}{2\sqrt{-ac}}ln\frac{\sqrt{c}+x\sqrt{-a}}{\sqrt{c}-x\sqrt{-a}}$	$x\sqrt{ax^2+c} \to \frac{1}{3a}\sqrt{(ax^2+c)^3}$
$\frac{1}{(ax^2+c)^n}(n>1) \to \frac{x}{2c(n-1)(ax^2+c)^{n-1}} + \frac{2n-3}{2c(n-1)}\int\frac{dx}{(ax^2+c)^{n-1}}$	
$\frac{x^n}{ax^2+c}(n \neq 1) \to \frac{x^{n-1}}{a(n-1)} - \frac{c}{a}\int\frac{x^{n-2}}{ax^2+c}dx$	$\frac{1}{x^2(ax^2+c)} \to \frac{-1}{cx} - \frac{a}{c}\int\frac{dx}{ax^2+c}$
$\frac{1}{x^2(ax^2+c)^n}(n \geq 2) \to \frac{1}{c}\int\frac{dx}{x^2(ax^2+c)^{n-1}} - \frac{a}{c}\int\frac{\mathrm{dx}}{(ax^2+c)^n}$	

$\sqrt{ax^2+c}(a>0) \to \frac{x}{2}\sqrt{ax^2+c} + \frac{\mathrm{c}}{2\sqrt{a}}\ln\left(x\sqrt{a}+\sqrt{ax^2+c}\right)$		
$\sqrt{ax^2+c}(a<0) \to \frac{x}{2}\sqrt{ax^2+c} + \frac{\mathrm{c}}{2\sqrt{-a}}\arcsin\left(x\sqrt{\frac{-a}{c}}\right)$		$\frac{1}{\sqrt{ax^2+c}}(a<0)$ $\to \frac{1}{\sqrt{-a}}\arcsin\left(x\sqrt{-\frac{a}{c}}\right)$
$\frac{1}{\sqrt{ax^2+c}}(a>0) \to \frac{1}{\sqrt{a}}\ln\left(x\sqrt{a}+\sqrt{ax^2+c}\right)$		
$\sin^2 ax \to \frac{x}{2} - \frac{1}{4a}\sin 2ax$	$\cos^2 ax \to \frac{x}{2} + \frac{1}{4a}\sin 2ax$	$\frac{1}{\sin ax} \to \frac{1}{a}\ln\tan\frac{ax}{2}$
$\frac{1}{\cos^2 ax} \to \frac{1}{a}\tan ax$	$\frac{1}{\cos ax} \to \frac{1}{a}\ln\tan\left(\frac{\pi}{4}+\frac{ax}{2}\right)$	$\ln(ax) \to xln(ax) - x$
$\sin^3 ax \to \frac{-1}{a}\cos ax + \frac{1}{3a}\cos^3 ax$	$\cos^3 ax \to \frac{1}{a}\sin ax - \frac{1}{3a}\sin^3 ax$	
$\frac{1}{\sin^2 ax} \to -\frac{1}{a}\cot ax$	$xln(ax) \to \frac{x^2}{2}\ln(ax) - \frac{x^2}{4}$	$\cos ax \to \frac{1}{a}\sin ax$
$x^2e^{ax} \to \frac{e^{ax}}{a^3}(a^2x^2-2ax+2)$	$(\ln(ax))^2 \to x(\ln(ax))^2 - 2xln(ax) + 2x$	
$x^2\ln(ax) \to \frac{x^3}{3}\ln(ax) - \frac{x^3}{9}$	$x^n\ln(ax) \to \frac{\mathrm{x}^{\mathrm{n+1}}}{n+1}\ln(ax) - \frac{x^{n+1}}{(n+1)^2}$	
$\sin(\ln ax) \to \frac{x}{2}[\sin(\ln ax) - \cos(\ln ax)]$	$\cos(\ln ax) \to \frac{x}{2}[\sin(\ln ax) + \cos(\ln ax)]$	

.vimrc

```
syntax on
set mp=g++\ -O2\ -Wall\ -Wno-unused-result\ %:r.cpp\ -o\ %:r
set si nu sw=4 ts=4
nmap <F2> :vs %:r.in <CR>
autocmd filetype cpp nmap <F5> :!time ./%:r < %:r.in <CR>
autocmd filetype cpp nmap <F8> :!./%:r <CR>
autocmd filetype cpp nmap <F9> :w <CR> :make<CR>

autocmd BufNewFile *.cpp silent! 0r ~/MahoushojoMiracle/Templates/starter.cpp

autocmd filetype java nmap <F5> :!time java %:r < %:r.in <CR>
autocmd filetype java nmap <F8> :!java %:r <CR>
autocmd filetype java nmap <F9> :w <CR> :!javac %:r.java <CR>

colo evening
```