1 计算几何

1.1 二维计算几何基本操作

```cpp
const double PI = 3.14159265358979323846264338327950288;
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();
```

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

1.2 圆的面积模板

```
1 struct Event { point p; double alpha; int add; // `构造函数省略`
2     bool operator < (const Event &other) const { return alpha < other.alpha; } };
3 void circleKCover(circle *c, int N, double *area) { // `$area[k]$ : 至少被覆盖$k$次`
4     static bool overlap[MAXN][MAXN], g[MAXN][MAXN];
5     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]);
6     Rep(i, 1, N) Rep(j, 1, N) g[i][j] = !(overlap[i][j] || overlap[j][i] || c[i].disjunct(c[j]));
7     Rep(i, 1, N) { static Event events[MAXN * 2 + 1]; int totE = 0, cnt = 1;
8         Rep(j, 1, N) if (j != i && overlap[j][i]) ++cnt;
9         Rep(j, 1, N) if (j != i && g[i][j]) {
10            circle &a = c[i], &b = c[j]; double l = (a.o − b.o).norm();
11            double s = ((a.r − b.r) * (a.r + b.r) / l + 1) * 0.5;
12            double t = sqrt(−(l − sqr(a.r − b.r)) * (l − sqr(a.r + b.r)) / (l * l * 4.0));
13            point dir = b.o − a.o, nDir = point(−dir.y, dir.x);
14            point aa = a.o + dir * s + nDir * t;
15            point bb = a.o + dir * s − nDir * t;
16            double A = atan2(aa.y − a.o.y, aa.x − a.o.x);
17            double B = atan2(bb.y − a.o.y, bb.x − a.o.x);
18            events[totE++] = Event(bb, B, 1); events[totE++] = Event(aa, A, −1); if (B > A) ++cnt;
19        } if (totE == 0) { area[cnt] += PI * c[i].rSquare; continue; }
20        sort(events, events + totE); events[totE] = events[0];
21        Foru(j, 0, totE) {
22            cnt += events[j].add; area[cnt] += 0.5 * det(events[j].p, events[j + 1].p);
23            double theta = events[j + 1].alpha − events[j].alpha; if (theta < 0) theta += 2.0 * PI;
24            area[cnt] += 0.5 * c[i].rSquare * (theta − sin(theta));
25        }}}
```

1.3 多边形相关

```
1 struct Polygon { // stored in [0, n)
2     int n; point ps[MAXN];
3     Polygon cut(const point &a, const point &b) {
4         static Polygon res; static point o; res.n = 0;
5         for (int i = 0; i < n; ++i) {
6             int s1 = sign(det(ps[i] − a, b − a));
7             int s2 = sign(det(ps[(i + 1) % n] − a, b − a));
8             if (s1 <= 0) res.ps[res.n++] = ps[i];
9             if (s1 * s2 < 0) {
10                lineIntersect(a, b, ps[i], ps[(i + 1) % n], o);
11                res.ps[res.n++] = o;
12            }
13        } return res;
14    }
15    bool contain(const point &p) const { // 1 if on border or inner, 0 if outter
16        static point A, B; int res = 0;
17        for (int i = 0; i < n; ++i) {
18            A = ps[i]; B = ps[(i + 1) % n];
19            if (p.onSeg(A, B)) return 1;
20            if (sign(A.y − B.y) <= 0) swap(A, B);
21            if (sign(p.y − A.y) > 0) continue;
22            if (sign(p.y − B.y) <= 0) continue;
23            res += (int)(sign(det(B − p, A − p)) > 0);
24        } return res & 1;
25    }
26 #define qs(x) (ps[x] − ps[0])
27    bool convexContain(point p) const { // `counter−clockwise`
28        point q = qs(n − 1); p = p − ps[0];
29        if (!p.inAngle(qs(1), q)) return false;
```

```
30        int L = 0, R = n − 1;
31        while (L + 1 < R) { int M((L + R) >> 1);
32            if (p.inAngle(qs(M), q)) L = M; else R = M;
33        } if (L == 0) return false; point l(qs(L)), r(qs(R));
34        return sign( fabs(det(l, p)) + fabs(det(p, r)) + fabs(det(r − l, p − l)) − det(l, r) ) == 0;
35    }
36 #undef qs
37    double isPLAtan2(const point &a, const point &b) {
38        double k = (b − a).alpha(); if (k < 0) k += 2 * PI;
39        return k;
40    }
41    point isPL_Get(const point &a, const point &b, const point &s1, const point &s2) {
42        double k1 = det(b − a, s1 − a), k2 = det(b − a, s2 − a);
43        if (sign(k1) == 0) return s1;
44        if (sign(k2) == 0) return s2;
45        return (s1 * k2 − s2 * k1) / (k2 − k1);
46    }
47    int isPL_Dic(const point &a, const point &b, int l, int r) {
48        int s = (det(b − a, ps[l] − a) < 0) ? −1 : 1;
49        while (l <= r) {
50            int mid = (l + r) / 2;
51            if (det(b − a, ps[mid] − a) * s <= 0) r = mid − 1;
52            else l = mid + 1;
53        }
54        return r + 1;
55    }
56    int isPL_Find(double k, double w[]) {
57        if (k <= w[0] || k > w[n − 1]) return 0;
58        int l = 0, r = n − 1, mid;
59        while (l <= r) {
60            mid = (l + r) / 2;
61            if (w[mid] >= k) r = mid − 1;
62            else l = mid + 1;
63        } return r + 1;
64    }
65    bool isPL(const point &a, const point &b, point &cp1, point &cp2) {  // `$O (log N)$`
66        static double w[MAXN * 2]; // `pay attention to the array size`
67        for (int i = 0; i <= n; ++i) ps[i + n] = ps[i];
68        for (int i = 0; i < n; ++i) w[i] = w[i + n] = isPLAtan2(ps[i], ps[i + 1]);
69        int i = isPL_Find(isPLAtan2(a, b), w);
70        int j = isPL_Find(isPLAtan2(b, a), w);
71        double k1 = det(b − a, ps[i] − a), k2 = det(b − a, ps[j] − a);
72        if (sign(k1) * sign(k2) > 0) return false; // `no intersection`
73        if (sign(k1) == 0 || sign(k2) == 0) { // `intersect with a point or a line in the convex`
74            if (sign(k1) == 0) {
75                if (sign(det(b − a, ps[i + 1] − a)) == 0) cp1 = ps[i], cp2 = ps[i + 1];
76                else cp1 = cp2 = ps[i];
77                return true;
78            }
79            if (sign(k2) == 0) {
80                if (sign(det(b − a, ps[j + 1] − a)) == 0) cp1 = ps[j], cp2 = ps[j + 1];
81                else cp1 = cp2 = ps[j];
82            }
83            return true;
84        }
85        if (i > j) swap(i, j);
86        int x = isPL_Dic(a, b, i, j), y = isPL_Dic(a, b, j, i + n);
87        cp1 = isPL_Get(a, b, ps[x − 1], ps[x]);
88        cp2 = isPL_Get(a, b, ps[y − 1], ps[y]);
89        return true;
90    }
91    double getI(const point &O) const {
92        if (n <= 2) return 0;
93        point G(0.0, 0.0);
94        double S = 0.0, I = 0.0;
95        for (int i = 0; i < n; ++i) {
96            const point &x = ps[i], &y = ps[(i + 1) % n];
97            double d = det(x, y);
98            G = G + (x + y) * d / 3.0;
99            S += d;
100       } G = G / S;
101       for (int i = 0; i < n; ++i) {
102           point x = ps[i] − G, y = ps[(i + 1) % n] − G;
103           I += fabs(det(x, y)) * (x.norm() + dot(x, y) + y.norm());
104       }
```

```
105        return I = I / 12.0 + fabs(S * 0.5) * (O - G).norm();
106    }
107 };
```

1.4 直线与凸包求交点

```
 1 int isPL(point a, point b, vector<point> &res) { // `点逆时针给出，无三点共线`
 2     static double theta[MAXN];
 3     for (int i = 0; i < n; ++i) theta[i] = (list[(i + 1) % n] - list[i]).atan2();
 4     double delta = theta[0];
 5     for (int i = 0; i < n; ++i) theta[i] = normalize(theta[i] - delta);
 6     int x = lower_bound(theta, theta + n, normalize((b - a).atan2() - delta)) - theta;
 7     int y = lower_bound(theta, theta + n, normalize((a - b).atan2() - delta)) - theta;
 8     for (int k = 0; k <= 1; ++k, swap(a, b), swap(x, y)) {
 9         if (y < x) y += n;
10         int l = x, r = y, m;
11         while (l + 1 < r) {
12             if (sign(det(b - a, list[(m = (l + r) / 2) % n] - a)) < 0) l = m;
13             else r = m;
14         }
15         l %= n, r %= n;
16         if (sign(det(b - a, list[r] - list[l])) == 0) {
17             if (sign(det(b - a, list[l] - a)) == 0)
18                 return -l; // `直线与 $(list[l], list[r])$ 重合`
19         }
20         else {
21             point p; lineIntersect(list[l], list[r], a, b, p);
22             if (p.onSeg(list[l], list[r]))
23                 res.push_back(p);
24         }
25     }
26     return res.size();
27 }
```

1.5 半平面交

```
 1 struct Border {
 2     point p1, p2; double alpha;
 3     Border() : p1(), p2(), alpha(0.0) {}
 4     Border(const point &a, const point &b): p1(a), p2(b), alpha( atan2(p2.y - p1.y, p2.x - p1.x) ) {}
 5     bool operator == (const Border &b) const { return sign(alpha - b.alpha) == 0; }
 6     bool operator < (const Border &b) const {
 7         int c = sign(alpha - b.alpha); if (c != 0) return c > 0;
 8         return sign(det(b.p2 - b.p1, p1 - b.p1)) >= 0;
 9     }
10 };
11 point isBorder(const Border &a, const Border &b) { // a and b should not be parallel
12     point is; lineIntersect(a.p1, a.p2, b.p1, b.p2, is); return is;
13 }
14 bool checkBorder(const Border &a, const Border &b, const Border &me) {
15     point is; lineIntersect(a.p1, a.p2, b.p1, b.p2, is);
16     return sign(det(me.p2 - me.p1, is - me.p1)) > 0;
17 }
18 double HPI(int N, Border border[]) {
19     static Border que[MAXN * 2 + 1]; static point ps[MAXN];
20     int head = 0, tail = 0, cnt = 0; // [head, tail)
21     sort(border, border + N); N = unique(border, border + N) - border;
22     for (int i = 0; i < N; ++i) {
23         Border &cur = border[i];
24         while (head + 1 < tail && !checkBorder(que[tail - 2], que[tail - 1], cur)) --tail;
25         while (head + 1 < tail && !checkBorder(que[head], que[head + 1], cur)) ++head;
26         que[tail++] = cur;
27     } while (head + 1 < tail && !checkBorder(que[tail - 2], que[tail - 1], que[head])) --tail;
28     while (head + 1 < tail && !checkBorder(que[head], que[head + 1], que[tail - 1])) ++head;
29     if (tail - head <= 2) return 0.0;
30     Foru(i, head, tail) ps[cnt++] = isBorder(que[i], que[(i + 1 == tail) ? (head) : (i + 1)]);
31     double area = 0; Foru(i, 0, cnt) area += det(ps[i], ps[(i + 1) % cnt]);
32     return fabs(area * 0.5); // or (-area * 0.5)
33 }
```

1.6 最大面积空凸包

```
 1 inline bool toUpRight(const point &a, const point &b) {
 2     int c = sign(b.y - a.y); if (c > 0) return true;
 3     return c == 0 && sign(b.x - a.x) > 0;
 4 }
 5 inline bool cmpByPolarAngle(const point &a, const point &b) { // `counter-clockwise, shorter first if they share
         the same polar angle`
 6     int c = sign(det(a, b)); if (c != 0) return c > 0;
 7     return sign(b.len() - a.len()) > 0;
 8 }
 9 double maxEmptyConvexHull(int N, point p[]) {
10     static double dp[MAXN][MAXN];
11     static point vec[MAXN];
12     static int seq[MAXN]; // `empty triangles formed with $(0, 0), vec[o], vec[ seq[i] ]$`
13     double ans = 0.0;
14     Rep(o, 1, N) {
15         int totVec = 0;
16         Rep(i, 1, N) if (toUpRight(p[o], p[i])) vec[++totVec] = p[i] - p[o];
17         sort(vec + 1, vec + totVec + 1, cmpByPolarAngle);
18         Rep(i, 1, totVec) Rep(j, 1, totVec) dp[i][j] = 0.0;
19         Rep(k, 2, totVec) {
20             int i = k - 1;
21             while (i > 0 && sign( det(vec[k], vec[i]) ) == 0) --i;
22             int totSeq = 0;
23             for (int j = i; j > 0; i = j) {
24                 seq[++totSeq] = i;
25                 for (j = i - 1; j > 0 && sign(det(vec[i] - vec[k], vec[j] - vec[k])) > 0; --j);
26                 double v = det(vec[i], vec[k]) * 0.5;
27                 if (j > 0) v += dp[i][j];
28                 dp[k][i] = v;
29                 cMax(ans, v);
30             } for (int i = totSeq - 1; i >= 1; --i) cMax( dp[k][ seq[i] ], dp[k][seq[i + 1]] );
31         }
32     } return ans;
33 }
```

1.7 最近点对

```
 1 int N; point p[maxn];
 2 bool cmpByX(const point &a, const point &b) { return sign(a.x - b.x) < 0; }
 3 bool cmpByY(const int &a, const int &b) { return p[a].y < p[b].y; }
 4 double minimalDistance(point *c, int n, int *ys) {
 5     double ret = 1e+20;
 6     if (n < 20) {
 7         Foru(i, 0, n) Foru(j, i + 1, n) cMin(ret, (c[i] - c[j]).len() );
 8         sort(ys, ys + n, cmpByY); return ret;
 9     } static int mergeTo[maxn];
10     int mid = n / 2; double xmid = c[mid].x;
11     ret = min(minimalDistance(c, mid, ys), minimalDistance(c + mid, n - mid, ys + mid));
12     merge(ys, ys + mid, ys + mid, ys + n, mergeTo, cmpByY);
13     copy(mergeTo, mergeTo + n, ys);
14     Foru(i, 0, n) {
15         while (i < n && sign(fabs(p[ys[i]].x - xmid) - ret) > 0) ++i;
16         int cnt = 0;
17         Foru(j, i + 1, n)
18             if (sign(p[ys[j]].y - p[ys[i]].y - ret) > 0) break;
19             else if (sign(fabs(p[ys[j]].x - xmid) - ret) <= 0) {
20                 ret = min(ret, (p[ys[i]] - p[ys[j]]).len());
21                 if (++cnt >= 10) break;
22             }
23     } return ret;
24 }
25 double work() {
26     sort(p, p + n, cmpByX); Foru(i, 0, n) ys[i] = i; return minimalDistance(p, n, ys);
27 }
```

1.8 凸包与点集直径

```cpp
1  vector<point> convexHull(int n, point ps[]) { // `counter-clockwise, strict`
2      static point qs[MAXN * 2];
3      sort(ps, ps + n, cmpByXY);
4      if (n <= 2) return vector<point>(ps, ps + n);
5      int k = 0;
6      for (int i = 0; i < n; qs[k++] = ps[i++])
7          while (k > 1 && det(qs[k - 1] - qs[k - 2], ps[i] - qs[k - 1]) < EPS) --k;
8      for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
9          while (k > t && det(qs[k - 1] - qs[k - 2], ps[i] - qs[k - 1]) < EPS) --k;
10     return vector<point>(qs, qs + k);
11 }
12 double convexDiameter(int n, point ps[]) {
13     if (n < 2) return 0; if (n == 2) return (ps[1] - ps[0]).len();
14     double k, ans = 0;
15     for (int x = 0, y = 1, nx, ny; x < n; ++x) {
16         for(nx = (x == n - 1) ? (0) : (x + 1); ; y = ny) {
17             ny = (y == n - 1) ? (0) : (y + 1);
18             if ( sign(k = det(ps[nx] - ps[x], ps[ny] - ps[y])) <= 0) break;
19         } ans = max(ans, (ps[x] - ps[y]).len());
20         if (sign(k) == 0) ans = max(ans, (ps[x] - ps[ny]).len());
21     } return ans;
22 }
```

1.9 Farmland

```cpp
1  struct node { int begin[MAXN], *end; } a[MAXN]; // `按对 $p[i]$ 的极角的 atan2 值排序`
2  bool check(int n, point p[], int b1, int b2, bool vis[MAXN][MAXN]) {
3      static pii l[MAXN * 2 + 1]; static bool used[MAXN];
4      int tp(0), *k, p, p1, p2; double area(0.0);
5      for (l[0] = pii(b1, b2); ; ) {
6          vis[p1 = l[tp].first][p2 = l[tp].second] = true;
7          area += det(p[p1], p[p2]);
8          for (k = a[p2].begin; k != a[p2].end; ++k) if (*k == p1) break;
9          k = (k == a[p2].begin) ? (a[p2].end - 1) : (k - 1);
10         if ((l[++tp] = pii(p2, *k)) == l[0]) break;
11     } if (sign(area) < 0 || tp < 3) return false;
12     Rep(i, 1, n) used[i] = false;
13     for (int i = 0; i < tp; ++i) if (used[p = l[i].first]) return false; else used[p] = true;
14     return true; // `a face with tp vertices`
15 }
16 int countFaces(int n, point p[]) {
17     static bool vis[MAXN][MAXN]; int ans = 0;
18     Rep(x, 1, n) Rep(y, 1, n) vis[x][y] = false;
19     Rep(x, 1, n) for (int *itr = a[x].begin; itr != a[x].end; ++itr) if (!vis[x][*itr])
20         if (check(n, p, x, *itr, vis)) ++ans;
21     return ans;
22 }
```

1.10 Voronoi 图

不能有重点, 点数应当不小于 2

```cpp
1  #define Oi(e) ((e)->oi)
2  #define Dt(e) ((e)->dt)
3  #define On(e) ((e)->on)
4  #define Op(e) ((e)->op)
5  #define Dn(e) ((e)->dn)
6  #define Dp(e) ((e)->dp)
7  #define Other(e, p) ((e)->oi == p ? (e)->dt : (e)->oi)
8  #define Next(e, p) ((e)->oi == p ? (e)->on : (e)->dn)
9  #define Prev(e, p) ((e)->oi == p ? (e)->op : (e)->dp)
10 #define V(p1, p2, u, v) (u = p2->x - p1->x, v = p2->y - p1->y)
11 #define C2(u1, v1, u2, v2) (u1 * v2 - v1 * u2)
12 #define C3(p1, p2, p3) ((p2->x - p1->x) * (p3->y - p1->y) - (p2->y - p1->y) * (p3->x - p1->x))
13 #define Dot(u1, v1, u2, v2) (u1 * u2 + v1 * v2)
14 #define dis(a,b) (sqrt( (a->x - b->x) * (a->x - b->x) + (a->y - b->y) * (a->y - b->y) ))
15 const int maxn = 110024;
16 const int aix = 4;
17 const double eps = 1e-7;
18 int n, M, k;
```

```cpp
19 struct gEdge {
20     int u, v; double w;
21     bool operator <(const gEdge &e1) const { return w < e1.w - eps; }
22 } E[aix * maxn], MST[maxn];
23 struct point {
24     double x, y; int index; edge *in;
25     bool operator <(const point &p1) const { return x < p1.x - eps || (abs(x - p1.x) <= eps && y < p1.y - eps); }
26 };
27 struct edge { point *oi, *dt; edge *on, *op, *dn, *dp; };
28
29 point p[maxn], *Q[maxn];
30 edge mem[aix * maxn], *elist[aix * maxn];
31 int nfree;
32 void Alloc_memory() { nfree = aix * n; edge *e = mem; for (int i = 0; i < nfree; i++) elist[i] = e++; }
33 void Splice(edge *a, edge *b, point *v) {
34     edge *next;
35     if (Oi(a) == v) next = On(a), On(a) = b; else next = Dn(a), Dn(a) = b;
36     if (Oi(next) == v) Op(next) = b; else Dp(next) = b;
37     if (Oi(b) == v) On(b) = next, Op(b) = a; else Dn(b) = next, Dp(b) = a;
38 }
39 edge *Make_edge(point *u, point *v) {
40     edge *e = elist[--nfree];
41     e->on = e->op = e->dn = e->dp = e; e->oi = u; e->dt = v;
42     if (!u->in) u->in = e;
43     if (!v->in) v->in = e;
44     return e;
45 }
46 edge *Join(edge *a, point *u, edge *b, point *v, int side) {
47     edge *e = Make_edge(u, v);
48     if (side == 1) {
49         if (Oi(a) == u) Splice(Op(a), e, u);
50         else Splice(Dp(a), e, u);
51         Splice(b, e, v);
52     } else {
53         Splice(a, e, u);
54         if (Oi(b) == v) Splice(Op(b), e, v);
55         else Splice(Dp(b), e, v);
56     } return e;
57 }
58 void Remove(edge *e) {
59     point *u = Oi(e), *v = Dt(e);
60     if (u->in == e) u->in = e->on;
61     if (v->in == e) v->in = e->dn;
62     if (Oi(e->on) == u) e->on->op = e->op; else e->on->dp = e->op;
63     if (Oi(e->op) == u) e->op->op = e->on; else e->op->dn = e->on;
64     if (Oi(e->dn) == v) e->dn->op = e->dp; else e->dn->dp = e->dp;
65     if (Oi(e->dp) == v) e->dp->on = e->dn; else e->dp->dn = e->dn;
66     elist[nfree++] = e;
67 }
68 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
       ) {
69     for (point *d_l = Other(e_l, o_l), *d_r = Other(e_r, o_r); ; )
70         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);
71         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);
72         else break;
73     *OL = o_l, *OR = o_r; *l_low = e_l, *r_low = e_r;
74 }
75 void Merge(edge *lr, point *s, edge *rl, point *u, edge **tangent) {
76     double l1, l2, l3, l4, r1, r2, r3, r4, cot_L, cot_R, u1, v1, u2, v2, n1, cot_n, P1, cot_P;
77     point *O, *D, *OR, *OL; edge *B, *L, *R;
78     Low_tangent(lr, s, rl, u, &L, &OL, &R, &OR);
79     for (*tangent = B = Join(L, OL, R, OR, 0), O = OL, D = OR; ; ) {
80         edge *El = Next(B, O), *Er = Prev(B, D), *next, *prev;
81         point *l = Other(El, O), *r = Other(Er, D);
82         V(l, O, l1, l2); V(l, D, l3, l4); V(r, O, r1, r2); V(r, D, r3, r4);
83         double cl = C2(l1, l2, l3, l4), cr = C2(r1, r2, r3, r4);
84         bool BL = cl > eps, BR = cr > eps;
85         if (!BL && !BR) break;
86         if (BL) {
87             double dl = Dot(l1, l2, l3, l4);
88             for (cot_L = dl / cl; ; Remove(El), El = next, cot_L = cot_n) {
89                 next = Next(El, O); V(Other(next, O), O, u1, v1); V(Other(next, O), D, u2, v2);
90                 n1 = C2(u1, v1, u2, v2); if (!(n1 > eps)) break;
91                 cot_n = Dot(u1, v1, u2, v2) / n1;
92                 if (cot_n > cot_L) break;
```

```
 93                }
 94            } if (BR) {
 95                double dr = Dot(r1, r2, r3, r4);
 96                for (cot_R = dr / cr; ; Remove(Er), Er = prev, cot_R = cot_P) {
 97                    prev = Prev(Er, D); V(Other(prev, D), O, u1, v1); V(Other(prev, D), D, u2, v2);
 98                    P1 = C2(u1, v1, u2, v2); if (!(P1 > eps)) break;
 99                    cot_P = Dot(u1, v1, u2, v2) / P1;
100                    if (cot_P > cot_R) break;
101                }
102            } l = Other(El, O); r = Other(Er, D);
103            if (!BL || (BL && BR && cot_R < cot_L)) B = Join(B, O, Er, r, 0), D = r;
104            else B = Join(El, l, B, D, 0), O = l;
105        }
106 }
107 void Divide(int s, int t, edge **L, edge **R) {
108     edge *a, *b, *c, *ll, *lr, *rl, *rr, *tangent;
109     int n = t − s + 1;
110     if (n == 2) *L = *R = Make_edge(Q[s], Q[t]);
111     else if (n == 3) {
112         a = Make_edge(Q[s], Q[s + 1]), b = Make_edge(Q[s + 1], Q[t]);
113         Splice(a, b, Q[s + 1]);
114         double v = C3(Q[s], Q[s + 1], Q[t]);
115         if (v > eps)        c = Join(a, Q[s], b, Q[t], 0), *L = a, *R = b;
116         else if (v < −eps) c = Join(a, Q[s], b, Q[t], 1), *L = c, *R = c;
117         else *L = a, *R = b;
118     } else if (n > 3) {
119         int split = (s + t) / 2;
120         Divide(s, split, &ll, &lr); Divide(split + 1, t, &rl, &rr);
121         Merge(lr, Q[split], rl, Q[split + 1], &tangent);
122         if (Oi(tangent) == Q[s]) ll = tangent;
123         if (Dt(tangent) == Q[t]) rr = tangent;
124         *L = ll; *R = rr;
125     }
126 }
127 void Make_Graph() {
128     edge *start, *e; point *u, *v;
129     for (int i = 0; i < n; i++) {
130         start = e = (u = &p[i])→in;
131         do{ v = Other(e, u);
132             if (u < v) E[M++].u = (u − p, v − p, dis(u, v)); // M < aix * maxn
133         } while ((e = Next(e, u)) != start);
134     }
135 }
136 int b[maxn];
137 int Find(int x) { while (x != b[x]) { b[x] = b[b[x]]; x = b[x]; } return x; }
138 void Kruskal() {
139     memset(b, 0, sizeof(b)); sort(E, E + M);
140     for (int i = 0; i < n; i++) b[i] = i;
141     for (int i = 0, kk = 0; i < M && kk < n − 1; i++) {
142         int m1 = Find(E[i].u), m2 = Find(E[i].v);
143         if (m1 != m2) b[m1] = m2, MST[kk++] = E[i];
144     }
145 }
146 void solve() {
147     scanf("%d", &n);
148     for (int i = 0; i < n; i++) scanf("%lf%lf", &p[i].x, &p[i].y), p[i].index = i, p[i].in = NULL;
149     Alloc_memory(); sort(p, p + n);
150     for (int i = 0; i < n; i++) Q[i] = p + i;
151     edge *L, *R; Divide(0, n − 1, &L, &R);
152     M = 0; Make_Graph(); Kruskal();
153 }
154 int main() { solve(); return 0; }
```

1.11　四边形双费马点

```
 1 typedef complex<double> Tpoint;
 2 const double eps = 1e−8;
 3 const double sqrt3 = sqrt(3.0);
 4 bool cmp(const Tpoint &a, const Tpoint &b) {
 5     return a.real() < b.real() − eps || (a.real() < b.real() + eps && a.imag() < b.imag());
 6 }
 7 Tpoint rotate(const Tpoint &a, const Tpoint &b, const Tpoint &c) {
 8     Tpoint d = b − a; d = Tpoint(−d.imag(), d.real());
```

```
 9     if (Sign(cross(a, b, c)) == Sign(cross(a, b, a + d))) d *= −1.0;
10     return unit(d);
11 }
12 Tpoint p[10], a[10], b[10];
13 int N, T;
14 double totlen(const Tpoint &p, const Tpoint &a, const Tpoint &b, const Tpoint &c) {
15     return abs(p − a) + abs(p − b) + abs(p − c);
16 }
17 double fermat(const Tpoint &x, const Tpoint &y, const Tpoint &z, Tpoint &cp) {
18     a[0] = a[3] = x; a[1] = a[4] = y; a[2] = a[5] = z;
19     double len = 1e100, len2;
20     for (int i = 0; i < 3; i++) {
21         len2 = totlen(a[i], x, y, z);
22         if (len2 < len) len = len2, cp = a[i];
23     }
24     for (int i = 0; i < 3; i++) {
25         b[i] = rotate(a[i + 1], a[i], a[i + 2]);
26         b[i] = (a[i + 1] + a[i]) / 2.0 + b[i] * (abs(a[i + 1] − a[i]) * sqrt3 / 2.0);
27     }
28     b[3] = b[0];
29     Tpoint cp2 = intersect(b[0], a[2], b[1], a[3]);
30     len2 = totlen(cp2, x, y, z);
31     if (len2 < len) len = len2, cp = cp2;
32     return len;
33 }
34 double getans(const Tpoint &a) {
35     double len = 0; for (int i = 0; i < N; i++) len += abs(a − p[i]);
36     return len;
37 }
38 double mindist(const Tpoint &p, const Tpoint &a, const Tpoint &b, const Tpoint &c, const Tpoint &d) {
39     return min( min(abs(p − a), abs(p − b)), min(abs(p − c), abs(p − d)));
40 }
41 int main() {
42     N = 4;
43     for (cin >> T; T; T−−) {
44         double ret = 1e100, len_cur, len_before, len1, len2, len;
45         Tpoint cp, cp1, cp2;
46         Foru(i, 0, N) cin >> p[i];
47         Foru(i, 0, N) ret = min(ret, getans(p[i]));
48         Foru(i, 1, N) Foru(j, 1, N) if (j != i) Foru(k, 1, N) if (k != i && k != j) {
49             cMin(ret, abs(p[0] − p[i]) + abs(p[j] − p[k])
50                 + min( min(abs(p[0] − p[j]), abs(p[0] − p[k])),
51                     min(abs(p[i] − p[j]), abs(p[i] − p[k]))
52                 ));
53             ret = min(ret, getans(intersect(p[0], p[i], p[j], p[k])));
54         }
55         Foru(i, 0, N) Foru(j, i + 1, N) Foru(k, j + 1, N) {
56             double len = fermat(p[i], p[j], p[k], cp);
57             ret = min(ret, len + mindist(p[6 − i − j − k], p[i], p[j], p[k], cp));
58         }
59         sort(p, p + N, cmp);
60         for (int i = 1; i < N; i++) {
61             cp1 = (p[0] + p[i]) / 2.0;
62             int j, k;
63             for (j = 1; j < N && j == i; j++);
64             for (k = 6 − i − j, len_before = 1e100; ; ) {
65                 len1 = fermat(cp1, p[j], p[k], cp2);
66                 len1 = fermat(cp2, p[0], p[i], cp1);
67                 len = len1 + abs(cp2 − p[j]) + abs(cp2 − p[k]);
68                 if (len < len_before − (1e−6)) len_before = len;
69                 else break;
70             } ret = min(ret, len_before);
71         } printf("%.4f\n", ret);
72     }
73     return 0;
74 }
```

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 $\triangle ABF$, $\triangle BCD$, $\triangle CAE$, 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

```
1  struct point { double x, y, z; // something omitted
2      friend point det(const point &a, const point &b) {
3          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);
4      }
5      friend double mix(const point &a, const point &b, const point &c) {
6          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;
7      }
8      double distLP(const point &p1, const point &p2) const {
9          return det(p2 − p1, *this − p1).len() / (p2 − p1).len();
10     }
11     double distFP(const point &p1, const point &p2, const point &p3) const {
12         point n = det(p2 − p1, p3 − p1); return fabs( dot(n, *this − p1) / n.len() );
13     }
14  };
15  double distLL(const point &p1, const point &p2, const point &q1, const point &q2) {
16      point p = q1 − p1, u = p2 − p1, v = q2 − q1;
17      double d = u.norm() * v.norm() − dot(u, v) * dot(u, v);
18      if (sign(d) == 0) return p1.distLP(q1, q2);
19      double s = (dot(p, u) * v.norm() − dot(p, v) * dot(u, v)) / d;
20      return (p1 + u * s).distLP(q1, q2);
21  }
22  double distSS(const point &p1, const point &p2, const point &q1, const point &q2) {
23      point p = q1 − p1, u = p2 − p1, v = q2 − q1;
24      double d = u.norm() * v.norm() − dot(u, v) * dot(u, v);
25      if (sign(d) == 0) return min( min((p1 − q1).len(), (p1 − q2).len()),
26                                    min((p2 − q1).len(), (p2 − q2).len()));
27      double s1 = (dot(p, u) * v.norm() − dot(p, v) * dot(u, v)) / d;
28      double s2 = (dot(p, v) * u.norm() − dot(p, u) * dot(u, v)) / d;
29      if (s1 < 0.0) s1 = 0.0; if (s1 > 1.0) s1 = 1.0;
30      if (s2 < 0.0) s2 = 0.0; if (s2 > 1.0) s2 = 1.0;
31      point r1 = p1 + u * s1; point r2 = q1 + v * s2;
32      return (r1 − r2).len();
33  }
34  bool isFL(const point &p, const point &o, const point &q1, const point &q2, point &res) {
35      double a = dot(o, q2 − p), b = dot(o, q1 − p), d = a − b;
36      if (sign(d) == 0) return false;
37      res = (q1 * a − q2 * b) / d;
38      return true;
39  }
40  bool isFF(const point &p1, const point &o1, const point &p2, const point &o2, point &a, point &b) {
41      point e = det(o1, o2), v = det(o1, e);
42      double d = dot(o2, v); if (sign(d) == 0) return false;
43      point q = p1 + v * (dot(o2, p2 − p1) / d);
44      a = q; b = q + e;
45      return true;
46  }
```

1.14 凸多面体切割

```
1  vector<vector<point> > convexCut(const vector<vector<point> > &pss, const point &p, const point &o) {
2      vector<vector<point> > res;
3      vector<point> sec;
4      for (unsigned itr = 0, size = pss.size(); itr < size; ++itr) {
5          const vector<point> &ps = pss[itr];
6          int n = ps.size();
7          vector<point> qs;
```

```
8          bool dif = false;
9          for (int i = 0; i < n; ++i) {
10             int d1 = sign( dot(o, ps[i] − p) );
11             int d2 = sign( dot(o, ps[(i + 1) % n] − p) );
12             if (d1 <= 0) qs.push_back(ps[i]);
13             if (d1 * d2 < 0) {
14                 point q;
15                 isFL(p, o, ps[i], ps[(i + 1) % n], q); // must return true
16                 qs.push_back(q);
17                 sec.push_back(q);
18             }
19             if (d1 == 0) sec.push_back(ps[i]);
20             else dif = true;
21             dif |= dot(o, det(ps[(i + 1) % n] − ps[i], ps[(i + 2) % n] − ps[i])) < −EPS;
22         }
23         if (!qs.empty() && dif)
24             res.insert(res.end(), qs.begin(), qs.end());
25     }
26     if (!sec.empty()) {
27         vector<point> tmp( convexHull2D(sec, o) );
28         res.insert(res.end(), tmp.begin(), tmp.end());
29     }
30     return res;
31  }
32
33  vector<vector<point> > initConvex() {
34      vector<vector<point> > pss(6, vector<point>(4));
35      pss[0][0] = pss[1][0] = pss[2][0] = point(−INF, −INF, −INF);
36      pss[0][3] = pss[1][1] = pss[5][2] = point(−INF, −INF,  INF);
37      pss[0][1] = pss[2][3] = pss[4][2] = point(−INF,  INF, −INF);
38      pss[0][2] = pss[5][3] = pss[4][1] = point(−INF,  INF,  INF);
39      pss[1][3] = pss[2][1] = pss[3][2] = point( INF, −INF, −INF);
40      pss[1][2] = pss[5][1] = pss[3][3] = point( INF, −INF,  INF);
41      pss[2][2] = pss[4][3] = pss[3][1] = point( INF,  INF, −INF);
42      pss[5][0] = pss[4][0] = pss[3][0] = point( INF,  INF,  INF);
43      return pss;
44  }
```

1.15 三维凸包

不能有重点

```
1  namespace ConvexHull3D {
2      #define volume(a, b, c, d) (mix(ps[b] − ps[a], ps[c] − ps[a], ps[d] − ps[a]))
3      vector<Facet> getHull(int n, point ps[]) {
4          static int mark[MAXN][MAXN], a, b, c;
5          int stamp = 0;
6          bool exist = false;
7          vector<Facet> facet;
8          random_shuffle(ps, ps + n);
9          for (int i = 2; i < n && !exist; i++) {
10             point ndir = det(ps[0] − ps[i], ps[1] − ps[i]);
11             if (ndir.len() < EPS) continue;
12             swap(ps[i], ps[2]);
13             for (int j = i + 1; j < n && !exist; j++)
14                 if (sign(volume(0, 1, 2, j)) != 0) {
15                     exist = true;
16                     swap(ps[j], ps[3]);
17                     facet.push_back(Facet(0, 1, 2));
18                     facet.push_back(Facet(0, 2, 1));
19                 }
20         }
21         if (!exist) return ConvexHull2D(n, ps);
22         for (int i = 0; i < n; ++i)
23             for (int j = 0; j < n; ++j)
24                 mark[i][j] = 0;
25         stamp = 0;
26         for (int v = 3; v < n; ++v) {
27             vector<Facet> tmp;
28             ++stamp;
29             for (unsigned i = 0; i < facet.size(); i++) {
30                 a = facet[i].a;
31                 b = facet[i].b;
32                 c = facet[i].c;
```

```
33              if (sign(volume(v, a, b, c)) < 0)
34                  mark[a][b] = mark[a][c] =
35                  mark[b][a] = mark[b][c] =
36                  mark[c][a] = mark[c][b] = stamp;
37              else tmp.push_back(facet[i]);
38          } facet = tmp;
39          for (unsigned i = 0; i < tmp.size(); i++) {
40              a = facet[i].a; b = facet[i].b; c = facet[i].c;
41              if (mark[a][b] == stamp) facet.push_back(Facet(b, a, v));
42              if (mark[b][c] == stamp) facet.push_back(Facet(c, b, v));
43              if (mark[c][a] == stamp) facet.push_back(Facet(a, c, v));
44          }
45      } return facet;
46  }
47  #undef volume
48  }
49  namespace Gravity {
50      using ConvexHull3D::Facet;
51      point findG(point ps[], const vector<Facet> &facet) {
52          double ws = 0; point res(0.0, 0.0, 0.0), o = ps[ facet[0].a ];
53          for (int i = 0, size = facet.size(); i < size; ++i) {
54              const point &a = ps[ facet[i].a ], &b = ps[ facet[i].b ], &c = ps[ facet[i].c ];
55              point p = (a + b + c + o) * 0.25;
56              double w = mix(a - o, b - o, c - o);
57              ws += w;
58              res = res + p * w;
59          } res = res / ws;
60          return res;
61      }
62  }
```

1.16 球面点表面点距离

```
1  double distOnBall(double lati1, double longi1, double lati2, double longi2, double R) {
2      lati1 *= PI / 180; longi1 *= PI / 180;
3      lati2 *= PI / 180; longi2 *= PI / 180;
4      double x1 = cos(lati1) * sin(longi1);
5      double y1 = cos(lati1) * cos(longi1);
6      double z1 = sin(lati1);
7      double x2 = cos(lati2) * sin(longi2);
8      double y2 = cos(lati2) * cos(longi2);
9      double z2 = sin(lati2);
10     double theta = acos(x1 * x2 + y1 * y2 + z1 * z2);
11     return R * theta;
12 }
```

1.17 长方体表面点距离

```
1  int r;
2  void turn(int i, int j, int x, int y, int z, int x0, int y0, int L, int W, int H) {
3      if (z == 0) r = min(r, x * x + y * y);
4      else {
5          if (i >= 0 && i < 2)  turn(i + 1, j, x0 + L + z, y, x0 + L - x, x0 + L, y0, H, W, L);
6          if (j >= 0 && j < 2)  turn(i, j + 1, x, y0 + W + z, y0 + W - y, x0, y0 + W, L, H, W);
7          if (i <= 0 && i > -2) turn(i - 1, j, x0 - z, y, x - x0, x0 - H, y0, H, W, L);
8          if (j <= 0 && j > -2) turn(i, j - 1, x, y0 - z, y - y0, x0, y0 - H, L, H, W);
9      }
10 }
11 int calc(int L, int H, int W, int x1, int y1, int z1, int x2, int y2, int z2) {
12     if (z1 != 0 && z1 != H)
13         if (y1 == 0 || y1 == W) swap(y1, z1), swap(y2, z2), swap(W, H);
14         else                    swap(x1, z1), swap(x2, z2), swap(L, H);
15     if (z1 == H) z1 = 0, z2 = H - z2;
16     r = INF; turn(0, 0, x2 - x1, y2 - y1, z2, -x1, -y1, L, W, H);
17     return r;
18 }
```

1.18 最小覆盖球

```
1  int outCnt; point out[4], res; double radius;
2  void ball() {
3      static point q[3];
4      static double m[3][3], sol[3], L[3], det;
5      int i, j; res = point(0.0, 0.0, 0.0); radius = 0.0;
6      switch (outCnt) {
7      case 1: res = out[0]; break;
8      case 2: res = (out[0] + out[1]) * 0.5; radius = (res - out[0]).norm();
9          break;
10     case 3:
11         q[0] = out[1] - out[0]; q[1] = out[2] - out[0];
12         for (i = 0; i < 2; ++i) for (j = 0; j < 2; ++j)
13             m[i][j] = dot(q[i], q[j]) * 2.0;
14         for (i = 0; i < 2; ++i) sol[i] = dot(q[i], q[i]);
15         det = m[0][0] * m[1][1] - m[0][1] * m[1][0];
16         if (sign(det) == 0) return;
17         L[0] = (sol[0] * m[1][1] - sol[1] * m[0][1]) / det;
18         L[1] = (sol[1] * m[0][0] - sol[0] * m[1][0]) / det;
19         res = out[0] + q[0] * L[0] + q[1] * L[1];
20         radius = (res - out[0]).norm();
21         break;
22     case 4:
23         q[0] = out[1] - out[0]; q[1] = out[2] - out[0]; q[2] = out[3] - out[0];
24         for (i = 0; i < 3; ++i) for (j = 0; j < 3; ++j) m[i][j] = dot(q[i], q[j]) * 2;
25         for (i = 0; i < 3; ++i) sol[i] = dot(q[i], q[i]);
26         det = m[0][0] * m[1][1] * m[2][2] + m[0][1] * m[1][2] * m[2][0]
27             + m[0][2] * m[2][1] * m[1][0] - m[0][2] * m[1][1] * m[2][0]
28             - m[0][1] * m[1][0] * m[2][2] - m[0][0] * m[1][2] * m[2][1];
29         if (sign(det) == 0) return;
30         for (j = 0; j < 3; ++j) { for (i = 0; i < 3; ++i) m[i][j] = sol[i];
31             L[j] = (m[0][0] * m[1][1] * m[2][2] + m[0][1] * m[1][2] * m[2][0]
32                 + m[0][2] * m[2][1] * m[1][0] - m[0][2] * m[1][1] * m[2][0]
33                 - m[0][1] * m[1][0] * m[2][2] - m[0][0] * m[1][2] * m[2][1]) / det;
34             for (i = 0; i < 3; ++i) m[i][j] = dot(q[i], q[j]) * 2;
35         } res = out[0];
36         for (i = 0; i < 3; ++i) res += q[i] * L[i]; radius = (res - out[0]).norm();
37     }
38 }
39 void minball(int n, point pt[]) {
40     ball();
41     if (outCnt < 4) for (int i = 0; i < n; ++i)
42         if ((res - pt[i]).norm() > +radius + EPS) {
43             out[outCnt] = pt[i]; ++outCnt; minball(i, pt); --outCnt;
44             if (i > 0) {
45                 point Tt = pt[i];
46                 memmove(&pt[1], &pt[0], sizeof(point) * i);
47                 pt[0] = Tt;
48             }
49         }
50 }
51 pair<point, double> main(int npoint, point pt[]) { // 0-based
52     random_shuffle(pt, pt + npoint); radius = -1;
53     for (int i = 0; i < npoint; i++) { if ((res - pt[i]).norm() > EPS + radius) {
54         outCnt = 1; out[0] = pt[i]; minball(i, pt); } }
55     return make_pair(res, sqrt(radius));
56 }
```

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:

$$\begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta \\ u_y u_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x\sin\theta \\ u_z u_x(1-\cos\theta) - u_y\sin\theta & u_z u_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix}$$

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

- 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 $H = I - 2\frac{vv^T}{v^T v}$,

- 点 a 对称点: $a' = a - 2\frac{v^T a}{v^T v} \cdot v$

1.20 立体角

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

2 数据结构

2.1 动态凸包 (只支持插入)

```
1  #define x first   // `upperHull $\leftarrow (x, y)$`
2  #define y second  // `lowerHull $\leftarrow (x, -y)$`
3  typedef map<int, int> mii;
4  typedef map<int, int>::iterator mit;
5  struct point { point(const mit &p): x(p->first), y(p->second) {} };
6  inline bool checkInside(mii &a, const point &p) { // `border inclusive`
7      int x = p.x, y = p.y; mit p1 = a.lower_bound(x);
8      if (p1 == a.end()) return false; if (p1->x == x) return y <= p1->y;
9      if (p1 == a.begin()) return false; mit p2(p1--);
10     return sign(det(p - point(p1), point(p2) - p)) >= 0;
11 } inline void addPoint(mii &a, const point &p) { // `no collinear points`
12     int x = p.x, y = p.y; mit pnt = a.insert(make_pair(x, y)).first, p1, p2;
13     for (pnt->y = y; ; a.erase(p2)) {
14         p1 = pnt; if (++p1 == a.end()) break;
15         p2 = p1;  if (++p1 == a.end()) break;
16         if (det(point(p2) - p, point(p1) - p) < 0) break;
17     } for ( ; ; a.erase(p2)) {
18         if ((p1 = pnt) == a.begin()) break;
19         if (--p1 == a.begin()) break; p2 = p1--;
20         if (det(point(p2) - p, point(p1) - p) > 0) break;
21     }
22 }
```

2.2 Rope 用法

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

2.3 Treap

```
1  struct node { int key, prio, size; node *ch[2]; } base[MAXN], *top, *root, *null, nil;
2  typedef node *tree;
3  tree newNode(int key) {
4      static int seed = 3312;
5      top->key = key; top->prio = seed = int(seed * 48271LL % 2147483647);
6      top->size = 1; top->ch[0] = top->ch[1] = null; return top++;
7  }
8  void Rotate(tree &x, int d) {
9      tree y = x->ch[!d]; x->ch[!d] = y->ch[d]; y->ch[d] = x; y->size = x->size;
10     x->size = x->ch[0]->size + 1 + x->ch[1]->size; x = y;
11 }
12 void Insert(tree &t, int key) {
13     if (t == null) t = newNode(key);
14     else { int d = t->key < key; Insert(t->ch[d], key); ++t->size;
```

```
15         if (t->ch[d]->prio < t->prio) Rotate(t, !d);
16     }
17 }
18 void Delete(tree &t, int key) {
19     if (t->key != key) { Delete(t->ch[t->key < key], key); --t->size; }
20     else if (t->ch[0] == null) t = t->ch[1];
21     else if (t->ch[1] == null) t = t->ch[0];
22     else { int d = t->ch[0]->prio < t->ch[1]->prio;
23         Rotate(t, d); Delete(t->ch[d], key); --t->size;
24     }
25 }
```

2.4 可持久化 Treap

```
1  inline bool randomBySize(int a, int b) {
2      static long long seed = 1;
3      return (seed = seed * 48271 % 2147483647) * (a + b) < 2147483647LL * a;
4  }
5  tree merge(tree x, tree y) {
6      if (x == null) return y; if (y == null) return x;
7      tree t = NULL;
8      if (randomBySize(x->size, y->size)) t = newNode(x), t->r = merge(x->r, y);
9      else t = newNode(y), t->l = merge(x, y->l);
10     update(t); return t;
11 }
12 void splitByKey(tree t, int k, tree &l, tree &r) { // `$[-\infty, k) [k, +infty)$`
13     if (t == null) l = r = null;
14     else if (t->key < k) l = newNode(t), splitByKey(t->r, k, l->r, r), update(l);
15     else                 r = newNode(t), splitByKey(t->l, k, l, r->l), update(r);
16 }
17 void splitBySize(tree t, int k, tree &l, tree &r) { // `$[1, k) [k, +\infty)$`
18     static int s; if (t == null) l = r = null;
19     else if ((s = t->l->size + 1) < k) l = newNode(t), splitBySize(t->r, k - s, l->r, r), update(l);
20     else                               r = newNode(t), splitBySize(t->l, k, l, r->l), update(r);
21 }
```

2.5 左偏树

```
1  tree merge(tree a, tree b) {
2      if (a == null) return b;
3      if (b == null) return a;
4      if (a->key > b->key) swap(a, b);
5      a->rc = merge(a->rc, b);
6      a->rc->fa = a;
7      if (a->lc->dist < a->rc->dist) swap(a->lc, a->rc);
8      a->dist = a->rc->dist + 1;
9      return a;
10 }
11 void erase(tree t) {
12     tree x = t->fa, y = merge(t->lc, t->rc);
13     if (y != null) y->fa = x;
14     if (x == null) root = y;
15     else
16     for ((x->lc == t ? x->lc : x->rc) = y; x != null; y = x, x = x->fa) {
17         if (x->lc->dist < x->rc->dist) swap(x->lc, x->rc);
18         if (x->rc->dist + 1 == x->dist) return;
19         x->dist = x->rc->dist + 1;
20     }
21 }
```

2.6 Link-Cut Tree

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

```
 6  inline void PushDown(tree t) { if (t→rev) { MakeRev(t→ch[0]); MakeRev(t→ch[1]); t→rev = 0; } }
 7  inline void Rotate(tree x) {
 8      tree y = x→pre; PushDown(y); PushDown(x);
 9      int d = isRight(x);
10      if (!isRoot(y)) y→pre→ch[isRight(y)] = x; x→pre = y→pre;
11      if ((y→ch[d] = x→ch[!d]) != null) y→ch[d]→pre = y;
12      x→ch[!d] = y; y→pre = x; Update(y);
13  }
14  inline void Splay(tree x) {
15      PushDown(x); for (tree y; !isRoot(x); Rotate(x)) {
16          y = x→pre; if (!isRoot(y)) Rotate(isRight(x) != isRight(y) ? x : y);
17      } Update(x);
18  }
19  inline void Splay(tree x, tree to) {
20      PushDown(x); for (tree y; (y = x→pre) != to; Rotate(x)) if (y→pre != to)
21          Rotate(isRight(x) != isRight(y) ? x : y);
22      Update(x);
23  }
24  inline tree Access(tree t) {
25      tree last = null; for (; t != null; last = t, t = t→pre) Splay(t),t→ch[1] = last, Update(t);
26      return last;
27  }
28  inline void MakeRoot(tree t) { Access(t); Splay(t); MakeRev(t); }
29  inline tree FindRoot(tree t) { Access(t); Splay(t); tree last = null;
30      for ( ; t != null; last = t, t = t→ch[0]) PushDown(t); Splay(last); return last;
31  }
32  inline void Join(tree x, tree y) { MakeRoot(y); y→pre = x; }
33  inline void Cut(tree t) {Access(t); Splay(t); t→ch[0]→pre = null; t→ch[0] = null; Update(t);}
34  inline void Cut(tree x, tree y) {
35      tree upper = (Access(x), Access(y));
36      if (upper == x) { Splay(x); y→pre = null; x→ch[1] = null; Update(x); }
37      else if (upper == y) { Access(x); Splay(y); x→pre = null; y→ch[1] = null; Update(y); }
38      else assert(0); // `impossible to happen`
39  }
40  inline int Query(tree a, tree b) { // `query the cost in path a <—> b, lca inclusive`
41      Access(a); tree c = Access(b); // c is lca
42      int v1 = c→ch[1]→maxCost; Access(a);
43      int v2 = c→ch[1]→maxCost;
44      return max(max(v1, v2), c→cost);
45  }
46  void Init() {
47      null = &nil; null→ch[0] = null→ch[1] = null→pre = null; null→rev = 0;
48      Rep(i, 1, N) { node &n = base[i]; n.rev = 0; n.pre = n.ch[0] = n.ch[1] = null; }
49  }
```

3 字符串相关

3.1 Manacher

```
 1  char t[1001];        // 要处理的字符串
 2  char s[1001 * 2];    // 中间插入特殊字符以后的
 3  int Z[1001 * 2], L, R;  // Gusfield's Algorithm
 4  // 由a往左，由b往右，对称地做字符匹配
 5  int match(int a, int b){
 6      int i = 0;
 7      while (a − i >= 0 && b + i < n && s[a − i] == s[b + i]) i++;
 8      return i;
 9  }
10  void longest_palindromic_substring()
11  {
12      int N = strlen(t);
13      /* 在 t 中插入特殊字符，存放到 s*/ memset(s, '.', N*2+1);
14      for (int i = 0; i < N; ++i) s[i * 2 + 1] = t[i];
15      N = N * 2 + 1;
16      Z[0] = 1; L = R = 0;
17      for (int i = 1; i < N; ++i) {
18          int ii = L − (i − L), n = R + 1 − i;   // i的映拉
19          if (i > R) Z[i] = match(i, i), L = i, R = i + Z[i] − 1;
20          else if (Z[ii] == n) Z[i] = n + match(i−n, i+n), L = i, R = i + Z[i] − 1;
21          else Z[i] = min(Z[ii], n);
22      }
23      // 寻找最长回文子字串的長度。
24      int n = 0, p = 0;
25      for (int i=0; i < N; ++i) if(Z[i] > n) n = Z[p = i];
26      // 記得去卓特殊字元。
27      cout << "最長迴文子字串的長度是" << (n − 1) / 2;
28      // 印出最長迴文字串，記得別卬特殊字元。
29      for (int i = p − Z[p] + 1; i <= p + Z[p] − 1; ++i) if (i & 1) cout << s[i];
30  }
```

3.2 最大回文正方形

```
 1  #include <cstdio>
 2  #include <cstdlib>
 3  #include <cstring>
 4  #include <cmath>
 5  #include <iostream>
 6  #include <fstream>
 7  #include <algorithm>
 8  #include <string>
 9
10  using namespace std;
11
12  const int MAX = 711;
13  int n, m, tmp, t;
14  string line[MAX], row[MAX], col[MAX];
15  int row_z[MAX][MAX * 2], col_z[MAX][MAX * 2];
16  int row_palinlen[MAX][MAX][2], col_palinlen[MAX][MAX][2];
17  int maxlen, sx, sy, ex, ey;
18
19  void match(int start, int &z, string &s,int len)
20  {
21      while (start − z >= 0 && start + z < len && s[start − z] == s[start + z])
22          z++;
23  }
24
25  void calc_z(string s, int *z)
26  {
27      static int mid, right, len, reflect_i, suply_pos;
28
29      z[0] = 1;
30      len = s.size();
31      mid = right = 0;
32
33      for (int i = 1; i < len; ++i) {
34          reflect_i = 2 * mid − i;
35          suply_pos = i + z[reflect_i] − 1;
36          if (i > right) {
37              match(i, z[i] = 0, s, len);
38              mid = i, right = i + z[i] − 1;
39          } else if (suply_pos == right) {
40              match(i, z[i] = z[reflect_i], s, len);
41              mid = i, right = i + z[i] − 1;
42          } else {
43              z[i] = min(z[reflect_i], right − i + 1);
44          }
45      }
46  }
47
48  int main()
49  {
50      ios::sync_with_stdio(false);
51
52      cin >> n >> m;
53      for (int i = 0; i < n; ++i) {
54          row[i] = "#";
55          cin >> line[i];
56          for (int j = 0; j < m; ++j) {
57              row[i] += line[i][j];
58              row[i] += '#';
59          }
60          calc_z(row[i], row_z[i]);
61          for (int j = 0, cnt = 1; j < m; ++j, cnt += 2) {
62              row_palinlen[i][j][1] = row_z[i][cnt] − 1;
63              row_palinlen[i][j][0] = row_z[i][cnt + 1] − 1;
```

```
64          }
65      }
66      for (int i = 0; i < m; ++i) {
67          col[i] = "#";
68          for (int j = 0; j < n; ++j) {
69              col[i] += line[j][i];
70              col[i] += "#";
71          }
72          calc_z(col[i], col_z[i]);
73          for (int j = 0, cnt = 1; j < n; ++j, cnt += 2) {
74              col_palinlen[j][i][1] = col_z[i][cnt] - 1;
75              col_palinlen[j][i][0] = col_z[i][cnt + 1] - 1;
76          }
77      }
78
79      tmp = min(n,m);
80      maxlen = 1;
81      sx = sy = ex = ey = 0;
82
83      for (int i = 0; i < n; ++i)
84          for (int j = 0; j < m; ++j) {
85              int k,c;
86              t = tmp;
87              for (k = 1, c = 0; k <= t; k += 2, c++) {
88                  if (i - c < 0 || i + c >= n || j - c < 0 || j + c >= m) break;
89                  t = min(t, row_palinlen[i - c][j][1]);
90                  t = min(t, row_palinlen[i + c][j][1]);
91                  t = min(t, col_palinlen[i][j - c][1]);
92                  t = min(t, col_palinlen[i][j + c][1]);
93                  if (k > t) break;
94                  if (t <= maxlen) break;
95              }
96              if (k - 2 > maxlen) {
97                  c--;
98                  maxlen = k - 2;
99                  sx = i - c; ex = i + c;
100                 sy = j - c; ey = j + c;
101             }
102
103             t = tmp;
104             for (k = 2, c = 0; k <= t; k += 2, c++) {
105                 if (i - c < 0 || i + c + 1 >= n || j - c < 0 || j + c + 1 >= m) break;
106                 t = min(t, row_palinlen[i - c][j][0]);
107                 t = min(t, row_palinlen[i + c + 1][j][0]);
108                 t = min(t, col_palinlen[i][j - c][0]);
109                 t = min(t, col_palinlen[i][j + c + 1][0]);
110                 if (k > t) break;
111                 if (t <= maxlen) break;
112             }
113             if (k - 2 > maxlen) {
114                 c--;
115                 maxlen = k - 2;
116                 sx = i - c; ex = i + c + 1;
117                 sy = j - c; ey = j + c + 1;
118             }
119         }
120     cout << sx + 1 << " " << sy + 1 << " "
121          << ex + 1 << " " << ey + 1 << endl;
122
123     return 0;
124 }
```

3.3 KMP

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

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

```
1  void KMP(char *a, int la, char *b, int lb, int *next, int *ext) {
2      --a; --b; --next; --ext;
3      for (int i = 2, j = next[1] = 0; i <= la; i++) {
4          while (j && a[j + 1] != a[i]) j = next[j]; if (a[j + 1] == a[i]) ++j; next[i] = j;
5      } for (int i = 1, j = 0; i <= lb; ++i) {
6          while (j && a[j + 1] != b[i]) j = next[j]; if (a[j + 1] == b[i]) ++j; ext[i] = j;
7          if (j == la) j = next[j];
```

```
8      }
9  } void ExKMP(char *a, int la, char *b, int lb, int *next, int *ext) {
10     next[0] = la; for (int &j = next[1] = 0; j + 1 < la && a[j] == a[j + 1]; ++j);
11     for (int i = 2, k = 1; i < la; ++i) {
12         int p = k + next[k], l = next[i - k]; if (l < p - i) next[i] = l;
13         else for (int &j = next[k = i] = max(0, p - i); i + j < la && a[j] == a[i + j]; ++j);
14     } for (int &j = ext[0] = 0; j < la && j < lb && a[j] == b[j]; ++j);
15     for (int i = 1, k = 0; i < lb; ++i) {
16         int p = k + ext[k], l = next[i - k]; if (l < p - i) ext[i] = l;
17         else for (int &j = ext[k = i] = max(0, p - i); j < la && i + j < lb && a[j] == b[i + j]; ++j);
18     }
19 }
```

3.4 Aho-Corasick 自动机

```
1  void construct() {
2      static tree Q[MAX_NODE]; int head = 0, tail = 0;
3      for (root->fail = root, Q[++tail] = root; head < tail; ) {
4          tree x = Q[++head];
5  //        if (x->fail->danger) x->danger = true;
6          Rep(d, 0, sigma - 1) if (!x->next[d])
7              x->next[d] = (x == root) ? (root) : (x->fail->next[d]);
8          else {
9              x->next[d]->fail = (x == root) ? (root) : (x->fail->next[d]);
10             Q[++tail] = x->next[d];
11         }
12     }
13 }
```

3.5 后缀自动机

```
1  struct SAM {
2      int in[Maxn * 2 + 1][Sigma], fa[Maxn * 2 + 1], max[Maxn * 2 + 1], tot, last;
3      void init(int n) {
4          tot = last = 0;
5          for(int i = 0; i <= 2 * n + 1; ++i)
6              memset(in[i], -1, sizeof in[i]), fa[i] = -1;
7      }
8      void add(int x) {
9          int v = last; ++tot, last = tot, max[last] = max[v] + 1;
10         while(v != -1 && in[v][x] == -1) in[v][x] = last, v = fa[v];
11         if(v == -1) { fa[last] = 0; return; }
12         int p = in[v][x];
13         if(max[p] == max[v] + 1) fa[last] = p;
14         else {
15             int np = ++tot;
16             max[np] = max[v] + 1; fa[np] = fa[p], fa[p] = np, fa[last] = np;
17             while(v != -1 && in[v][x] == p) in[v][x] = np, v = fa[v];
18             memcpy(in[np], in[p], sizeof in[p]);
19 }}};
```

3.6 后缀数组

待排序的字符串放在 $r[0 \ldots n - 1]$ 中, 最大值小于 m.

$r[0 \ldots n - 2] > 0, r[n - 1] = 0$.

结果放在 $sa[0 \ldots n - 1]$.

```
1  namespace SuffixArrayDoubling {
2      int wa[MAXN], wb[MAXN], wv[MAXN], ws[MAXN];
3      int cmp(int *r, int a, int b, int l) { return r[a] == r[b] && r[a + l] == r[b + l]; }
4      void da(int *r, int *sa, int n, int m) {
5          int i, j, p, *x = wa, *y = wb, *t;
6          for (i = 0; i < m; i++) ws[i] = 0;
7          for (i = 0; i < n; i++) ws[x[i] = r[i]]++;
8          for (i = 1; i < m; i++) ws[i] += ws[i - 1];
9          for (i = n - 1; i >= 0; i--) sa[--ws[x[i]]] = i;
10         for (j = 1, p = 1; p < n; j *= 2, m = p) {
```

```
11          for (p = 0, i = n − j; i < n; i++) y[p++] = i;
12          for (i = 0; i < n; i++) if (sa[i] >= j) y[p++] = sa[i] − j;
13          for (i = 0; i < n; i++) wv[i] = x[y[i]];
14          for (i = 0; i < m; i++) ws[i] = 0;
15          for (i = 0; i < n; i++) ws[wv[i]]++;
16          for (i = 1; i < m; i++) ws[i] += ws[i − 1];
17          for (i = n − 1; i >= 0; i−−) sa[−−ws[wv[i]]] = y[i];
18          for (t = x, x = y, y = t, p = 1, x[sa[0]] = 0, i = 1; i < n; i++)
19              x[sa[i]] = cmp(y, sa[i − 1], sa[i], j) ? p − 1 : p++;
20 }}}
21 namespace SuffixArrayDC3 { // `r 与 sa 大小需 3 倍`
22      #define F(x) ((x) / 3 + ((x) % 3 == 1 ? 0 : tb))
23      #define G(x) ((x) < tb ? (x) * 3 + 1 : ((x) − tb) * 3 + 2)
24      int wa[MAXN], wb[MAXN], wv[MAXN], ws[MAXN];
25      int c0(int *r, int a, int b) {
26          return r[a] == r[b] && r[a + 1] == r[b + 1] && r[a + 2] == r[b + 2];
27      }
28      int c12(int k, int *r, int a, int b) {
29          if (k == 2) return r[a] < r[b] || (r[a] == r[b] && c12(1, r, a + 1, b + 1));
30          else        return r[a] < r[b] || (r[a] == r[b] && wv[a + 1] < wv[b + 1]);
31      }
32      void sort(int *r, int *a, int *b, int n, int m) {
33          for (int i = 0; i < n; i++) wv[i] = r[a[i]];
34          for (int i = 0; i < m; i++) ws[i] = 0;
35          for (int i = 0; i < n; i++) ws[wv[i]]++;
36          for (int i = 1; i < m; i++) ws[i] += ws[i − 1];
37          for (int i = n − 1; i >= 0; i−−) b[−−ws[wv[i]]] = a[i];
38      }
39      void dc3(int *r, int *sa, int n, int m) {
40          int i, j, *rn = r + n, *san = sa + n, ta = 0, tb = (n + 1) / 3, tbc = 0, p;
41          r[n] = r[n + 1] = 0;
42          for (i = 0; i < n; i++) if (i % 3 != 0) wa[tbc++] = i;
43          sort(r + 2, wa, wb, tbc, m);
44          sort(r + 1, wb, wa, tbc, m);
45          sort(r, wa, wb, tbc, m);
46          for (p = 1, rn[F(wb[0])] = 0, i = 1; i < tbc; i++)
47              rn[F(wb[i])] = c0(r, wb[i − 1], wb[i]) ? p − 1 : p++;
48          if (p < tbc) dc3(rn, san, tbc, p);
49          else for (i = 0; i < tbc; i++) san[rn[i]] = i;
50          for (i = 0; i < tbc; i++) if (san[i] < tb) wb[ta++] = san[i] * 3;
51          if (n % 3 == 1) wb[ta++] = n − 1;
52          sort(r, wb, wa, ta, m);
53          for (i = 0; i < tbc; i++) wv[wb[i] = G(san[i])] = i;
54          for (i = 0, j = 0, p = 0; i < ta && j < tbc; p++)
55              sa[p] = c12(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j++];
56          for (; i < ta; p++) sa[p] = wa[i++];
57          for (; j < tbc; p++) sa[p] = wb[j++];
58      }
59      #undef F
60      #undef G
61 }
62 namespace CalcHeight {
63      int rank[MAXN], height[MAXN];
64      void calheight(int *r, int *sa, int n) {
65          int i, j, k = 0; for (i = 1; i <= n; i++) rank[sa[i]] = i;
66          for (i = 0; i < n; height[rank[i++]] = k)
67              for (k ? k−− : 0, j = sa[rank[i] − 1]; r[i + k] == r[j + k]; k++);
68      }
69      void init(int len)
70      {
71          for(int i = 0; i <= len + 10; ++i)
72              rank[i] = height[i] = 0;
73      }
74 }
75 //Sample
76 int r[MAXN]; char s[MAXN];
77 int main()
78 {
79      int len;
80      scanf("%s", s);
81      len = strlen(s);
82      for(int i = 0; i < len; ++i) r[i] = s[i] − 'a' + 1;
83      r[len] = 0;
84      SuffixArrayDoubling::da(r, sa, n + 1, 30);
85      CalcHeight::calheight(r, sa, n);
```

```
86      //Then the value of sa[0~len−1] is 1 ~ n, so init RMQ carefully(1~n not 0~n−1)
87      return 0;
88 }
89 }
```

3.7 环串最小表示

```
1 int minimalRepresentation(int N, char *s) { // s must be double−sized and 0−based
2      int i, j, k, l; for (i = 0; i < N; ++i) s[i + N] = s[i]; s[N + N] = 0;
3      for (i = 0, j = 1; j < N; ) {
4          for (k = 0; k < N && s[i + k] == s[j + k]; ++k);
5          if (k >= N) break; if (s[i + k] < s[j + k]) j += k + 1;
6          else l = i + k, i = j, j = max(l, j) + 1;
7      } return i; // [i, i + N) is the minimal representation
8 }
```

3.8 回文自动机

```
1 #include <cstdlib>
2 #include <cstdio>
3 #include <cstring>
4 #include <algorithm>
5
6 const int C = 26, N = 100000, S = N + 2 + C;
7 char string[N + 2];
8 int s, length[S], suffix[S], go[S][C];
9 int extend(int p, int i) {
10      while (string[i − 1 − length[p]] != string[i]) {p = suffix[p];}
11      int q = suffix[p];
12      while (string[i − 1 − length[q]] != string[i]) {q = suffix[q];}
13      int c = string[i] − 'a';
14      int pp = go[p][c];int qq = go[q][c];
15      if (pp == −1) {
16          length[pp = go[p][c] = s ++] = length[p] + 2;
17          suffix[pp] = qq;
18          memset(go[pp], −1, sizeof(go[pp]));
19      } return pp;
20 }
21 int main() {
22      int tests; scanf("%d", &tests);
23      for (int t = 1; t <= tests; ++ t) {
24          printf("Case #%d: ", t);
25          for (int i = 0; i < C + 2; ++ i) {
26              suffix[i] = 1; length[i] = std::min(i − 1, 1);
27              memset(go[i], −1, sizeof(go[i]));
28          }
29          suffix[0] = suffix[1] = 0;
30          for (int i = 0; i < C; ++ i)
31              go[0][i] = 2 + i;
32          s = C + 2; string[0] = '#';
33          scanf("%s", string + 1);
34          int n = strlen(string + 1); int p = 0;
35          for (int i = 1; i <= n; ++i)
36              p = extend(p, i);
37          int result = s − (C + 2); std::sort(string + 1, string + n + 1);
38          result += std::unique(string + 1, string + n + 1) − string − 1;
39          printf("%d\n", result);
40      }
41      return 0;
42 }
```

4 图论

4.1 Dominator Tree

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <cstring>
4  #include <iostream>
5  #include <algorithm>
6  #include <vector>
7
8  using namespace std;
9
10 const int oo = 1073741819;
11
12 const int Maxn = 200000;
13 const int Maxm = 200000;
14
15 vector<int> g[Maxn];
16
17 //idom[i] is the dominator of i, node id— 1 based(1 ~ n), n is the source
18 class DominatorTree
19 {
20 public:
21     int tail[4][Maxm], n, m;
22     int Next[4][Maxm], sora[4][Maxm];
23     int ss[4], top, w_time;
24     int rel[Maxn], semi[Maxn], b[Maxn], idom[Maxn], best[Maxn], st[Maxn], pre[Maxn];
25     void origin()
26     {
27         for (int e = 0; e <= 3; e++) ss[e] = n;
28         for (int i = 1; i <= n; i++) {
29             for (int e = 0; e <= 3; e++)
30                 tail[e][i] = i, Next[e][i] = 0;
31             rel[i] = 0, semi[i] = idom[i] = pre[i] = 0, best[i] = i;
32             b[i] = i;
33         }
34         rel[0] = oo;
35     }
36     void link(int e, int x, int y)
37     {
38         ++ss[e], Next[e][tail[e][x]] = ss[e], tail[e][x] = ss[e], sora[e][ss[e]] = y, Next[e][ss[e]] = 0;
39     }
40     void dfs(int x, int y)
41     {
42         ++w_time, rel[x] = w_time;
43         st[++top] = x, pre[x] = y;
44         for (int i = x, ne; Next[0][i];) {
45             i = Next[0][i], ne = sora[0][i];
46             if (!rel[ne]) dfs(ne, x);
47         }
48     }
49     int find(int x)
50     {
51         int y = b[x];
52         if (b[x] != x) b[x] = find(b[x]);
53         if (rel[semi[best[y]]]<rel[semi[best[x]]])
54             best[x] = best[y];
55         return b[x];
56     }
57     //n— number of vertex, m— number of edges, e— edge set
58     void init(int _n, int _m, const vector<pair<int, int> > &e)
59     {
60         n = _n, m = _m;
61         origin();
62         for (int i = 0; i < m; i++) {
63             link(0, e[i].first, e[i].second);
64             link(1, e[i].second, e[i].first);
65         }
66         w_time = 0, top = 0;
67         dfs(n, 0);
68     }
69
70     void work()
71     {
72         for (int i = top; i >= 1; i—) {
73             int ne = st[i];
74             for (int j = ne, na; Next[1][j];) {
75                 j = Next[1][j], na = sora[1][j];
76                 if (!rel[na]) continue;
77                 int y;
78                 if (rel[na]>rel[ne]) {
79                     find(na);
80                     y = semi[best[na]];
81                 }
82                 else y = na;
83                 if (rel[y]<rel[semi[ne]]) semi[ne] = y;
84             }
85             if (ne != n) link(2, semi[ne], ne);
86             for (int j = ne, na; Next[2][j];) {
87                 j = Next[2][j], na = sora[2][j];
88                 find(na);
89                 int y = best[na];
90                 if (semi[y] == semi[na]) idom[na] = semi[na];
91                 else idom[na] = y;
92             }
93             for (int j = ne, na; Next[0][j];) {
94                 j = Next[0][j], na = sora[0][j];
95                 if (pre[na] == ne) {
96                     na = find(na);
97                     b[na] = ne;
98                 }
99             }
100        }
101        for (int i = 2; i <= top; i++) {
102            int ne = st[i];
103            if (idom[ne] != semi[ne]) idom[ne] = idom[idom[ne]];
104            link(3, idom[ne], ne);
105        }
106    }
107 }dom;
```

4.2 树 Hash

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <cstring>
4  #include <cmath>
5  #include <iostream>
6  #include <algorithm>
7  #include <vector>
8  #include <map>
9  #include <queue>
10
11 using namespace std;
12
13 const int mm = 1051697, p = 1e9 + 9, q = 1e9 + 7;
14 const int N = 100000 + 10;
15 vector<int> vec[N];
16 int n, size[N], mark[N], deg[N], father[N];
17 long long f[N], g[N], rtp[N], rtq[N];
18 map<pair<long long, long long>, int> mp;
19
20 struct Node {
21     int a, b, v;
22     Node() {}
23     Node(int _a, int _b, int _v) {
24         a = _a, b = _b, v = _v;
25     }
26     bool operator < (const Node &rhs) const {
27         if (a == rhs.a )
28             return b < rhs.b;
29         return a < rhs.a;
30     }
31 };
32
33 struct HashNode {
34     int pos;
35     long long val1, val2;
36     HashNode() {}
37     HashNode(int _pos, long long _val1, long long _val2) {
38         pos = _pos;
39         val1 = _val1;
40         val2 = _val2;
41     }
42     bool operator < (const HashNode &rhs) const {
43         if (val1 == rhs.val1)
44             return val2 < rhs.val2;
45         return val1 < rhs.val1;
```

```
46          }
47  };
48
49  void hashwork(int u)
50  {
51      vector<Node> data;
52      size[u] = 1;
53      for (int i = 0; i < (int)vec[u].size(); ++i) {
54          int v = vec[u][i];
55          hashwork(v);
56          data.push_back(Node(f[v], g[v], size[v]));
57          size[u] += size[v];
58      }
59      data.push_back(Node(1, 1, size[u]));
60      sort(data.begin(), data.end());
61
62      int len = 0;
63      f[u] = 1;
64      for (int i = 0; i < (int)data.size(); ++i) {
65          f[u] = ((f[u] * data[i].a) % p * rtp[len]) % p;
66          g[u] = ((g[u] * data[i].b) % q + rtq[len]) % q;
67          len += data[i].v;
68      }
69  }
70
71  int main()
72  {
73      ios::sync_with_stdio(false);
74      rtp[0] = rtq[0] = 1;
75      for (int i = 1; i < N; ++i) {
76          rtp[i] = (rtp[i - 1] * mm) % p;
77          rtq[i] = (rtq[i - 1] * mm) % q;
78      }
79
80      queue<int> que;
81      cin >> n;
82      for (int v = 2; v <= n; ++v) {
83          int u;
84          cin >> u;
85          vec[u].push_back(v);
86          father[v] = u;
87          deg[u]++;
88      }
89      memset(size, 0, sizeof(size));
90      memset(f, 0, sizeof(f));
91      memset(g, 0, sizeof(g));
92      for (int i = 1; i <= n; ++i)
93          if (deg[i] == 0)
94              que.push(i);
95      while (!que.empty()) {
96          int u = que.front();
97          //cout << u << endl;
98          que.pop();
99
100         deg[father[u]]--;
101         if (deg[father[u]] == 0) que.push(father[u]);
102
103         vector<Node> data;
104         size[u] = 1;
105         for (int i = 0; i < (int)vec[u].size(); ++i) {
106             int v = vec[u][i];
107             //hashwork(v);
108             data.push_back(Node(f[v], g[v], size[v]));
109             size[u] += size[v];
110         }
111         data.push_back(Node(1, 1, size[u]));
112         sort(data.begin(), data.end());
113
114         int len = 0;
115         f[u] = 1;
116         for (int i = 0; i < (int)data.size(); ++i) {
117             f[u] = ((f[u] * data[i].a) % p * rtp[len]) % p;
118             g[u] = ((g[u] * data[i].b) % q + rtq[len]) % q;
119             len += data[i].v;
120         }
121     }
122
123     //hashwork(1);
```

```
124     /*
125         vector<HashNode> ans;
126         for (int i = 1; i <= n; ++i) {
127         ans.push_back(HashNode(i, f[i], g[i]));
128     }
129         sort(ans.begin(), ans.end());
130         int tot = 0;
131         for (int i = 0, j; i < (int)ans.size(); i = j) {
132         ++tot;
133         for (j = i; j < (int)ans.size() && (ans[j].val1 == ans[i].val1 && ans[j].val2 == ans[i].val2); ++j)
134             mark[ans[j].pos] = tot;
135     }
136     */
137     int tot = 0;
138     for (int i = 1; i <= n; ++i) {
139         pair<long long, long long> pr = make_pair(f[i], g[i]);
140         if (mp.count(pr) == 0) {
141             mp[pr] = ++tot;
142             mark[i] = tot;
143         } else {
144             mark[i] = mp[pr];
145         }
146     }
147     for (int i = 1; i <= n; ++i) {
148         cout << mark[i];
149         if (i == n) cout << endl;
150         else cout << " ";
151     }
152     return 0;
153 }
```

4.3 带花树

```
1  namespace Blossom {
2      int n, head, tail, S, T, lca;
3      int match[MAXN], Q[MAXN], pred[MAXN], label[MAXN], inq[MAXN], inb[MAXN];
4      vector<int> link[MAXN];
5      inline void push(int x) { Q[tail++] = x; inq[x] = true; }
6      int findCommonAncestor(int x, int y) {
7          static bool inPath[MAXN]; for (int i = 0; i < n; ++i) inPath[i] = 0;
8          for ( ; ; x = pred[ match[x] ]) { x = label[x]; inPath[x] = true; if (x == S) break; }
9          for ( ; ; y = pred[ match[y] ]) { y = label[y]; if (inPath[y]) break; } return y;
10     }
11     void resetTrace(int x, int lca) {
12         while (label[x] != lca) { int y = match[x]; inb[ label[x] ] = inb[ label[y] ] = true;
13             x = pred[y]; if (label[x] != lca) pred[x] = y; }}
14     void blossomContract(int x, int y) {
15         lca = findCommonAncestor(x, y);
16         Foru(i, 0, n) inb[i] = 0; resetTrace(x, lca); resetTrace(y, lca);
17         if (label[x] != lca) pred[x] = y; if (label[y] != lca) pred[y] = x;
18         Foru(i, 0, n) if (inb[ label[i] ]) { label[i] = lca; if (!inq[i]) push(i); }
19     }
20     bool findAugmentingPath() {
21         Foru(i, 0, n) pred[i] = -1, label[i] = i, inq[i] = 0;
22         int x, y, z; head = tail = 0;
23         for (push(S); head < tail; ) for (int i = (int)link[x = Q[head++]].size() - 1; i >= 0; --i) {
24             y = link[x][i]; if (label[x] == label[y] || x == match[y]) continue;
25             if (y == S || (match[y] >= 0 && pred[ match[y] ] >= 0)) blossomContract(x, y);
26             else if (pred[y] == -1) {
27                 pred[y] = x; if (match[y] >= 0) push(match[y]);
28                 else {
29                     for (x = y; x >= 0; x = z) {
30                         y = pred[x], z = match[y]; match[x] = y, match[y] = x;
31                     } return true; }}} return false;
32     }
33     int findMaxMatching() {
34         int ans = 0; Foru(i, 0, n) match[i] = -1;
35         for (S = 0; S < n; ++S) if (match[S] == -1) if (findAugmentingPath()) ++ans;
36         return ans;
37     }
38 }
```

4.4 KM

```
1  int N, Tcnt, w[MAXN][MAXN], slack[MAXN];
2  int lx[MAXN], linkx[MAXN], visy[MAXN], ly[MAXN], linky[MAXN], visx[MAXN]; // `初值全为0`
3  bool DFS(int x) { visx[x] = Tcnt;
4      Rep(y, 1, N) if(visy[y] != Tcnt) { int t = lx[x] + ly[y] − w[x][y];
5          if (t == 0) { visy[y] = Tcnt;
6              if (!linky[y] || DFS(linky[y])) { linkx[x] = y; linky[y] = x; return true; }
7          } else cMin(slack[y], t);
8      } return false;
9  } void KM() {
10     Tcnt = 0; Rep(x, 1, N) Rep(y, 1, N) cMax(lx[x], w[x][y]);
11     Rep(S, 1, N) { Rep(i, 1, N) slack[i] = INF;
12         for (++Tcnt; !DFS(S); ++Tcnt) { int d = INF;
13             Rep(y, 1, N) if(visy[y] != Tcnt) cMin(d, slack[y]);
14             Rep(x, 1, N) if(visx[x] == Tcnt) lx[x] −= d;
15             Rep(y, 1, N) if(visy[y] == Tcnt) ly[y] += d; else slack[y] −= d;
16         }
17     }
18 }
```

4.5 双连通分量

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <cstdlib>
5  #include <vector>
6
7  using namespace std;
8
9  const int MAXN = 100000 + 10;
10
11 int dfn[MAXN], low[MAXN], bccno[MAXN], dfn_clock, bcc_cnt, Top;
12 vector <int> G[MAXN], bcc[MAXN];
13 pair <int, int> stk[MAXN];
14 bool iscut[MAXN];
15 int n, m;
16
17 void dfs(int p, int fa) {
18     low[p] = dfn[p] = ++dfn_clock;
19     int child = 0;
20     for (int i = 0; i < G[p].size(); ++i) {
21         int v = G[p][i];
22         if (!dfn[v]) {
23             stk[++Top] = make_pair(p, v);
24             dfs(v, p);
25             child++;
26             low[p] = min(low[p], low[v]);
27             if (low[v] >= dfn[p]) {
28                 iscut[p] = 1;
29                 ++bcc_cnt;
30                 bcc[bcc_cnt].clear();
31                 for (;;) {
32                     pair <int, int> x = stk[Top];
33                     −−Top;
34                     if (bccno[x.first] != bcc_cnt) {
35                         bccno[x.first] = bcc_cnt;
36                         bcc[bcc_cnt].push_back(x.first);
37                     }
38                     if (bccno[x.second] != bcc_cnt) {
39                         bccno[x.second] = bcc_cnt;
40                         bcc[bcc_cnt].push_back(x.second);
41                     }
42                     if (x.first == p && x.second == v)
43                         break;
44                 }
45             }
46         }
47         else
48             if (dfn[v] < dfn[p] && v != fa) {
49                 stk[++Top] = make_pair(p, v);
50                 low[p] = min(low[p], dfn[v]);
51             }
```

```
52         }
53         if (fa < 0 && child == 1) iscut[p] = 0;
54 }
55
56 void find_bcc(int n) {
57     for (int i = 1; i <= n; ++i) dfn[i] = 0;
58     for (int i = 1; i <= n; ++i) iscut[i] = 0;
59     for (int i = 1; i <= n; ++i) bccno[i] = 0;
60     dfn_clock = bcc_cnt = 0;
61     for (int i = 1; i <= n; ++i)
62         if (!dfn[i])
63             dfs(i, −1);
64 }
65
66 int main() {
67     scanf("%d%d", &n, &m);
68     for (int a, b, i = 1; i <= m; ++i) {
69         scanf("%d%d", &a, &b);
70         G[a].push_back(b);
71         G[b].push_back(a);
72     }
73     find_bcc(n);
74
75     return 0;
76 }
```

4.6 强连通分量

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <cstdlib>
5  #include <vector>
6  #include <algorithm>
7
8  using namespace std;
9
10 const int MAXN = 100000 + 10;
11
12 vector <int> G[MAXN];
13 int n, m;
14 int dfn[MAXN], low[MAXN], stk[MAXN], Top, scc_cnt, sccno[MAXN], dfn_clock;
15
16 void dfs(int p) {
17     dfn[p] = low[p] = ++dfn_clock;
18     stk[++Top] = p;
19     for (int i = 0; i < (int)G[p].size(); ++i) {
20         int v = G[p][i];
21         if (!dfn[v]) {
22             dfs(v);
23             low[p] = min(low[p], low[v]);
24         }
25         else if (!sccno[v])
26             low[p] = min(low[p], dfn[v]);
27     }
28     if (low[p] == dfn[p]) {
29         scc_cnt++;
30         for (;;) {
31             int x = stk[Top];
32             −−Top;
33             sccno[x] = scc_cnt;
34             if (x == p) break;
35         }
36     }
37 }
38
39 void find_scc(int n) {
40     dfn_clock = scc_cnt = 0;
41     for (int i = 1; i <= n; ++i) sccno[i] = 0;
42     for (int i = 1; i <= n; ++i) dfn[i] = low[i] = 0;
43     for (int i = 1; i <= n; ++i)
44         if (!dfn[i])
45             dfs(i);
46 }
```

4.7 2-SAT 与 Kosaraju

注意 Kosaraju 需要建反图

```
1  namespace SCC {
2      int code[MAXN * 2], seq[MAXN * 2], sCnt;
3      void DFS_1(int x) { code[x] = 1;
4          for (edge e(fir[x]); e; e = e→next) if (code[e→to] == −1) DFS_1(e→to);
5          seq[++sCnt] = x;
6      } void DFS_2(int x) { code[x] = sCnt;
7          for (edge e(fir2[x]); e; e = e→next) if (code[e→to] == −1) DFS_2(e→to); }
8      void SCC(int N) {
9          sCnt = 0; for (int i = 1; i <= N; ++i) code[i] = −1;
10         for (int i = 1; i <= N; ++i) if (code[i] == −1) DFS_1(i);
11         sCnt = 0; for (int i = 1; i <= N; ++i) code[i] = −1;
12         for (int i = N; i >= 1; −−i) if (code[seq[i]] == −1) {
13             ++sCnt; DFS_2(seq[i]); }
14     }
15 }// true   −   2i − 1
16 //   false   −   2i
17 bool TwoSat() { SCC::SCC(N + N);
18     // if code[2i − 1] = code[2i]: no solution
19     // if code[2i − 1] > code[2i]: i selected. else i not selected
20 }
```

4.8 全局最小割 Stoer-Wagner

```
1  int minCut(int N, int G[MAXN][MAXN]) { // 0−based
2      static int weight[MAXN], used[MAXN]; int ans = INT_MAX;
3      while (N > 1) {
4          for (int i = 0; i < N; ++i) used[i] = false; used[0] = true;
5          for (int i = 0; i < N; ++i) weight[i] = G[i][0];
6          int S = −1, T = 0;
7          for (int _r = 2; _r <= N; ++_r) { // N − 1 selections
8              int x = −1;
9              for (int i = 0; i < N; ++i) if (!used[i])
10                 if (x == −1 || weight[i] > weight[x]) x = i;
11             for (int i = 0; i < N; ++i) weight[i] += G[x][i];
12             S = T; T = x; used[x] = true;
13         } ans = min(ans, weight[T]);
14         for (int i = 0; i < N; ++i) G[i][S] += G[i][T], G[S][i] += G[i][T];
15         G[S][S] = 0; −−N;
16         for (int i = 0; i <= N; ++i) swap(G[i][T], G[i][N]);
17         for (int i = 0; i < N; ++i) swap(G[T][i], G[N][i]);
18     } return ans;
19 }
```

4.9 欧拉路

```
1  vector<int> eulerianWalk(int N, int S) {
2      static int res[MAXM], stack[MAXN]; static edge cur[MAXN];
3      int rcnt = 0, top = 0, x; for (int i = 1; i <= N; ++i) cur[i] = fir[i];
4      for (stack[top++] = S; top; ) {
5          for (x = stack[−−top]; ; ) {
6              edge &e = cur[x]; if (e == NULL) break;
7              stack[top++] = x; x = e→to; e = e→next;
8              // 对于无向图需要删掉反向边
9          } res[rcnt++] = x;
10     } reverse(res, res + rcnt); return vector<int>(res, res + rcnt);
11 }
```

4.10 稳定婚姻

```
1  namespace StableMatching {
2      int pairM[MAXN], pairW[MAXN], p[MAXN];
3      // init: pairM[0...n − 1] = pairW[0...n − 1] = −1, p[0...n − 1] = 0
4      void stableMatching(int n, int orderM[MAXN][MAXN], int preferW[MAXN][MAXN]) {
5          for (int i = 0; i < n; i++) while (pairM[i] < 0) {
6              int w = orderM[i][p[i]++], m = pairW[w];
7              if (m == −1) pairM[i] = w, pairW[w] = i;
8              else if (preferW[w][i] < preferW[w][m])
9                  pairM[m] = −1, pairM[i] = w, pairW[w] = i, i = m;
10         }
11     }
12 }
```

4.11 最大团搜索

```
1  namespace MaxClique { // 1−based
2      int g[MAXN][MAXN], len[MAXN], list[MAXN][MAXN], mc[MAXN], ans, found;
3      void DFS(int size) {
4          if (len[size] == 0) { if (size > ans) ans = size, found = true; return; }
5          for (int k = 0; k < len[size] && !found; ++k) {
6              if (size + len[size] − k <= ans) break;
7              int i = list[size][k]; if (size + mc[i] <= ans) break;
8              for (int j = k + 1, len[size + 1] = 0; j < len[size]; ++j) if (g[i][list[size][j]])
9                  list[size + 1][len[size + 1]++] = list[size][j];
10             DFS(size + 1);
11         }
12     }
13     int work(int n) {
14         mc[n] = ans = 1; for (int i = n − 1; i; −−i) { found = false; len[1] = 0;
15             for (int j = i + 1; j <= n; ++j) if (g[i][j]) list[1][len[1]++] = j;
16             DFS(1); mc[i] = ans;
17         } return ans;
18     }
19 }
```

4.12 极大团计数

```
1  namespace MaxCliqueCounting {
2      int n, ans;
3      int ne[MAXN], ce[MAXN];
4      int g[MAXN][MAXN], list[MAXN][MAXN];
5      void dfs(int size) {
6          int i, j, k, t, cnt, best = 0;
7          bool bb;
8          if (ne[size] == ce[size]) {
9              if (ce[size] == 0)
10                 ++ans;
11             return;
12         }
13         for (t = 0, i = 1; i <= ne[size]; ++i) {
14             for (cnt = 0, j = ne[size] + 1; j <= ce[size]; ++j)
15                 if (!g[list[size][i]][list[size][j]])
16                     ++cnt;
17             if (t == 0 || cnt < best)
18                 t = i, best = cnt;
19         }
20         if (t && best <= 0)
21             return;
22         for (k = ne[size] + 1; k <= ce[size]; ++k) {
23             if (t > 0) {
24                 for (i = k; i <= ce[size]; ++i)
25                     if (!g[list[size][t]][list[size][i]])
26                         break;
27                 swap(list[size][k], list[size][i]);
28             }
29             i = list[size][k];
30             ne[size + 1] = ce[size + 1] = 0;
31             for (j = 1; j < k; ++j)
32                 if (g[i][list[size][j]])
33                     list[size + 1][++ne[size + 1]] = list[size][j];
34             for (ce[size + 1] = ne[size + 1], j = k + 1; j <= ce[size]; ++j)
35                 if (g[i][list[size][j]])
36                     list[size + 1][++ce[size + 1]] = list[size][j];
37             dfs(size + 1);
```

```
38          ++ne[size];
39          --best;
40          for (j = k + 1, cnt = 0; j <= ce[size]; ++j)
41              if (!g[i][list[size][j]])
42                  ++cnt;
43          if (t == 0 || cnt < best)
44              t = k, best = cnt;
45          if (t && best <= 0)
46              break;
47      }
48  }
49  void work() {
50      int i;
51      ne[0] = 0;
52      ce[0] = 0;
53      for (i = 1; i <= n; ++i)
54          list[0][++ce[0]] = i;
55      ans = 0;
56      dfs(0);
57  }
58 }
```

4.13 最小树形图

```
1  namespace EdmondsAlgorithm { // O(ElogE + V^2) !!! 0-based !!!
2      struct enode { int from, c, key, delta, dep; enode *ch[2], *next;
3      } ebase[maxm], *etop, *fir[maxn], nil, *null, *inEdge[maxn], *chs[maxn];
4      typedef enode *edge; typedef enode *tree;
5      int n, m, setFa[maxn], deg[maxn], que[maxn];
6      inline void pushDown(tree x) { if (x->delta) {
7          x->ch[0]->key += x->delta; x->ch[0]->delta += x->delta;
8          x->ch[1]->key += x->delta; x->ch[1]->delta += x->delta; x->delta = 0;
9      }}
10     tree merge(tree x, tree y) {
11         if (x == null) return y; if (y == null) return x;
12         if (x->key > y->key) swap(x, y); pushDown(x); x->ch[1] = merge(x->ch[1], y);
13         if (x->ch[0]->dep < x->ch[1]->dep) swap(x->ch[0], x->ch[1]);
14         x->dep = x->ch[1]->dep + 1; return x;
15     }
16     void addEdge(int u, int v, int w) {
17         etop->from = u; etop->c = etop->key = w; etop->delta = etop->dep = 0;
18         etop->next = fir[v]; etop->ch[0] = etop->ch[1] = null;
19         fir[v] = etop; inEdge[v] = merge(inEdge[v], etop++);
20     }
21     void deleteMin(tree &r) { pushDown(r); r = merge(r->ch[0], r->ch[1]); }
22     int findSet(int x) { return setFa[x] == x ? x : setFa[x] = findSet(setFa[x]); }
23     void clear(int V, int E) {
24         null = &nil; null->ch[0] = null->ch[1] = null; null->dep = -1;
25         n = V; m = E; etop = ebase; Foru(i, 0, V) fir[i] = NULL; Foru(i, 0, V) inEdge[i] = null;
26     }
27     int solve(int root) { int res = 0, head, tail;
28         for (int i = 0; i < n; ++i) setFa[i] = i;
29         for ( ; ; ) { memset(deg, 0, sizeof(int) * n); chs[root] = inEdge[root];
30             for (int i = 0; i < n; ++i) if (i != root && setFa[i] == i) {
31                 while (findSet(inEdge[i]->from) == findSet(i)) deleteMin(inEdge[i]);
32                 ++deg[ findSet((chs[i] = inEdge[i])->from) ];
33             }
34             for (int i = head = tail = 0; i < n; ++i)
35                 if (i != root && setFa[i] == i && deg[i] == 0) que[tail++] = i;
36             while (head < tail) {
37                 int x = findSet(chs[que[head++]]->from);
38                 if (--deg[x] == 0) que[tail++] = x;
39             } bool found = false;
40             for (int i = 0; i < n; ++i) if (i != root && setFa[i] == i && deg[i] > 0) {
41                 int j = i; tree temp = null; found = true;
42                 do {setFa[j = findSet(chs[j]->from)] = i;
43                     deleteMin(inEdge[j]); res += chs[j]->key;
44                     inEdge[j]->key -= chs[j]->key; inEdge[j]->delta -= chs[j]->key;
45                     temp = merge(temp, inEdge[j]);
46                 } while (j != i); inEdge[i] = temp;
47             } if (!found) break;
48         } for (int i = 0; i < n; ++ i) if (i != root && setFa[i] == i) res += chs[i]->key;
```

```
49          return res;
50      }
51 }
52 namespace ChuLiu { // O(V ^ 3) !!! 1-based !!!
53     int n, used[maxn], pass[maxn], eg[maxn], more, que[maxn], g[maxn][maxn];
54     void combine(int id, int &sum) { int tot = 0, from, i, j, k;
55         for ( ; id != 0 && !pass[id]; id = eg[id]) que[tot++] = id, pass[id] = 1;
56         for (from = 0; from < tot && que[from] != id; from++);
57         if (from == tot) return; more = 1;
58         for (i = from; i < tot; i++) {
59             sum += g[eg[que[i]]][que[i]]; if (i == from) continue;
60             for (j = used[que[i]] = 1; j <= n; j++) if (!used[j])
61                 if (g[que[i]][j] < g[id][j]) g[id][j] = g[que[i]][j];
62         }
63         for (i = 1; i <= n; i++) if (!used[i] && i != id)
64             for (j = from; j < tot; j++) {
65                 k = que[j]; if (g[i][id] > g[i][k] - g[eg[k]][k])
66                 g[i][id] = g[i][k] - g[eg[k]][k];
67             }
68     }
69     void clear(int V) { n = V; Rep(i, 1, V) Rep(j, 1, V) g[i][j] = inf; }
70     int solve(int root) {
71         int i, j, k, sum = 0; memset(used, 0, sizeof(int) * (n + 1));
72         for (more = 1; more; ) {
73             more = 0; memset(eg, 0, sizeof(int) * (n + 1));
74             for (i = 1; i <= n; i++) if (!used[i] && i != root) {
75                 for (j = 1, k = 0; j <= n; j++) if (!used[j] && i != j)
76                     if (k == 0 || g[j][i] < g[k][i]) k = j;
77                 eg[i] = k;
78             } memset(pass, 0, sizeof(int) * (n + 1));
79             for (i = 1; i <= n; i++) if (!used[i] && !pass[i] && i != root)
80                 combine(i, sum);
81         } for (i = 1; i <= n; i++) if (!used[i] && i != root) sum += g[eg[i]][i];
82         return sum;
83     }
84 }
```

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.

```
1  const int maxn = 100000 + 5;
2  const int maxm = 1000000 + 5;
3  const int maxq = 1000000 + 5;
4  const int qsize = maxm + 3 * maxq;
5  int n, m, Q, x[qsize], y[qsize], z[qsize], qx[maxq], qy[maxq], a[maxn], *tz;
6  int kx[maxn], ky[maxn], kt, vd[maxn], id[maxm], app[maxm];
7  bool extra[maxm];
8  void init() {
9      scanf("%d%d", &n, &m); for (int i = 0; i < m; i++) scanf("%d%d%d", x + i, y + i, z + i);
10     scanf("%d", &Q); for (int i = 0; i < Q; i++) { scanf("%d%d", qx + i, qy + i); qx[i]--; }
11 }
12 int find(int x) {
13     int root = x, next; while (a[root]) root = a[root];
14     while ((next = a[x]) != 0) a[x] = root, x = next; return root;
15 }
16 inline bool cmp(const int &a, const int &b) { return tz[a] < tz[b]; }
17 void solve(int *qx, int *qy, int Q, int n, int *x, int *y, int *z, int m, long long ans) {
18     int ri, rj;
19     if (Q == 1) {
20         for (int i = 1; i <= n; i++) a[i] = 0; z[qx[0]] = qy[0];
21         for (int i = 0; i < m; i++) id[i] = i;
22         tz = z; sort(id, id + m, cmp);
23         for (int i = 0; i < m; i++) {
24             ri = find(x[id[i]]); rj = find(y[id[i]]);
25             if (ri != rj) ans += z[id[i]], a[ri] = rj;
26         } printf("%I64d\n", ans);
27         return;
28     } int tm = kt = 0, n2 = 0, m2 = 0;
29     for (int i = 1; i <= n; i++) a[i] = 0;
30     for (int i = 0; i < Q; i++) {
31         ri = find(x[qx[i]]); rj = find(y[qx[i]]); if (ri != rj) a[ri] = rj;
```

```
32        }
33        for (int i = 0; i < m; i++) extra[i] = true;
34        for (int i = 0; i < Q; i++) extra[qx[i]] = false;
35        for (int i = 0; i < m; i++) if (extra[i]) id[tm++] = i;
36        tz = z; sort(id, id + tm, cmp);
37        for (int i = 0; i < tm; i++) {
38            ri = find(x[id[i]]); rj = find(y[id[i]]);
39            if (ri != rj)
40                a[ri] = rj, ans += z[id[i]], kx[kt] = x[id[i]], ky[kt] = y[id[i]], kt++;
41        }
42        for (int i = 1; i <= n; i++) a[i] = 0;
43        for (int i = 0; i < kt; i++) a[find(kx[i])] = find(ky[i]);
44        for (int i = 1; i <= n; i++) if (a[i] == 0) vd[i] = ++n2;
45        for (int i = 1; i <= n; i++) if (a[i] != 0) vd[i] = vd[find(i)];
46        int *Nx = x + m, *Ny = y + m, *Nz = z + m;
47        for (int i = 0; i < m; i++) app[i] = -1;
48        for (int i = 0; i < Q; i++)
49            if (app[qx[i]] == -1)
50                Nx[m2] = vd[x[qx[i]]], Ny[m2] = vd[y[qx[i]]], Nz[m2] = z[qx[i]], app[qx[i]] = m2, m2++;
51        for (int i = 0; i < Q; i++) {
52            z[qx[i]] = qy[i];
53            qx[i] = app[qx[i]];
54        }
55        for (int i = 1; i <= n2; i++) a[i] = 0;
56        for (int i = 0; i < tm; i++) {
57            ri = find(vd[x[id[i]]]); rj = find(vd[y[id[i]]]);
58            if (ri != rj)
59                a[ri] = rj, Nx[m2] = vd[x[id[i]]], Ny[m2] = vd[y[id[i]]], Nz[m2] = z[id[i]], m2++;
60        }
61        int mid = Q / 2;
62        solve(qx, qy, mid, n2, Nx, Ny, Nz, m2, ans);
63        solve(qx + mid, qy + mid, Q - mid, n2, Nx, Ny, Nz, m2, ans);
64    }
65    void work() { if (Q) solve(qx, qy, Q, n, x, y, z, m, 0); }
66    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 $N(v) = \{w|w$与v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 $|N(v)| + 1 \le |N(w)|$ 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)

```
1  class Chordal { // 1-Based, G is the Graph, must be sorted before call Check_Chordal
2  public: // Construct will sort it automatically
3      int v[Maxn], id[Maxn];  bool inseq[Maxn]; priority_queue<pair<int, int> > pq;
4      vector<int> Construct_Perfect_Elimination_Sequence(vector<int> *G, int n) { // O(m + nlogn)
5          vector<int> seq(n + 1, 0);
6          for (int i = 0; i <= n; ++i) inseq[i] = false, sort(G[i].begin(), G[i].end()), v[i] = 0;
7          int cur = n; pair<int, int> Mx; while(!pq.empty()) pq.pop(); pq.push(make_pair(0, 1));
8          for (int i = n; i >= 1; --i) {
9              while (!pq.empty() && (Mx = pq.top(), inseq[Mx.second] || Mx.first != v[Mx.second])) pq.pop();
10             id[Mx.second] = cur;
11             int x = seq[cur--] = Mx.second, sz = (int)G[Mx.second].size(); inseq[x] = true;
12             for (int j = 0; j < sz; ++j) {
13                 int y = G[x][j]; if(!inseq[y]) pq.push(make_pair(++v[y], y));
14             }
15         } return seq;
16     }
17     bool Check_Chordal(vector<int> *G, vector<int> &seq, int n) { // O(n + mlogn), plz gen seq first
18         bool isChordal = true;
19         for (int i = n - 1; i >= 1 && isChordal; --i) {
20             int x = seq[i], sz, y = -1;
21             if ((sz = (int)G[x].size()) == 0) continue;
22             for(int j = 0; j < sz; ++j) {
23                 if (id[G[x][j]] < i) continue;
24                 if (y == -1 || id[y] > id[G[x][j]]) y = G[x][j];
25             } if (y == -1) continue;
26             for (int j = 0; j < sz; ++j) {
27                 int y1 = G[x][j]; if (id[y1] < i) continue;
28                 if (y1 == y || binary_search(G[y].begin(), G[y].end(), y1)) continue;
29                 isChordal = false; break;
30             }
31         } return isChordal;
32     }
33 };
```

4.16 K 短路 (允许重复)

```
1  #define for_each(it, v) for (vector<Edge*>::iterator it = (v).begin(); it != (v).end(); ++it)
2  const int MAX_N = 10000, MAX_M = 50000, MAX_K = 10000, INF = 1000000000;
3  struct Edge { int from, to, weight; };
4  struct HeapNode { Edge* edge; int depth; HeapNode* child[4]; }; // child[0..1] for heap G, child[2..3] for heap
        out edge
5
6  int n, m, k, s, t; Edge* edge[MAX_M];
7  int dist[MAX_N]; Edge* prev[MAX_N];
8  vector<Edge*> graph[MAX_N]; vector<Edge*> graphR[MAX_N];
9  HeapNode* nullNode; HeapNode* heapTop[MAX_N];
10
11 HeapNode* createHeap(HeapNode* curNode, HeapNode* newNode) {
12     if (curNode == nullNode) return newNode; HeapNode* rootNode = new HeapNode;
13     memcpy(rootNode, curNode, sizeof(HeapNode));
14     if (newNode->edge->weight < curNode->edge->weight) {
15         rootNode->edge = newNode->edge; rootNode->child[2] = newNode->child[2]; rootNode->child[3] = newNode->
            child[3];
16         newNode->edge = curNode->edge; newNode->child[2] = curNode->child[2]; newNode->child[3] = curNode->child
            [3];
17     } if (rootNode->child[0]->depth < rootNode->child[1]->depth) rootNode->child[0] = createHeap(rootNode->child
            [0], newNode);
18     else rootNode->child[1] = createHeap(rootNode->child[1], newNode);
19     rootNode->depth = max(rootNode->child[0]->depth, rootNode->child[1]->depth) + 1;
20     return rootNode;
21 }
22 bool heapNodeMoreThan(HeapNode* node1, HeapNode* node2) { return node1->edge->weight > node2->edge->weight; }
23
24 int main() {
25     scanf("%d%d%d", &n, &m, &k); scanf("%d%d", &s, &t); s--, t--;
26     while (m--) { Edge* newEdge = new Edge;
27         int i, j, w; scanf("%d%d%d", &i, &j, &w);
28         i--, j--; newEdge->from = i; newEdge->to = j; newEdge->weight = w;
29         graph[i].push_back(newEdge); graphR[j].push_back(newEdge);
30     }
31     //Dijkstra
32     queue<int> dfsOrder; memset(dist, -1, sizeof(dist));
33     typedef pair<int, pair<int, Edge*> > DijkstraQueueItem;
34     priority_queue<DijkstraQueueItem, vector<DijkstraQueueItem>, greater<DijkstraQueueItem> > dq;
35     dq.push(make_pair(0, make_pair(t, (Edge*) NULL)));
36     while (!dq.empty()) {
37         int d = dq.top().first; int i = dq.top().second.first;
38         Edge* edge = dq.top().second.second; dq.pop();
39         if (dist[i] != -1) continue;
40         dist[i] = d; prev[i] = edge; dfsOrder.push(i);
41         for_each(it, graphR[i]) dq.push(make_pair(d + (*it)->weight, make_pair((*it)->from, *it)));
42     }
43     //Create edge heap
44     nullNode = new HeapNode; nullNode->depth = 0; nullNode->edge = new Edge; nullNode->edge->weight = INF;
45     fill(nullNode->child, nullNode->child + 4, nullNode);
46     while (!dfsOrder.empty()) {
```

```
47        int i = dfsOrder.front(); dfsOrder.pop();
48        if (prev[i] == NULL) heapTop[i] = nullNode;
49        else heapTop[i] = heapTop[prev[i]→to];
50        vector<HeapNode*> heapNodeList;
51        for_each(it, graph[i]) { int j = (*it)→to; if (dist[j] == −1) continue;
52            (*it)→weight += dist[j] − dist[i]; if (prev[i] != *it) {
53                HeapNode* curNode = new HeapNode;
54                fill(curNode→child, curNode→child + 4, nullNode);
55                curNode→depth = 1; curNode→edge = *it;
56                heapNodeList.push_back(curNode);
57            }
58        } if (!heapNodeList.empty()) { //Create heap out
59            make_heap(heapNodeList.begin(), heapNodeList.end(), heapNodeMoreThan);
60            int size = heapNodeList.size();
61            for (int p = 0; p < size; p++) {
62                heapNodeList[p]→child[2] = 2 * p + 1 < size ? heapNodeList[2 * p + 1] : nullNode;
63                heapNodeList[p]→child[3] = 2 * p + 2 < size ? heapNodeList[2 * p + 2] : nullNode;
64            } heapTop[i] = createHeap(heapTop[i], heapNodeList.front());
65        }
66    } //Walk on DAG
67    typedef pair<long long, HeapNode*> DAGQueueItem;
68    priority_queue<DAGQueueItem, vector<DAGQueueItem>, greater<DAGQueueItem> > aq;
69    if (dist[s] == −1) printf("NO\n");
70    else { printf("%d\n", dist[s]);
71        if (heapTop[s] != nullNode) aq.push(make_pair(dist[s] + heapTop[s]→edge→weight, heapTop[s]));
72    } k−−; while (k−−) {
73        if (aq.empty()) { printf("NO\n"); continue; }
74        long long d = aq.top().first; HeapNode* curNode = aq.top().second; aq.pop();
75        printf("%I64d\n", d);
76        if (heapTop[curNode→edge→to] != nullNode)
77            aq.push(make_pair(d + heapTop[curNode→edge→to]→edge→weight, heapTop[curNode→edge→to]));
78        for (int i = 0; i < 4; i++) if (curNode→child[i] != nullNode)
79            aq.push(make_pair(d − curNode→edge→weight + curNode→child[i]→edge→weight, curNode→child[i]));
80    } return 0;
81 }
```

4.17   K 短路 (不允许重复)

```
1  int Num[10005][205], Path[10005][205], dev[10005], from[10005], value[10005], dist[205], Next[205], Graph
       [205][205];
2  int N, M, K, s, t, tot, cnt; bool forbid[205], hasNext[10005][205];
3  struct cmp {
4      bool operator()(const int &a, const int &b) {
5          int *i, *j; if (value[a] != value[b]) return value[a] > value[b];
6          for (i = Path[a], j = Path[b]; (*i) == (*j); i++, j++);
7          return (*i) > (*j);
8      }
9  };
10 void Check(int idx, int st, int *path, int &res) {
11     int i, j; for (i = 0; i < N; i++) dist[i] = 1000000000, Next[i] = t;
12     dist[t] = 0; forbid[t] = true; j = t;
13     for ( ; ; ) {
14         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])))
15             Next[i] = j, dist[i] = dist[j] + Graph[i][j];
16         j = −1; for (i = 0; i < N; i++) if (!forbid[i] && (j == −1 || dist[i] < dist[j])) j = i;
17         if (j == −1) break; forbid[j] = 1; if (j == st) break;
18     } res += dist[st]; for (i = st; i != t; i = Next[i], path++) (*path) = i; (*path) = i;
19 }
20 int main() {
21     int i, j, k, l;
22     while (scanf("%d%d%d%d%d", &N, &M, &K, &s, &t) && N) {
23         priority_queue<int, vector<int>, cmp> Q;
24         for (i = 0; i < N; i++) for (j = 0; j < N; j++) Graph[i][j] = 1000000000;
25         for (i = 0; i < M; i++) { scanf("%d%d%d", &j, &k, &l); Graph[j − 1][k − 1] = l; }
26         s−−; t−−;
27         memset(forbid, false, sizeof(forbid)); memset(hasNext[0], false, sizeof(hasNext[0]));
28         Check(0, s, Path[0], value[0]); dev[0] = 0; from[0] = 0; Num[0][0] = 0; Q.push(0);
29         cnt = 1; tot = 1;
30         for (i = 0; i < K; i++) {
31             if (Q.empty()) break; l = Q.top(); Q.pop();
32             for (j = 0; j <= dev[l]; j++) Num[l][j] = Num[from[l]][j];
33             for (; Path[l][j] != t; j++) {
34                 memset(hasNext[tot], false, sizeof(hasNext[tot])); Num[l][j] = tot++;
35             } for (j = 0; Path[l][j] != t; j++) hasNext[Num[l][j]][Path[l][j + 1]] = true;
36             for (j = dev[l]; Path[l][j] != t; j++) {
37                 memset(forbid, false, sizeof(forbid)); value[cnt] = 0;
38                 for (k = 0; k < j; k++) {
39                     forbid[Path[l][k]] = true;
40                     Path[cnt][k] = Path[l][k];
41                     value[cnt] += Graph[Path[l][k]][Path[l][k + 1]];
42                 } Check(Num[l][j], Path[l][j], &Path[cnt][j], value[cnt]);
43                 if (value[cnt] > 2000000) continue;
44                 dev[cnt] = j; from[cnt] = l; Q.push(cnt); cnt++;
45             }
46         }
47         if (i < K || value[l] > 2000000) printf("None\n");
48         else {
49             for (i = 0; Path[l][i] != t; i++) printf("%d−", Path[l][i] + 1);
50             printf("%d\n", t + 1);
51         }
52     } return 0;
53 }
```

4.18   小知识

- 平面图: 一定存在一个度小于等于 5 的点. $E \leq 3V − 6$. 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 $= \sum_{P_u > 0} P_u − C[S, T]$, 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

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

```cpp
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 = 0; 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

```
1  double area(const double &left, const double &right) {
2      double mid = (left + right) / 2;
3      return (right − left) * (calc(left) + 4 * calc(mid) + calc(right)) / 6;
4  }
5
6  double simpson(const double &left, const double &right,
7                 const double &eps, const double &area_sum) {
8      double mid = (left + right) / 2;
9      double area_left = area(left, mid);
10     double area_right = area(mid, right);
11     double area_total = area_left + area_right;
12     if (std::abs(area_total − area_sum) < 15 * eps) {
13         return area_total + (area_total − area_sum) / 15;
14     }
15     return simpson(left, mid, eps / 2, area_left)
16         + simpson(mid, right, eps / 2, area_right);
17 }
18
19 double simpson(const double &left, const double &right, const double &eps) {
20     return simpson(left, right, eps, area(left, right));
21 }
```

5.4　FFT

```
1  namespace FFT {
2      #define mul(a, b) (Complex(a.x * b.x − a.y * b.y, a.x * b.y + a.y * b.x))
3      struct Complex {}; // `something omitted`
4      void FFT(Complex P[], int n, int oper) {
5          for (int i = 1, j = 0; i < n−1; i++) {
6              for (int s = n; j ^= s >>= 1, ~j & s; );
7              if (i < j) swap(P[i], P[j]);
8          }
9          for (int d = 0; (1 << d) < n; d++) {
10             int m = 1 << d, m2 = m * 2;
11             double p0 = PI / m * oper;
12             Complex unit_p0(cos(p0), sin(p0));
13             for (int i = 0; i < n; i += m2) {
14                 Complex unit(1.0, 0.0);
15                 for (int j = 0; j < m; j++) {
16                     Complex &P1 = P[i + j + m], &P2 = P[i + j];
17                     Complex t = mul(unit, P1);
18                     P1 = Complex(P2.x − t.x, P2.y − t.y);
19                     P2 = Complex(P2.x + t.x, P2.y − t.y);
20                     unit = mul(unit, unit_p0);
21     }}}}
22     vector<int> doFFT(const vector<int> &a, const vector<int> &b) {
23         vector<int> ret(max(0, (int) a.size() + (int) b.size() − 1), 0);
24         static Complex A[MAXB], B[MAXB], C[MAXB];
25         int len = 1; while (len < (int)ret.size()) len *= 2;
26         for (int i = 0; i < len; i++) A[i] = i < (int)a.size() ? a[i] : 0;
27         for (int i = 0; i < len; i++) B[i] = i < (int)b.size() ? b[i] : 0;
28         FFT(A, len, 1); FFT(B, len, 1);
29         for (int i = 0; i < len; i++) C[i] = mul(A[i], B[i]);
30         FFT(C, len, −1);
31         for (int i = 0; i < (int)ret.size(); i++)
32             ret[i] = (int) (C[i].x / len + 0.5);
33         return ret;
34     }
35 }
```

5.5　整数 FFT

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

```
7              if (t >= downLimit && isPrime(t)) return t;
8      }}
9      int modInv(int a) { return a <= 1 ? a : (long long) (MOD − MOD / a) * modInv(MOD % a) % MOD; }
10     void NTT(int P[], int n, int oper) {
11         for (int i = 1, j = 0; i < n − 1; i++) {
12             for (int s = n; j ^= s >>= 1, ~j & s;);
13             if (i < j) swap(P[i], P[j]);
14         }
15         for (int d = 0; (1 << d) < n; d++) {
16             int m = 1 << d, m2 = m * 2;
17             long long unit_p0 = powMod(PRIMITIVE_ROOT, (MOD − 1) / m2);
18             if (oper < 0) unit_p0 = modInv(unit_p0);
19             for (int i = 0; i < n; i += m2) {
20                 long long unit = 1;
21                 for (int j = 0; j < m; j++) {
22                     int &P1 = P[i + j + m], &P2 = P[i + j];
23                     int t = unit * P1 % MOD;
24                     P1 = (P2 − t + MOD) % MOD; P2 = (P2 + t) % MOD;
25                     unit = unit * unit_p0 % MOD;
26     }}}}
27     vector<int> mul(const vector<int> &a, const vector<int> &b) {
28         vector<int> ret(max(0, (int) a.size() + (int) b.size() − 1), 0);
29         static int A[MAXB], B[MAXB], C[MAXB];
30         int len = 1; while (len < (int)ret.size()) len <<= 1;
31         for (int i = 0; i < len; i++) A[i] = i < (int)a.size() ? a[i] : 0;
32         for (int i = 0; i < len; i++) B[i] = i < (int)b.size() ? b[i] : 0;
33         NTT(A, len, 1); NTT(B, len, 1);
34         for (int i = 0; i < len; i++) C[i] = (long long) A[i] * B[i] % MOD;
35         NTT(C, len, −1); for (int i = 0, inv = modInv(len); i < (int)ret.size(); i++) ret[i] = (long long) C[i] *
           inv % MOD;
36         return ret;
37     }
38 }
```

5.6　扩展欧几里得

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

```
1  void exgcd(LL x, LL y, LL &a0, LL &b0, LL &g) {
2      LL a1 = b0 = 0, b1 = a0 = 1, t;
3      while (y != 0) {
4          t = a0 − x / y * a1, a0 = a1, a1 = t;
5          t = b0 − x / y * b1, b0 = b1, b1 = t;
6          t = x % y, x = y, y = t;
7      } if (x < 0) a0 = −a0, b0 = −b0, x = −x;
8      g = x;
9  }
```

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

```
1  bool test(LL n, int base) {
2      LL m = n − 1, ret = 0; int s = 0;
3      for ( ; m % 2 == 0; ++s) m >>= 1; ret = pow_mod(base, m, n);
4      if (ret == 1 || ret == n − 1) return true;
5      for (−−s; s >= 0; −−s) {
6          ret = multiply_mod(ret, ret, n); if (ret == n − 1) return true;
7      } return false;
8  }
```

```
 9  LL special[7] = {
10      1373653LL,        25326001LL,
11      3215031751LL,     25000000000LL,
12      2152302898747LL,  3474749660383LL, 341550071728321LL};
13  /*
14   * n < 2047                    test[] = {2}
15   * n < 1,373,653               test[] = {2, 3}
16   * n < 9,080,191               test[] = {31, 73}
17   * n < 25,326,001              test[] = {2, 3, 5}
18   * n < 4,759,123,141           test[] = {2, 7, 61}
19   * n < 1,122,004,669,633       test[] = {2, 13, 23, 1662803}
20   * n < 2,152,302,898,747       test[] = {2, 3, 5, 7, 11}
21   * n < 3,474,749,660,383       test[] = {2, 3, 5, 7, 11, 13}
22   * n < 341,550,071,728,321     test[] = {2, 3, 5, 7, 11, 13, 17}
23   * n < 3,825,123,056,546,413,051  test[] = {2, 3, 5, 7, 11, 13, 17, 19, 23}
24   */
25  bool is_prime(LL n) {
26      if (n < 2) return false;
27      if (n < 4) return true;
28      if (!test(n, 2) || !test(n, 3)) return false;
29      if (n < special[0]) return true;
30      if (!test(n, 5)) return false;
31      if (n < special[1]) return true;
32      if (!test(n, 7)) return false;
33      if (n == special[2]) return false;
34      if (n < special[3]) return true;
35      if (!test(n, 11)) return false;
36      if (n < special[4]) return true;
37      if (!test(n, 13)) return false;
38      if (n < special[5]) return true;
39      if (!test(n, 17)) return false;
40      if (n < special[6]) return true;
41      return test(n, 19) && test(n, 23) && test(n, 29) && test(n, 31) && test(n, 37);
42  }
```

5.9 PollardRho

```
 1  LL pollardRho(LL n, LL seed) {
 2      LL x, y, head = 1, tail = 2; x = y = random() % (n - 1) + 1;
 3      for ( ; ; ) {
 4          x = addMod(multiplyMod(x, x, n), seed, n);
 5          if (x == y) return n; LL d = gcd(myAbs(x - y), n);
 6          if (1 < d && d < n) return d;
 7          if (++head == tail) y = x, tail <<= 1;
 8  }} vector<LL> divisors;
 9  void factorize(LL n) { // `需要保证 n > 1`
10      if (isPrime(n)) divisors.push_back(n);
11      else { LL d = n;
12          while (d >= n) d = pollardRho(n, random() % (n - 1) + 1);
13          factorize(n / d); factorize(d);
14  }}
```

5.10 多项式求根

```
 1  const double error = 1e-12;
 2  const double infi = 1e+12;
 3  int n; double a[10], x[10];
 4  double f(double a[], int n, double x) {
 5      double tmp = 1, sum = 0;
 6      for (int i = 0; i <= n; i++) sum = sum + a[i] * tmp, tmp = tmp * x;
 7      return sum;
 8  }
 9  double binary(double l, double r, double a[], int n) {
10      int sl = sign(f(a, n, l)), sr = sign(f(a, n, r));
11      if (sl == 0) return l; if (sr == 0) return r;
12      if (sl * sr > 0) return infi;
13      while (r - l > error) {
14          double mid = (l + r) / 2;
15          int ss = sign(f(a, n, mid));
16          if (ss == 0) return mid;
17          if (ss * sl > 0) l = mid; else r = mid;
```

```
18      } return l;
19  }
20  void solve(int n, double a[], double x[], int &nx) {
21      if (n == 1) { x[1] = -a[0] / a[1]; nx = 1; return; }
22      double da[10], dx[10]; int ndx;
23      for (int i = n; i >= 1; i--) da[i - 1] = a[i] * i;
24      solve(n - 1, da, dx, ndx); nx = 0;
25      if (ndx == 0) {
26          double tmp = binary(-infi, infi, a, n);
27          if (tmp < infi) x[++nx] = tmp; return;
28      } double tmp = binary(-infi, dx[1], a, n);
29      if (tmp < infi) x[++nx] = tmp;
30      for (int i = 1; i <= ndx - 1; i++) {
31          tmp = binary(dx[i], dx[i + 1], a, n);
32          if (tmp < infi) x[++nx] = tmp;
33      } tmp = binary(dx[ndx], infi, a, n);
34      if (tmp < infi) x[++nx] = tmp;
35  }
36  int main() {
37      scanf("%d", &n);
38      for (int i = n; i >= 0; i--) scanf("%lf", &a[i]);
39      int nx; solve(n, a, x, nx);
40      for (int i = 1; i <= nx; i++) printf("%0.6f\n", x[i]);
41      return 0;
42  }
```

5.11 线性递推

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

```
 1  vector<int> recFormula(int n, int k[], int m) {
 2      vector<int> c(n + 1, 0);
 3      if (m < n) c[m] = 1;
 4      else {
 5          static int a[MAX_K * 2 + 1];
 6          vector<int> b = recFormula(n, k, m >> 1);
 7          for (int i = 0; i < n + n; ++i) a[i] = 0;
 8          int s = m & 1;
 9          for (int i = 0; i < n; i++) {
10              for (int j = 0; j < n; j++) a[i + j + s] += b[i] * b[j];
11              c[n] += b[i];
12          } c[n] = (c[n] + 1) * b[n];
13          for (int i = n * 2 - 1; i >= n; i--) {
14              int add = a[i]; if (add == 0) continue;
15              for (int j = 0; j < n; j++) a[i - n + j] += k[j] * add;
16              c[n] += add;
17          } for (int i = 0; i < n; ++i) c[i] = a[i];
18      } return c;
19  }
```

5.12 离散对数

$A^x \equiv B \pmod{C}$, 对非质数 C 也适用.

```
 1  int modLog(int A, int B, int C) {
 2      static pii baby[MAX_SQRT_C + 11];
 3      int d = 0; LL k = 1, D = 1; B %= C;
 4      for (int i = 0; i < 100; ++i, k = k * A % C) // `$[0, \log C]$`
 5          if (k == B) return i;
 6      for (int g; ; ++d) {
 7          g = gcd(A, C); if (g == 1) break;
 8          if (B % g != 0) return -1;
 9          B /= g; C /= g; D = (A / g * D) % C;
10      } int m = (int) ceil(sqrt((double) C)); k = 1;
11      for (int i = 0; i <= m; ++i, k = k * A % C) baby[i] = pii(k, i);
12      sort(baby, baby + m + 1); // [0, m]
13      int n = unique(baby, baby + m + 1, equalFirst) - baby, am = powMod(A, m, C);
14      for (int i = 0; i <= m; ++i) {
15          LL e, x, y; exgcd(D, C, x, y, e); e = x * B % C;
16          if (e < 0) e += C;
17          if (e >= 0) {
```

```
18          int k = lower_bound(baby, baby + n, pii(e, -1)) - baby;
19          if (baby[k].first == e) return i * m + baby[k].second + d;
20        } D = D * am % C;
21    } return -1;
22  }
```

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

```
1  inline int normalize(LL a, int P) { a %= P; return a < 0 ? a + P : a; }
2  vector<int> QuadraticResidue(LL a, LL b, LL c, int P) {
3      int h, t; LL r1, r2, delta, pb = 0;
4      a = normalize(a, P); b = normalize(b, P); c = normalize(c, P);
5      if (P == 2) { vector<int> res;
6          if (c % P == 0) res.push_back(0);
7          if ((a + b + c) % P == 0) res.push_back(1);
8          return res;
9      } delta = b * rev(a + a, P) % P;
10     a = normalize(-c * rev(a, P) + delta * delta, P);
11     if (powMod(a, P / 2, P) + 1 == P) return vector<int>(0);
12     for (t = 0, h = P / 2; h % 2 == 0; ++t, h /= 2);
13     r1 = powMod(a, h / 2, P);
14     if (t > 0) { do b = random() % (P - 2) + 2;
15         while (powMod(b, P / 2, P) + 1 != P); }
16     for (int i = 1; i <= t; ++i) {
17         LL d = r1 * r1 % P * a % P;
18         for (int j = 1; j <= t - i; ++j) d = d * d % P;
19         if (d + 1 == P) r1 = r1 * pb % P; pb = pb * pb % P;
20     } r1 = a * r1 % P; r2 = P - r1;
21     r1 = normalize(r1 - delta, P); r2 = normalize(r2 - delta, P);
22     if (r1 > r2) swap(r1, r2); vector<int> res(1, r1);
23     if (r1 != r2) res.push_back(r2);
24     return res;
25  }
```

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

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

```
8      for (int i = 0, delta = B / d; i < d; ++i) {
9          x = (x + delta) % B; ret.push_back((int)powMod(g, x, p));
10     } sort(ret.begin(), ret.end());
11     ret.resize(unique(ret.begin(), ret.end()) - ret.begin());
12     return ret;
13  }
```

5.15 Pell 方程

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

```
1  pair<ULL, ULL> Pell(int n) {
2      static ULL p[50] = {0, 1}, q[50] = {1, 0}, g[50] = {0, 0}, h[50] = {0, 1}, a[50];
3      ULL t = a[2] = Sqrt(n);
4      for (int i = 2; ; ++i) {
5          g[i] = -g[i - 1] + a[i] * h[i - 1];
6          h[i] = (n - g[i] * g[i]) / h[i - 1];
7          a[i + 1] = (g[i] + t) / h[i];
8          p[i] = a[i] * p[i - 1] + p[i - 2];
9          q[i] = a[i] * q[i - 1] + q[i - 2];
10         if (p[i] * p[i] - n * q[i] * q[i] == 1) return make_pair(p[i], q[i]);
11     } return make_pair(-1, -1);
12  }
```

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, $s(n,k) = (-1)^{n-k} \left[{n \atop k}\right]$.

  - $(x)^{(n)} = \sum_{k=0}^{n} \left[{n \atop k}\right] x^k$, $(x)_n = \sum_{k=0}^{n} s(n,k) x^k$

  - $\left[{n \atop k}\right] = n\left[{n-1 \atop k}\right] + \left[{n-1 \atop k-1}\right]$, $\left[{0 \atop 0}\right] = 1$, $\left[{n \atop 0}\right] = \left[{0 \atop n}\right] = 0$

  - $\left[{n \atop n-2}\right] = \frac{1}{4}(3n-1)\binom{n}{3}$, $\left[{n \atop n-3}\right] = \binom{n}{2}\binom{n}{4}$

  - $\sum_{k=0}^{a} \left[{n \atop k}\right] = n! - \sum_{k=0}^{n} \left[{n \atop k+a+1}\right]$

  - $\sum_{p=k}^{n} \left[{n \atop p}\right]\binom{p}{k} = \left[{n+1 \atop k+1}\right]$

- 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.

  - $\left\{{n \atop k}\right\} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^j \binom{k}{j}(k-j)^n$

  - $\left\{{n+1 \atop k}\right\} = k\left\{{n \atop k}\right\} + \left\{{n \atop k-1}\right\}$, $\left\{{0 \atop 0}\right\} = 1$, $\left\{{n \atop 0}\right\} = \left\{{0 \atop n}\right\} = 0$

  - 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

  - $B_0 = B_1 = 1$, $B_n = \sum_{k=0}^{n-1} \binom{n-1}{k} B_k$

  - $B_n = \sum_{k=0}^{n} \left\{{n \atop k}\right\}$

  - 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

  - $B_0 = 1$, $B_1 = \frac{1}{2}$, $B_2 = \frac{1}{6}$, $B_4 = -\frac{1}{30}$, $B_6 = \frac{1}{42}$, $B_8 = B_4$, $B_{10} = \frac{5}{66}$

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

  - $B_m = 1 - \sum_{k=0}^{m-1} \binom{m}{k} \frac{B_k}{m-k+1}$

- 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

```
1 int josephus(int n, int m, int k) { int x = −1;
2     for (int i = n − k + 1; i <= n; i++) x = (x + m) % i; return x;
3 }
4 int invJosephus(int n, int m, int x) {
5     for (int i = n; ; i−−) { if (x == i) return n − i; x = (x − m % i + i) % i; }
6 }
```

6.5 直线下的整点个数

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

```
1 LL count(LL n, LL a, LL b, LL m) {
2     if (b == 0) return n * (a / m);
3     if (a >= m) return n * (a / m) + count(n, a % m, b, m);
4     if (b >= m) return (n − 1) * n / 2 * (b / m) + count(n, a, b % m, m);
5     return count((a + b * n) / m, (a + b * n) % m, m, b);
6 }
```

6.6 Java 多项式

```
1 class Polynomial {
2     final static Polynomial ZERO = new Polynomial(new int[] { 0 });
3     final static Polynomial ONE = new Polynomial(new int[] { 1 });
4     final static Polynomial X = new Polynomial(new int[] { 0, 1 });
5     int[] coef;
6     static Polynomial valueOf(int val) { return new Polynomial(new int[] { val }); }
7     Polynomial(int[] coef) { this.coef = Arrays.copyOf(coef, coef.length); }
8     Polynomial add(Polynomial o, int mod); // omitted
9     Polynomial subtract(Polynomial o, int mod); // omitted
10    Polynomial multiply(Polynomial o, int mod); // omitted
11    Polynomial scale(int o, int mod); // omitted
12    public String toString() {
13        int n = coef.length; String ret = "";
14        for (int i = n − 1; i > 0; −−i) if (coef[i] != 0)
15            ret += coef[i] + "x^" + i + "+";
16        return ret + coef[0];
17    }
18    static Polynomial lagrangeInterpolation(int[] x, int[] y, int mod) {
19        int n = x.length; Polynomial ret = Polynomial.ZERO;
20        for (int i = 0; i < n; ++i) {
21            Polynomial poly = Polynomial.valueOf(y[i]);
22            for (int j = 0; j < n; ++j) if (i != j) {
23                poly = poly.multiply(
24                    Polynomial.X.subtract(Polynomial.valueOf(x[j]), mod), mod);
25                poly = poly.scale(powMod(x[i] − x[j] + mod, mod − 2, mod), mod);
26            } ret = ret.add(poly, mod);
27        } return ret;
28    }
29 }
```

6.7 long long 乘法取模

```
1 LL multiplyMod(LL a, LL b, LL P) { // `需要保证 a 和 b 非负`
2     LL t = (a * b − LL((long double)a / P * b + 1e−3) * P) % P;
3     return t < 0 ? t + P : t;
4 }
```

6.8 重复覆盖

```
1 namespace DLX {
2     struct node { int x, y; node *l, *r, *u, *d; } base[MAX * MAX], *top, *head;
3     typedef node *link;
4     int row, col, nGE, ans, stamp, cntc[MAX], vis[MAX];
5     vector<link> eachRow[MAX], eachCol[MAX];
6     inline void addElement(int x, int y) {
7         top→x = x; top→y = y; top→l = top→r = top→u = top→d = NULL;
8         eachRow[x].push_back(top); eachCol[y].push_back(top++);
9     }
10    void init(int _row, int _col, int _nGE) {
11        row = _row; col = _col; nGE = _nGE; top = base; stamp = 0;
12        for (int i = 0; i <= col; ++i) vis[i] = 0;
13        for (int i = 0; i <= row; ++i) eachRow[i].clear();
14        for (int i = 0; i <= col; ++i) eachCol[i].clear();
15        for (int i = 0; i <= col; ++i) addElement(0, i);
16        head = eachCol[0].front();
17    }
18    void build() {
19        for (int i = 0; i <= row; ++i) {
20            vector<link> &v = eachRow[i];
21            sort(v.begin(), v.end(), cmpByY);
22            int s = v.size();
23            for (int j = 0; j < s; ++j) {
24                link l = v[j], r = v[(j + 1) % s]; l→r = r; r→l = l;
25            }
26        }
27        for (int i = 0; i <= col; ++i) {
28            vector<link> &v = eachCol[i];
29            sort(v.begin(), v.end(), cmpByX);
30            int s = v.size();
31            for (int j = 0; j < s; ++j) {
32                link u = v[j], d = v[(j + 1) % s]; u→d = d; d→u = u;
33            }
34        } for (int i = 0; i <= col; ++i) cntc[i] = (int) eachCol[i].size() − 1;
35    }
36    void removeExact(link c) {
37        c→l→r = c→r; c→r→l = c→l;
38        for (link i = c→d; i != c; i = i→d)
39            for (link j = i→r; j != i; j = j→r) {
40                j→d→u = j→u; j→u→d = j→d; −−cntc[j→y];
41            }
42    }
43    void resumeExact(link c) {
44        for (link i = c→u; i != c; i = i→u)
45            for (link j = i→l; j != i; j = j→l) {
46                j→d→u = j; j→u→d = j; ++cntc[j→y];
47            }
48        c→l→r = c; c→r→l = c;
49    }
50    void removeRepeat(link c) {
51        for (link i = c→d; i != c; i = i→d) {
52            i→l→r = i→r; i→r→l = i→l;
53        }
54    }
55    void resumeRepeat(link c) {
56        for (link i = c→u; i != c; i = i→u) {
57            i→l→r = i; i→r→l = i;
58        }
59    }
60    int calcH() {
61        int y, res = 0; ++stamp;
62        for (link c = head→r; (y = c→y) <= row && c != head; c = c→r) {
63            if (vis[y] != stamp) {
64                vis[y] = stamp; ++res;
65                for (link i = c→d; i != c; i = i→d)
66                    for (link j = i→r; j != i; j = j→r) vis[j→y] = stamp;
67            }
68        } return res;
69    }
70    void DFS(int dep) { if (dep + calcH() >= ans) return;
71        if (head→r→y > nGE || head→r == head) {
72            if (ans > dep) ans = dep; return;
73        } link c = NULL;
74        for (link i = head→r; i→y <= nGE && i != head; i = i→r)
75            if (!c || cntc[i→y] < cntc[c→y]) c = i;
```

```
76      for (link i = c→d; i != c; i = i→d) {
77          removeRepeat(i);
78          for (link j = i→r; j != i; j = j→r) if (j→y <= nGE) removeRepeat(j);
79          for (link j = i→r; j != i; j = j→r) if (j→y  > nGE) removeExact(base + j→y);
80          DFS(dep + 1);
81          for (link j = i→l; j != i; j = j→l) if (j→y  > nGE) resumeExact(base + j→y);
82          for (link j = i→l; j != i; j = j→l) if (j→y <= nGE) resumeRepeat(j);
83          resumeRepeat(i);
84      }
85  }
86  int solve() { build(); ans = INF; DFS(0); return ans; }
87 }
```

6.9 星期几判定

```
1 int getDay(int y, int m, int d) {
2     if (m <= 2) m += 12, y—;
3     if (y < 1752 || (y == 1752 && m < 9) || (y == 1752 && m == 9 && d < 3))
4         return (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 + 5) % 7 + 1;
5     return (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 — y / 100 + y / 400) % 7 + 1;
6 }
```

6.10 LCSequence Fast

```
1 ULL *a, *b, *s, c, d;
2 for (i = 0, a = appear[(int)B[k]], b = row[max(k − 1, 0)], s = X; i < bitSetLen; ++i)
3     *s++ = *a++ | *b++;                      // `X = row[i − 1] or appear[ B[i] ]`
4 for (i = 0, a = dp, c = d = 0; i < bitSetLen; ++a, c = d, ++i)
5     d = *a >> 63, *a = ~((*a << 1) + c);     // `row[i] = —((row[i] << 1) + 1)`
6 for (i = 0, a = dp, b = X, c = 0; i < bitSetLen; ++a, ++b, ++i)
7     d = *b + c, c = (*a >=—d), *a += d;      // `row[i] += X`
8 for (i = 0, a = dp, b = X; i < bitSetLen; ++a, ++b, ++i)
9     *a = (*a ^ *b) & *b;                     // `row[i] = X and (row[i] xor X)`
```

6.11 C Split

```
1 for (char *tok = strtok(ins, delimiters); tok; tok = strtok(NULL, delimiters))
2     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++

```
1  #pragma comment(linker, "/STACK:10240000")
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <algorithm>
7  #define Rep(i, a, b) for(int i = (a); i <= (b); ++i)
8  #define Foru(i, a, b) for(int i = (a); i < (b); ++i)
9  using namespace std;
10 typedef long long LL;
11 typedef pair<int, int> pii;
12 namespace BufferedReader {
13     char buff[MAX_BUFFER + 5], *ptr = buff, c; bool flag;
14     bool nextChar(char &c) {
15         if ( (c = *ptr++) == 0 ) {
16             int tmp = fread(buff, 1, MAX_BUFFER, stdin);
17             buff[tmp] = 0; if (tmp == 0) return false;
18             ptr = buff; c = *ptr++;
19         } return true;
20     }
21     bool nextUnsignedInt(unsigned int &x) {
22         for (;;){if (!nextChar(c)) return false; if ('0'<=c && c<='9') break;}
23         for (x=c—'0'; nextChar(c); x = x * 10 + c — '0') if (c < '0' || c > '9') break;
24         return true;
25     }
26     bool nextInt(int &x) {
27         for (;;) { if (!nextChar(c)) return false; if (c=='—' || ('0'<=c && c<='9')) break; }
28         for ((c=='—') ? (x=0,flag=true) : (x=c—'0',flag=false); nextChar(c); x=x*10+c—'0')
29             if (c<'0' || c>'9') break;
30         if (flag) x=—x; return true;
31     }
32 };
33 #endif
```

7.3 Java

```
1  import java.io.*;
2  import java.util.*;
3  import java.math.*;
4
5  public class Main {
6      public void solve() {}
7      public void run() {
8          tokenizer = null; out = new PrintWriter(System.out);
9          in = new BufferedReader(new InputStreamReader(System.in));
10         solve();
11         out.close();
12     }
13     public static void main(String[] args) {
14         new Main().run();
15     }
16     public StringTokenizer tokenizer;
17     public BufferedReader in;
18     public PrintWriter out;
19     public String next() {
20         while (tokenizer == null || !tokenizer.hasMoreTokens()) {
21             try { tokenizer = new StringTokenizer(in.readLine()); }
22             catch (IOException e) { throw new RuntimeException(e); }
23         } return tokenizer.nextToken();
24     }
25 }
```

7.4 gcc 配置

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