Contents

1 计算几何

1.1 二维计算几何基本操作

```cpp
const double PI = 3.1415926535897932384626433832795028 8;
double arcSin(const double &a) { return a <= -1.0 ? -PI / 2 : (a >= 1.0 ? PI / 2 : asin(a)); }
double arcCos(const double &a) { return a <= -1.0 ? PI : (a >= 1.0 ? 0 : acos(a)); }
struct point {
    double x, y; // `something omitted`
    point rot(const double &a) const { // `counter-clockwise`
        return point(x * cos(a) - y * sin(a), x * sin(a) + y * cos(a));}
    point rot90() const { // `counter-clockwise`
        return point(-y, x);}
    point project(const point &p1, const point &p2) const {
        const point &q = *this;
        return p1 + (p2 - p1) * (dot(p2 - p1, q - p1) / (p2 - p1).norm());}
    bool onSeg(const point &a, const point &b) const { // `a, b inclusive`
        const point &c = *this;
        return sign(dot(a - c, b - c)) <= 0 && sign(det(b - a, c - a)) == 0;}
    double distLP(const point &p1, const point &p2) const { // `dist from *this to line p1->p2`
        const point &q = *this;
        return fabs(det(p2 - p1, q - p1)) / (p2 - p1).len(); }
    double distSP(const point &p1, const point &p2) const { // `dist from *this to segment [p1, p2]`
        const point &q = *this;
        if (dot(p2 - p1, q - p1) < EPS) return (q - p1).len();
        if (dot(p1 - p2, q - p2) < EPS) return (q - p2).len();
        return distLP(p1, p2); }
    bool inAngle(const point &p1, const point &p2) const { // `det(p1, p2) $\ge$ 0`
        const point &q = *this; return det(p1, q) > -EPS && det(p2, q) < EPS;}
};
bool lineIntersect(const point &a, const point &b, const point &c, const point &d, point &e) {
    double s1 = det(c - a, d - a), s2 = det(d - b, c - b);
    if (!sign(s1 + s2)) return false;
    e = (b - a) * (s1 / (s1 + s2)) + a; return true; }
int segIntersectCheck(const point &a, const point &b, const point &c, const point &d, point &o) {
    static double s1, s2, s3, s4; static int iCnt;
    int d1 = sign(s1 = det(b - a, c - a)), d2 = sign(s2 = det(b - a, d - a));
    int d3 = sign(s3 = det(d - c, a - c)), d4 = sign(s4 = det(d - c, b - c));
    if ((d1 ^ d2) == -2 && (d3 ^ d4) == -2) {
        o = (c * s2 - d * s1) / (s2 - s1); return true; }
    iCnt = 0;
    if (d1 == 0 && c.onSeg(a, b)) o = c, ++iCnt;
    if (d2 == 0 && d.onSeg(a, b)) o = d, ++iCnt;
    if (d3 == 0 && a.onSeg(c, d)) o = a, ++iCnt;
    if (d4 == 0 && b.onSeg(c, d)) o = b, ++iCnt;
    return iCnt ? 2 : 0; // `不相交返回0, 严格相交返回1, 非严格相交返回2`
}
struct circle {
    point o; double r, rSqure;
    bool inside(const point &a) { // `非严格`
        return (a - o).len() < r + EPS; }
    bool contain(const circle &b) const { // `非严格`
        return sign(b.r + (o - b.o).len() - r) <= 0; }
    bool disjunct(const circle &b) const { // `非严格`
        return sign(b.r + r - (o - b.o).len()) <= 0; }
    int isCL(const point &p1, const point &p2, point &a, point &b) const {
        double x = dot(p1 - o, p2 - p1), y = (p2 - p1).norm();
        double d = x * x - y * ((p1 - o).norm() - rSqure);
        if (d < -EPS) return 0; if (d < 0) d = 0;
        point q1 = p1 - (p2 - p1) * (x / y);
        point q2 = (p2 - p1) * (sqrt(d) / y);
        a = q1 - q2; b = q1 + q2;
        return q2.len() < EPS ? 1 : 2; }
    int tanCP(const point &p, point &a, point &b) const { // `返回切点, 注意可能与 $p$ 重合`
        double x = (p - o).norm(), d = x - rSqure;
        if (d < -EPS) return 0; if (d < 0) d = 0;
        point q1 = (p - o) * (rSqure / x);
        point q2 = ((p - o) * (-r * sqrt(d) / x)).rot90();
        a = o + (q1 - q2); b = o + (q1 + q2);
        return q2.len() < EPS ? 1 : 2; }};
bool checkCrossCS(const circle &cir, const point &p1, const point &p2) { // `非严格`
    const point &c = cir.o;
    const double &r = cir.r;
    return c.distSP(p1, p2) < r + EPS && (r < (c - p1).len() + EPS || r < (c - p2).len() + EPS); }
bool checkCrossCC(const circle &cir1, const circle &cir2) { // `非严格`
    const double &r1 = cir1.r, &r2 = cir2.r, d = (cir1.o - cir2.o).len();
    return d < r1 + r2 + EPS && fabs(r1 - r2) < d + EPS; }
int isCC(const circle &cir1, const circle &cir2, point &a, point &b) {
    const point &c1 = cir1.o, &c2 = cir2.o;
    double x = (c1 - c2).norm(), y = ((cir1.rSqure - cir2.rSqure) / x + 1) / 2;
    double d = cir1.rSqure / x - y * y;
    if (d < -EPS) return 0; if (d < 0) d = 0;
    point q1 = c1 + (c2 - c1) * y;
    point q2 = ((c2 - c1) * sqrt(d)).rot90();
    a = q1 - q2; b = q1 + q2;
    return q2.len() < EPS ? 1 : 2; }
vector<pair<point, point> > tanCC(const circle &cir1, const circle &cir2) {
    // `注意: 如果只有三条切线, 即 $s1 = 1, s2 = 1$, 返回的切线可能重复, 切点没有问题`
    vector<pair<point, point> > list;
    if (cir1.contain(cir2) || cir2.contain(cir1)) return list;
    const point &c1 = cir1.o, &c2 = cir2.o;
    double r1 = cir1.r, r2 = cir2.r;
    point p, a1, b1, a2, b2; int s1, s2;
    if (sign(r1 - r2) == 0) {
        p = c2 - c1; p = (p * (r1 / p.len())).rot90();
        list.push_back(make_pair(c1 + p, c2 + p)); list.push_back(make_pair(c1 - p, c2 - p));
    } else {
        p = (c2 * r1 - c1 * r2) / (r1 - r2);
        s1 = cir1.tanCP(p, a1, b1); s2 = cir2.tanCP(p, a2, b2);
        if (s1 >= 1 && s2 >= 1) {
            list.push_back(make_pair(a1, a2)); list.push_back(make_pair(b1, b2)); } }
    p = (c1 * r2 + c2 * r1) / (r1 + r2);
    s1 = cir1.tanCP(p, a1, b1); s2 = cir2.tanCP(p, a2, b2);
    if (s1 >= 1 && s2 >= 1) {
        list.push_back(make_pair(a1, a2)); list.push_back(make_pair(b1, b2)); }
    return list; }
bool distConvexPIn(const point &p1, const point &p2, const point &p3, const point &p4, const point &q) {
    point o12 = (p1 - p2).rot90(), o23 = (p2 - p3).rot90(), o34 = (p3 - p4).rot90();
    return (q - p1).inAngle(o12, o23) || (q - p3).inAngle(o23, o34)
        || ((q - p2).inAngle(o23, p3 - p2) && (q - p3).inAngle(p2 - p3, o23));
}
double distConvexP(int n, point ps[], const point &q) { // `外部点到多边形的距离`
    int left = 0, right = n;
    while (right - left > 1) {
        int mid = (left + right) / 2;
        if (distConvexPIn(ps[(left + n - 1) % n], ps[left], ps[mid], ps[(mid + 1) % n], q)) right = mid;
        else left = mid; }
    return q.distSP(ps[left], ps[right % n]); }
double areaCT(const circle &cir, point pa, point pb) {
    pa = pa - cir.o; pb = pb - cir.o; double R = cir.r;
    if (pa.len() < pb.len()) swap(pa, pb);
    if (pb.len() < EPS) return 0; point pc = pb - pa;
    double a = pa.len(), b = pb.len(), c = pc.len(), S, h, theta;
    double cosB = dot(pb, pc) / b / c, B = acos(cosB);
    double cosC = dot(pa, pb) / a / b, C = acos(cosC);
    if (b > R) {
        S = C * 0.5 * R * R; h = b * a * sin(C) / c;
        if (h < R && B < PI * 0.5)
            S -= acos(h / R) * R * R - h * sqrt(R * R - h * h);
    } else if (a > R) {
        theta = PI - B - asin(sin(B) / R * b);
        S = 0.5 * b * R * sin(theta) + (C - theta) * 0.5 * R * R;
    } else S = 0.5 * sin(C) * b * a;
    return S; }
circle minCircle(const point &a, const point &b) {
    return circle((a + b) * 0.5, (b - a).len() * 0.5); }
circle minCircle(const point &a, const point &b, const point &c) { // `钝角三角形没有被考虑`
    double a2( (b - c).norm() ), b2( (a - c).norm() ), c2( (a - b).norm() );
    if (b2 + c2 <= a2 + EPS) return minCircle(b, c);
    if (a2 + c2 <= b2 + EPS) return minCircle(a, c);
    if (a2 + b2 <= c2 + EPS) return minCircle(a, b);
    double A = 2.0 * (a.x - b.x), B = 2.0 * (a.y - b.y);
    double D = 2.0 * (a.x - c.x), E = 2.0 * (a.y - c.y);
    double C = a.norm() - b.norm(), F = a.norm() - c.norm();
```

```
    point p((C * E - B * F) / (A * E - B * D), (A * F - C * D) / (A * E - B * D));
    return circle(p, (p - a).len()); }
circle minCircle(point P[], int N) { // `1-based`
    if (N == 1) return circle(P[1], 0.0);
    random_shuffle(P + 1, P + N + 1); circle O = minCircle(P[1], P[2]);
    Rep(i, 1, N) if(!O.inside(P[i])) { O = minCircle(P[1], P[i]);
        Foru(j, 1, i) if(!O.inside(P[j])) { O = minCircle(P[i], P[j]);
            Foru(k, 1, j) if(!O.inside(P[k])) O = minCircle(P[i], P[j], P[k]); }
    } return O; }
```

1.2 圆的面积模板

```
struct Event { point p; double alpha; int add; // `构造函数省略`
    bool operator < (const Event &other) const { return alpha < other.alpha; } };
void circleKCover(circle *c, int N, double *area) { // `$area[k]$ : 至少被覆盖$k$次`
    static bool overlap[MAXN][MAXN], g[MAXN][MAXN];
    Rep(i, 0, N + 1) area[i] = 0.0; Rep(i, 1, N) Rep(j, 1, N) overlap[i][j] = c[i].contain(c[j]);
    Rep(i, 1, N) Rep(j, 1, N) g[i][j] = !(overlap[i][j] || overlap[j][i] || c[i].disjunct(c[j]));
    Rep(i, 1, N) { static Event events[MAXN * 2 + 1]; int totE = 0, cnt = 1;
        Rep(j, 1, N) if (j != i && overlap[j][i]) ++cnt;
        Rep(j, 1, N) if (j != i && g[i][j]) {
            circle &a = c[i], &b = c[j]; double l = (a.o - b.o).norm();
            double s = ((a.r - b.r) * (a.r + b.r) / l + 1) * 0.5;
            double t = sqrt(-(l - sqr(a.r - b.r)) * (l - sqr(a.r + b.r)) / (l * l * 4.0));
            point dir = b.o - a.o, nDir = point(-dir.y, dir.x);
            point aa = a.o + dir * s + nDir * t;
            point bb = a.o + dir * s - nDir * t;
            double A = atan2(aa.y - a.o.y, aa.x - a.o.x);
            double B = atan2(bb.y - a.o.y, bb.x - a.o.x);
            events[totE++] = Event(bb, B, 1); events[totE++] = Event(aa, A, -1); if (B > A) ++cnt;
        } if (totE == 0) { area[cnt] += PI * c[i].rSquare; continue; }
        sort(events, events + totE); events[totE] = events[0];
        Foru(j, 0, totE) {
            cnt += events[j].add; area[cnt] += 0.5 * det(events[j].p, events[j + 1].p);
            double theta = events[j + 1].alpha - events[j].alpha; if (theta < 0) theta += 2.0 * PI;
            area[cnt] += 0.5 * c[i].rSquare * (theta - sin(theta));
        }}}
```

1.3 多边形相关

```
struct Polygon { // stored in [0, n)
    int n; point ps[MAXN];
    Polygon cut(const point &a, const point &b) {
        static Polygon res; static point o; res.n = 0;
        for (int i = 0; i < n; ++i) {
            int s1 = sign(det(ps[i] - a, b - a));
            int s2 = sign(det(ps[(i + 1) % n] - a, b - a));
            if (s1 <= 0) res.ps[res.n++] = ps[i];
            if (s1 * s2 < 0) {
                lineIntersect(a, b, ps[i], ps[(i + 1) % n], o);
                res.ps[res.n++] = o;
            }
        } return res;
    }
    bool contain(const point &p) const { // 1 if on border or inner, 0 if outter
        static point A, B; int res = 0;
        for (int i = 0; i < n; ++i) {
            A = ps[i]; B = ps[(i + 1) % n];
            if (p.onSeg(A, B)) return 1;
            if (sign(A.y - B.y) <= 0) swap(A, B);
            if (sign(p.y - A.y) > 0) continue;
            if (sign(p.y - B.y) <= 0) continue;
            res += (int)(sign(det(B - p, A - p)) > 0);
        } return res & 1;
    }
#define qs(x) (ps[x] - ps[0])
    bool convexContain(point p) const { // `counter-clockwise`
        point q = qs(n - 1); p = p - ps[0];
        if (!p.inAngle(qs(1), q)) return false;
        int L = 0, R = n - 1;
        while (L + 1 < R) { int M((L + R) >> 1);
            if (p.inAngle(qs(M), q)) L = M; else R = M;
        } if (L == 0) return false; point l(qs(L)), r(qs(R));
        return sign( fabs(det(l, p)) + fabs(det(p, r)) + fabs(det(r - l, p - l)) - det(l, r) ) == 0;
    }
#undef qs
    double isPLAtan2(const point &a, const point &b) {
        double k = (b - a).alpha(); if (k < 0) k += 2 * PI;
        return k;
    }
    point isPL_Get(const point &a, const point &b, const point &s1, const point &s2) {
        double k1 = det(b - a, s1 - a), k2 = det(b - a, s2 - a);
        if (sign(k1) == 0) return s1;
        if (sign(k2) == 0) return s2;
        return (s1 * k2 - s2 * k1) / (k2 - k1);
    }
    int isPL_Dic(const point &a, const point &b, int l, int r) {
        int s = (det(b - a, ps[l] - a) < 0) ? -1 : 1;
        while (l <= r) {
            int mid = (l + r) / 2;
            if (det(b - a, ps[mid] - a) * s <= 0) r = mid - 1;
            else l = mid + 1;
        }
        return r + 1;
    }
    int isPL_Find(double k, double w[]) {
        if (k <= w[0] || k > w[n - 1]) return 0;
        int l = 0, r = n - 1, mid;
        while (l <= r) {
            mid = (l + r) / 2;
            if (w[mid] >= k) r = mid - 1;
            else l = mid + 1;
        } return r + 1;
    }
    bool isPL(const point &a, const point &b, point &cp1, point &cp2) {  // `$O (log N)$`
        static double w[MAXN * 2]; // `pay attention to the array size`
        for (int i = 0; i <= n; ++i) ps[i + n] = ps[i];
        for (int i = 0; i < n; ++i) w[i] = w[i + n] = isPLAtan2(ps[i], ps[i + 1]);
        int i = isPL_Find(isPLAtan2(a, b), w);
        int j = isPL_Find(isPLAtan2(b, a), w);
        double k1 = det(b - a, ps[i] - a), k2 = det(b - a, ps[j] - a);
        if (sign(k1) * sign(k2) > 0) return false; // `no intersection`
        if (sign(k1) == 0 || sign(k2) == 0) { // `intersect with a point or a line in the convex`
            if (sign(k1) == 0) {
                if (sign(det(b - a, ps[i + 1] - a)) == 0) cp1 = ps[i], cp2 = ps[i + 1];
                else cp1 = cp2 = ps[i];
                return true;
            }
            if (sign(k2) == 0) {
                if (sign(det(b - a, ps[j + 1] - a)) == 0) cp1 = ps[j], cp2 = ps[j + 1];
                else cp1 = cp2 = ps[j];
            }
            return true;
        }
        if (i > j) swap(i, j);
        int x = isPL_Dic(a, b, i, j), y = isPL_Dic(a, b, j, i + n);
        cp1 = isPL_Get(a, b, ps[x - 1], ps[x]);
        cp2 = isPL_Get(a, b, ps[y - 1], ps[y]);
        return true;
    }
    double getI(const point &O) const {
        if (n <= 2) return 0;
        point G(0.0, 0.0);
        double S = 0.0, I = 0.0;
        for (int i = 0; i < n; ++i) {
            const point &x = ps[i], &y = ps[(i + 1) % n];
            double d = det(x, y);
            G = G + (x + y) * d / 3.0;
            S += d;
        } G = G / S;
        for (int i = 0; i < n; ++i) {
            point x = ps[i] - G, y = ps[(i + 1) % n] - G;
            I += fabs(det(x, y)) * (x.norm() + dot(x, y) + y.norm());
        }
```

```
        return I = I / 12.0 + fabs(S * 0.5) * (O - G).norm();
    }
};
```

1.4 直线与凸包求交点

```
int isPL(point a, point b, vector<point> &res) { // `点逆时针给出，无三点共线`
    static double theta[MAXN];
    for (int i = 0; i < n; ++i) theta[i] = (list[(i + 1) % n] - list[i]).atan2();
    double delta = theta[0];
    for (int i = 0; i < n; ++i) theta[i] = normalize(theta[i] - delta);
    int x = lower_bound(theta, theta + n, normalize((b - a).atan2() - delta)) - theta;
    int y = lower_bound(theta, theta + n, normalize((a - b).atan2() - delta)) - theta;
    for (int k = 0; k <= 1; ++k, swap(a, b), swap(x, y)) {
        if (y < x) y += n;
        int l = x, r = y, m;
        while (l + 1 < r) {
            if (sign(det(b - a, list[(m = (l + r) / 2) % n] - a)) < 0) l = m;
            else r = m;
        }
        l %= n, r %= n;
        if (sign(det(b - a, list[r] - list[l])) == 0) {
            if (sign(det(b - a, list[l] - a)) == 0)
            return -l; // `直线与 $(list[l], list[r])$ 重合`
        }
        else {
            point p; lineIntersect(list[l], list[r], a, b, p);
            if (p.onSeg(list[l], list[r]))
            res.push_back(p);
        }
    }
    return res.size();
}
```

1.5 半平面交

```
struct Border {
    point p1, p2; double alpha;
    Border() : p1(), p2(), alpha(0.0) {}
    Border(const point &a, const point &b): p1(a), p2(b), alpha( atan2(p2.y - p1.y, p2.x - p1.x) ) {}
    bool operator == (const Border &b) const { return sign(alpha - b.alpha) == 0; }
    bool operator < (const Border &b) const {
        int c = sign(alpha - b.alpha); if (c != 0) return c > 0;
        return sign(det(b.p2 - b.p1, p1 - b.p1)) >= 0;
    }
};
point isBorder(const Border &a, const Border &b) { // a and b should not be parallel
    point is; lineIntersect(a.p1, a.p2, b.p1, b.p2, is); return is;
}
bool checkBorder(const Border &a, const Border &b, const Border &me) {
    point is; lineIntersect(a.p1, a.p2, b.p1, b.p2, is);
    return sign(det(me.p2 - me.p1, is - me.p1)) > 0;
}
double HPI(int N, Border border[]) {
    static Border que[MAXN * 2 + 1]; static point ps[MAXN];
    int head = 0, tail = 0, cnt = 0; // [head, tail)
    sort(border, border + N); N = unique(border, border + N) - border;
    for (int i = 0; i < N; ++i) {
        Border &cur = border[i];
        while (head + 1 < tail && !checkBorder(que[tail - 2], que[tail - 1], cur)) --tail;
        while (head + 1 < tail && !checkBorder(que[head], que[head + 1], cur)) ++head;
        que[tail++] = cur;
    } while (head + 1 < tail && !checkBorder(que[tail - 2], que[tail - 1], que[head])) --tail;
    while (head + 1 < tail && !checkBorder(que[head], que[head + 1], que[tail - 1])) ++head;
    if (tail - head <= 2) return 0.0;
    Foru(i, head, tail) ps[cnt++] = isBorder(que[i], que[(i + 1 == tail) ? (head) : (i + 1)]);
    double area = 0; Foru(i, 0, cnt) area += det(ps[i], ps[(i + 1) % cnt]);
    return fabs(area * 0.5); // or (-area * 0.5)
}
```

1.6 最大面积空凸包

```
inline bool toUpRight(const point &a, const point &b) {
    int c = sign(b.y - a.y); if (c > 0) return true;
    return c == 0 && sign(b.x - a.x) > 0;
}
inline bool cmpByPolarAngle(const point &a, const point &b) { // `counter-clockwise, shorter first if they share the same polar angle`
    int c = sign(det(a, b)); if (c != 0) return c > 0;
    return sign(b.len() - a.len()) > 0;
}
double maxEmptyConvexHull(int N, point p[]) {
    static double dp[MAXN][MAXN];
    static point vec[MAXN];
    static int seq[MAXN]; // `empty triangles formed with $(0, 0), vec[o], vec[ seq[i] ]$`
    double ans = 0.0;
    Rep(o, 1, N) {
        int totVec = 0;
        Rep(i, 1, N) if (toUpRight(p[o], p[i])) vec[++totVec] = p[i] - p[o];
        sort(vec + 1, vec + totVec + 1, cmpByPolarAngle);
        Rep(i, 1, totVec) Rep(j, 1, totVec) dp[i][j] = 0.0;
        Rep(k, 2, totVec) {
            int i = k - 1;
            while (i > 0 && sign( det(vec[k], vec[i]) ) == 0) --i;
            int totSeq = 0;
            for (int j; i > 0; i = j) {
                seq[++totSeq] = i;
                for (j = i - 1; j > 0 && sign(det(vec[i] - vec[k], vec[j] - vec[k])) > 0; --j);
                double v = det(vec[i], vec[k]) * 0.5;
                if (j > 0) v += dp[i][j];
                dp[k][i] = v;
                cMax(ans, v);
            } for (int i = totSeq - 1; i >= 1; --i) cMax( dp[k][ seq[i] ], dp[k][seq[i + 1]] );
        }
    } return ans;
}
```

1.7 最近点对

```
int N; point p[maxn];
bool cmpByX(const point &a, const point &b) { return sign(a.x - b.x) < 0;  }
bool cmpByY(const int &a, const int &b) { return p[a].y < p[b].y;  }
double minimalDistance(point *c, int n, int *ys) {
    double ret = 1e+20;
    if (n < 20) {
        Foru(i, 0, n) Foru(j, i + 1, n) cMin(ret, (c[i] - c[j]).len() );
        sort(ys, ys + n, cmpByY); return ret;
    } static int mergeTo[maxn];
    int mid = n / 2; double xmid = c[mid].x;
    ret = min(minimalDistance(c, mid, ys), minimalDistance(c + mid, n - mid, ys + mid));
    merge(ys, ys + mid, ys + mid, ys + n, mergeTo, cmpByY);
    copy(mergeTo, mergeTo + n, ys);
    Foru(i, 0, n) {
        while (i < n && sign(fabs(p[ys[i]].x - xmid) - ret) > 0) ++i;
        int cnt = 0;
        Foru(j, i + 1, n)
            if (sign(p[ys[j]].y - p[ys[i]].y - ret) > 0) break;
            else if (sign(fabs(p[ys[j]].x - xmid) - ret) <= 0) {
                ret = min(ret, (p[ys[i]] - p[ys[j]]).len());
                if (++cnt >= 10) break;
            }
    } return ret;
}
double work() {
    sort(p, p + n, cmpByX); Foru(i, 0, n) ys[i] = i; return minimalDistance(p, n, ys);
}
```

1.8 凸包与点集直径

```
vector<point> convexHull(int n, point ps[]) { // `counter-clockwise, strict`
    static point qs[MAXN * 2];
    sort(ps, ps + n, cmpByXY);
    if (n <= 2) return vector<point>(ps, ps + n);
    int k = 0;
    for (int i = 0; i < n; qs[k++] = ps[i++])
        while (k > 1 && det(qs[k - 1] - qs[k - 2], ps[i] - qs[k - 1]) < EPS) --k;
    for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
        while (k > t && det(qs[k - 1] - qs[k - 2], ps[i] - qs[k - 1]) < EPS) --k;
    return vector<point>(qs, qs + k);
}
double convexDiameter(int n, point ps[]) {
    if (n < 2) return 0; if (n == 2) return (ps[1] - ps[0]).len();
    double k, ans = 0;
    for (int x = 0, y = 1, nx, ny; x < n; ++x) {
        for(nx = (x == n - 1) ? (0) : (x + 1); ; y = ny) {
            ny = (y == n - 1) ? (0) : (y + 1);
            if ( sign(k = det(ps[nx] - ps[x], ps[ny] - ps[y])) <= 0) break;
        } ans = max(ans, (ps[x] - ps[y]).len());
        if (sign(k) == 0) ans = max(ans, (ps[x] - ps[ny]).len());
    } return ans;
}
```

1.9 Farmland

```
struct node { int begin[MAXN], *end; } a[MAXN]; // `按对 $p[i]$ 的极角的 atan2 值排序`
bool check(int n, point p[], int b1, int b2, bool vis[MAXN][MAXN]) {
    static pii l[MAXN * 2 + 1]; static bool used[MAXN];
    int tp(0), *k, p, p1, p2; double area(0.0);
    for (l[0] = pii(b1, b2); ; ) {
        vis[p1 = l[tp].first][p2 = l[tp].second] = true;
        area += det(p[p1], p[p2]);
        for (k = a[p2].begin; k != a[p2].end; ++k) if (*k == p1) break;
        k = (k == a[p2].begin) ? (a[p2].end - 1) : (k - 1);
        if ((l[++tp] = pii(p2, *k)) == l[0]) break;
    } if (sign(area) < 0 || tp < 3) return false;
    Rep(i, 1, n) used[i] = false;
    for (int i = 0; i < tp; ++i) if (used[p = l[i].first]) return false; else used[p] = true;
    return true; // `a face with tp vertices`
}
int countFaces(int n, point p[]) {
    static bool vis[MAXN][MAXN]; int ans = 0;
    Rep(x, 1, n) Rep(y, 1, n) vis[x][y] = false;
    Rep(x, 1, n) for (int *itr = a[x].begin; itr != a[x].end; ++itr) if (!vis[x][*itr])
        if (check(n, p, x, *itr, vis)) ++ans;
    return ans;
}
```

1.10 Voronoi 图

不能有重点, 点数应当不小于 2

```
#define Oi(e) ((e)->oi)
#define Dt(e) ((e)->dt)
#define On(e) ((e)->on)
#define Op(e) ((e)->op)
#define Dn(e) ((e)->dn)
#define Dp(e) ((e)->dp)
#define Other(e, p) ((e)->oi == p ? (e)->dt : (e)->oi)
#define Next(e, p) ((e)->oi == p ? (e)->on : (e)->dn)
#define Prev(e, p) ((e)->oi == p ? (e)->op : (e)->dp)
#define V(p1, p2, u, v) (u = p2->x - p1->x, v = p2->y - p1->y)
#define C2(u1, v1, u2, v2) (u1 * v2 - v1 * u2)
#define C3(p1, p2, p3) ((p2->x - p1->x) * (p3->y - p1->y) - (p2->y - p1->y) * (p3->x - p1->x))
#define Dot(u1, v1, u2, v2) (u1 * u2 + v1 * v2)
#define dis(a,b) (sqrt( (a->x - b->x) * (a->x - b->x) + (a->y - b->y) * (a->y - b->y) ))
const int maxn = 110024;
const int aix = 4;
const double eps = 1e-7;
int n, M, k;
struct gEdge {
    int u, v; double w;
    bool operator <(const gEdge &e1) const { return w < e1.w - eps; }
} E[aix * maxn], MST[maxn];
struct point {
    double x, y; int index; edge *in;
    bool operator <(const point &p1) const { return x < p1.x - eps || (abs(x - p1.x) <= eps && y < p1.y - eps); }
};
struct edge { point *oi, *dt; edge *on, *op, *dn, *dp; };

point p[maxn], *Q[maxn];
edge mem[aix * maxn], *elist[aix * maxn];
int nfree;
void Alloc_memory() { nfree = aix * n; edge *e = mem; for (int i = 0; i < nfree; i++) elist[i] = e++; }
void Splice(edge *a, edge *b, point *v) {
    edge *next;
    if (Oi(a) == v) next = On(a), On(a) = b; else next = Dn(a), Dn(a) = b;
    if (Oi(next) == v) Op(next) = b; else Dp(next) = b;
    if (Oi(b) == v) On(b) = next, Op(b) = a; else Dn(b) = next, Dp(b) = a;
}
edge *Make_edge(point *u, point *v) {
    edge *e = elist[--nfree];
    e->on = e->op = e->dn = e->dp = e; e->oi = u; e->dt = v;
    if (!u->in) u->in = e;
    if (!v->in) v->in = e;
    return e;
}
edge *Join(edge *a, point *u, edge *b, point *v, int side) {
    edge *e = Make_edge(u, v);
    if (side == 1) {
        if (Oi(a) == u) Splice(Op(a), e, u);
        else Splice(Dp(a), e, u);
        Splice(b, e, v);
    } else {
        Splice(a, e, u);
        if (Oi(b) == v) Splice(Op(b), e, v);
        else Splice(Dp(b), e, v);
    } return e;
}
void Remove(edge *e) {
    point *u = Oi(e), *v = Dt(e);
    if (u->in == e) u->in = e->on;
    if (v->in == e) v->in = e->dn;
    if (Oi(e->on) == u) e->on->op = e->op; else e->on->dp = e->op;
    if (Oi(e->op) == u) e->op->on = e->on; else e->op->dn = e->on;
    if (Oi(e->dn) == v) e->dn->op = e->dp; else e->dn->dp = e->dp;
    if (Oi(e->dp) == v) e->dp->on = e->dn; else e->dp->dn = e->dn;
    elist[nfree++] = e;
}
void Low_tangent(edge *e_l, point *o_l, edge *e_r, point *o_r, edge **l_low, point **OL, edge **r_low, point **OR
    ) {
    for (point *d_l = Other(e_l, o_l), *d_r = Other(e_r, o_r); ; )
        if (C3(o_l, o_r, d_l) < -eps)      e_l = Prev(e_l, d_l), o_l = d_l, d_l = Other(e_l, o_l);
        else if (C3(o_l, o_r, d_r) < -eps) e_r = Next(e_r, d_r), o_r = d_r, d_r = Other(e_r, o_r);
        else break;
    *OL = o_l, *OR = o_r; *l_low = e_l, *r_low = e_r;
}
void Merge(edge *lr, point *s, edge *rl, point *u, edge **tangent) {
    double l1, l2, l3, l4, r1, r2, r3, r4, cot_L, cot_R, u1, v1, u2, v2, n1, cot_n, P1, cot_P;
    point *O, *D, *OR, *OL; edge *B, *L, *R;
    Low_tangent(lr, s, rl, u, &L, &OL, &R, &OR);
    for (*tangent = B = Join(L, OL, R, OR, 0), O = OL, D = OR; ; ) {
        edge *El = Next(B, O), *Er = Prev(B, D), *next, *prev;
        point *l = Other(El, O), *r = Other(Er, D);
        V(l, O, l1, l2); V(l, D, l3, l4); V(r, O, r1, r2); V(r, D, r3, r4);
        double cl = C2(l1, l2, l3, l4), cr = C2(r1, r2, r3, r4);
        bool BL = cl > eps, BR = cr > eps;
        if (!BL && !BR) break;
        if (BL) {
            double dl = Dot(l1, l2, l3, l4);
            for (cot_L = dl / cl; ; Remove(El), El = next, cot_L = cot_n) {
                next = Next(El, O); V(Other(next, O), O, u1, v1); V(Other(next, O), D, u2, v2);
                n1 = C2(u1, v1, u2, v2); if (!(n1 > eps)) break;
                cot_n = Dot(u1, v1, u2, v2) / n1;
                if (cot_n > cot_L) break;
```

```cpp
            }
        } if (BR) {
            double dr = Dot(r1, r2, r3, r4);
            for (cot_R = dr / cr; ; Remove(Er), Er = prev, cot_R = cot_P) {
                prev = Prev(Er, D); V(Other(prev, D), O, u1, v1); V(Other(prev, D), D, u2, v2);
                P1 = C2(u1, v1, u2, v2); if (!(P1 > eps)) break;
                cot_P = Dot(u1, v1, u2, v2) / P1;
                if (cot_P > cot_R) break;
            }
        } l = Other(El, O); r = Other(Er, D);
        if (!BL || (BL && BR && cot_R < cot_L)) B = Join(B, O, Er, r, 0), D = r;
        else B = Join(El, l, B, D, 0), O = l;
    }
}
void Divide(int s, int t, edge **L, edge **R) {
    edge *a, *b, *c, *ll, *lr, *rl, *rr, *tangent;
    int n = t - s + 1;
    if (n == 2) *L = *R = Make_edge(Q[s], Q[t]);
    else if (n == 3) {
        a = Make_edge(Q[s], Q[s + 1]), b = Make_edge(Q[s + 1], Q[t]);
        Splice(a, b, Q[s + 1]);
        double v = C3(Q[s], Q[s + 1], Q[t]);
        if (v > eps)       c = Join(a, Q[s], b, Q[t], 0), *L = a, *R = b;
        else if (v < -eps) c = Join(a, Q[s], b, Q[t], 1), *L = c, *R = c;
        else *L = a, *R = b;
    } else if (n > 3) {
        int split = (s + t) / 2;
        Divide(s, split, &ll, &lr); Divide(split + 1, t, &rl, &rr);
        Merge(lr, Q[split], rl, Q[split + 1], &tangent);
        if (Oi(tangent) == Q[s]) ll = tangent;
        if (Dt(tangent) == Q[t]) rr = tangent;
        *L = ll; *R = rr;
    }
}
void Make_Graph() {
    edge *start, *e; point *u, *v;
    for (int i = 0; i < n; i++) {
        start = e = (u = &p[i])->in;
        do{ v = Other(e, u);
            if (u < v) E[M++].u = (u - p, v - p, dis(u, v)); // M < aix * maxn
        } while ((e = Next(e, u)) != start);
    }
}
int b[maxn];
int Find(int x) { while (x != b[x]) { b[x] = b[b[x]]; x = b[x]; } return x; }
void Kruskal() {
    memset(b, 0, sizeof(b)); sort(E, E + M);
    for (int i = 0; i < n; i++) b[i] = i;
    for (int i = 0, kk = 0; i < M && kk < n - 1; i++) {
        int m1 = Find(E[i].u), m2 = Find(E[i].v);
        if (m1 != m2) b[m1] = m2, MST[kk++] = E[i];
    }
}
void solve() {
    scanf("%d", &n);
    for (int i = 0; i < n; i++) scanf("%lf%lf", &p[i].x, &p[i].y), p[i].index = i, p[i].in = NULL;
    Alloc_memory(); sort(p, p + n);
    for (int i = 0; i < n; i++) Q[i] = p + i;
    edge *L, *R; Divide(0, n - 1, &L, &R);
    M = 0; Make_Graph(); Kruskal();
}
int main() { solve(); return 0; }
```

1.11 四边形双费马点

```cpp
typedef complex<double> Tpoint;
const double eps = 1e-8;
const double sqrt3 = sqrt(3.0);
bool cmp(const Tpoint &a, const Tpoint &b) {
    return a.real() < b.real() - eps || (a.real() < b.real() + eps && a.imag() < b.imag());
}
Tpoint rotate(const Tpoint &a, const Tpoint &b, const Tpoint &c) {
    Tpoint d = b - a; d = Tpoint(-d.imag(), d.real());
    if (Sign(cross(a, b, c)) == Sign(cross(a, b, a + d))) d *= -1.0;
    return unit(d);
}
Tpoint p[10], a[10], b[10];
int N, T;
double totlen(const Tpoint &p, const Tpoint &a, const Tpoint &b, const Tpoint &c) {
    return abs(p - a) + abs(p - b) + abs(p - c);
}
double fermat(const Tpoint &x, const Tpoint &y, const Tpoint &z, Tpoint &cp) {
    a[0] = a[3] = x; a[1] = a[4] = y; a[2] = a[5] = z;
    double len = 1e100, len2;
    for (int i = 0; i < 3; i++) {
        len2 = totlen(a[i], x, y, z);
        if (len2 < len) len = len2, cp = a[i];
    }
    for (int i = 0; i < 3; i++) {
        b[i] = rotate(a[i + 1], a[i], a[i + 2]);
        b[i] = (a[i + 1] + a[i]) / 2.0 + b[i] * (abs(a[i + 1] - a[i]) * sqrt3 / 2.0);
    }
    b[3] = b[0];
    Tpoint cp2 = intersect(b[0], a[2], b[1], a[3]);
    len2 = totlen(cp2, x, y, z);
    if (len2 < len) len = len2, cp = cp2;
    return len;
}
double getans(const Tpoint &a) {
    double len = 0; for (int i = 0; i < N; i++) len += abs(a - p[i]);
    return len;
}
double mindist(const Tpoint &p, const Tpoint &a, const Tpoint &b, const Tpoint &c, const Tpoint &d) {
    return min( min(abs(p - a), abs(p - b)), min(abs(p - c), abs(p - d)));
}
int main() {
    N = 4;
    for (cin >> T; T; T--) {
        double ret = 1e100, len_cur, len_before, len1, len2, len;
        Tpoint cp, cp1, cp2;
        Foru(i, 0, N) cin >> p[i];
        Foru(i, 0, N) ret = min(ret, getans(p[i]));
        Foru(i, 1, N) Foru(j, 1, N) if (j != i) Foru(k, 1, N) if (k != i && k != j) {
            cMin(ret, abs(p[0] - p[i]) + abs(p[j] - p[k])
                    + min( min(abs(p[0] - p[j]), abs(p[0] - p[k])),
                           min(abs(p[i] - p[j]), abs(p[i] - p[k]))
                    ));
            ret = min(ret, getans(intersect(p[0], p[i], p[j], p[k])));
        }
        Foru(i, 0, N) Foru(j, i + 1, N) Foru(k, j + 1, N) {
            double len = fermat(p[i], p[j], p[k], cp);
            ret = min(ret, len + mindist(p[6 - i - j - k], p[i], p[j], p[k], cp));
        }
        sort(p, p + N, cmp);
        for (int i = 1; i < N; i++) {
            cp1 = (p[0] + p[i]) / 2.0;
            int j, k;
            for (j = 1; j < N && j == i; j++);
            for (k = 6 - i - j, len_before = 1e100; ; ) {
                len1 = fermat(cp1, p[j], p[k], cp2);
                len1 = fermat(cp2, p[0], p[i], cp1);
                len = len1 + abs(cp2 - p[j]) + abs(cp2 - p[k]);
                if (len < len_before - (1e-6)) len_before = len;
                else break;
            } ret = min(ret, len_before);
        } printf("%.4f\n", ret);
    }
    return 0;
}
```

1.12 三角形和四边形的费马点

- 费马点：距几个顶点距离之和最小的点
- 三角形:

  - 若每个角都小于 120°：以每条边向外作正三角形，得到 ΔABF, ΔBCD, ΔCAE，连接 AD, BE, CF，三线必共点于费马点. 该点对三边的张角必然是 120°，也必然是三个三角形外接圆的交点
  - 否则费马点一定是那个大于等于 120° 的顶角
- 四边形:
  - 在凸四边形中，费马点为对角线的交点
  - 在凹四边形中，费马点位凹顶点

1.13 三维计算几何基本操作

```
struct point { double x, y, z; // something omitted
    friend point det(const point &a, const point &b) {
        return point(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
    }
    friend double mix(const point &a, const point &b, const point &c) {
        return a.x * b.y * c.z + a.y * b.z * c.x + a.z * b.x * c.y - a.z * b.y * c.x - a.x * b.z * c.y - a.y * b.x * c.z;
    }
    double distLP(const point &p1, const point &p2) const {
        return det(p2 - p1, *this - p1).len() / (p2 - p1).len();
    }
    double distFP(const point &p1, const point &p2, const point &p3) const {
        point n = det(p2 - p1, p3 - p1); return fabs( dot(n, *this - p1) / n.len() );
    }
};
double distLL(const point &p1, const point &p2, const point &q1, const point &q2) {
    point p = q1 - p1, u = p2 - p1, v = q2 - q1;
    double d = u.norm() * v.norm() - dot(u, v) * dot(u, v);
    if (sign(d) == 0) return p1.distLP(q1, q2);
    double s = (dot(p, u) * v.norm() - dot(p, v) * dot(u, v)) / d;
    return (p1 + u * s).distLP(q1, q2);
}
double distSS(const point &p1, const point &p2, const point &q1, const point &q2) {
    point p = q1 - p1, u = p2 - p1, v = q2 - q1;
    double d = u.norm() * v.norm() - dot(u, v) * dot(u, v);
    if (sign(d) == 0) return min( min((p1 - q1).len(), (p1 - q2).len()),
                                  min((p2 - q1).len(), (p2 - q2).len()));
    double s1 = (dot(p, u) * v.norm() - dot(p, v) * dot(u, v)) / d;
    double s2 = (dot(p, v) * u.norm() - dot(p, u) * dot(u, v)) / d;
    if (s1 < 0.0) s1 = 0.0; if (s1 > 1.0) s1 = 1.0;
    if (s2 < 0.0) s2 = 0.0; if (s2 > 1.0) s2 = 1.0;
    point r1 = p1 + u * s1; point r2 = q1 + v * s2;
    return (r1 - r2).len();
}
bool isFL(const point &p, const point &o, const point &q1, const point &q2, point &res) {
    double a = dot(o, q2 - p), b = dot(o, q1 - p), d = a - b;
    if (sign(d) == 0) return false;
    res = (q1 * a - q2 * b) / d;
    return true;
}
bool isFF(const point &p1, const point &o1, const point &p2, const point &o2, point &a, point &b) {
    point e = det(o1, o2), v = det(o1, e);
    double d = dot(o2, v); if (sign(d) == 0) return false;
    point q = p1 + v * (dot(o2, p2 - p1) / d);
    a = q; b = q + e;
    return true;
}
```

1.14 凸多面体切割

```
vector<vector<point> > convexCut(const vector<vector<point> > &pss, const point &p, const point &o) {
    vector<vector<point> > res;
    vector<point> sec;
    for (unsigned itr = 0, size = pss.size(); itr < size; ++itr) {
        const vector<point> &ps = pss[itr];
        int n = ps.size();
        vector<point> qs;
        bool dif = false;
        for (int i = 0; i < n; ++i) {
            int d1 = sign( dot(o, ps[i] - p) );
            int d2 = sign( dot(o, ps[(i + 1) % n] - p) );
            if (d1 <= 0) qs.push_back(ps[i]);
            if (d1 * d2 < 0) {
                point q;
                isFL(p, o, ps[i], ps[(i + 1) % n], q); // must return true
                qs.push_back(q);
                sec.push_back(q);
            }
            if (d1 == 0) sec.push_back(ps[i]);
            else dif = true;
            dif |= dot(o, det(ps[(i + 1) % n] - ps[i], ps[(i + 2) % n] - ps[i])) < -EPS;
        }
        if (!qs.empty() && dif)
            res.insert(res.end(), qs.begin(), qs.end());
    }
    if (!sec.empty()) {
        vector<point> tmp( convexHull2D(sec, o) );
        res.insert(res.end(), tmp.begin(), tmp.end());
    }
    return res;
}

vector<vector<point> > initConvex() {
    vector<vector<point> > pss(6, vector<point>(4));
    pss[0][0] = pss[1][0] = pss[2][0] = point(-INF, -INF, -INF);
    pss[0][3] = pss[1][1] = pss[5][2] = point(-INF, -INF,  INF);
    pss[0][1] = pss[2][3] = pss[4][2] = point(-INF,  INF, -INF);
    pss[0][2] = pss[5][3] = pss[4][1] = point(-INF,  INF,  INF);
    pss[1][3] = pss[2][1] = pss[3][2] = point( INF, -INF, -INF);
    pss[1][2] = pss[5][1] = pss[3][3] = point( INF, -INF,  INF);
    pss[2][2] = pss[4][3] = pss[3][1] = point( INF,  INF, -INF);
    pss[5][0] = pss[4][0] = pss[3][0] = point( INF,  INF,  INF);
    return pss;
}
```

1.15 三维凸包

不能有重点

```
namespace ConvexHull3D {
    #define volume(a, b, c, d) (mix(ps[b] - ps[a], ps[c] - ps[a], ps[d] - ps[a]))
    vector<Facet> getHull(int n, point ps[]) {
        static int mark[MAXN][MAXN], a, b, c;
        int stamp = 0;
        bool exist = false;
        vector<Facet> facet;
        random_shuffle(ps, ps + n);
        for (int i = 2; i < n && !exist; i++) {
            point ndir = det(ps[0] - ps[i], ps[1] - ps[i]);
            if (ndir.len() < EPS) continue;
            swap(ps[i], ps[2]);
            for (int j = i + 1; j < n && !exist; j++)
                if (sign(volume(0, 1, 2, j)) != 0) {
                    exist = true;
                    swap(ps[j], ps[3]);
                    facet.push_back(Facet(0, 1, 2));
                    facet.push_back(Facet(0, 2, 1));
                }
        }
        if (!exist) return ConvexHull2D(n, ps);
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < n; ++j)
                mark[i][j] = 0;
        stamp = 0;
        for (int v = 3; v < n; ++v) {
            vector<Facet> tmp;
            ++stamp;
            for (unsigned i = 0; i < facet.size(); i++) {
                a = facet[i].a;
                b = facet[i].b;
                c = facet[i].c;
```

```
                if (sign(volume(v, a, b, c)) < 0)
                    mark[a][b] = mark[a][c] =
                    mark[b][a] = mark[b][c] =
                    mark[c][a] = mark[c][b] = stamp;
                else tmp.push_back(facet[i]);
            } facet = tmp;
            for (unsigned i = 0; i < tmp.size(); i++) {
                a = facet[i].a; b = facet[i].b; c = facet[i].c;
                if (mark[a][b] == stamp) facet.push_back(Facet(b, a, v));
                if (mark[b][c] == stamp) facet.push_back(Facet(c, b, v));
                if (mark[c][a] == stamp) facet.push_back(Facet(a, c, v));
            }
        } return facet;
    }
    #undef volume
}
namespace Gravity {
    using ConvexHull3D::Facet;
    point findG(point ps[], const vector<Facet> &facet) {
        double ws = 0; point res(0.0, 0.0, 0.0), o = ps[ facet[0].a ];
        for (int i = 0, size = facet.size(); i < size; ++i) {
            const point &a = ps[ facet[i].a ], &b = ps[ facet[i].b ], &c = ps[ facet[i].c ];
            point p = (a + b + c + o) * 0.25;
            double w = mix(a - o, b - o, c - o);
            ws += w;
            res = res + p * w;
        } res = res / ws;
        return res;
    }
}
```

1.16 球面点表面点距离

```
double distOnBall(double lati1, double longi1, double lati2, double longi2, double R) {
    lati1 *= PI / 180; longi1 *= PI / 180;
    lati2 *= PI / 180; longi2 *= PI / 180;
    double x1 = cos(lati1) * sin(longi1);
    double y1 = cos(lati1) * cos(longi1);
    double z1 = sin(lati1);
    double x2 = cos(lati2) * sin(longi2);
    double y2 = cos(lati2) * cos(longi2);
    double z2 = sin(lati2);
    double theta = acos(x1 * x2 + y1 * y2 + z1 * z2);
    return R * theta;
}
```

1.17 长方体表面点距离

```
int r;
void turn(int i, int j, int x, int y, int z, int x0, int y0, int L, int W, int H) {
    if (z == 0) r = min(r, x * x + y * y);
    else {
        if (i >= 0 && i < 2)  turn(i + 1, j, x0 + L + z, y, x0 + L - x, x0 + L, y0, H, W, L);
        if (j >= 0 && j < 2)  turn(i, j + 1, x, y0 + W + z, y0 + W - y, x0, y0 + W, L, H, W);
        if (i <= 0 && i > -2) turn(i - 1, j, x0 - z, y, x - x0, x0 - H, y0, H, W, L);
        if (j <= 0 && j > -2) turn(i, j - 1, x, y0 - z, y - y0, x0, y0 - H, L, H, W);
    }
}
int calc(int L, int H, int W, int x1, int y1, int z1, int x2, int y2, int z2) {
    if (z1 != 0 && z1 != H)
        if (y1 == 0 || y1 == W) swap(y1, z1), swap(y2, z2), swap(W, H);
        else                    swap(x1, z1), swap(x2, z2), swap(L, H);
    if (z1 == H) z1 = 0, z2 = H - z2;
    r = INF; turn(0, 0, x2 - x1, y2 - y1, z2, -x1, -y1, L, W, H);
    return r;
}
```

1.18 最小覆盖球

```
int outCnt; point out[4], res; double radius;
void ball() {
    static point q[3];
    static double m[3][3], sol[3], L[3], det;
    int i, j; res = point(0.0, 0.0, 0.0); radius = 0.0;
    switch (outCnt) {
    case 1: res = out[0]; break;
    case 2: res = (out[0] + out[1]) * 0.5; radius = (res - out[0]).norm();
        break;
    case 3:
        q[0] = out[1] - out[0]; q[1] = out[2] - out[0];
        for (i = 0; i < 2; ++i) for (j = 0; j < 2; ++j)
            m[i][j] = dot(q[i], q[j]) * 2.0;
        for (i = 0; i < 2; ++i) sol[i] = dot(q[i], q[i]);
        det = m[0][0] * m[1][1] - m[0][1] * m[1][0];
        if (sign(det) == 0) return;
        L[0] = (sol[0] * m[1][1] - sol[1] * m[0][1]) / det;
        L[1] = (sol[1] * m[0][0] - sol[0] * m[1][0]) / det;
        res = out[0] + q[0] * L[0] + q[1] * L[1];
        radius = (res - out[0]).norm();
        break;
    case 4:
        q[0] = out[1] - out[0]; q[1] = out[2] - out[0]; q[2] = out[3] - out[0];
        for (i = 0; i < 3; ++i) for (j = 0; j < 3; ++j) m[i][j] = dot(q[i], q[j]) * 2;
        for (i = 0; i < 3; ++i) sol[i] = dot(q[i], q[i]);
        det = m[0][0] * m[1][1] * m[2][2] + m[0][1] * m[1][2] * m[2][0]
            + m[0][2] * m[2][1] * m[1][0] - m[0][2] * m[1][1] * m[2][0]
            - m[0][1] * m[1][0] * m[2][2] - m[0][0] * m[1][2] * m[2][1];
        if (sign(det) == 0) return;
        for (j = 0; j < 3; ++j) { for (i = 0; i < 3; ++i) m[i][j] = sol[i];
            L[j] = (m[0][0] * m[1][1] * m[2][2] + m[0][1] * m[1][2] * m[2][0]
                  + m[0][2] * m[2][1] * m[1][0] - m[0][2] * m[1][1] * m[2][0]
                  - m[0][1] * m[1][0] * m[2][2] - m[0][0] * m[1][2] * m[2][1]) / det;
            for (i = 0; i < 3; ++i) m[i][j] = dot(q[i], q[j]) * 2;
        } res = out[0];
        for (i = 0; i < 3; ++i) res += q[i] * L[i]; radius = (res - out[0]).norm();
    }
}
void minball(int n, point pt[]) {
    ball();
    if (outCnt < 4) for (int i = 0; i < n; ++i)
        if ((res - pt[i]).norm() > +radius + EPS) {
            out[outCnt] = pt[i]; ++outCnt; minball(i, pt); --outCnt;
            if (i > 0) {
                point Tt = pt[i];
                memmove(&pt[1], &pt[0], sizeof(point) * i);
                pt[0] = Tt;
            }
        }
}
pair<point, double> main(int npoint, point pt[]) { // 0-based
    random_shuffle(pt, pt + npoint); radius = -1;
    for (int i = 0; i < npoint; i++) { if ((res - pt[i]).norm() > EPS + radius) {
        outCnt = 1; out[0] = pt[i]; minball(i, pt); } }
    return make_pair(res, sqrt(radius));
}
```

1.19 三维向量操作矩阵

- 绕单位向量 $u=(u_x,u_y,u_z)$ 右手方向旋转 θ 度的矩阵:

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

$$=\cos\theta I+\sin\theta\begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}+(1-\cos\theta)\begin{bmatrix} u_x^2 & u_xu_y & u_xu_z \\ u_yu_x & u_y^2 & u_yu_z \\ u_zu_x & u_zu_y & u_z^2 \end{bmatrix}$$

- 点 a 绕单位向量 $u=(u_x,u_y,u_z)$ 右手方向旋转 θ 度的对应点为 $a'=a\cos\theta+(u\times a)\sin\theta+(u\otimes u)a(1-\cos\theta)$

- 关于向量 v 作对称变换的矩阵 $H = I - 2\frac{vv^T}{v^Tv}$,
- 点 a 对称点: $a' = a - 2\frac{v^Ta}{v^Tv} \cdot v$

1.20 立体角

对于任意一个四面体 $OABC$, 从 O 点观察 ΔABC 的立体角 $\tan\frac{\Omega}{2} = \frac{\mathrm{mix}(\vec{a},\vec{b},\vec{c})}{|a||b||c|+(\vec{a}\cdot\vec{b})|c|+(\vec{a}\cdot\vec{c})|b|+(\vec{b}\cdot\vec{c})|a|}$.

2 数据结构

2.1 动态凸包（只支持插入）

```
#define x first   // `upperHull $\leftarrow (x, y)$`
#define y second  // `lowerHull $\leftarrow (x, -y)$`
typedef map<int, int> mii;
typedef map<int, int>::iterator mit;
struct point { point(const mit &p): x(p->first), y(p->second) {} };
inline bool checkInside(mii &a, const point &p) { // `border inclusive`
    int x = p.x, y = p.y; mit p1 = a.lower_bound(x);
    if (p1 == a.end()) return false; if (p1->x == x) return y <= p1->y;
    if (p1 == a.begin()) return false; mit p2(p1--);
    return sign(det(p - point(p1), point(p2) - p)) >= 0;
} inline void addPoint(mii &a, const point &p) { // `no collinear points`
    int x = p.x, y = p.y; mit pnt = a.insert(make_pair(x, y)).first, p1, p2;
    for (pnt->y = y; ; a.erase(p2)) {
        p1 = pnt; if (++p1 == a.end()) break;
        p2 = p1;  if (++p1 == a.end()) break;
        if (det(point(p2) - p, point(p1) - p) < 0) break;
    } for ( ; ; a.erase(p2)) {
        if ((p1 = pnt) == a.begin()) break;
        if (--p1 == a.begin()) break; p2 = p1--;
        if (det(point(p2) - p, point(p1) - p) > 0) break;
    }
}
```

2.2 Rope 用法

```
#include <ext/rope>
using __gnu_cxx::crope; using __gnu_cxx::rope;
a = b.substr(from, len);        // `$[from, from + len)$`
a = b.substr(from);             // `$[from, from]$`
b.c_str();                      // `might lead to memory leaks`
b.delete_c_str();               // `delete the c\_{}str that created before`
a.insert(p, str);               // `insert str before position $p$`
a.erase(i, n);                  // `erase $[i, i + n)$`
```

2.3 Treap

```
struct node { int key, prio, size; node *ch[2]; } base[MAXN], *top, *root, *null, nil;
typedef node *tree;
tree newNode(int key) {
    static int seed = 3312;
    top->key = key; top->prio = seed = int(seed * 48271LL % 2147483647);
    top->size = 1; top->ch[0] = top->ch[1] = null; return top++;
}
void Rotate(tree &x, int d) {
    tree y = x->ch[!d]; x->ch[!d] = y->ch[d]; y->ch[d] = x; y->size = x->size;
    x->size = x->ch[0]->size + 1 + x->ch[1]->size; x = y;
}
void Insert(tree &t, int key) {
    if (t == null) t = newNode(key);
    else { int d = t->key < key; Insert(t->ch[d], key); ++t->size;
        if (t->ch[d]->prio < t->prio) Rotate(t, !d);
    }
}
void Delete(tree &t, int key) {
    if (t->key != key) { Delete(t->ch[t->key < key], key); --t->size; }
    else if (t->ch[0] == null) t = t->ch[1];
    else if (t->ch[1] == null) t = t->ch[0];
    else { int d = t->ch[0]->prio < t->ch[1]->prio;
        Rotate(t, d); Delete(t->ch[d], key); --t->size;
    }
}
```

2.4 可持久化 Treap

```
inline bool randomBySize(int a, int b) {
    static long long seed = 1;
    return (seed = seed * 48271 % 2147483647) * (a + b) < 2147483647LL * a;
}
tree merge(tree x, tree y) {
    if (x == null) return y; if (y == null) return x;
    tree t = NULL;
    if (randomBySize(x->size, y->size)) t = newNode(x), t->r = merge(x->r, y);
    else t = newNode(y), t->l = merge(x, y->l);
    update(t); return t;
}
void splitByKey(tree t, int k, tree &l, tree &r) { // `$[-\infty, k) [k, +\infty)$`
    if (t == null) l = r = null;
    else if (t->key < k) l = newNode(t), splitByKey(t->r, k, l->r, r), update(l);
    else                 r = newNode(t), splitByKey(t->l, k, l, r->l), update(r);
}
void splitBySize(tree t, int k, tree &l, tree &r) { // `$[1, k) [k, +\infty)$`
    static int s; if (t == null) l = r = null;
    else if ((s = t->l->size + 1) < k) l = newNode(t), splitBySize(t->r, k - s, l->r, r), update(l);
    else                                r = newNode(t), splitBySize(t->l, k, l, r->l), update(r);
}
```

2.5 左偏树

```
tree merge(tree a, tree b) {
    if (a == null) return b;
    if (b == null) return a;
    if (a->key > b->key) swap(a, b);
    a->rc = merge(a->rc, b);
    a->rc->fa = a;
    if (a->lc->dist < a->rc->dist) swap(a->lc, a->rc);
    a->dist = a->rc->dist + 1;
    return a;
}
void erase(tree t) {
    tree x = t->fa, y = merge(t->lc, t->rc);
    if (y != null) y->fa = x;
    if (x == null) root = y;
    else
    for ((x->lc == t ? x->lc : x->rc) = y; x != null; y = x, x = x->fa) {
        if (x->lc->dist < x->rc->dist) swap(x->lc, x->rc);
        if (x->rc->dist + 1 == x->dist) return;
        x->dist = x->rc->dist + 1;
    }
}
```

2.6 Link-Cut Tree

```
struct node { int rev; node *pre, *ch[2]; } base[MAXN], nil, *null;
typedef node *tree;
#define isRoot(x) (x->pre->ch[0] != x && x->pre->ch[1] != x)
#define isRight(x) (x->pre->ch[1] == x)
inline void MakeRev(tree t) { if (t != null) { t->rev ^= 1; swap(t->ch[0], t->ch[1]); } }
```

```
inline void PushDown(tree t) { if (t->rev) { MakeRev(t->ch[0]); MakeRev(t->ch[1]); t->rev = 0; } }
inline void Rotate(tree x) {
    tree y = x->pre; PushDown(y); PushDown(x);
    int d = isRight(x);
    if (!isRoot(y)) y->pre->ch[isRight(y)] = x; x->pre = y->pre;
    if ((y->ch[d] = x->ch[!d]) != null) y->ch[d]->pre = y;
    x->ch[!d] = y; y->pre = x; Update(y);
}
inline void Splay(tree x) {
    PushDown(x); for (tree y; !isRoot(x); Rotate(x)) {
        y = x->pre; if (!isRoot(y)) Rotate(isRight(x) != isRight(y) ? x : y);
    } Update(x);
}
inline void Splay(tree x, tree to) {
    PushDown(x); for (tree y; (y = x->pre) != to; Rotate(x)) if (y->pre != to)
        Rotate(isRight(x) != isRight(y) ? x : y);
    Update(x);
}
inline tree Access(tree t) {
    tree last = null; for (; t != null; last = t, t = t->pre) Splay(t),t->ch[1] = last, Update(t);
    return last;
}
inline void MakeRoot(tree t) { Access(t); Splay(t); MakeRev(t); }
inline tree FindRoot(tree t) { Access(t); Splay(t); tree last = null;
    for ( ; t != null; last = t, t = t->ch[0]) PushDown(t); Splay(last); return last;
}
inline void Join(tree x, tree y) { MakeRoot(y); y->pre = x; }
inline void Cut(tree t) {Access(t); Splay(t); t->ch[0]->pre = null; t->ch[0] = null; Update(t);}
inline void Cut(tree x, tree y) {
    tree upper = (Access(x), Access(y));
    if (upper == x) { Splay(x); y->pre = null; x->ch[1] = null; Update(x); }
    else if (upper == y) { Access(x); Splay(y); x->pre = null; y->ch[1] = null; Update(y); }
    else assert(0); // `impossible to happen`
}
inline int Query(tree a, tree b) { // `query the cost in path a <-> b, lca inclusive`
    Access(a); tree c = Access(b); // c is lca
    int v1 = c->ch[1]->maxCost; Access(a);
    int v2 = c->ch[1]->maxCost;
    return max(max(v1, v2), c->cost);
}
void Init() {
    null = &nil; null->ch[0] = null->ch[1] = null->pre = null; null->rev = 0;
    Rep(i, 1, N) { node &n = base[i]; n.rev = 0; n.pre = n.ch[0] = n.ch[1] = null; }
}
```

3 字符串相关

3.1 Manacher

```
char t[1001];       // 要处理的字符串
char s[1001 * 2];   // 中间插入特殊字符以后的
int Z[1001 * 2], L, R;  // Gusfield's Algorithm
// 由a往左，由b往右，对称地做字符匹配
int match(int a, int b){
    int i = 0;
    while (a - i >= 0 && b + i < n && s[a - i] == s[b + i]) i++;
    return i;
}
void longest_palindromic_substring()
{
    int N = strlen(t);
    /* 在 t 中插入特殊字符，存放到 s*/ memset(s, '.', N*2+1);
    for (int i = 0; i < N; ++i) s[i * 2 + 1] = t[i];
    N = N * 2 + 1;
    Z[0] = 1; L = R = 0;
    for (int i = 1; i < N; ++i) {
        int ii = L - (i - L), n = R + 1 - i;   // i的映射位
        if (i > R) Z[i] = match(i, i), L = i, R = i + Z[i] - 1;
        else if (Z[ii] == n) Z[i] = n + match(i-n, i+n), L = i, R = i + Z[i] - 1;
        else Z[i] = min(Z[ii], n);
    }
    // 尋找最長迴文子字串的長度。
    int n = 0, p = 0;
    for (int i=0; i < N; ++i) if(Z[i] > n) n = Z[p = i];
    // 記得去掉特殊字元。
    cout << "最長迴文子字串的長度是" << (n - 1) / 2;
    // 印出最長迴文子字串，記得別印特殊字元。
    for (int i = p - Z[p] + 1; i <= p + Z[p] - 1; ++i) if (i & 1) cout << s[i];
}
```

3.2 KMP

$$next[i] = \max\{len|A[0\ldots len-1] = A\text{的第 i 位向前或后的长度为 } len \text{ 的串}\}$$
$$ext[i] = \max\{len|A[0\ldots len-1] = B\text{的第 i 位向前或后的长度为 } len \text{ 的串}\}$$

```
void KMP(char *a, int la, char *b, int lb, int *next, int *ext) {
    --a; --b; --next; --ext;
    for (int i = 2, j = next[1] = 0; i <= la; i++) {
        while (j && a[j + 1] != a[i]) j = next[j]; if (a[j + 1] == a[i]) ++j; next[i] = j;
    } for (int i = 1, j = 0; i <= lb; ++i) {
        while (j && a[j + 1] != b[i]) j = next[j]; if (a[j + 1] == b[i]) ++j; ext[i] = j;
        if (j == la) j = next[j];
    }
} void ExKMP(char *a, int la, char *b, int lb, int *next, int *ext) {
    next[0] = la; for (int &j = next[1] = 0; j + 1 < la && a[j] == a[j + 1]; ++j);
    for (int i = 2, k = 1; i < la; ++i) {
        int p = k + next[k], l = next[i - k]; if (l < p - i) next[i] = l;
        else for (int &j = next[k = i] = max(0, p - i); i + j < la && a[j] == a[i + j]; ++j);
    } for (int &j = ext[0] = 0; j < la && j < lb && a[j] == b[j]; ++j);
    for (int i = 1, k = 0; i < lb; ++i) {
        int p = k + ext[k], l = next[i - k]; if (l < p - i) ext[i] = l;
        else for (int &j = ext[k = i] = max(0, p - i); j < la && i + j < lb && a[j] == b[i + j]; ++j);
    }
}
```

3.3 Aho-Corasick 自动机

```
void construct() {
    static tree Q[MAX_NODE]; int head = 0, tail = 0;
    for (root->fail = root, Q[++tail] = root; head < tail; ) {
        tree x = Q[++head];
//      if (x->fail->danger) x->danger = true;
        Rep(d, 0, sigma - 1) if (!x->next[d])
            x->next[d] = (x == root) ? (root) : (x->fail->next[d]);
        else {
            x->next[d]->fail = (x == root) ? (root) : (x->fail->next[d]);
            Q[++tail] = x->next[d];
        }
    }
}
```

3.4 后缀自动机

```
struct SAM {
    int in[Maxn * 2 + 1][Sigma], fa[Maxn * 2 + 1], max[Maxn * 2 + 1], tot, last;
    void init(int n) {
        tot = last = 0;
        for(int i = 0; i <= 2 * n + 1; ++i)
            memset(in[i], -1, sizeof in[i]), fa[i] = -1;
    }
    void add(int x) {
        int v = last; ++tot, last = tot, max[last] = max[v] + 1;
        while(v != -1 && in[v][x] == -1) in[v][x] = last, v = fa[v];
        if(v == -1) { fa[last] = 0; return; }
        int p = in[v][x];
        if(max[p] == max[v] + 1) fa[last] = p;
        else {
```

```cpp
            int np = ++tot;
            max[np] = max[v] + 1; fa[np] = fa[p], fa[p] = np, fa[last] = np;
            while(v != -1 && in[v][x] == p) in[v][x] = np, v = fa[v];
            memcpy(in[np], in[p], sizeof in[p]);
}}};
```

3.5 后缀数组

待排序的字符串放在 $r[0 \dots n-1]$ 中，最大值小于 m.
$r[0 \dots n-2] > 0$, $r[n-1] = 0$.
结果放在 $sa[0 \dots n-1]$.

```cpp
namespace SuffixArrayDoubling {
    int wa[MAXN], wb[MAXN], wv[MAXN], ws[MAXN];
    int cmp(int *r, int a, int b, int l) { return r[a] == r[b] && r[a + l] == r[b + l]; }
    void da(int *r, int *sa, int n, int m) {
        int i, j, p, *x = wa, *y = wb, *t;
        for (i = 0; i < m; i++) ws[i] = 0;
        for (i = 0; i < n; i++) ws[x[i] = r[i]]++;
        for (i = 1; i < m; i++) ws[i] += ws[i - 1];
        for (i = n - 1; i >= 0; i--) sa[--ws[x[i]]] = i;
        for (j = 1, p = 1; p < n; j *= 2, m = p) {
            for (p = 0, i = n - j; i < n; i++) y[p++] = i;
            for (i = 0; i < n; i++) if (sa[i] >= j) y[p++] = sa[i] - j;
            for (i = 0; i < n; i++) wv[i] = x[y[i]];
            for (i = 0; i < m; i++) ws[i] = 0;
            for (i = 0; i < n; i++) ws[wv[i]]++;
            for (i = 1; i < m; i++) ws[i] += ws[i - 1];
            for (i = n - 1; i >= 0; i--) sa[--ws[wv[i]]] = y[i];
            for (t = x, x = y, y = t, p = 1, x[sa[0]] = 0, i = 1; i < n; i++)
                x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;
}}}
namespace SuffixArrayDC3 { // `r 与 sa 大小需 3 倍`
    #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], ws[MAXN];
    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) {
        for (int i = 0; i < n; i++) wv[i] = r[a[i]];
        for (int i = 0; i < m; i++) ws[i] = 0;
        for (int i = 0; i < n; i++) ws[wv[i]]++;
        for (int i = 1; i < m; i++) ws[i] += ws[i - 1];
        for (int i = n - 1; i >= 0; i--) b[--ws[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++];
    }
    #undef F
    #undef G
}
namespace CalcHeight {
    int rank[MAXN], height[MAXN];
    void calheight(int *r, int *sa, int n) {
        int i, j, k = 0; for (i = 1; i <= n; i++) rank[sa[i]] = i;
        for (i = 0; i < n; height[rank[i++]] = k)
            for (k ? k-- : 0, j = sa[rank[i] - 1]; r[i + k] == r[j + k]; k++);
}
    void init(int len)
    {
        for(int i = 0; i <= len + 10; ++i)
            rank[i] = height[i] = 0;
    }
}
//Sample
int r[MAXN]; char s[MAXN];
int main()
{
    int len;
    scanf("%s", s);
    len = strlen(s);
    for(int i = 0; i < len; ++i) r[i] = s[i] - 'a' + 1;
    r[len] = 0;
    SuffixArrayDoubling::da(r, sa, n + 1, 30);
    CalcHeight::calheight(r, sa, n);
    //Then the value of sa[0~len-1] is 1 ~ n, so init RMQ carefully(1~n not 0~n-1)
    return 0;
}
}
```

3.6 环串最小表示

```cpp
int minimalRepresentation(int N, char *s) { // s must be double-sized and 0-based
    int i, j, k, l; for (i = 0; i < N; ++i) s[i + N] = s[i]; s[N + N] = 0;
    for (i = 0, j = 1; j < N; ) {
        for (k = 0; k < N && s[i + k] == s[j + k]; ++k);
        if (k >= N) break; if (s[i + k] < s[j + k]) j += k + 1;
        else l = i + k, i = j, j = max(l, j) + 1;
    } return i; // [i, i + N) is the minimal representation
}
```

3.7 回文自动机

```cpp
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <algorithm>

const int C = 26, N = 100000, S = N + 2 + C;
char string[N + 2];
int s, length[S], suffix[S], go[S][C];
int extend(int p, int i) {
    while (string[i - 1 - length[p]] != string[i]) {p = suffix[p];}
    int q = suffix[p];
    while (string[i - 1 - length[q]] != string[i]) {q = suffix[q];}
    int c = string[i] - 'a';
    int pp = go[p][c];int qq = go[q][c];
    if (pp == -1) {
        length[pp = go[p][c] = s ++] = length[p] + 2;
        suffix[pp] = qq;
        memset(go[pp], -1, sizeof(go[pp]));
    } return pp;
}
int main() {
    int tests; scanf("%d", &tests);
    for (int t = 1; t <= tests; ++ t) {
        printf("Case #%d: ", t);
        for (int i = 0; i < C + 2; ++ i) {
            suffix[i] = 1; length[i] = std::min(i - 1, 1);
            memset(go[i], -1, sizeof(go[i]));
        }
```

```
        suffix[0] = suffix[1] = 0;
        for (int i = 0; i < C; ++ i)
            go[0][i] = 2 + i;
        s = C + 2; string[0] = '#';
        scanf("%s", string + 1);
        int n = strlen(string + 1); int p = 0;
        for (int i = 1; i <= n; ++i)
            p = extend(p, i);
        int result = s - (C + 2); std::sort(string + 1, string + n + 1);
        result += std::unique(string + 1, string + n + 1) - string - 1;
        printf("%d\n", result);
    }
    return 0;
}
```

4 图论

4.1 Dominator Tree

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <algorithm>
#include <vector>

using namespace std;

const int oo = 1073741819;

const int Maxn = 200000;
const int Maxm = 200000;

vector<int> g[Maxn];

//idom[i] is the dominator of i, node id -- 1 based(1 ~ n), n is the source
class DominatorTree
{
public:
    int tail[4][Maxm], n, m;
    int Next[4][Maxm], sora[4][Maxm];
    int ss[4], top, w_time;
    int rel[Maxn], semi[Maxn], b[Maxn], idom[Maxn], best[Maxn], st[Maxn], pre[Maxn];
    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;
        }
        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][ss[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];
    }
    //n -- number of vertex, m -- number of edges, e -- edge set
    void init(int _n, int _m, const vector<pair<int, int> > &e)
    {
        n = _n, m = _m;
        origin();
        for (int i = 0; i < m; i++) {
            link(0, e[i].first, e[i].second);
            link(1, e[i].second, e[i].first);
        }
        w_time = 0, top = 0;
        dfs(n, 0);
    }

    void work()
    {
        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;
                int y;
                if (rel[na]>rel[ne]) {
                    find(na);
                    y = semi[best[na]];
                }
                else 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);
        }
    }
}dom;
```

4.2 带花树

```
namespace Blossom {
    int n, head, tail, S, T, lca;
    int match[MAXN], Q[MAXN], pred[MAXN], label[MAXN], inq[MAXN], inb[MAXN];
    vector<int> link[MAXN];
    inline void push(int x) { Q[tail++] = x; inq[x] = true; }
    int findCommonAncestor(int x, int y) {
        static bool inPath[MAXN]; for (int i = 0; i < n; ++i) inPath[i] = 0;
        for ( ; ; x = pred[ match[x] ]) { x = label[x]; inPath[x] = true; if (x == S) break; }
        for ( ; ; y = pred[ match[y] ]) { y = label[y]; if (inPath[y]) break; } return y;
    }
    void resetTrace(int x, int lca) {
        while (label[x] != lca) { int y = match[x]; inb[ label[x] ] = inb[ label[y] ] = true;
            x = pred[y]; if (label[x] != lca) pred[x] = y; }}
    void blossomContract(int x, int y) {
        lca = findCommonAncestor(x, y);
        Foru(i, 0, n) inb[i] = 0; resetTrace(x, lca); resetTrace(y, lca);
        if (label[x] != lca) pred[x] = y; if (label[y] != lca) pred[y] = x;
        Foru(i, 0, n) if (inb[ label[i] ]) { label[i] = lca; if (!inq[i]) push(i); }
    }
    bool findAugmentingPath() {
```

```
        Foru(i, 0, n) pred[i] = -1, label[i] = i, inq[i] = 0;
        int x, y, z; head = tail = 0;
        for (push(S); head < tail; ) for (int i = (int)link[x = Q[head++]].size() - 1; i >= 0; --i) {
            y = link[x][i]; if (label[x] == label[y] || x == match[y]) continue;
            if (y == S || (match[y] >= 0 && pred[ match[y] ] >= 0)) blossomContract(x, y);
            else if (pred[y] == -1) {
                pred[y] = x; if (match[y] >= 0) push(match[y]);
                else {
                    for (x = y; x >= 0; x = z) {
                    y = pred[x], z = match[y]; match[x] = y, match[y] = x;
                } return true; }}} return false;
    }
    int findMaxMatching() {
        int ans = 0; Foru(i, 0, n) match[i] = -1;
        for (S = 0; S < n; ++S) if (match[S] == -1) if (findAugmentingPath()) ++ans;
        return ans;
    }
}
```

4.3 KM

```
int N, Tcnt, w[MAXN][MAXN], slack[MAXN];
int lx[MAXN], linkx[MAXN], visy[MAXN], ly[MAXN], linky[MAXN], visx[MAXN]; // `初值全为0`
bool DFS(int x) { visx[x] = Tcnt;
    Rep(y, 1, N) if(visy[y] != Tcnt) { int t = lx[x] + ly[y] - w[x][y];
        if (t == 0) { visy[y] = Tcnt;
            if (!linky[y] || DFS(linky[y])) { linkx[x] = y; linky[y] = x; return true; }
        } else cMin(slack[y], t);
    } return false;
} void KM() {
    Tcnt = 0; Rep(x, 1, N) Rep(y, 1, N) cMax(lx[x], w[x][y]);
    Rep(S, 1, N) { Rep(i, 1, N) slack[i] = INF;
        for (++Tcnt; !DFS(S); ++Tcnt) { int d = INF;
            Rep(y, 1, N) if(visy[y] != Tcnt) cMin(d, slack[y]);
            Rep(x, 1, N) if(visx[x] == Tcnt) lx[x] -= d;
            Rep(y, 1, N) if(visy[y] == Tcnt) ly[y] += d; else slack[y] -= d;
        }
    }
}
```

4.4 双连通分量

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <vector>

using namespace std;

const int MAXN = 100000 + 10;

int dfn[MAXN], low[MAXN], bccno[MAXN], dfn_clock, bcc_cnt, Top;
vector <int> G[MAXN], bcc[MAXN];
pair <int, int> stk[MAXN];
bool iscut[MAXN];
int n, m;

void dfs(int p, int fa) {
    low[p] = dfn[p] = ++dfn_clock;
    int child = 0;
    for (int i = 0; i < G[p].size(); ++i) {
        int v = G[p][i];
        if (!dfn[v]) {
            stk[++Top] = make_pair(p, v);
            dfs(v, p);
            child++;
            low[p] = min(low[p], low[v]);
            if (low[v] >= dfn[p]) {
                iscut[p] = 1;
                ++bcc_cnt;
                bcc[bcc_cnt].clear();
                for (;;) {
                    pair <int, int> x = stk[Top];
                    --Top;
                    if (bccno[x.first] != bcc_cnt) {
                        bccno[x.first] = bcc_cnt;
                        bcc[bcc_cnt].push_back(x.first);
                    }
                    if (bccno[x.second] != bcc_cnt) {
                        bccno[x.second] = bcc_cnt;
                        bcc[bcc_cnt].push_back(x.second);
                    }
                    if (x.first == p && x.second == v)
                        break;
                }
            }
        }
        else
            if (dfn[v] < dfn[p] && v != fa) {
                stk[++Top] = make_pair(p, v);
                low[p] = min(low[p], dfn[v]);
            }
    }
    if (fa < 0 && child == 1) iscut[p] = 0;
}

void find_bcc(int n) {
    for (int i = 1; i <= n; ++i) dfn[i] = 0;
    for (int i = 1; i <= n; ++i) iscut[i] = 0;
    for (int i = 1; i <= n; ++i) bccno[i] = 0;
    dfn_clock = bcc_cnt = 0;
    for (int i = 1; i <= n; ++i)
        if (!dfn[i])
            dfs(i, -1);
}

int main() {
    scanf("%d%d", &n, &m);
    for (int a, b, i = 1; i <= m; ++i) {
        scanf("%d%d", &a, &b);
        G[a].push_back(b);
        G[b].push_back(a);
    }
    find_bcc(n);

    return 0;
}
```

4.5 强连通分量

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <vector>
#include <algorithm>

using namespace std;

const int MAXN = 100000 + 10;

vector <int> G[MAXN];
int n, m;
int dfn[MAXN], low[MAXN], stk[MAXN], Top, scc_cnt, sccno[MAXN], dfn_clock;

void dfs(int p) {
    dfn[p] = low[p] = ++dfn_clock;
    stk[++Top] = p;
    for (int i = 0; i < (int)G[p].size(); ++i) {
        int v = G[p][i];
        if (!dfn[v]) {
            dfs(v);
            low[p] = min(low[p], low[v]);
        }
        else if (!sccno[v])
            low[p] = min(low[p], dfn[v]);
    }
    if (low[p] == dfn[p]) {
```

```
        scc_cnt++;
        for (;;) {
            int x = stk[Top];
            --Top;
            sccno[x] = scc_cnt;
            if (x == p) break;
        }
    }
}

void find_scc(int n) {
    dfn_clock = scc_cnt = 0;
    for (int i = 1; i <= n; ++i) sccno[i] = 0;
    for (int i = 1; i <= n; ++i) dfn[i] = low[i] = 0;
    for (int i = 1; i <= n; ++i)
        if (!dfn[i])
            dfs(i);
}
```

4.6 2-SAT 与 Kosaraju

注意 Kosaraju 需要建反图

```
namespace SCC {
    int code[MAXN * 2], seq[MAXN * 2], sCnt;
    void DFS_1(int x) { code[x] = 1;
        for (edge e(fir[x]); e; e = e->next) if (code[e->to] == -1) DFS_1(e->to);
        seq[++sCnt] = x;
    } void DFS_2(int x) { code[x] = sCnt;
        for (edge e(fir2[x]); e; e = e->next) if (code[e->to] == -1) DFS_2(e->to); }
    void SCC(int N) {
        sCnt = 0; for (int i = 1; i <= N; ++i) code[i] = -1;
        for (int i = 1; i <= N; ++i) if (code[i] == -1) DFS_1(i);
        sCnt = 0; for (int i = 1; i <= N; ++i) code[i] = -1;
        for (int i = N; i >= 1; --i) if (code[seq[i]] == -1) {
            ++sCnt; DFS_2(seq[i]); }
    }
}//  true   -   2i - 1
//   false  -   2i
bool TwoSat() { SCC::SCC(N + N);
    // if code[2i - 1] = code[2i]: no solution
    // if code[2i - 1] > code[2i]: i selected. else i not selected
}
```

4.7 全局最小割 Stoer-Wagner

```
int minCut(int N, int G[MAXN][MAXN]) { // 0-based
    static int weight[MAXN], used[MAXN]; int ans = INT_MAX;
    while (N > 1) {
        for (int i = 0; i < N; ++i) used[i] = false; used[0] = true;
        for (int i = 0; i < N; ++i) weight[i] = G[i][0];
        int S = -1, T = 0;
        for (int _r = 2; _r <= N; ++_r) { // N - 1 selections
            int x = -1;
            for (int i = 0; i < N; ++i) if (!used[i])
                if (x == -1 || weight[i] > weight[x]) x = i;
            for (int i = 0; i < N; ++i) weight[i] += G[x][i];
            S = T; T = x; used[x] = true;
        } ans = min(ans, weight[T]);
        for (int i = 0; i < N; ++i) G[i][S] += G[i][T], G[S][i] += G[i][T];
        G[S][S] = 0; --N;
        for (int i = 0; i <= N; ++i) swap(G[i][T], G[i][N]);
        for (int i = 0; i < N; ++i) swap(G[T][i], G[N][i]);
    } return ans;
}
```

4.8 重口味费用流

```
struct MinCostFlow {
    int e[M], succ[M], last[N], val[M], cost[M], sum, 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;
```

4.9 欧拉路

```
vector<int> eulerianWalk(int N, int S) {
    static int res[MAXM], stack[MAXN]; static edge cur[MAXN];
    int rcnt = 0, top = 0, x; for (int i = 1; i <= N; ++i) cur[i] = fir[i];
    for (stack[top++] = S; top; ) {
        for (x = stack[--top]; ; ) {
            edge &e = cur[x]; if (e == NULL) break;
            stack[top++] = x; x = e->to; e = e->next;
            // 对于无向图需要删掉反向边
        } res[rcnt++] = x;
    } reverse(res, res + rcnt); return vector<int>(res, res + rcnt);
}
```

4.10 稳定婚姻

```cpp
namespace StableMatching {
    int pairM[MAXN], pairW[MAXN], p[MAXN];
    // init: pairM[0...n - 1] = pairW[0...n - 1] = -1, p[0...n - 1] = 0
    void stableMatching(int n, int orderM[MAXN][MAXN], int preferW[MAXN][MAXN]) {
        for (int i = 0; i < n; i++) while (pairM[i] < 0) {
            int w = orderM[i][p[i]++], m = pairW[w];
            if (m == -1) pairM[i] = w, pairW[w] = i;
            else if (preferW[w][i] < preferW[w][m])
                pairM[m] = -1, pairM[i] = w, pairW[w] = i, i = m;
        }
    }
}
```

4.11 最大团搜索

```cpp
namespace MaxClique { // 1-based
    int g[MAXN][MAXN], len[MAXN], list[MAXN][MAXN], mc[MAXN], ans, found;
    void DFS(int size) {
        if (len[size] == 0) { if (size > ans) ans = size, found = true; return; }
        for (int k = 0; k < len[size] && !found; ++k) {
            if (size + len[size] - k <= ans) break;
            int i = list[size][k]; if (size + mc[i] <= ans) break;
            for (int 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);
        }
    }
    int work(int n) {
        mc[n] = ans = 1; for (int i = n - 1; i; --i) { found = false; len[1] = 0;
            for (int j = i + 1; j <= n; ++j) if (g[i][j]) list[1][len[1]++] = j;
            DFS(1); mc[i] = ans;
        } return ans;
    }
}
```

4.12 极大团计数

```cpp
namespace MaxCliqueCounting {
    int n, ans;
    int ne[MAXN], ce[MAXN];
    int g[MAXN][MAXN], list[MAXN][MAXN];
    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);
    }
}
```

4.13 最小树形图

```cpp
namespace EdmondsAlgorithm { // O(ElogE + V^2) !!! 0-based !!!
    struct enode { int from, c, key, delta, dep; enode *ch[2], *next;
    } ebase[maxm], *etop, *fir[maxn], nil, *null, *inEdge[maxn], *chs[maxn];
    typedef enode *edge; typedef enode *tree;
    int n, m, setFa[maxn], deg[maxn], que[maxn];
    inline void pushDown(tree x) { if (x->delta) {
        x->ch[0]->key += x->delta; x->ch[0]->delta += x->delta;
        x->ch[1]->key += x->delta; x->ch[1]->delta += x->delta; x->delta = 0;
    }}
    tree merge(tree x, tree y) {
        if (x == null) return y; if (y == null) return x;
        if (x->key > y->key) swap(x, y); pushDown(x); x->ch[1] = merge(x->ch[1], y);
        if (x->ch[0]->dep < x->ch[1]->dep) swap(x->ch[0], x->ch[1]);
        x->dep = x->ch[1]->dep + 1; return x;
    }
    void addEdge(int u, int v, int w) {
        etop->from = u; etop->c = etop->key = w; etop->delta = etop->dep = 0;
        etop->next = fir[v]; etop->ch[0] = etop->ch[1] = null;
        fir[v] = etop; inEdge[v] = merge(inEdge[v], etop++);
    }
    void deleteMin(tree &r) { pushDown(r); r = merge(r->ch[0], r->ch[1]); }
    int findSet(int x) { return setFa[x] == x ? x : setFa[x] = findSet(setFa[x]); }
    void clear(int V, int E) {
        null = &nil; null->ch[0] = null->ch[1] = null; null->dep = -1;
        n = V; m = E; etop = ebase; Foru(i, 0, V) fir[i] = NULL; Foru(i, 0, V) inEdge[i] = null;
    }
    int solve(int root) { int res = 0, head, tail;
        for (int i = 0; i < n; ++i) setFa[i] = i;
        for ( ; ; ) { memset(deg, 0, sizeof(int) * n); chs[root] = inEdge[root];
            for (int i = 0; i < n; ++i) if (i != root && setFa[i] == i) {
                while (findSet(inEdge[i]->from) == findSet(i)) deleteMin(inEdge[i]);
                ++deg[ findSet((chs[i] = inEdge[i])->from) ];
            }
            for (int i = head = tail = 0; i < n; ++i)
                if (i != root && setFa[i] == i && deg[i] == 0) que[tail++] = i;
            while (head < tail) {
                int x = findSet(chs[que[head++]]->from);
                if (--deg[x] == 0) que[tail++] = x;
            } bool found = false;
            for (int i = 0; i < n; ++i) if (i != root && setFa[i] == i && deg[i] > 0) {
                int j = i; tree temp = null; found = true;
                do {setFa[j = findSet(chs[j]->from)] = i;
```

```
                    deleteMin(inEdge[j]); res += chs[j]->key;
                    inEdge[j]->key -= chs[j]->key; inEdge[j]->delta -= chs[j]->key;
                    temp = merge(temp, inEdge[j]);
                } while (j != i); inEdge[i] = temp;
            } if (!found) break;
        } for (int i = 0; i < n; ++ i) if (i != root && setFa[i] == i) res += chs[i]->key;
        return res;
    }
}
namespace ChuLiu { // O(V ^ 3) !!! 1-based !!!
    int n, used[maxn], pass[maxn], eg[maxn], more, que[maxn], g[maxn][maxn];
    void combine(int id, int &sum) { int tot = 0, from, i, j, k;
        for ( ; id != 0 && !pass[id]; id = eg[id]) que[tot++] = id, pass[id] = 1;
        for (from = 0; from < tot && que[from] != id; from++);
        if (from == tot) return; more = 1;
        for (i = from; i < tot; i++) {
            sum += g[eg[que[i]]][que[i]]; if (i == from) continue;
            for (j = used[que[i]] = 1; j <= n; j++) if (!used[j])
                if (g[que[i]][j] < g[id][j]) g[id][j] = g[que[i]][j];
        }
        for (i = 1; i <= n; i++) if (!used[i] && i != id)
            for (j = from; j < tot; j++) {
                k = que[j]; if (g[i][id] > g[i][k] - g[eg[k]][k])
                g[i][id] = g[i][k] - g[eg[k]][k];
            }
    }
    void clear(int V) { n = V; Rep(i, 1, V) Rep(j, 1, V) g[i][j] = inf; }
    int solve(int root) {
        int i, j, k, sum = 0; memset(used, 0, sizeof(int) * (n + 1));
        for (more = 1; more; ) {
            more = 0; memset(eg, 0, sizeof(int) * (n + 1));
            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(int) * (n + 1));
            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;
    }
}
```

4.14 离线动态最小生成树

$O(Qlog^2Q)$. $(qx[i], qy[i])$ 表示将编号为 $qx[i]$ 的边的权值改为 $qy[i]$, 删除一条边相当于将其权值改为 ∞, 加入一条边相当于将其权值从 ∞ 变成某个值.

```
const int maxn = 100000 + 5;
const int maxm = 1000000 + 5;
const int maxq = 1000000 + 5;
const int qsize = maxm + 3 * maxq;
int n, m, Q, x[qsize], y[qsize], z[qsize], qx[maxq], qy[maxq], a[maxn], *tz;
int kx[maxn], ky[maxn], kt, vd[maxn], id[maxm], app[maxm];
bool extra[maxm];
void init() {
    scanf("%d%d", &n, &m); for (int i = 0; i < m; i++) scanf("%d%d%d", x + i, y + i, z + i);
    scanf("%d", &Q); for (int i = 0; i < Q; i++) { scanf("%d%d", qx + i, qy + i); qx[i]--; }
}
int find(int x) {
    int root = x, next; while (a[root]) root = a[root];
    while ((next = a[x]) != 0) a[x] = root, x = next; return root;
}
inline bool cmp(const int &a, const int &b) { return tz[a] < tz[b]; }
void solve(int *qx, int *qy, int Q, int n, int *x, int *y, int *z, int m, long long ans) {
    int ri, rj;
    if (Q == 1) {
        for (int i = 1; i <= n; i++) a[i] = 0; z[qx[0]] = qy[0];
        for (int i = 0; i < m; i++) id[i] = i;
        tz = z; sort(id, id + m, cmp);
        for (int i = 0; i < m; i++) {
            ri = find(x[id[i]]); rj = find(y[id[i]]);
            if (ri != rj) ans += z[id[i]], a[ri] = rj;
        } printf("%I64d\n", ans);
        return;
    } int tm = kt = 0, n2 = 0, m2 = 0;
    for (int i = 1; i <= n; i++) a[i] = 0;
    for (int i = 0; i < Q; i++) {
        ri = find(x[qx[i]]); rj = find(y[qx[i]]); if (ri != rj) a[ri] = rj;
    }
    for (int i = 0; i < m; i++) extra[i] = true;
    for (int i = 0; i < Q; i++) extra[qx[i]] = false;
    for (int i = 0; i < m; i++) if (extra[i]) id[tm++] = i;
    tz = z; sort(id, id + tm, cmp);
    for (int i = 0; i < tm; i++) {
        ri = find(x[id[i]]); rj = find(y[id[i]]);
        if (ri != rj)
            a[ri] = rj, ans += z[id[i]], kx[kt] = x[id[i]], ky[kt] = y[id[i]], kt++;
    }
    for (int i = 1; i <= n; i++) a[i] = 0;
    for (int i = 0; i < kt; i++) a[find(kx[i])] = find(ky[i]);
    for (int i = 1; i <= n; i++) if (a[i] == 0) vd[i] = ++n2;
    for (int i = 1; i <= n; i++) if (a[i] != 0) vd[i] = vd[find(i)];
    int *Nx = x + m, *Ny = y + m, *Nz = z + m;
    for (int i = 0; i < m; i++) app[i] = -1;
    for (int i = 0; i < Q; i++)
        if (app[qx[i]] == -1)
            Nx[m2] = vd[x[qx[i]]], Ny[m2] = vd[y[qx[i]]], Nz[m2] = z[qx[i]], app[qx[i]] = m2, m2++;
    for (int i = 0; i < Q; i++) {
        z[qx[i]] = qy[i];
        qx[i] = app[qx[i]];
    }
    for (int i = 1; i <= n2; i++) a[i] = 0;
    for (int i = 0; i < tm; i++) {
        ri = find(vd[x[id[i]]]); rj = find(vd[y[id[i]]]);
        if (ri != rj)
            a[ri] = rj, Nx[m2] = vd[x[id[i]]], Ny[m2] = vd[y[id[i]]], Nz[m2] = z[id[i]], m2++;
    }
    int mid = Q / 2;
    solve(qx, qy, mid, n2, Nx, Ny, Nz, m2, ans);
    solve(qx + mid, qy + mid, Q - mid, n2, Nx, Ny, Nz, m2, ans);
}
void work() { if (Q) solve(qx, qy, Q, n, x, y, z, m, 0); }
int main() { init(); work(); return 0; }
```

4.15 弦图

- 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.
- 设第 i 个点在弦图的完美消除序列第 $p(i)$ 个. 令 $N(v) = \{w|w\text{与}v\text{相邻且}p(w) > p(v)\}$ 弦图的极大团一定是 $v \cup N(v)$ 的形式.
- 弦图最多有 n 个极大团.
- 设 $next(v)$ 表示 $N(v)$ 中最前的点. 令 $w*$ 表示所有满足 $A \in B$ 的 w 中最后的一个点. 判断 $v \cup N(v)$ 是否为极大团, 只需判断是否存在一个 w, 满足 $Next(w) = v$ 且 $|N(v)| + 1 \leq |N(w)|$ 即可.
- 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)
- 最大独立集: 完美消除序列从前往后能选就选.
- 最小团覆盖: 设最大独立集为 $\{p_1, p_2, \ldots, p_t\}$, 则 $\{p_1 \cup N(p_1), \ldots, p_t \cup N(p_t)\}$ 为最小团覆盖. (最大独立集数 = 最小团覆盖数)

```
class Chordal { // 1-Based, G is the Graph, must be sorted before call Check_Chordal
public: // Construct will sort it automatically
    int v[Maxn], id[Maxn];  bool inseq[Maxn]; priority_queue<pair<int, int> > pq;
    vector<int> Construct_Perfect_Elimination_Sequence(vector<int> *G, int n) { // O(m + nlogn)
        vector<int> seq(n + 1, 0);
        for (int i = 0; i <= n; ++i) inseq[i] = false, sort(G[i].begin(), G[i].end()), v[i] = 0;
        int cur = n; pair<int, int> Mx; while(!pq.empty()) pq.pop(); pq.push(make_pair(0, 1));
```

```
        for (int i = n; i >= 1; --i) {
            while (!pq.empty() && (Mx = pq.top(), inseq[Mx.second] || Mx.first != v[Mx.second])) pq.pop();
            id[Mx.second] = cur;
            int x = seq[cur--] = Mx.second, sz = (int)G[Mx.second].size(); inseq[x] = true;
            for (int j = 0; j < sz; ++j) {
                int y = G[x][j]; if(!inseq[y]) pq.push(make_pair(++v[y], y));
            }
        } return seq;
    }
    bool Check_Chordal(vector<int> *G, vector<int> &seq, int n) { // O(n + mlogn), plz gen seq first
        bool isChordal = true;
        for (int i = n - 1; i >= 1 && isChordal; --i) {
            int x = seq[i], sz, y = -1;
            if ((sz = (int)G[x].size()) == 0) continue;
            for(int j = 0; j < sz; ++j) {
                if (id[G[x][j]] < i) continue;
                if (y == -1 || id[y] > id[G[x][j]]) y = G[x][j];
            } if (y == -1) continue;
            for (int j = 0; j < sz; ++j) {
                int y1 = G[x][j]; if (id[y1] < i) continue;
                if (y1 == y || binary_search(G[y].begin(), G[y].end(), y1)) continue;
                isChordal = false; break;
            }
        } return isChordal;
    }
};
```

4.16 K 短路 (允许重复)

```
#define for_each(it, v) for (vector<Edge*>::iterator it = (v).begin(); it != (v).end(); ++it)
const int MAX_N = 10000, MAX_M = 50000, MAX_K = 10000, INF = 1000000000;
struct Edge { int from, to, weight; };
struct HeapNode { Edge* edge; int depth; HeapNode* child[4]; }; // child[0..1] for heap G, child[2..3] for heap
      out edge

int n, m, k, s, t; Edge* edge[MAX_M];
int dist[MAX_N]; Edge* prev[MAX_N];
vector<Edge*> graph[MAX_N]; vector<Edge*> graphR[MAX_N];
HeapNode* nullNode; HeapNode* heapTop[MAX_N];

HeapNode* createHeap(HeapNode* curNode, HeapNode* newNode) {
    if (curNode == nullNode) return newNode; HeapNode* rootNode = new HeapNode;
    memcpy(rootNode, curNode, sizeof(HeapNode));
    if (newNode->edge->weight < curNode->edge->weight) {
        rootNode->edge = newNode->edge; rootNode->child[2] = newNode->child[2]; rootNode->child[3] = newNode->
      child[3];
        newNode->edge = curNode->edge; newNode->child[2] = curNode->child[2]; newNode->child[3] = curNode->child
      [3];
    } if (rootNode->child[0]->depth < rootNode->child[1]->depth) rootNode->child[0] = createHeap(rootNode->child
      [0], newNode);
    else rootNode->child[1] = createHeap(rootNode->child[1], newNode);
    rootNode->depth = max(rootNode->child[0]->depth, rootNode->child[1]->depth) + 1;
    return rootNode;
}
bool heapNodeMoreThan(HeapNode* node1, HeapNode* node2) { return node1->edge->weight > node2->edge->weight; }

int main() {
    scanf("%d%d%d", &n, &m, &k); scanf("%d%d", &s, &t); s--, t--;
    while (m--) { Edge* newEdge = new Edge;
        int i, j, w; scanf("%d%d%d", &i, &j, &w);
        i--, j--; newEdge->from = i; newEdge->to = j; newEdge->weight = w;
        graph[i].push_back(newEdge); graphR[j].push_back(newEdge);
    }
    //Dijkstra
    queue<int> dfsOrder; memset(dist, -1, sizeof(dist));
    typedef pair<int, pair<int, Edge*> > DijkstraQueueItem;
    priority_queue<DijkstraQueueItem, vector<DijkstraQueueItem>, greater<DijkstraQueueItem> > dq;
    dq.push(make_pair(0, make_pair(t, (Edge*) NULL)));
    while (!dq.empty()) {
        int d = dq.top().first; int i = dq.top().second.first;
        Edge* edge = dq.top().second.second; dq.pop();
        if (dist[i] != -1) continue;
        dist[i] = d; prev[i] = edge; dfsOrder.push(i);
        for_each(it, graphR[i]) dq.push(make_pair(d + (*it)->weight, make_pair((*it)->from, *it)));
    }
    //Create edge heap
    nullNode = new HeapNode; nullNode->depth = 0; nullNode->edge = new Edge; nullNode->edge->weight = INF;
    fill(nullNode->child, nullNode->child + 4, nullNode);
    while (!dfsOrder.empty()) {
        int i = dfsOrder.front(); dfsOrder.pop();
        if (prev[i] == NULL) heapTop[i] = nullNode;
        else heapTop[i] = heapTop[prev[i]->to];
        vector<HeapNode*> heapNodeList;
        for_each(it, graph[i]) { int j = (*it)->to; if (dist[j] == -1) continue;
            (*it)->weight += dist[j] - dist[i]; if (prev[i] != *it) {
                HeapNode* curNode = new HeapNode;
                fill(curNode->child, curNode->child + 4, nullNode);
                curNode->depth = 1; curNode->edge = *it;
                heapNodeList.push_back(curNode);
            }
        } if (!heapNodeList.empty()) { //Create heap out
            make_heap(heapNodeList.begin(), heapNodeList.end(), heapNodeMoreThan);
            int size = heapNodeList.size();
            for (int p = 0; p < size; p++) {
                heapNodeList[p]->child[2] = 2 * p + 1 < size ? heapNodeList[2 * p + 1] : nullNode;
                heapNodeList[p]->child[3] = 2 * p + 2 < size ? heapNodeList[2 * p + 2] : nullNode;
            } heapTop[i] = createHeap(heapTop[i], heapNodeList.front());
        }
    } //Walk on DAG
    typedef pair<long long, HeapNode*> DAGQueueItem;
    priority_queue<DAGQueueItem, vector<DAGQueueItem>, greater<DAGQueueItem> > aq;
    if (dist[s] == -1) printf("NO\n");
    else { printf("%d\n", dist[s]);
        if (heapTop[s] != nullNode) aq.push(make_pair(dist[s] + heapTop[s]->edge->weight, heapTop[s]));
    } k--; while (k--) {
        if (aq.empty()) { printf("NO\n"); continue; }
        long long d = aq.top().first; HeapNode* curNode = aq.top().second; aq.pop();
        printf("%I64d\n", d);
        if (heapTop[curNode->edge->to] != nullNode)
            aq.push(make_pair(d + heapTop[curNode->edge->to]->edge->weight, heapTop[curNode->edge->to]));
        for (int i = 0; i < 4; i++) if (curNode->child[i] != nullNode)
            aq.push(make_pair(d - curNode->edge->weight + curNode->child[i]->edge->weight, curNode->child[i]));
    } return 0;
}
```

4.17 K 短路 (不允许重复)

```
int Num[10005][205], Path[10005][205], dev[10005], from[10005], value[10005], dist[205], Next[205], Graph
      [205][205];
int N, M, K, s, t, tot, cnt; bool forbid[205], hasNext[10005][205];
struct cmp {
    bool operator()(const int &a, const int &b) {
        int *i, *j; if (value[a] != value[b]) return value[a] > value[b];
        for (i = Path[a], j = Path[b]; (*i) == (*j); i++, j++);
        return (*i) > (*j);
    }
};
void Check(int idx, int st, int *path, int &res) {
    int i, j; for (i = 0; i < N; i++) dist[i] = 1000000000, Next[i] = t;
    dist[t] = 0; forbid[t] = true; j = t;
    for ( ; ; ) {
        for (i = 0; i < N; i++) if (!forbid[i] && (i != st || !hasNext[idx][j]) && (dist[j] + Graph[i][j] < dist[
      i] || (dist[j] + Graph[i][j] == dist[i] && j < Next[i])))
            Next[i] = j, dist[i] = dist[j] + Graph[i][j];
        j = -1; for (i = 0; i < N; i++) if (!forbid[i] && (j == -1 || dist[i] < dist[j])) j = i;
        if (j == -1) break; forbid[j] = 1; if (j == st) break;
    } res += dist[st]; for (i = st; i != t; i = Next[i], path++) (*path) = i; (*path) = i;
}
int main() {
    int i, j, k, l;
    while (scanf("%d%d%d%d%d", &N, &M, &K, &s, &t) && N) {
        priority_queue<int, vector<int>, cmp> Q;
        for (i = 0; i < N; i++) for (j = 0; j < N; j++) Graph[i][j] = 1000000000;
        for (i = 0; i < M; i++) { scanf("%d%d%d", &j, &k, &l); Graph[j - 1][k - 1] = l; }
        s--; t--;
```

```
        memset(forbid, false, sizeof(forbid)); memset(hasNext[0], false, sizeof(hasNext[0]));
        Check(0, s, Path[0], value[0]); dev[0] = 0; from[0] = 0; Num[0][0] = 0; Q.push(0);
        cnt = 1; tot = 1;
        for (i = 0; i < K; i++) {
            if (Q.empty()) break; l = Q.top(); Q.pop();
            for (j = 0; j <= dev[l]; j++) Num[l][j] = Num[from[l]][j];
            for (; Path[l][j] != t; j++) {
                memset(hasNext[tot], false, sizeof(hasNext[tot])); Num[l][j] = tot++;
            } for (j = 0; Path[l][j] != t; j++) hasNext[Num[l][j]][Path[l][j + 1]] = true;
            for (j = dev[l]; Path[l][j] != t; j++) {
                memset(forbid, false, sizeof(forbid)); value[cnt] = 0;
                for (k = 0; k < j; k++) {
                    forbid[Path[l][k]] = true;
                    Path[cnt][k] = Path[l][k];
                    value[cnt] += Graph[Path[l][k]][Path[l][k + 1]];
                } Check(Num[l][j], Path[l][j], &Path[cnt][j], value[cnt]);
                if (value[cnt] > 2000000) continue;
                dev[cnt] = j; from[cnt] = l; Q.push(cnt); cnt++;
            }
        }
        if (i < K || value[l] > 2000000) printf("None\n");
        else {
            for (i = 0; Path[l][i] != t; i++) printf("%d-", Path[l][i] + 1);
            printf("%d\n", t + 1);
        }
    } return 0;
}
```

4.18 小知识

- 平面图: 一定存在一个度小于等于 5 的点. $E \le 3V - 6$. 欧拉公式: $V + F - E = 1 +$ 连通块数
- 图连通度:
  1. $k-$ 连通 (k-*connected*): 对于任意一对结点都至少存在结点各不相同的 k 条路
  2. 点连通度 (*vertex connectivity*): 把图变成非连通图所需删除的最少点数
  3. Whitney 定理: 一个图是 $k-$ 连通的当且仅当它的点连通度至少为 k
- Lindstroem-Gessel-Viennot Lemma: 给定一个图的 n 个起点和 n 个终点, 令 $A_{ij} =$ 第 i 个起点到第 j 个终点的路径条数, 则从起点到终点的不相交路径条数为 $det(A)$
- 欧拉回路与树形图的联系: 对于出度等于入度的连通图 $s(G) = t_i(G)\prod_{j=1}^{n}(d^+(v_j) - 1)!$
- 密度子图: 给定无向图, 选取点集及其导出子图, 最大化 $W_e + P_v$ (点权可负).
  - $(S, u) = U$, $(u, T) = U - 2P_u - D_u$, $(u, v) = (v, u) = W_e$
  - ans $= \frac{Un - C[S,T]}{2}$, 解集为 $S - \{s\}$
- 最大权闭合图: 选 a 则 a 的后继必须被选
  - $P_u > 0$, $(S, u) = P_u$, $P_u < 0$, $(u, T) = -P_u$
  - ans $= \sum\limits_{P_u > 0} P_u - C[S, T]$, 解集为 $S - \{s\}$
- 判定边是否属于最小割:
  - 可能属于最小割: (u, v) 不属于同一 SCC
  - 一定在所有最小割中: (u, v) 不属于同一 SCC, 且 S, u 在同一 SCC, u, T 在同一 SCC
- 图同构 Hash: $F_t(i) = (F_{t-1}(i) \times A + \sum_{i \to j} F_{t-1}(j) \times B + \sum_{j \leftarrow i} F_{t-1}(j) \times C + D \times (i = a)) \pmod P$, 枚举点 a, 迭代 K 次后求得的 $F_k(a)$ 就是 a 点所对应的 Hash 值.

5 数学

5.1 博弈论相关

1. Anti-SG:
   规则与 Nim 基本相同，取最后一个的输。
   先手必胜当且仅当:
   (1) 所有堆的石子数都为 1 且游戏的 SG 值为 0;
   (2) 有些堆的石子数大于 1 且游戏的 SG 值不为 0。

2. SJ 定理:
   对于任意一个 Anti-SG 游戏，如果我们规定当局面中，所有的单一游戏的 SG 值为 0 时，游戏结束，则先手必胜当且仅当:
   (1) 游戏的 SG 函数不为 0 且游戏中某个单一游戏的 SG 函数大于 1;
   (2) 游戏的 SG 函数为 0 且游戏中没有单一游戏的 SG 函数大于 1。

3. Multi-SG 游戏:
   可以将一堆石子分成多堆.

4. Every-SG 游戏:
   每一个可以移动的棋子都要移动.
   对于我们可以赢的单一游戏，我们一定要拿到这一场游戏的胜利.
   只需要考虑如何让我们必胜的游戏尽可能长的玩下去，对手相反。
   于是就来一个 DP,
   step[v] = 0;（v 为终止状态）
   step[v] = maxstep[u] + 1;（sg[v]>0,sg[u]=0）
   step[v] = minstep[u] + 1;（sg[v]==0）

5. 翻硬币游戏:
   N 枚硬币排成一排，有的正面朝上，有的反面朝上。游戏者根据某些约束翻硬币 (如：每次只能翻一或两枚，或者每次只能翻连续的几枚)，但他所翻动的硬币中，最右边的必须是从正面翻到反面。谁不能翻谁输。
   结论：局面的 SG 值为局面中每个正面朝上的棋子单一存在时的 SG 值的异或和。可用数学归纳法证明。

6. 无向树删边游戏:
   规则如下:
   给出一个有 N 个点的树，有一个点作为树的根节点。游戏者轮流从树中删去边，删去一条边后，不与根节点相连的部分将被移走。谁无路可走谁输。
   结论:
   叶子节点的 SG 值为 0；中间节点的 SG 值为它的所有子节点的 SG 值加 1 后的异或和。是用数学归纳法证明。

7. Christmas Game(PKU3710):
   题目大意:
   有 N 个局部联通的图。Harry 和 Sally 轮流从图中删边，删去一条边后，不与根节点相连的部分将被移走。Sally 为先手。图是通过从基础树中加一些边得到的。所有形成的环保证不共用边，且只与基础树有一个公共点。谁无路可走谁输。环的处理成为了解题的关键。
   性质:
   (1) 对于长度为奇数的环，去掉其中任意一个边之后，剩下的两个链长度同奇偶，抑或之后的 SG 值不可能为奇数，所以它的 SG 值为 1;
   (2) 对于长度为偶数的环，去掉其中任意一个边之后，剩下的两个链长度异奇偶，抑或之后的 SG 值不可能为 0，所以它的 SG 值为 0；所以我们可以去掉所有的偶环，将所有的奇环变为长短为 1 的链。
   这样的话，我们已经将这道题改造成了上一节的模型。

8. 无向图的删边游戏:
   我们将 Christmas Game 这道题进行一步拓展——去掉对环的限制条件，这个模型应该怎样处理?
   无向图的删边游戏:
   一个无向联通图，有一个点作为图的根。游戏者轮流从图中删去边，删去一条边后，不与根节点相连的部分将被移走。谁无路可走谁输。
   结论:
   对无向图做如下改动：将图中的任意一个偶环缩成一个新点，任意一个奇环缩成一个新点加一个新边；所有连

到原先环上的边全部改为与新点相连。这样的改动不会影响图的 SG 值。

9. Staircase nim:
楼梯从地面由下向上编号为 0 到 n。游戏者在每次操作时可以将楼梯 j(1<=j<=n) 上的任意多但至少一个硬币移动到楼梯 j-1 上。将最后一枚硬币移至地上的人获胜。
结论:
设该游戏 Sg 函数为奇数格棋子数的 Xor 和 S。
如果 S=0，则先手必败，否则必胜。

5.2 单纯形 Cpp

max $\{cx | Ax \le b, x \ge 0\}$

```
const int MAXN = 11000, MAXM = 1100;
//  `here MAXN is the MAX number of conditions, MAXM is the MAX number of vars`

int avali[MAXM], avacnt;
double A[MAXN][MAXM];
double b[MAXN], c[MAXM];
double* simplex(int n, int m) {
//  `here n is the number of conditions, m is the number of vars`
    m++;
    int r = n, s = m - 1;
    static double D[MAXN + 2][MAXM + 1];
    static int ix[MAXN + MAXM];
    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];
            for (int j = 0; j <= m; j++) if (j != s) D[r][j] *= -D[r][s];
            avacnt = 0;
            for (int i = 0; i <= m; ++i)
                if(fabs(D[r][i]) > EPS)
                    avali[avacnt++] = i;
            for (int i = 0; i <= n + 1; i++) if (i != r) {
                if(fabs(D[i][s]) < EPS) continue;
                double *cur1 = D[i], *cur2 = D[r], tmp = D[i][s];
                //for (int j = 0; j <= m; j++) if (j != s) cur1[j] += cur2[j] * tmp;
                for(int j = 0; j < avacnt; ++j) if(avali[j] != s) cur1[avali[j]] += cur2[avali[j]] * tmp;
                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 null; // `非有界`
    }
    if (D[n + 1][m] < -EPS) return null; // `无法执行`
    static double x[MAXM - 1];
    for (int i = m; i < n + m; i++) if (ix[i] < m - 1) x[ix[i]] = D[i - m][m];
    return x; // `值为 $D[n][m]$`
}
```

5.3 自适应辛普森

```
double area(const double &left, const double &right) {
    double mid = (left + right) / 2;
    return (right - left) * (calc(left) + 4 * calc(mid) + calc(right)) / 6;
}

double simpson(const double &left, const double &right,
               const double &eps, const double &area_sum) {
    double mid = (left + right) / 2;
    double area_left = area(left, mid);
    double area_right = area(mid, right);
    double area_total = area_left + area_right;
    if (std::abs(area_total - area_sum) < 15 * eps) {
        return area_total + (area_total - area_sum) / 15;
    }
    return simpson(left, mid, eps / 2, area_left)
         + simpson(mid, right, eps / 2, area_right);
}

double simpson(const double &left, const double &right, const double &eps) {
    return simpson(left, right, eps, area(left, right));
}
```

5.4 FFT

```
namespace FFT {
    #define mul(a, b) (Complex(a.x * b.x - a.y * b.y, a.x * b.y + a.y * b.x))
    struct Complex {}; // `something omitted`
    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, 0.0);
                for (int j = 0; j < m; j++) {
                    Complex &P1 = P[i + j + m], &P2 = P[i + j];
                    Complex t = mul(unit, P1);
                    P1 = Complex(P2.x - t.x, P2.y - t.y);
                    P2 = Complex(P2.x + t.x, P2.y + t.y);
                    unit = mul(unit, unit_p0);
    }}}}
    vector<int> doFFT(const vector<int> &a, const vector<int> &b) {
        vector<int> ret(max(0, (int) a.size() + (int) b.size() - 1), 0);
        static Complex A[MAXB], B[MAXB], C[MAXB];
        int len = 1; while (len < (int)ret.size()) len *= 2;
        for (int i = 0; i < len; i++) A[i] = i < (int)a.size() ? a[i] : 0;
        for (int i = 0; i < len; i++) B[i] = i < (int)b.size() ? b[i] : 0;
        FFT(A, len, 1); FFT(B, len, 1);
        for (int i = 0; i < len; i++) C[i] = mul(A[i], B[i]);
        FFT(C, len, -1);
        for (int i = 0; i < (int)ret.size(); i++)
            ret[i] = (int) (C[i].x / len + 0.5);
        return ret;
    }
}
```

5.5 整数 FFT

```
namespace FFT {
//  `替代方案: $23068673( = 11 * 2 ^ {21} + 1)$, 原根为 $3$`
    const int MOD = 786433, PRIMITIVE_ROOT = 10; // `$3 * 2 ^ {18} + 1$`
    const int MAXB = 1 << 20;
    int getMod(int downLimit) { // `或者现场自己找一个MOD`
        for (int c = 3; ; ++c) { int t = (c << 21) | 1;
```

```
            if (t >= downLimit && isPrime(t)) return t;
    }}
    int modInv(int a) { return a <= 1 ? a : (long long) (MOD - MOD / a) * modInv(MOD % a) % MOD; }
    void NTT(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;
            long long unit_p0 = powMod(PRIMITIVE_ROOT, (MOD - 1) / m2);
            if (oper < 0) unit_p0 = modInv(unit_p0);
            for (int i = 0; i < n; i += m2) {
                long long unit = 1;
                for (int j = 0; j < m; j++) {
                    int &P1 = P[i + j + m], &P2 = P[i + j];
                    int t = unit * P1 % MOD;
                    P1 = (P2 - t + MOD) % MOD; P2 = (P2 + t) % MOD;
                    unit = unit * unit_p0 % MOD;
    }}}}
    vector<int> mul(const vector<int> &a, const vector<int> &b) {
        vector<int> ret(max(0, (int) a.size() + (int) b.size() - 1), 0);
        static int A[MAXB], B[MAXB], C[MAXB];
        int len = 1; while (len < (int)ret.size()) len <<= 1;
        for (int i = 0; i < len; i++) A[i] = i < (int)a.size() ? a[i] : 0;
        for (int i = 0; i < len; i++) B[i] = i < (int)b.size() ? b[i] : 0;
        NTT(A, len, 1); NTT(B, len, 1);
        for (int i = 0; i < len; i++) C[i] = (long long) A[i] * B[i] % MOD;
        NTT(C, len, -1); for (int i = 0, inv = modInv(len); i < (int)ret.size(); i++) ret[i] = (long long) C[i] *
      inv % MOD;
        return ret;
    }
}
```

5.6 扩展欧几里得

$ax + by = g = gcd(x, y)$

```
void exgcd(LL x, LL y, LL &a0, LL &b0, LL &g) {
    LL a1 = b0 = 0, b1 = a0 = 1, t;
    while (y != 0) {
        t = a0 - x / y * a1, a0 = a1, a1 = t;
        t = b0 - x / y * b1, b0 = b1, b1 = t;
        t = x % y, x = y, y = t;
    } if (x < 0) a0 = -a0, b0 = -b0, x = -x;
    g = x;
}
```

5.7 线性同余方程

- 中国剩余定理: 设 $m_1, m_2, \cdots, m_k$ 两两互素, 则同余方程组 $x \equiv a_i \pmod{m_i}$for $i = 1, 2, \cdots, k$ 在 $[0, M = m_1 m_2 \cdots m_k)$ 内有唯一解. 记 $M_i = M/m_i$, 找出 p_i 使得 $M_i p_i \equiv 1 \pmod{m_i}$, 记 $e_i = M_i p_i$, 则 $x \equiv e_1 a_1 + e_2 a_2 + \cdots + e_k a_k \pmod{M}$
- 多变元线性同余方程组: 方程的形式为 $a_1 x_1 + a_2 x_2 + \cdots + a_n x_n + b \equiv 0 \pmod{m}$, 令 $d = (a_1, a_2, \cdots, a_n, m)$, 有解的充要条件是 $d|b$, 解的个数为 $m^{n-1} d$

5.8 Miller-Rabin 素性测试

```
bool test(LL n, int base) {
    LL m = n - 1, ret = 0; int s = 0;
    for ( ; m % 2 == 0; ++s) m >>= 1; ret = pow_mod(base, m, n);
    if (ret == 1 || ret == n - 1) return true;
    for (--s; s >= 0; --s) {
        ret = multiply_mod(ret, ret, n); if (ret == n - 1) return true;
    } return false;
}
LL special[7] = {
    1373653LL,          25326001LL,
    3215031751LL,       25000000000LL,
    2152302898747LL,    3474749660383LL, 341550071728321LL};
/*
 * n < 2047                          test[] = {2}
 * n < 1,373,653                     test[] = {2, 3}
 * n < 9,080,191                     test[] = {31, 73}
 * n < 25,326,001                    test[] = {2, 3, 5}
 * n < 4,759,123,141                 test[] = {2, 7, 61}
 * n < 1,122,004,669,633             test[] = {2, 13, 23, 1662803}
 * n < 2,152,302,898,747             test[] = {2, 3, 5, 7, 11}
 * n < 3,474,749,660,383             test[] = {2, 3, 5, 7, 11, 13}
 * n < 341,550,071,728,321           test[] = {2, 3, 5, 7, 11, 13, 17}
 * n < 3,825,123,056,546,413,051     test[] = {2, 3, 5, 7, 11, 13, 17, 19, 23}
 */
bool is_prime(LL n) {
    if (n < 2) return false;
    if (n < 4) return true;
    if (!test(n, 2) || !test(n, 3)) return false;
    if (n < special[0]) return true;
    if (!test(n, 5)) return false;
    if (n < special[1]) return true;
    if (!test(n, 7)) return false;
    if (n == special[2]) return false;
    if (n < special[3]) return true;
    if (!test(n, 11)) return false;
    if (n < special[4]) return true;
    if (!test(n, 13)) return false;
    if (n < special[5]) return true;
    if (!test(n, 17)) return false;
    if (n < special[6]) return true;
    return test(n, 19) && test(n, 23) && test(n, 29) && test(n, 31) && test(n, 37);
}
```

5.9 PollardRho

```
LL pollardRho(LL n, LL seed) {
    LL x, y, head = 1, tail = 2; x = y = random() % (n - 1) + 1;
    for ( ; ; ) {
        x = addMod(multiplyMod(x, x, n), seed, n);
        if (x == y) return n; LL d = gcd(myAbs(x - y), n);
        if (1 < d && d < n) return d;
        if (++head == tail) y = x, tail <<= 1;
}} vector<LL> divisors;
void factorize(LL n) { // `需要保证 n > 1`
    if (isPrime(n)) divisors.push_back(n);
    else { LL d = n;
        while (d >= n) d = pollardRho(n, random() % (n - 1) + 1);
        factorize(n / d); factorize(d);
}}
```

5.10 多项式求根

```
const double error = 1e-12;
const double infi = 1e+12;
int n; double a[10], x[10];
double f(double a[], int n, double x) {
    double tmp = 1, sum = 0;
    for (int i = 0; i <= n; i++) sum = sum + a[i] * tmp, tmp = tmp * x;
    return sum;
}
double binary(double l, double r, double a[], int n) {
    int sl = sign(f(a, n, l)), sr = sign(f(a, n, r));
    if (sl == 0) return l; if (sr == 0) return r;
    if (sl * sr > 0) return infi;
    while (r - l > error) {
        double mid = (l + r) / 2;
        int ss = sign(f(a, n, mid));
        if (ss == 0) return mid;
        if (ss * sl > 0) l = mid; else r = mid;
```

```
    } return l;
}
void solve(int n, double a[], double x[], int &nx) {
    if (n == 1) { x[1] = -a[0] / a[1]; nx = 1; return; }
    double da[10], dx[10]; int ndx;
    for (int i = n; i >= 1; i--) da[i - 1] = a[i] * i;
    solve(n - 1, da, dx, ndx); nx = 0;
    if (ndx == 0) {
        double tmp = binary(-infi, infi, a, n);
        if (tmp < infi) x[++nx] = tmp; return;
    } double tmp = binary(-infi, dx[1], a, n);
    if (tmp < infi) x[++nx] = tmp;
    for (int i = 1; i <= ndx - 1; i++) {
        tmp = binary(dx[i], dx[i + 1], a, n);
        if (tmp < infi) x[++nx] = tmp;
    } tmp = binary(dx[ndx], infi, a, n);
    if (tmp < infi) x[++nx] = tmp;
}
int main() {
    scanf("%d", &n);
    for (int i = n; i >= 0; i--) scanf("%lf", &a[i]);
    int nx; solve(n, a, x, nx);
    for (int i = 1; i <= nx; i++) printf("%0.6f\n", x[i]);
    return 0;
}
```

5.11 线性递推

for $a_{i+n} = (\sum_{i=0}^{n-1} k_j a_{i+j}) + d$, $a_m = (\sum_{i=0}^{n-1} c_i a_i) + c_n d$

```
vector<int> recFormula(int n, int k[], int m) {
    vector<int> c(n + 1, 0);
    if (m < n) c[m] = 1;
    else {
        static int a[MAX_K * 2 + 1];
        vector<int> b = recFormula(n, k, m >> 1);
        for (int i = 0; i < n + n; ++i) a[i] = 0;
        int s = m & 1;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) a[i + j + s] += b[i] * b[j];
            c[n] += b[i];
        } c[n] = (c[n] + 1) * b[n];
        for (int i = n * 2 - 1; i >= n; i--) {
            int add = a[i]; if (add == 0) continue;
            for (int j = 0; j < n; j++) a[i - n + j] += k[j] * add;
            c[n] += add;
        } for (int i = 0; i < n; ++i) c[i] = a[i];
    } return c;
}
```

5.12 离散对数

$A^x \equiv B \pmod C$, 对非质数 C 也适用.

```
int modLog(int A, int B, int C) {
    static pii baby[MAX_SQRT_C + 11];
    int d = 0; LL k = 1, D = 1; B %= C;
    for (int i = 0; i < 100; ++i, k = k * A % C) // `$[0, \log C]$`
        if (k == B) return i;
    for (int g; ; ++d) {
        g = gcd(A, C); if (g == 1) break;
        if (B % g != 0) return -1;
        B /= g; C /= g; D = (A / g * D) % C;
    } int m = (int) ceil(sqrt((double) C)); k = 1;
    for (int i = 0; i <= m; ++i, k = k * A % C) baby[i] = pii(k, i);
    sort(baby, baby + m + 1); // [0, m]
    int n = unique(baby, baby + m + 1, equalFirst) - baby, am = powMod(A, m, C);
    for (int i = 0; i <= m; ++i) {
        LL e, x, y; exgcd(D, C, x, y, e); e = x * B % C;
        if (e < 0) e += C;
        if (e >= 0) {
            int k = lower_bound(baby, baby + n, pii(e, -1)) - baby;
            if (baby[k].first == e) return i * m + baby[k].second + d;
        } D = D * am % C;
    } return -1;
}
```

5.13 平方剩余

- Legrendre Symbol: 对奇质数 p, $(\frac{a}{p}) = \begin{cases} 1 & \text{是平方剩余} \\ -1 & \text{是非平方剩余} = a^{\frac{p-1}{2}} \bmod p \\ 0 & a \equiv 0 \pmod p \end{cases}$
- 若 p 是奇质数, $(\frac{-1}{p}) = 1$ 当且仅当 $p \equiv 1 \pmod 4$
- 若 p 是奇质数, $(\frac{2}{p}) = 1$ 当且仅当 $p \equiv \pm 1 \pmod 8$
- 若 p, q 是奇素数且互质, $(\frac{p}{q})(\frac{q}{p}) = (-1)^{\frac{p-1}{2} \times \frac{q-1}{2}}$
- Jacobi Symbol: 对奇数 $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$, $(\frac{a}{n}) = (\frac{a}{p_1})^{\alpha_1} (\frac{a}{p_2})^{\alpha_2} \cdots (\frac{a}{p_k})^{\alpha_k}$
- Jacobi Symbol 为 -1 则一定不是平方剩余, 所有平方剩余的 Jacobi Symbol 都是 1, 但 1 不一定是平方剩余

$ax^2 + bx + c \equiv 0 \pmod p$, 其中 $a \neq 0 \pmod p$, 且 p 是质数

```
inline int normalize(LL a, int P) { a %= P; return a < 0 ? a + P : a; }
vector<int> QuadraticResidue(LL a, LL b, LL c, int P) {
    int h, t; LL r1, r2, delta, pb = 0;
    a = normalize(a, P); b = normalize(b, P); c = normalize(c, P);
    if (P == 2) { vector<int> res;
        if (c % P == 0) res.push_back(0);
        if ((a + b + c) % P == 0) res.push_back(1);
        return res;
    } delta = b * rev(a + a, P) % P;
    a = normalize(-c * rev(a, P) + delta * delta, P);
    if (powMod(a, P / 2, P) + 1 == P) return vector<int>(0);
    for (t = 0, h = P / 2; h % 2 == 0; ++t, h /= 2);
    r1 = powMod(a, h / 2, P);
    if (t > 0) { do b = random() % (P - 2) + 2;
        while (powMod(b, P / 2, P) + 1 != P); }
    for (int i = 1; i <= t; ++i) {
        LL d = r1 * r1 % P * a % P;
        for (int j = 1; j <= t - i; ++j) d = d * d % P;
        if (d + 1 == P) r1 = r1 * pb % P; pb = pb * pb % P;
    } r1 = a * r1 % P; r2 = P - r1;
    r1 = normalize(r1 - delta, P); r2 = normalize(r2 - delta, P);
    if (r1 > r2) swap(r1, r2); vector<int> res(1, r1);
    if (r1 != r2) res.push_back(r2);
    return res;
}
```

5.14 N 次剩余

- 若 p 为奇质数, a 为 p 的 n 次剩余的充要条件是 $a^{\frac{p-1}{(a,p-1)}} \equiv 1 \pmod p$.

$x^N \equiv a \pmod p$, 其中 p 是质数

```
vector<int> solve(int p, int N, int a) {
    if ((a %= p) == 0) return vector<int>(1, 0);
    int g = findPrimitiveRoot(p), m = modLog(g, a, p); // g ^ m = a (mod p)
    if (m == -1) return vector<int>(0);
    LL B = p - 1, x, y, d; exgcd(N, B, x, y, d);
    if (m % d != 0) return vector<int>(0);
    vector<int> ret; x = (x * (m / d) % B + B) % B; // g ^ B mod p = g ^ (p - 1) mod p = 1
```

```cpp
    for (int i = 0, delta = B / d; i < d; ++i) {
        x = (x + delta) % B; ret.push_back((int)powMod(g, x, p));
    } sort(ret.begin(), ret.end());
    ret.resize(unique(ret.begin(), ret.end()) - ret.begin());
    return ret;
}
```

5.15 Pell 方程

$\binom{x_k}{y_k} = \begin{pmatrix} x_1 & dy_1 \\ y_1 & x_1 \end{pmatrix}^{k-1} \binom{x_1}{y_1}$

```cpp
pair<ULL, ULL> Pell(int n) {
    static ULL p[50] = {0, 1}, q[50] = {1, 0}, g[50] = {0, 0}, h[50] = {0, 1}, a[50];
    ULL t = a[2] = Sqrt(n);
    for (int i = 2; ; ++i) {
        g[i] = -g[i - 1] + a[i] * h[i - 1];
        h[i] = (n - g[i] * g[i]) / h[i - 1];
        a[i + 1] = (g[i] + t) / h[i];
        p[i] = a[i] * p[i - 1] + p[i - 2];
        q[i] = a[i] * q[i - 1] + q[i - 2];
        if (p[i] * p[i] - n * q[i] * q[i] == 1) return make_pair(p[i], q[i]);
    } return make_pair(-1, -1);
}
```

5.16 小知识

- 勾股数：设正整数 n 的质因数分解为 $n = \prod p_i^{a_i}$，则 $x^2 + y^2 = n$ 有整数解的充要条件是 n 中不存在形如 $p_i \equiv 3 \pmod 4$ 且指数 a_i 为奇数的质因数 p_i. $(\frac{a-b}{2})^2 + ab = (\frac{a+b}{2})^2$.
- 素勾股数：若 m 和 n 互质，而且 m 和 n 中有一个是偶数，则 $a = m^2 - n^2$, $b = 2mn$, $c = m^2 + n^2$，则 a、b、c 是素勾股数.
- Stirling 公式：$n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$
- Pick 定理：简单多边形，不自交，顶点如果全是整点. 则：严格在多边形内部的整点数 $+\frac{1}{2}$ 在边上的整点数 $-1=$ 面积
- Mersenne 素数：p 是素数且 $2^p - 1$ 的数是素数. (10000 以内的 p 有：2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217, 4253, 4423, 9689, 9941)
- 序列差分表：差分表的第 0 条对角线确定原序列. 设原序列为 h_i，第 0 条对角线为 $c_0, c_1, \ldots, c_p, 0, 0, \ldots$. 有这样两个公式：$h_n = \binom{n}{0}c_0 + \binom{n}{1}c_1 + \ldots + \binom{n}{p}c_p$, $\sum_{k=0}^{n} h_k = \binom{n+1}{1}c_0 + \binom{n+1}{2}c_2 + \ldots + \binom{n+1}{p+1}c_p$
- GCD: $\gcd(2^a - 1, 2^b - 1) = 2^{\gcd(a,b)} - 1$
- Fermat 分解算法：从 $t = \sqrt{n}$ 开始，依次检查 $t^2 - n, (t+1)^2 - n, (t+2)^2 - n, \ldots$，直到出现一个平方数 y，由于 $t^2 - y^2 = n$，因此分解得 $n = (t-y)(t+y)$. 显然，当两个因数很接近时这个方法能很快找到结果，但如果遇到一个素数，则需要检查 $\frac{n+1}{2} - \sqrt{n}$ 个整数
- 牛顿迭代：$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$
- 球与盒子的动人故事：(n 个球，m 个盒子，S 为第二类斯特林数)
  1. 球同，盒同，无空：dp
  2. 球同，盒同，可空：dp
  3. 球同，盒不同，无空：$\binom{n-1}{m-1}$
  4. 球同，盒不同，可空：$\binom{n+m-1}{n-1}$
  5. 球不同，盒同，无空：$S(n, m)$
  6. 球不同，盒同，可空：$\sum_{k=1}^{m} S(n, k)$
  7. 球不同，盒不同，无空：$m!S(n, m)$
  8. 球不同，盒不同，可空：m^n
- 组合数奇偶性：若 $(n\&m) = m$，则 $\binom{n}{m}$ 为奇数，否则为偶数
- 格雷码 $G(x) = x \otimes (x >> 1)$
- Fibonacci 数：
  - $F_0 = F_1 = 1$, $F_i = F_{i-1} + F_{i-2}$, $F_{-i} = (-1)^{i-1}F_i$
  - $F_i = \frac{1}{\sqrt{5}}((\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n)$
  - $\gcd(F_n, F_m) = F_{\gcd(n,m)}$
  - $F_{i+1}F_i - F_i^2 = (-1)^i$
  - $F_{n+k} = F_k F_{n+1} + F_{k-1} F_n$
- 第一类 Stirling 数：$\left[{n \atop k}\right]$ 代表第一类无符号 Stirling 数，代表将 n 阶置换群中有 k 个环的置换个数；$s(n, k)$ 代表有符号型，$s(n, k) = (-1)^{n-k}\left[{n \atop k}\right]$.
  - $(x)^{(n)} = \sum\limits_{k=0}^{n} \left[{n \atop k}\right] x^k$, $(x)_n = \sum\limits_{k=0}^{n} s(n, k)x^k$
  - $\left[{n \atop k}\right] = n\left[{n-1 \atop k}\right] + \left[{n-1 \atop k-1}\right]$, $\left[{0 \atop 0}\right] = 1$, $\left[{n \atop 0}\right] = \left[{0 \atop n}\right] = 0$
  - $\left[{n \atop n-2}\right] = \frac{1}{4}(3n-1)\binom{n}{3}$, $\left[{n \atop n-3}\right] = \binom{n}{2}\binom{n}{4}$
  - $\sum\limits_{k=0}^{a} \left[{n \atop k}\right] = n! - \sum\limits_{k=0}^{n} \left[{n \atop k+a+1}\right]$
  - $\sum\limits_{p=k}^{n} \left[{n \atop p}\right]\binom{p}{k} = \left[{n+1 \atop k+1}\right]$
- 第二类 Stirling 数：$\left\{{n \atop k}\right\} = S(n, k)$ 代表 n 个不同的球，放到 k 个相同的盒子里，盒子非空.
  - $\left\{{n \atop k}\right\} = \frac{1}{k!}\sum\limits_{j=0}^{k}(-1)^j\binom{k}{j}(k-j)^n$
  - $\left\{{n+1 \atop k}\right\} = k\left\{{n \atop k}\right\} + \left\{{n \atop k-1}\right\}$, $\left\{{0 \atop 0}\right\} = 1$, $\left\{{n \atop 0}\right\} = \left\{{0 \atop n}\right\} = 0$
  - 奇偶性：$(n-k)\&\frac{k-1}{2} == 0$
- Bell 数：B_n 代表将 n 个元素划分成若干个非空集合的方案数
  - $B_0 = B_1 = 1$, $B_n = \sum\limits_{k=0}^{n-1}\binom{n-1}{k}B_k$
  - $B_n = \sum\limits_{k=0}^{n}\left\{{n \atop k}\right\}$
  - Bell 三角形：$a_{1,1} = 1$, $a_{n,1} = a_{n-1,n-1}$, $a_{n,m} = a_{n,m-1} + a_{n-1,m-1}$, $B_n = a_{n,1}$

  - 对质数 p, $B_{n+p} \equiv B_n + B_{n+1} \pmod p$
  - 对质数 p, $B_{n+p^m} \equiv mB_n + B_{n+1} \pmod p$
  - 对质数 p, 模的周期一定是 $\frac{p^p-1}{p-1}$ 的约数, $p \le 101$ 时就是这个值
  - 从 B_0 开始, 前几项是 $1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975 \cdots$

- Bernoulli 数
  - $B_0 = 1, B_1 = \frac{1}{2}, B_2 = \frac{1}{6}, B_4 = -\frac{1}{30}, B_6 = \frac{1}{42}, B_8 = B_4, B_{10} = \frac{5}{66}$
  - $\sum\limits_{k=1}^{n} k^m = \frac{1}{m+1} \sum\limits_{k=0}^{m} \binom{m+1}{k} B_k n^{m+1-k}$
  - $B_m = 1 - \sum\limits_{k=0}^{m-1} \binom{m}{k} \frac{B_k}{m-k+1}$

- 完全数: x 是偶完全数等价于 $x = 2^{n-1}(2^n - 1)$, 且 $2^n - 1$ 是质数.

6 其他

6.1 Extended LIS

```
int G[MAXN][MAXN];
void insertYoung(int v) {
    for (int x = 1, y = INT_MAX; ; ++x) {
        Down(y, *G[x]); while (y > 0 && G[x][y] >= v) --y;
        if (++y > *G[x]) { ++*G[x]; G[x][y] = v; break; }
        else swap(G[x][y], v);
    }
}
int solve(int N, int seq[]) {
    Rep(i, 1, N) *G[i] = 0;
    Rep(i, 1, N) insertYoung(seq[i]);
    printf("%d\n", *G[1] + *G[2]);
    return 0;
}
```

6.2 生成 nCk

```
void nCk(int n, int k) {
    for (int comb = (1 << k) - 1; comb < (1 << n); ) {
        int x = comb & -comb, y = comb + x;
        comb = (((comb & ~y) / x) >> 1) | y;
    }
}
```

6.3 nextPermutation

```
boolean nextPermutation(int[] is) {
    int n = is.length;
    for (int i = n - 1; i > 0; i--) {
        if (is[i - 1] < is[i]) {
            int j = n; while (is[i - 1] >= is[--j]);
            swap(is, i - 1, j); // swap is[i - 1], is[j]
            rev(is, i, n); // reverse is[i, n)
            return true;
        }
    } rev(is, 0, n);
    return false;
}
```

6.4 Josephus 数与逆 Josephus 数

```
int josephus(int n, int m, int k) { int x = -1;
    for (int i = n - k + 1; i <= n; i++) x = (x + m) % i; return x;
}
int invJosephus(int n, int m, int x) {
    for (int i = n; ; i--) { if (x == i) return n - i; x = (x - m % i + i) % i; }
}
```

6.5 直线下的整点个数

求 $\sum_{i=0}^{n-1} \lfloor \frac{a+bi}{m} \rfloor$

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

6.6 Java 多项式

```
class Polynomial {
    final static Polynomial ZERO = new Polynomial(new int[] { 0 });
    final static Polynomial ONE = new Polynomial(new int[] { 1 });
    final static Polynomial X = new Polynomial(new int[] { 0, 1 });
    int[] coef;
    static Polynomial valueOf(int val) { return new Polynomial(new int[] { val }); }
    Polynomial(int[] coef) { this.coef = Arrays.copyOf(coef, coef.length); }
    Polynomial add(Polynomial o, int mod); // omitted
    Polynomial subtract(Polynomial o, int mod); // omitted
    Polynomial multiply(Polynomial o, int mod); // omitted
    Polynomial scale(int o, int mod); // omitted
    public String toString() {
        int n = coef.length; String ret = "";
        for (int i = n - 1; i > 0; --i) if (coef[i] != 0)
            ret += coef[i] + "x^" + i + "+";
        return ret + coef[0];
    }
    static Polynomial lagrangeInterpolation(int[] x, int[] y, int mod) {
        int n = x.length; Polynomial ret = Polynomial.ZERO;
        for (int i = 0; i < n; ++i) {
            Polynomial poly = Polynomial.valueOf(y[i]);
            for (int j = 0; j < n; ++j) if (i != j) {
                poly = poly.multiply(
                    Polynomial.X.subtract(Polynomial.valueOf(x[j]), mod), mod);
                poly = poly.scale(powMod(x[i] - x[j] + mod, mod - 2, mod), mod);
            } ret = ret.add(poly, mod);
        } return ret;
    }
}
```

6.7 long long 乘法取模

```
LL multiplyMod(LL a, LL b, LL P) { // `需要保证 a 和 b 非负`
    LL t = (a * b - LL((long double)a / P * b + 1e-3) * P) % P;
    return t < 0 : t + P : t;
}
```

6.8 重复覆盖

```cpp
namespace DLX {
    struct node { int x, y; node *l, *r, *u, *d; } base[MAX * MAX], *top, *head;
    typedef node *link;
    int row, col, nGE, ans, stamp, cntc[MAX], vis[MAX];
    vector<link> eachRow[MAX], eachCol[MAX];
    inline void addElement(int x, int y) {
        top->x = x; top->y = y; top->l = top->r = top->u = top->d = NULL;
        eachRow[x].push_back(top); eachCol[y].push_back(top++);
    }
    void init(int _row, int _col, int _nGE) {
        row = _row; col = _col; nGE = _nGE; top = base; stamp = 0;
        for (int i = 0; i <= col; ++i) vis[i] = 0;
        for (int i = 0; i <= row; ++i) eachRow[i].clear();
        for (int i = 0; i <= col; ++i) eachCol[i].clear();
        for (int i = 0; i <= col; ++i) addElement(0, i);
        head = eachCol[0].front();
    }
    void build() {
        for (int i = 0; i <= row; ++i) {
            vector<link> &v = eachRow[i];
            sort(v.begin(), v.end(), cmpByY);
            int s = v.size();
            for (int j = 0; j < s; ++j) {
                link l = v[j], r = v[(j + 1) % s]; l->r = r; r->l = l;
            }
        }
        for (int i = 0; i <= col; ++i) {
            vector<link> &v = eachCol[i];
            sort(v.begin(), v.end(), cmpByX);
            int s = v.size();
            for (int j = 0; j < s; ++j) {
                link u = v[j], d = v[(j + 1) % s]; u->d = d; d->u = u;
            }
        } for (int i = 0; i <= col; ++i) cntc[i] = (int) eachCol[i].size() - 1;
    }
    void removeExact(link c) {
        c->l->r = c->r; c->r->l = c->l;
        for (link i = c->d; i != c; i = i->d)
            for (link j = i->r; j != i; j = j->r) {
                j->d->u = j->u; j->u->d = j->d; --cntc[j->y];
            }
    }
    void resumeExact(link c) {
        for (link i = c->u; i != c; i = i->u)
            for (link j = i->l; j != i; j = j->l) {
                j->d->u = j; j->u->d = j; ++cntc[j->y];
            }
        c->l->r = c; c->r->l = c;
    }
    void removeRepeat(link c) {
        for (link i = c->d; i != c; i = i->d) {
            i->l->r = i->r; i->r->l = i->l;
        }
    }
    void resumeRepeat(link c) {
        for (link i = c->u; i != c; i = i->u) {
            i->l->r = i; i->r->l = i;
        }
    }
    int calcH() {
        int y, res = 0; ++stamp;
        for (link c = head->r; (y = c->y) <= row && c != head; c = c->r) {
            if (vis[y] != stamp) {
                vis[y] = stamp; ++res;
                for (link i = c->d; i != c; i = i->d)
                    for (link j = i->r; j != i; j = j->r) vis[j->y] = stamp;
            }
        } return res;
    }
    void DFS(int dep) { if (dep + calcH() >= ans) return;
        if (head->r->y > nGE || head->r == head) {
            if (ans > dep) ans = dep; return;
        } link c = NULL;
        for (link i = head->r; i->y <= nGE && i != head; i = i->r)
            if (!c || cntc[i->y] < cntc[c->y]) c = i;
        for (link i = c->d; i != c; i = i->d) {
            removeRepeat(i);
            for (link j = i->r; j != i; j = j->r) if (j->y <= nGE) removeRepeat(j);
            for (link j = i->r; j != i; j = j->r) if (j->y  > nGE) removeExact(base + j->y);
            DFS(dep + 1);
            for (link j = i->l; j != i; j = j->l) if (j->y  > nGE) resumeExact(base + j->y);
            for (link j = i->l; j != i; j = j->l) if (j->y <= nGE) resumeRepeat(j);
            resumeRepeat(i);
        }
    }
    int solve() { build(); ans = INF; DFS(0); return ans; }
}
```

6.9 星期几判定

```cpp
int getDay(int y, int m, int d) {
    if (m <= 2) m += 12, y--;
    if (y < 1752 || (y == 1752 && m < 9) || (y == 1752 && m == 9 && d < 3))
        return (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 + 5) % 7 + 1;
    return (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 - y / 100 + y / 400) % 7 + 1;
}
```

6.10 LCSequence Fast

```cpp
ULL *a, *b, *s, c, d;
for (i = 0, a = appear[(int)B[k]], b = row[max(k - 1, 0)], s = X; i < bitSetLen; ++i)
    *s++ = *a++ | *b++;                    // `X = row[i - 1] or appear[ B[i] ]`
for (i = 0, a = dp, c = d = 0; i < bitSetLen; ++a, c = d, ++i)
    d = *a >> 63, *a = ~((*a << 1) + c);    // `row[i] = -((row[i] << 1) + 1)`
for (i = 0, a = dp, b = X, c = 0; i < bitSetLen; ++a, ++b, ++i)
    d = *b + c, c = (*a >= -d), *a += d;    // `row[i] += X`
for (i = 0, a = dp, b = X; i < bitSetLen; ++a, ++b, ++i)
    *a = (*a ^ *b) & *b;                    // `row[i] = X and (row[i] xor X)`
```

6.11 C Split

```cpp
for (char *tok = strtok(ins, delimiters); tok; tok = strtok(NULL, delimiters))
    puts(tok); // '会破坏原字符串ins'
```

6.12 builtin 系列

- int ___builtin_ffs (unsigned int x) 返回 x 的最后一位 1 的是从后向前第几位, 比如 7368(1110011001000) 返回 4.
- int ___builtin_clz (unsigned int x) 返回前导的 0 的个数.
- int ___builtin_ctz (unsigned int x) 返回后面的 0 个个数, 和 ___builtin_clz 相对.
- int ___builtin_popcount (unsigned int x) 返回二进制表示中 1 的个数.
- int ___builtin_parity (unsigned int x) 返回 x 的奇偶校验位, 也就是 x 的 1 的个数模 2 的结果.

7 Templates

7.1 Eclipse 配置

Exec=env UBUNTU_MENUPROXY= /opt/eclipse/eclipse

preference general keys 把 word completion 设置成 alt+c, 把 content assistant 设置成 alt + /

7.2 C++

```
#pragma comment(linker, "/STACK:10240000")
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <algorithm>
#define Rep(i, a, b) for(int i = (a); i <= (b); ++i)
#define Foru(i, a, b) for(int i = (a); i < (b); ++i)
using namespace std;
typedef long long LL;
typedef pair<int, int> pii;
namespace BufferedReader {
    char buff[MAX_BUFFER + 5], *ptr = buff, c; bool flag;
    bool nextChar(char &c) {
        if ( (c = *ptr++) == 0 ) {
            int tmp = fread(buff, 1, MAX_BUFFER, stdin);
            buff[tmp] = 0; if (tmp == 0) return false;
            ptr = buff; c = *ptr++;
        } return true;
    }
    bool nextUnsignedInt(unsigned int &x) {
        for (;;){if (!nextChar(c)) return false; if ('0'<=c && c<='9') break;}
        for (x=c-'0'; nextChar(c); x = x * 10 + c - '0') if (c < '0' || c > '9') break;
        return true;
    }
    bool nextInt(int &x) {
        for (;;) { if (!nextChar(c)) return false; if (c=='-' || ('0'<=c && c<='9')) break; }
        for ((c=='-') ? (x=0,flag=true) : (x=c-'0',flag=false); nextChar(c); x=x*10+c-'0')
            if (c<'0' || c>'9') break;
        if (flag) x=-x; return true;
    }
};
#endif
```

7.3 Java

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

public class Main {
    public void solve() {}
    public void run() {
        tokenizer = null; out = new PrintWriter(System.out);
        in = new BufferedReader(new InputStreamReader(System.in));
        solve();
        out.close();
    }
    public static void main(String[] args) {
        new Main().run();
    }
    public StringTokenizer tokenizer;
    public BufferedReader in;
    public PrintWriter out;
    public String next() {
        while (tokenizer == null || !tokenizer.hasMoreTokens()) {
            try { tokenizer = new StringTokenizer(in.readLine()); }
            catch (IOException e) { throw new RuntimeException(e); }
        } return tokenizer.nextToken();
    }
}
```

7.4 gcc 配置

在.bashrc 中加入 export CXXFLAGS=”-Wall -Wconversion -Wextra -g3”