Shanghai Jiao Tong University

All-in at the River

Lady luck is smilin'.

Version: EDG

Contents

1. Geometry

1.1 凸包

```cpp
#define cp const point &
bool turn_left(cp a, cp b, cp c) {
|  return sgn (det (b - a, c - a)) >= 0; }
vector <point> convex_hull (vector <point> a) {
|  int n = (int) a.size (), cnt = 0;
|  if (n < 2) return a;
|  sort (a.begin(), a.end()); // less<pair>
|  vector <point> ret;
|  for (int i = 0; i < n; ++i) {
|  |  while (cnt > 1
|  |  && turn_left (ret[cnt - 2], a[i], ret[cnt - 1])) {
|  |  |  --cnt; ret.pop_back ();
|  |  }
|  |  ret.push_back (a[i]); ++cnt;
|  }
|  int fixed = cnt;
|  for (int i = n - 2; i >= 0; --i) {
|  |  while (cnt > fixed
|  |  && turn_left (ret[cnt - 2], a[i], ret[cnt - 1])) {
|  |  |  --cnt; ret.pop_back ();
|  |  }
|  |  ret.push_back (a[i]); ++cnt;
|  }
|  ret.pop_back (); return ret;
} // counter-clockwise, ret[0] = min(pair(x, y))
```

1.2 闵可夫斯基和

```cpp
// a[0..1]: 逆时针凸包. 结果不是严格凸包
for (int i = 0; i < 2; i++) a[i].push_back(a[i].front());
int i[2] = {0, 0},
  len[2] = {(int)a[0].size() - 1, (int)a[1].size() - 1};
vector<point> mnk;
mnk.push_back(a[0][0] + a[1][0]);
do { // 输入不合法时会死循环; 存在精度问题, 考虑用整数
|  int d = sgn(det(a[1][i[1] + 1] - a[1][i[1]],
                   a[0][i[0] + 1] - a[0][i[0]])) >= 0;
|  mnk.push_back(a[d][i[d] + 1] - a[d][i[d]] + mnk.back());
|  i[d] = (i[d] + 1) % len[d];
} while(i[0] || i[1]);
```

1.3 最小覆盖圆

```cpp
struct circle {
|  point p; double r;
|  circle () {}
|  circle (point pp, double rr) {p = pp, r = rr;} };
bool in_circle(cp a, const circle &b) {
|  return (b.p - a).len() <= b.r; }
circle make_circle(point u, point v) {
|  point p = (u + v) / 2;
| return circle(p, (u - p).len()); }
circle make_circle(cp a, cp b, cp c) {
| point p = b - a, q = c - a,
| | s(dot(p, p) / 2, dot(q, q) / 2);
| double d = det(p, q);
| p = point( det(s, point(p.y, q.y)),
| | det(point(p.x, q.x), s) ) / d;
| return circle(a + p, p.len());
} // make_circle : 过参数点的最小圆
circle min_circle (vector <point> p) {
| circle ret;
| random_shuffle (p.begin (), p.end ());
| for (int i = 0; i < (int) p.size (); ++i)
| | if (!in_circle (p[i], ret)) {
| | | ret = circle (p[i], 0);
| | | for (int j = 0; j < i; ++j) if (!in_circle (p[j],
  ↪ret)) {
| | | | ret = make_circle (p[j], p[i]);
| | | | for (int k = 0; k < j; ++k)
| | | | | if (!in_circle (p[k], ret))
| | | | | | ret = make_circle(p[i],p[j],p[k]);
| | } } return ret; }
```

1.4 二维几何基础操作

```
bool point_on_segment(cp a,cl b){
return sgn (det (a - b.s, b.t - b.s)) == 0//pol
| && sgn (dot (b.s - a, b.t - a)) <= 0;}
bool two_side(cp a,cp b,cl c) {
| return sgn (det (a - c.s, c.t - c.s))
| | * sgn (det (b - c.s, c.t - c.s)) < 0; }
bool intersect_judge(cl a,cl b) {
| if (point_on_segment (b.s, a)
| || point_on_segment (b.t, a)) return true;
| if (point_on_segment (a.s, b)
| || point_on_segment (a.t, b)) return true;
| return two_side (a.s, a.t, b)
| | && two_side (b.s, b.t, a); }
point line_intersect(cl a, cl b) {
| double s1 = det (a.t - a.s, b.s - a.s);
| double s2 = det (a.t - a.s, b.t - a.s);
| return (b.s * s2 - b.t * s1) / (s2 - s1); }
bool point_on_ray (cp a, cl b) {
| return sgn (det (a - b.s, b.t - b.s)) == 0
| && sgn (dot (a - b.s, b.t - b.s)) >= 0; }
bool ray_intersect_judge(line a, line b) {
| LL s1, s2;
| s1 = det(a.t - a.s, b.s - a.s);
| s2 = det(a.t - a.s, b.t - a.s);
| if (sgn(s1) == 0 && sgn(s2) == 0) {
| | return sgn(dot(a.t - a.s, b.s - a.s)) >= 0
| | | || sgn(dot(b.t - b.s, a.s - b.s)) >= 0; }
| if (!sgn(s1 - s2) || sgn(s1) == sgn(s2 - s1)) return 0;
| swap(a, b);
| s1 = det(a.t - a.s, b.s - a.s);
| s2 = det(a.t - a.s, b.t - a.s);
| return sgn(s1) != sgn(s2 - s1); }
double point_to_line (cp a, cl b) {
| return abs (det (b.t-b.s, a-b.s)) / dis (b.s, b.t); }
point project_to_line (cp a, cl b) {
| return b.s + (b.t - b.s)
| * (dot (a - b.s, b.t - b.s) / (b.t - b.s).norm2 ()); }
double point_to_segment (cp a, cl b) {
| if (sgn (dot (b.s - a, b.t - b.s))
| * sgn (dot (b.t - a, b.t - b.s)) <= 0)
| | return abs(det(b.t - b.s, a - b.s)) / dis(b.s, b.t);
| return min (dis (a, b.s), dis (a, b.t)); }
bool in_polygon (cp p, const vector <point> & po) {
| int n = (int) po.size (); int counter = 0;
| for (int i = 0; i < n; ++i) {
| | point a = po[i], b = po[(i + 1) % n];
| | if (point_on_segment (p, line (a, b))) return true;
| | int x = sgn (det (p - a, b - a)),
| | | y = sgn (a.y - p.y), z = sgn (b.y - p.y);
| | if (x > 0 && y <= 0 && z > 0) ++counter;
| | if (x < 0 && z <= 0 && y > 0) --counter; }
| return counter != 0; }
vector <point> line_circle_intersect (cl a, cc b) {
| if (sgn (point_to_line (b.c, a) - b.r) > 0)
| | return vector <point> ();
| double x = sqrt(sq(b.r)-sq(point_to_line (b.c, a)));
| return vector <point>
| ({project_to_line (b.c, a) + (a.s - a.t).unit () * x,
| project_to_line (b.c, a) - (a.s - a.t).unit () * x});}
double circle_intersect_area (cc a, cc b) {
| double d = dis (a.c, b.c);
| if (sgn (d - (a.r + b.r)) >= 0) return 0;
| if (sgn (d - abs(a.r - b.r)) <= 0) {
| | double r = min (a.r, b.r);
| | return r * r * PI; }
| double x = (d * d + a.r * a.r - b.r * b.r) / (2 * d),
| |   t1 = acos (min (1., max (-1., x / a.r))),
| |   t2 = acos (min (1., max (-1., (d - x) / b.r)));
| return sq(a.r)*t1 + sq(b.r)*t2 - d*a.r*sin(t1);}
vector <point> circle_intersect (cc a, cc b) {
| if (a.c == b.c
| | || sgn (dis (a.c, b.c) - a.r - b.r) > 0
| | || sgn (dis (a.c, b.c) - abs (a.r - b.r)) < 0)
| | return {};
| point r = (b.c - a.c).unit ();
| double d = dis (a.c, b.c);
| double x = ((sqr (a.r) - sqr (b.r)) / d + d) / 2;
| double h = sqrt (sqr (a.r) - sqr (x));
| if (sgn (h) == 0) return {a.c + r * x};
| return {a.c + r * x + r.rot90 () * h,
| | |  a.c + r * x - r.rot90 () * h}; }
// 返回按照顺时针方向
vector <point> tangent (cp a, cc b) {
| circle p = make_circle (a, b.c);
| return circle_intersect (p, b); }
vector <line> extangent (cc a, cc b) {
| vector <line> ret;
| if (sgn(dis (a.c, b.c)-abs (a.r - b.r))<=0) return ret;
| if (sgn (a.r - b.r) == 0) {
| | point dir = b.c - a.c;
| | dir = (dir * a.r / dir.norm ()).rot90 ();
| | ret.push_back (line (a.c + dir, b.c + dir));
| | ret.push_back (line (a.c - dir, b.c - dir));
| } else {
| | point p = (b.c * a.r - a.c * b.r) / (a.r - b.r);
| | vector pp = tangent (p, a), qq = tangent (p, b);
| | if (pp.size () == 2 && qq.size () == 2) {
| | | if (sgn (a.r-b.r) < 0)
| | | | swap (pp[0], pp[1]), swap (qq[0], qq[1]);
| | | ret.push_back(line (pp[0], qq[0]));
| | | ret.push_back(line (pp[1], qq[1])); } }
| return ret; }
vector <line> intangent (cc a, cc b) {
| point p = (b.c * a.r + a.c * b.r) / (a.r + b.r);
| vector pp = tangent (p, a), qq = tangent (p, b);
| if (pp.size () == 2 && qq.size () == 2) {
| | ret.push_back (line (pp[0], qq[0]));
| | ret.push_back (line (pp[1], qq[1])); }
| return ret; }
vector <point> cut (const vector<point> &c, line p) {
| vector <point> ret;
| if (c.empty ()) return ret;
| for (int i = 0; i < (int) c.size (); ++i) {
| | int j = (i + 1) % (int) c.size ();
| | if (turn_left (p.s, p.t, c[i])) ret.push_back (c[i]);
| | if (two_side (c[i], c[j], p))
| | | ret.push_back (line_intersect (p, line (c[i],
  ↪c[j]))); }
| return ret; }
```

1.5 直线半平面交

```
bool turn_left (const line &l, const point &p) {
| return turn_left (l.s, l.t, p); }
vector <point> half_plane_intersect (vector <line> h) {
| typedef pair <double, line> polar;
| vector <polar> g; // use atan2, caution precision
| for (auto &i : h) {
| | point v = i.t - i.s;
| | g.push_back({atan2 (v.y, v.x), i}); }
| sort (g.begin(), g.end(), [] (const polar &a, const
  ↪polar &b) {
| | if (!sgn (a.first - b.first))
| | | return sgn (det (a.second.t - a.second.s,
  ↪b.second.t - a.second.s)) < 0;
| | else return sgn (a.first - b.first) < 0; });
| h.resize (unique (g.begin(), g.end(),
| [] (const polar &a, const polar &b)
| { return ! sgn (a.first - b.first); }) - g.begin());
| for (int i = 0; i < (int) h.size(); ++i)
```

```
    h[i] = g[i].second;
  int fore = 0, rear = -1;
  vector <line> ret;
  for (int i = 0; i < (int) h.size(); ++i) {
    while (fore < rear && !turn_left (h[i], line_intersect (ret[rear - 1], ret[rear]))) {
      --rear; ret.pop_back(); }
    while (fore < rear && !turn_left (h[i], line_intersect (ret[fore], ret[fore + 1])))
      ++fore;
    ++rear;
    ret.push_back (h[i]); }
  while (rear - fore > 1 && !turn_left (ret[fore], line_intersect (ret[rear - 1], ret[rear]))) {
    --rear; ret.pop_back(); }
  while (rear - fore > 1 && !turn_left (ret[rear], line_intersect (ret[fore], ret[fore + 1])))
    ++fore;
  if (rear - fore < 2) return vector <point>();
  vector <point> ans; ans.resize (rear - fore);
  for (int i = 0; i < (int) ans.size(); ++i)
    ans[i] = line_intersect (ret[fore + i],
          ret[fore + (i + 1) % ans.size()]);
  return ans; }
```

1.6 凸包快速询问

```
/* 给定凸包, log n 内完成各种询问, 具体操作有 :
1. 判定一个点是否在凸包内
2. 询问凸包外的点到凸包的两个切点
3. 询问一个向量关于凸包的切点
4. 询问一条直线和凸包的交点
INF 为坐标范围, 需要定义点类 operator < > 为 pair(x, y)
改成实数只需修改 sgn 函数, 以及把 LL 改为 double 即可
传入凸包要求无重点, 面积非空, pair(x,y) 最小点放在第一个 */
const int INF = 1e9;
struct Convex {
  int n;
  vector<point> a, upper, lower;
  Convex(vector<point> _a) : a(_a) {
    n = a.size(); int k = 0;
    for(int i = 1; i < n; ++ i) if (a[k] < a[i]) k = i;
    for(int i = 0; i <= k; ++ i) lower.push_back(a[i]);
    for(int i = k; i < n; ++ i) upper.push_back(a[i]);
    upper.push_back(a[0]);
  }
  pair <LL, int> get_tan(vector <point> & con, point vec) {
    int l = 0, r = (int) con.size() - 2;
    for ( ; l + 1 < r; ) {
      int mid = (l + r) / 2;
      if (sgn(det(con[mid + 1] - con[mid], vec)) > 0) r = mid;
      else l = mid;
    }
    return max(make_pair (det(vec, con[r]), r), make_pair(det(vec, con[0]), 0));
  }
  void upd_tan(cp p, int id, int &i0, int &i1) {
    if (det(a[i0] - p, a[id] - p) > 0) i0 = id;
    if (det(a[i1] - p, a[id] - p) < 0) i1 = id;
  }
  void search(int l, int r, point p, int &i0, int &i1) {
    if (l == r) return;
    upd_tan(p, l % n, i0, i1);
    int sl = sgn(det(a[l % n] - p, a[(l + 1) % n] - p));
    for ( ; l + 1 < r; ) {
      int mid = (l + r) / 2;
      int smid = sgn(det(a[mid % n] - p, a[(mid + 1) % n] - p));
      if (smid == sl) l = mid;
      else r = mid;
    }
    upd_tan(p, r % n, i0, i1);
  }
  int search(point u, point v, int l, int r) {
    int sl = sgn(det(v - u, a[l % n] - u));
    for ( ; l + 1 < r; ) {
      int mid = (l + r) / 2;
      int smid = sgn(det(v - u, a[mid % n] - u));
      if (smid == sl) l = mid;
      else r = mid;
    }
    return l % n;
  }
  // 判定点是否在凸包内, 在边界返回 true
  bool contain(point p) {
    if (p.x < lower[0].x || p.x > lower.back().x) return false;
    int id = lower_bound(lower.begin(), lower.end(), point(p.x, -INF)) - lower.begin();
    if (lower[id].x == p.x) {
      if (lower[id].y > p.y) return false;
    } else if (det(lower[id - 1] - p, lower[id] - p) < 0) return false;
    id = lower_bound(upper.begin(), upper.end(), point(p.x, INF), greater<point>()) - upper.begin();
    if (upper[id].x == p.x) {
      if (upper[id].y < p.y) return false;
    } else if (det(upper[id - 1] - p, upper[id] - p) < 0) return false;
    return true;
  }
  // 求点 p 关于凸包的两个切点, 如果在凸包外则有序返回编号, 共线的多个切点返回任意一个, 否则返回 false
  bool get_tan(point p, int &i0, int &i1) {
    i0 = i1 = 0;
    int id = int(lower_bound(lower.begin(), lower.end(), p) - lower.begin());
    search(0, id, p, i0, i1);
    search(id, (int)lower.size(), p, i0, i1);
    id = int(lower_bound(upper.begin(), upper.end(), p, greater <point> ()) - upper.begin());
    search((int)lower.size() - 1, (int) lower.size() - 1 + id, p, i0, i1);
    search((int)lower.size() - 1 + id, (int) lower.size() - 1 + (int)upper.size(), p, i0, i1);
    return true;
  }
  // 求凸包上和向量 vec 叉积最大的点, 返回编号, 共线的多个切点返回任意一个
  int get_tan(point vec) {
    pair<LL, int> ret = get_tan(upper, vec);
    ret.second = (ret.second + (int)lower.size() - 1) % n;
    ret = max(ret, get_tan(lower, vec));
    return ret.second;
  }
  // 求凸包和直线 u,v 的交点, 如果无严格相交返回 false. 如果有则是和 (i,next(i)) 的交点, 两个点无序, 交在点上不确定返回前后两条线段其中之一
  bool get_inter(point u, point v, int &i0, int &i1) {
    int p0 = get_tan(u - v), p1 = get_tan(v - u);
    if (sgn(det(v - u, a[p0] - u)) * sgn(det(v - u, a[p1] - u)) < 0) {
      if (p0 > p1) swap(p0, p1);
      i0 = search(u, v, p0, p1);
      i1 = search(u, v, p1, p0 + n);
      return true;
    } else return false;
  }};
```

1.7 三相之力

```
point incenter (cp a, cp b, cp c) {
  double p = dis (a, b) + dis (b, c) + dis (c, a);
  return (a * dis (b, c) + b * dis (c, a) + c * dis (a, b)) / p; }
point circumcenter (cp a, cp b, cp c) {
  point p = b - a, q = c - a, s (dot (p, p) / 2, dot (q, q) / 2);
  double d = det (p, q);
  return a + point (det (s, point (p.y, q.y)), det (point (p.x, q.x), s)) / d; }
point orthocenter (cp a, cp b, cp c) {
  return a + b + c - circumcenter (a, b, c) * 2.0; }
point fermat_point (cp a, cp b, cp c) {
  if (a == b) return a; if (b == c) return b;
  if (c == a) return c;
  double ab = dis (a, b), bc = dis (b, c), ca = dis (c, a);
  double cosa = dot (b - a, c - a) / ab / ca;
  double cosb = dot (a - b, c - b) / ab / bc;
  double cosc = dot (b - c, a - c) / ca / bc;
  double sq3 = PI / 3.0; point mid;
  if (sgn (cosa + 0.5) < 0) mid = a;
  else if (sgn (cosb + 0.5) < 0) mid = b;
```

```
    else if (sgn (cosc + 0.5) < 0) mid = c;
    else if (sgn (det (b - a, c - a)) < 0)
        mid = line_intersect (line (a, b + (c - b).rot (sq3)), line (b, c + (a - c).rot (sq3)));
    else
        mid = line_intersect (line (a, c + (b - c).rot (sq3)), line (c, b + (a - b).rot (sq3)));
    return mid; } // minimize(|A-x|+|B-x|+|C-x|)
```

1.8 圆并

```
int C; circle c[MAXN]; double area[MAXN];
struct event {
    point p; double ang; int delta;
    event (point p = point (), double ang = 0, int delta = 0) : p(p), ang(ang), delta(delta) {}
    bool operator < (const event &a) { return ang < a.ang; } };
void addevent(cc a, cc b, vector<event> &evt, int &cnt) {
    double d2 = (a.c - b.c).dis2(), dRatio = ((a.r - b.r) * (a.r + b.r) / d2 + 1) / 2,
        pRatio = sqrt (max (0., -(d2 - sqr(a.r - b.r)) * (d2 - sqr(a.r + b.r)) / (d2 * d2 * 4)));
    point d = b.c - a.c, p = d.rot(PI / 2),
        q0 = a.c + d * dRatio + p * pRatio,
        q1 = a.c + d * dRatio - p * pRatio;
    double ang0 = atan2 ((q0 - a.c).y, (q0 - a.c).x), ang1 = atan2 ((q1 - a.c).y, (q1 - a.c).x);
    evt.emplace_back(q1,ang1,1); evt.emplace_back(q0,ang0,-1);
    cnt += ang1 > ang0; }
bool issame(cc a, cc b) {
    return sgn((a.c-b.c).dis()) == 0 && sgn(a.r-b.r) == 0; }
bool overlap(cc a, cc b) {
    return sgn(a.r - b.r - (a.c - b.c).dis()) >= 0; }
bool intersect(cc a, cc b) {
    return sgn((a.c - b.c).dis() - a.r - b.r) < 0; }
void solve() {
    fill (area, area + C + 2, 0);
    for (int i = 0; i < C; ++i) { int cnt = 1;
        vector<event> evt;
        for (int j=0; j<i; ++j) if (issame(c[i],c[j])) ++cnt;
        for (int j = 0; j < C; ++j)
            if (j != i && !issame(c[i], c[j]) && overlap(c[j], c[i])) ++cnt;
        for (int j = 0; j < C; ++j)
            if (j != i && !overlap(c[j], c[i]) && !overlap(c[i], c[j]) && intersect(c[i], c[j]))
                addevent(c[i], c[j], evt, cnt);
        if (evt.empty()) area[cnt] += PI * c[i].r * c[i].r;
        else {
            sort(evt.begin(), evt.end());
            evt.push_back(evt.front());
            for (int j = 0; j + 1 < (int)evt.size(); ++j) {
                cnt += evt[j].delta;
                area[cnt] += det(evt[j].p,evt[j + 1].p) / 2;
                double ang = evt[j + 1].ang - evt[j].ang;
                if (ang < 0) ang += PI * 2;
                area[cnt] += ang * c[i].r * c[i].r / 2 - sin(ang) * c[i].r * c[i].r / 2; } } } }
```

1.9 多边形与圆交

```
double sector_area (cp a, cp b, double r) {
    double c = (2.0 * r * r - (a - b).norm2 ()) / (2.0 * r * r);
    double al = acos (c);
    return r * r * al / 2.0; }
double area(cp a,cp b, double r) { // point() : (0, 0)
    double dA = dot (a, a), dB = dot (b, b), dC = point_to_segment (point (), line (a, b)), ans = 0;
    if (sgn (dA - r * r) <= 0 && sgn (dB - r * r) <= 0) return det (a, b) / 2.0;
    point tA = a.unit () * r;
    point tB = b.unit () * r;
    if (sgn (dC - r) >= 0) return sector_area (tA, tB, r);
    pair <point, point> ret = line_circle_intersect (line (a, b), circle (point (), r));
    if (sgn (dA - r * r) > 0 && sgn (dB - r * r) > 0) {
        ans += sector_area (tA, ret.first, r);
        ans += det (ret.first, ret.second) / 2.0;
        ans += sector_area (ret.second, tB, r);
        return ans; }
    if (sgn (dA - r * r) > 0)
        return det (ret.first, b) / 2.0 + sector_area (tA, ret.first, r);
    else
        return det (a, ret.second) / 2.0 + sector_area (ret.second, tB, r); }
double solve(const vector<point> &p, cc c) {//p 逆时针
    double ret = 0;
    for (int i = 0; i < (int) p.size (); ++i) {
        int s = sgn (det (p[i] - c.c, p[ (i + 1) % p.size ()] - c.c));
        if (s > 0)
            ret += area (p[i] - c.c, p[ (i + 1) % p.size ()] - c.c, c.r);
        else
            ret -= area (p[ (i + 1) % p.size ()] - c.c, p[i] - c.c, c.r); }
    return abs (ret); }
```

1.10 阿波罗尼茨圆

```
硬币问题：两两相切的圆 r1, r2, r3, 求与他们都相切的圆 r4
分母取负号，答案再取绝对值，为外切圆半径
分母取正号为内切圆半径
// $r_4^{\pm} = \frac{r_1 r_2 r_3}{r_1 r_2 + r_1 r_3 + r_2 r_3 \pm 2\sqrt{r_1 r_2 r_3 (r_1 + r_2 + r_3)}}$
```

1.11 圆幂 圆反演 根轴

圆幂: 半径为 R 的圆 O, 任意一点 P 到 O 的幂为 $h = OP^2 - R^2$
圆幂定理: 过 P 的直线交圆在 A 和 B 两点, 则 $PA \cdot PB = |h|$
根轴: 到两圆等幂点的轨迹是一条垂直于连心线的直线
反演: 已知一圆 C, 圆心为 O, 半径为 r, 如果 P 与 P' 在过圆心 O 的直线上, 且 $OP \cdot OP' = r^2$, 则称 P 与 P' 关于 O 互为反演. 一般 C 取单位圆.
反演的性质:
不过反演中心的直线反形是过反演中心的圆, 反之亦然.
不过反演中心的圆, 它的反形是一个不过反演中心的圆.
两条直线在交点 A 的夹角, 等于它们的反形在相应点 A' 的夹角, 但方向相反.
两个相交圆周在交点 A 的夹角等于它们的反形在相应点 A' 的夹角, 但方向相反.
直线和圆周在交点 A 的夹角等于它们的反演图形在相应点 A' 的夹角, 但方向相反.
正交圆反形也正交. 相切圆反形也相切, 当切点为反演中心时, 反形为两条平行线.

1.12 球面基础

球面距离: 连接球面两点的大圆劣弧 (所有曲线中最短)
球面角: 球面两个大圆弧所在半平面形成的二面角
球面凸多边形: 把一个球面多边形任意一边向两方无限延长成大圆, 其余边都在此大圆的同旁.
球面角盈 E: 球面凸n边形的内角和与 $(n-2)\pi$ 的差
离北极夹角 θ, 距离 h 的球冠: $S = 2\pi Rh = 2\pi R^2(1-\cos\theta)$, $V = \frac{\pi h^2}{3}(3R-h)$
球面凸n边形面积: $S = ER^2$

1.13 经纬度球面距离

```
// lontitude 经度范围: ±π, latitude 纬度范围: ±π/2
double sphereDis(double lon1, double lat1, double lon2, double lat2, double R) {
    return R * acos(cos(lat1) * cos(lat2) * cos(lon1 - lon2) + sin(lat1) * sin(lat2)); }
```

1.14 圆上整点

```
vector <LL> solve(LL r) {
    vector <LL> ret; // non-negative Y pos
    ret.push_back(0);
    LL l = 2 * r, s = sqrt(l);
    for (LL d=1; d<=s; d++) if (l%d==0) {
        LL lim=LL(sqrt(l/(2*d)));
        for (LL a = 1; a <= lim; a++) {
            LL b = sqrt(l/d-a*a);
            if (a*a+b*b==l/d && __gcd(a,b)==1 && a!=b)
                ret.push_back(d*a*b);
        } if (d*d==l) break;
        lim = sqrt(d/2);
        for (LL a=1; a<=lim; a++) {
            LL b = sqrt(d - a * a);
            if (a*a+b*b==d && __gcd(a,b)==1 && a!=b)
                ret.push_back(l/d*a*b);
    } } return ret; }
```

1.15 相关公式

1.15.1 Heron's Formula

$$S = \sqrt{p(p-a)(p-b)(p-c)}$$
$$p = \frac{a+b+c}{2}$$

1.15.2 四面体内接球球心

假设 s_i 是第 i 个顶点相对面的面积，则有

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

体积可以使用 1/6 混合积求，内接球半径为

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

1.15.3 三角形内心

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

1.15.4 三角形外心

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

1.15.5 三角形垂心

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

1.15.6 三角形偏心

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

剩余两点的同理.

1.15.7 三角形内接外接圆半径

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

1.15.8 Pick's Theorem

$$S = I + \frac{B}{2} - 1$$

S is the area of lattice polygon, I is the number of lattice interior points, and B is the number of lattice boundary points.

1.15.9 Euler's Formula

For convex polyhedron: $V - E + F = 2$.

For planar graph: $|F| = |E| - |V| + n + 1$, n denotes the number of connected components.

1.16 三角公式

$$\sin(a \pm b) = \sin a \cos b \pm \cos a \sin b$$
$$\cos(a \pm b) = \cos a \cos b \mp \sin a \sin b$$
$$\tan(a \pm b) = \frac{\tan(a) \pm \tan(b)}{1 \mp \tan(a)\tan(b)}$$
$$\tan(a) \pm \tan(b) = \frac{\sin(a \pm b)}{\cos(a)\cos(b)}$$
$$\sin(a) + \sin(b) = 2\sin(\frac{a+b}{2})\cos(\frac{a-b}{2})$$
$$\sin(a) - \sin(b) = 2\cos(\frac{a+b}{2})\sin(\frac{a-b}{2})$$
$$\cos(a) + \cos(b) = 2\cos(\frac{a+b}{2})\cos(\frac{a-b}{2})$$
$$\cos(a) - \cos(b) = -2\sin(\frac{a+b}{2})\sin(\frac{a-b}{2})$$
$$\sin(na) = n\cos^{n-1}a\sin a - \tbinom{n}{3}\cos^{n-3}a\sin^3 a + \tbinom{n}{5}\cos^{n-5}a\sin^5 a - \dots$$
$$\cos(na) = \cos^n a - \binom{n}{2}\cos^{n-2}a\sin^2 a + \binom{n}{4}\cos^{n-4}a\sin^4 a - \dots$$

1.16.1 超球坐标系

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

1.16.2 三维旋转公式

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

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

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

1.16.3 立体角公式

$$\phi: \text{二面角}$$
$$\Omega = (\phi_{ab} + \phi_{bc} + \phi_{ac})\ \text{rad} - \pi\ \text{sr}$$
$$\tan\left(\frac{1}{2}\Omega/\text{rad}\right) = \frac{\left|\vec{a}\ \vec{b}\ \vec{c}\right|}{abc + \left(\vec{a}\cdot\vec{b}\right)c + (\vec{a}\cdot\vec{c})\,b + \left(\vec{b}\cdot\vec{c}\right)a}$$
$$\theta_s = \frac{\theta_a + \theta_b + \theta_c}{2}$$

1.16.4 常用体积公式

- Pyramid $V = \frac{1}{3}Sh$.
- Sphere $V = \frac{4}{3}\pi R^3$.
- Frustum $V = \frac{1}{3}h(S_1 + \sqrt{S_1S_2} + S_2)$.
- Ellipsoid $V = \frac{4}{3}\pi abc$.

1.16.5 高维球体积

$$V_2 = \pi R^2,\ S_2 = 2\pi R$$
$$V_3 = \frac{4}{3}\pi R^3,\ S_3 = 4\pi R^2$$
$$V_4 = \frac{1}{2}\pi^2 R^4,\ S_4 = 2\pi^2 R^3$$
$$\text{Generally, } V_n = \frac{2\pi}{n}V_{n-2},\ S_{n-1} = \frac{2\pi}{n-2}S_{n-3}$$
$$\text{Where, } S_0 = 2,\ V_1 = 2,\ S_1 = 2\pi,\ V_2 = \pi$$

1.17 三维几何基础操作

```
/* 右手系逆时针绕轴旋转，$(x, y, z)A = (x_{new}, y_{new}, z_{new})$
new[i] += old[j] * A[j][i] */
void calc(p3 n, double cosw) {
|   double sinw = sqrt(1 - cosw * cosw);
|   n.normalize();
|   for (int i = 0; i < 3; i++) {
|   |   int j = (i + 1) % 3, k = (j + 1) % 3;
|   |   double x = n[i], y = n[j], z = n[k];
|   |   A[i][i] = (y * y + z * z) * cosw + x * x;
|   |   A[i][j] = x * y * (1 - cosw) + z * sinw;
|   |   A[i][k] = x * z * (1 - cosw) - y * sinw; } }
p3 cross (const p3 & a, const p3 & b) {
|   return p3(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x); }
double mix(p3 a, p3 b, p3 c) {
|   return dot(cross(a, b), c); }
struct Line { p3 s, t; };
struct Plane { // nor 为单位法向量，离原点距离 m
|   p3 nor; double m;
|   Plane(p3 r, p3 a) : nor(r){
|   |   nor = 1 / r.len() * r;
|   |   m = dot(nor, a); } };
// 以下函数注意除以0的情况
// 点到平面投影
p3 project_to_plane(p3 a, Plane b) {
return a + (b.m - dot(a, b.nor)) * b.nor; }
// 点到直线投影
p3 project_to_line(p3 a, Line b) {
return b.s + dot(a - b.s, b.t - b.s) / dot(b.t - b.s, b.t - b.s) * (b.t - b.s); }
// 直线与直线最近点
pair<p3, p3> closest_two_lines(Line x, Line y) {
double a = dot(x.t - x.s, x.t - x.s);
double b = dot(x.t - x.s, y.t - y.s);
double e = dot(y.t - y.s, y.t - y.s);
double d = a*e - b*b; p3 r = x.s - y.s;
double c = dot(x.t - x.s, r), f = dot(y.t - y.s, r);
double s = (b*f - c*e) / d, t = (a*f - c*b) / d;
return {x.s + s*(x.t - x.s), y.s + t*(y.t - y.s)}; }
// 直线与平面交点
p3 intersect(Plane a, Line b) {
double t = dot(a.nor, a.m * a.nor - b.s) / dot(a.nor, b.t - b.s);
return b.s + t * (b.t - b.s); }
// 平面与平面求交线
Line intersect(Plane a, Plane b) {
p3 d=cross(a.nor,b.nor), d2=cross(b.nor,d);
double t = dot(d2, a.nor);
p3 s = 1 / t * (a.m - dot(b.m * b.nor, a.nor)) * d2 + b.m * b.nor;
return (Line) {s, s + d}; }
// 三个平面求交点
p3 intersect(Plane a, Plane b, Plane c) {
return intersect(a, intersect(b, c));
p3 c1 (a.nor.x, b.nor.x, c.nor.x);
p3 c2 (a.nor.y, b.nor.y, c.nor.y);
p3 c3 (a.nor.z, b.nor.z, c.nor.z);
p3 c4 (a.m, b.m, c.m);
return 1 / mix(c1, c2, c3) * p3(mix(c4, c2, c3), mix(c1, c4, c3), mix(c1, c2, c4)); }
```

1.18 三维凸包

```
vector <p3> p;
int mark[N][N], stp;
typedef array <int, 3> Face;
vector <Face> face;
double volume (int a, int b, int c, int d) {
|   return mix (p[b] - p[a], p[c] - p[a], p[d] - p[a]); }
void ins(int a, int b, int c) {face.push_back({a, b, c});}
void add(int v) {
|   vector <Face> tmp; int a, b, c; stp++;
|   for (auto f : face) {
|   |   if (sgn(volume(v, f[0], f[1], f[2])) < 0) {
|   |   |   for (auto i : f) for (auto j : f)
|   |   |   |   mark[i][j] = stp; }
|   |   else {
|   |   |   tmp.push_back(f);}
|   } face = tmp;
|   for (int i = 0; i < (int) tmp.size(); i++) {
|   |   a = tmp[i][0], b = tmp[i][1], c = tmp[i][2];
|   |   if (mark[a][b] == stp) ins(b, a, v);
|   |   if (mark[b][c] == stp) ins(c, b, v);
|   |   if (mark[c][a] == stp) ins(a, c, v); } }
bool Find(int n) {
|   for (int i = 2; i < n; i++) {
|   |   p3 ndir = cross (p[0] - p[i], p[1] - p[i]);
|   |   if (ndir == p3(0,0,0)) continue;
|   |   swap(p[i], p[2]);
|   |   for (int j = i + 1; j < n; j++) {
|   |   |   if (sgn(volume(0, 1, 2, j)) != 0) {
|   |   |   |   swap(p[j], p[3]);
|   |   |   |   ins(0, 1, 2);
|   |   |   |   ins(0, 2, 1);
|   |   |   |   return 1;
|   } } } return 0; }
mt19937 rng;
bool solve() {
|   face.clear();
|   int n = (int) p.size();
|   shuffle(p.begin(), p.end(), rng);
|   if (!Find(n)) return 0;
|   for (int i = 3; i < n; i++) add(i);
|   return 1; }
```

1.19 最小覆盖球

```
vector<p3> vec;
Circle calc() {
|   if(vec.empty()) { return Circle(p3(0, 0, 0), 0);
|   }else if(1 == (int)vec.size()) {return Circle(vec[0], 0);
|   }else if(2 == (int)vec.size()) {
|   |   return Circle(0.5 * (vec[0] + vec[1]), 0.5 * (vec[0] - vec[1]).len());
|   }else if(3 == (int)vec.size()) {
|   |   double r = (vec[0] - vec[1]).len() * (vec[1] - vec[2]).len() * (vec[2] - vec[0]).len() / 2 / fabs(cross(vec[0] - vec[2], vec[1] - vec[2]).len());
|   |   Plane ppp1 = Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0]));
|   |   return Circle(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])), Plane(vec[2] - vec[1], 0.5 * (vec[2] + vec[1])), Plane(cross(vec[1] - vec[0], vec[2] - vec[0]), vec[0])), r);
|   }else {
|   |   p3 o(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])), Plane(vec[2] - vec[0], 0.5 * (vec[2] + vec[0])), Plane(vec[3] - vec[0], 0.5 * (vec[3] + vec[0]))));
|   |   return Circle(o, (o - vec[0]).len()); } }
Circle miniBall(int n) {
|   Circle res(calc());
|   for(int i(0); i < n; i++) {
|   |   if(!in_circle(a[i], res)) { vec.push_back(a[i]);
|   |   |   res = miniBall(i); vec.pop_back();
|   |   |   if(i) { p3 tmp(a[i]);
|   |   |   |   memmove(a + 1, a, sizeof(p3) * i);
|   |   |   |   a[0] = tmp; } } }
|   return res; }
int main() {
|   int n; scanf("%d", &n);
|   for(int i(0); i < n; i++) a[i].scan();
|   sort(a, a + n); n = unique(a, a + n) - a;
|   vec.clear(); random_shuffle(a, a + n);
|   printf("%.10f\n", miniBall(n).r); }
```

2. Tree & Graph

2.1 Hopcroft-Karp $O(\sqrt{V}E)$

```
// 左侧n个点，右侧k个点 ,1-base, 初始化将mx[],my[] 都置为0
int n, m, k, q[N], dx[N], dy[N], mx[N], my[N];
vector <int> E[N];
bool bfs() { bool flag = 0; int qt = 0, qh = 0;
|  for(int i = 1; i <= k; ++ i) dy[i] = 0;
|  for(int i = 1; i <= n; ++ i) { dx[i] = 0;
|  |  if (! mx[i]) q[qt ++] = i; }
|  while (qh < qt) { int u = q[qh ++];
|  |  for(auto v : E[u]) {
|  |  |  if (! dy[v]) { dy[v] = dx[u] + 1;
|  |  |  |  if (! my[v]) flag = 1; else {
|  |  |  |  |  dx[my[v]] = dx[u] + 2;
|  |  |  |  |  q[qt ++] = my[v]; } } } }
|  return flag; }
bool dfs(int u) {
|  for(auto v : E[u]) {
|  |  if (dy[v] == dx[u] + 1) { dy[v] = 0;
|  |  |  if (! my[v] || dfs(my[v])) {
|  |  |  |  mx[u] = v; my[v] = u; return 1; }}}
|  return 0; }
void hk() {
fill(mx + 1, mx + n + 1, 0); fill(my + 1, my + k + 1, 0);
while (bfs()) for(int i=1; i<=n; ++i) if (!mx[i]) dfs(i);}
```

2.2 Hungarian $O(VE)$

```
int n, k, mat[N], vis[N]; vector<int> E[N];
bool dfs(int tim, int x) {
|  for (auto y : E[x]) {
|  |  if (vis[y] == tim) continue;
|  |  vis[y] = tim;
|  |  if (!mat[y] || dfs(tim, mat[y])) {
|  |  |  mat[y] = x; return 1; } }
|  return 0; }
int solve() {
|  fill(vis + 1, vis + k + 1, 0);
|  int ans = 0;
|  for (int i = 1; i <= n; ++i)
|  |  if (dfs(i, i)) ans++;
|  return ans; }
```

2.3 Shuffle 一般图最大匹配 $O(VE)$

```
mt19937 rng(233);
int n, m, mat[N], vis[N]; vector<int> E[N];
bool dfs(int tim, int x) {
|  shuffle(E[x].begin(), E[x].end(), rng);
|  vis[x] = tim;
|  for (auto y : E[x]) {
|  |  int z = mat[y]; if (vis[z] == tim) continue;
|  |  mat[x] = y, mat[y] = x, mat[z] = 0;
|  |  if (!z || dfs(tim, z)) return true;
|  |  mat[x] = 0, mat[y] = z, mat[z] = y; }
|  return false; }
int main() {
|  for (int T = 0; T < 10; ++T) {
|  |  fill(vis + 1, vis + n + 1, 0);
|  |  for (int i = 1; i <= n; ++i)
|  |  |  if (!mat[i]) dfs(i, i); } }
```

2.4 极大团计数

```
// 0下标，需删除自环（即确保 $E_{ii} = false$，补图要特别注意）
// 极大团计数，最坏情况O(3^(n/3))
ll ans; ull E[64]; #define bit(i) (1ULL << (i))
void dfs(ull P, ull X, ull R) { // 不要方案时可去掉R
|  if (!P && !X) { ++ans; sol.pb(R); return; }
|  ull Q = P & ~E[__builtin_ctzll(P | X)];
|  for (int i; i = __builtin_ctzll(Q), Q; Q &= ~bit(i)) {
|  |  dfs(P & E[i], X & E[i], R | bit(i));
|  |  P &= ~bit(i), X |= bit(i); }}
ans = 0; dfs(n == 64 ? ~0ULL : bit(n) - 1, 0, 0);
```

2.5 有根树同构 Hash

```
ULL get(vector <ULL> ha) {
|  sort(ha.begin(), ha.end());
|  ULL ret = 0xdeadbeef;
|  for (auto i : ha) {
|  |  ret = ret * P + i;
|  |  ret ^= ret << 17; }
|  return ret * 997; }
```

2.6 KM 最大权匹配 $O(V^3)$

```
struct KM {
int n, nl, nr;
LL a[N][N];
LL hl[N], hr[N], slk[N];
int fl[N], fr[N], vl[N], vr[N], pre[N], q[N], ql, qr;
int check(int i) {
|  if (vl[i] = 1, fl[i] != -1)
|  |  return vr[q[qr++] = fl[i]] = 1;
|  while (i != -1) swap(i, fr[fl[i] = pre[i]]);
|  return 0; }
void bfs(int s) {
|  fill(slk, slk + n, INF);
|  fill(vl, vl + n, 0); fill(vr, vr + n, 0);
|  q[ql = 0] = s; vr[s] = qr = 1;
|  for (LL d;;) {
|  |  for (; ql < qr; ++ql)
|  |  for (int i = 0, j = q[ql]; i < n; ++i)
|  |  if (d=hl[i]+hr[j]-a[i][j], !vl[i] && slk[i] >= d) {
|  |  |  if (pre[i] = j, d) slk[i] = d;
|  |  |  else if (!check(i)) return; }
|  |  d = INF;
|  |  for (int i = 0; i < n; ++i)
|  |  if (!vl[i] && d > slk[i]) d = slk[i];
|  |  for (int i = 0; i < n; ++i) {
|  |  |  if (vl[i]) hl[i] += d; else slk[i] -= d;
|  |  |  if (vr[i]) hr[i] -= d; }
|  |  for (int i = 0; i < n; ++i)
|  |  |  if (!vl[i] && !slk[i] && !check(i)) return; } }
void solve() {
|  n = max(nl, nr);
|  fill(pre, pre + n, -1); fill(hr, hr + n, 0);
|  fill(fl, fl + n, -1); fill(fr, fr + n, -1);
|  for (int i = 0; i < n; ++i)
|  |  hl[i] = *max_element(a[i], a[i] + n);
|  for (int i = 0; i < n; ++i)
|  |  bfs(i); }
LL calc() {
|  LL ans = 0;
|  for (int i = 0; i < nl; ++i)
|  |  if (~fl[i]) ans += a[i][fl[i]];
|  return ans; }
void output() {
|  for (int i = 0; i < nl; ++i)
|  printf("%d ", (~fl[i] && a[i][fl[i]] ? fl[i] + 1 : 0));
} } km;
```

2.7 Tarjan 点双，边双

```
/** 边双 **/
int n, m, head[N], nxt[M << 1], to[M << 1], ed;
int dfn[N], low[N], bcc_id[N], bcc_cnt, stp;
bool bri[M << 1], vis[N];
vector<int> bcc[N];
void tar(int now, int fa) {
|  dfn[now] = low[now] = ++stp;
|  for (int i = head[now]; ~i; i = nxt[i]) {
|  |  if (!dfn[to[i]]) {
|  |  |  tar(to[i], now);
|  |  |  low[now] = min(low[now], low[to[i]]);
|  |  |  if (low[to[i]] > dfn[now])
|  |  |  |  bri[i] = bri[i ^ 1] = 1; }
|  |  else if (dfn[to[i]] < dfn[now] && to[i] != fa)
|  |  |  low[now] = min(low[now], dfn[to[i]]); } }
void DFS(int now) {
|  vis[now] = 1;
|  bcc_id[now] = bcc_cnt;
|  bcc[bcc_cnt].push_back(now);
|  for (int i = head[now]; ~i; i = nxt[i]) {
|  |  if (bri[i]) continue;
|  |  if (!vis[to[i]]) DFS(to[i]); } }
void EBCC() {// clear dfn low bri bcc_id vis
|  bcc_cnt = stp = 0;
|  for (int i = 1; i <= n; ++i) if (!dfn[i]) tar(i, 0);
|  for (int i = 1; i <= n; ++i)
|  |  if (!vis[i]) ++bcc_cnt, DFS(i); }
/** 点双 **/
vector<int> G[N],bcc[N];
```

```
int dfn[N], low[N], bcc_id[N], bcc_cnt, stp;
bool iscut[N]; pii stk[N]; int top;
void tar(int now, int fa) {
|  int child = 0;
|  dfn[now] = low[now] = ++stp;
|  for (int to: G[now]) {
|  |  if (!dfn[to]) {
|  |  |  stk[++top] = mkpair(now, to); ++child;
|  |  |  tar(to, now);
|  |  |  low[now] = min(low[now], low[to]);
|  |  |  if (low[to] >= dfn[now]) {
|  |  |  |  iscut[now] = 1;
|  |  |  |  bcc[++bcc_cnt].clear();
|  |  |  |  while (1) {
|  |  |  |  |  pii tmp = stk[top--];
|  |  |  |  |  if (bcc_id[tmp.first] != bcc_cnt) {
|  |  |  |  |  |  bcc[bcc_cnt].push_back(tmp.first);
|  |  |  |  |  |  bcc_id[tmp.first] = bcc_cnt; }
|  |  |  |  |  if (bcc_id[tmp.second] != bcc_cnt) {
|  |  |  |  |  |  bcc[bcc_cnt].push_back(tmp.second);
|  |  |  |  |  |  bcc_id[tmp.second] = bcc_cnt; }
|  |  |  |  |  if (tmp.first == now && tmp.second == to)
|  |  |  |  |  |  break; } } }
|  |  else if (dfn[to] < dfn[now] && to != fa) {
|  |  |  stk[++top] = mkpair(now, to);
|  |  |  low[now] = min(low[now], dfn[to]); } }
|  if (!fa && child == 1) iscut[now] = 0; }
void PBCC() { // clear dfn low iscut bcc_id
|  stp = bcc_cnt = top = 0;
|  for (int i = 1; i <= n; ++i) if (!dfn[i]) tar(i, 0); }
```

2.8 2-SAT

```
int stp, comps, top;//清点清边要两倍
int dfn[N], low[N], comp[N], stk[N];
void add(int x, int a, int y, int b) {
//取 $X_a$ 则必须取 $Y_b$, 则 $X_a$ 向 $Y_b$ 连边
//注意连边是对称的, 即, 此时实际上 $X_b$ 也必须向 $Y_a$ 连边.
|  E[x << 1 | a].push_back(y << 1 | b); }
void tarjan(int x) {
|  dfn[x] = low[x] = ++stp;
|  stk[top++] = x;
|  for (auto y : E[x]) {
|  |  if (!dfn[y])
|  |  |  tarjan(y), low[x] = min(low[x], low[y]);
|  |  else if (!comp[y])
|  |  |  low[x] = min(low[x], dfn[y]);
|  }
|  if (low[x] == dfn[x]) {
|  |  comps++;
|  |  do {int y = stk[--top];
|  |  |  comp[y] = comps;
|  |  } while (stk[top] != x);
|  } }
bool answer[N];
bool solve() {
|  int cnt = n + n + 1;
|  stp = top = comps = 0;
|  fill(dfn, dfn + cnt, 0);
|  fill(comp, comp + cnt, 0);
|  for (int i = 0; i < cnt; ++i) if (!dfn[i]) tarjan(i);
|  for (int i = 0; i < n; ++i) {
|  |  if (comp[i << 1] == comp[i << 1 | 1]) return false;
|  |  answer[i] = (comp[i << 1 | 1] < comp[i << 1]); }
|  return true; }
```

2.9 Dominator Tree 支配树

```
struct Dominator_Tree{
//n为点数,s为起点,e[] 中记录每条边
int n,s,cnt;int dfn[N],id[N],pa[N],
  ↪ semi[N],idom[N],p[N],mn[N];
vector<int>e[N],dom[N],be[N];
void dfs(int x){//先得到DFS树
|  dfn[x]=++cnt;id[cnt]=x;
|  for(auto i:e[x]){
|  |  if(!dfn[i])dfs(i),pa[dfn[i]]=dfn[x];
|  |  be[dfn[i]].push_back(dfn[x]);
|  }}
int get(int x){//带权并查集
|  if(p[x]!=p[p[x]]){
|  |  if(semi[mn[x]]>semi[get(p[x])]) mn[x]=get(p[x]);
|  |  p[x]=p[p[x]];
|  }return mn[x];
}void LT(){//求出semi和idom得到支配树
|  for(int i=cnt;i>1;i--){
|  |  for(auto j:be[i]) semi[i]=min(semi[i],semi[get(j)]);
|  |  dom[semi[i]].push_back(i); int x=p[i]=pa[i];
|  |  for(auto j:dom[x]) idom[j]=(semi[get(j)]<x?get(j):x);
|  |  dom[x].clear();
|  }for(int i=2;i<=cnt;i++){
|  |  if(idom[i]!=semi[i])idom[i]=idom[idom[i]];
|  |  dom[id[idom[i]]].push_back(id[i]);
|  }
}void build(){//建立支配树
|  for(int i=1;i<=n;i++) dfn[i]=0,dom[i].clear(),
|  |  be[i].clear(),p[i]=mn[i]=semi[i]=i;
|  cnt=0;dfs(s);LT();
} }G;
```

2.10 Minimum Mean Cycle

```
// 点标号为 $1, 2, \cdots, n$, 0为虚拟源点向其他点连权值为0的单向边.
// f[i][v] : 从 0 到 v 恰好经过 i 条路的最短路
ll f[N][N] = {Inf}; int u[M], v[M], w[M]; f[0][0] = 0;
for(int i = 1; i <= n + 1; i ++)
|  for(int j = 0; j < m; j ++)
|  |  f[i][v[j]] = min(f[i][v[j]], f[i - 1][u[j]] + w[j]);
double ans = Inf;
for(int i = 1; i <= n; i ++) {
|  double t = -Inf;
|  for(int j = 1; j <= n; j ++)
|  |  t = max(t, (f[n][i] - f[j][i]) / (double)(n - j));
|  ans = min(t, ans); }
```

2.11 弦图

弦图的定义 连接环中不相邻的两个点的边称为弦. 一个无向图称为弦图, 当图中任意长度都大于 3 的环都至少有一个弦.

单纯点 一个点称为单纯点当 $\{v\} \cup A(v)$ 的导出子图为一个团. 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.

完美消除序列 一个序列 $v_1, v_2, ..., v_n$ 满足 v_i 在 $v_i, \cdots, v_n$ 的诱导子图中为一个单纯点. 一个无向图是弦图当且仅当它有一个完美消除序列.

最大势算法 从 n 到 1 的顺序依次给点标号. 设 label_i 表示第 i 个点与多少个已标号的点相邻, 每次选择 label 最大的未标号的点进行标号. 用桶维护优先队列可以做到 $O(n+m)$.

弦图的判定 判定最大势算法输出是否合法即可. 如果依次判断是否构成团, 时间复杂度为 $O(nm)$. 考虑优化, 设 $v_{i+1}, \cdots, v_n$ 中所有与 v_i 相邻的点依次为 $N(v_i) = \{v_{j1}, \cdots, v_{jk}\}$. 只需判断 v_{j1} 是否与 $v_{j2}, \cdots, v_{jk}$ 相邻即可. 时间复杂度 $O(n+m)$.

弦图的染色 完美消除序列从后往前染色, 染上出度的 mex.

最大独立集 完美消除序列从前往后能选就选.

团数 最大团的点数. 一般图团数 $\leq$ 色数, 弦图团数 $=$ 色数.

极大团 弦图的极大团一定为 $\{x\} \cup N(x)$.

最小团覆盖 用最少的团覆盖所有的点. 设最大独立集为 $\{p_1, \ldots, p_t\}$, 则 $\{p_1 \cup N(p_1), \ldots, p_t \cup N(p_t)\}$ 为最小团覆盖.

弦图 k 染色计数 $\prod_{v \in V} k - N(v) + 1$.

区间图 每个顶点代表一个区间, 有边当且仅当区间有交. 区间图是弦图, 一个完美消除序列是右端点排序.

```

vector <int> L[N];
int seq[N], lab[N], col[N], id[N], vis[N];
void mcs() {
|  for (int i = 0; i < n; i++) L[i].clear();
|  fill(lab + 1, lab + n + 1, 0);
|  fill(id + 1, id + n + 1, 0);
|  for (int i = 1; i <= n; i++) L[0].push_back(i);
|  int top = 0;
|  for (int k = n; k; k--) {
|  |  int x = -1;
|  |  for ( ; ; ) {
|  |  |  if (L[top].empty()) top --;
|  |  |  else {
|  |  |  |  x = L[top].back(), L[top].pop_back();
|  |  |  |  if (lab[x] == top) break;
|  |  |  }
|  |  }
|  |  seq[k] = x; id[x] = k;
|  |  for (auto v : E[x]) {
|  |  |  if (!id[v]) {
|  |  |  |  L[++lab[v]].push_back(v);
|  |  |  |  top = max(top, lab[v]);
|  |  |  } } } }
bool check() {
|  fill(vis + 1, vis + n + 1, 0);
|  for (int i = n; i; i--) {
|  |  int x = seq[i];
|  |  vector <int> to;
|  |  for (auto v : E[x])
|  |  |  if (id[v] > i) to.push_back(v);
|  |  if (to.empty()) continue;
|  |  int w = to.front();
|  |  for (auto v : to) if (id[v] < id[w]) w = v;
|  |  for (auto v : E[w]) vis[v] = i;
|  |  for (auto v : to)
|  |  |  if (v != w && vis[v] != i) return false;
|  } return true; }
void color() {
|  fill(vis + 1, vis + n + 1, 0);
|  for (int i = n; i; i--) {
|  |  int x = seq[i];
|  |  for (auto v : E[x]) vis[col[v]] = x;
|  |  for (int c = 1; !col[x]; c++)
|  |  |  if (vis[c] != x) col[x] = c;
|  } }
```

2.12 欧拉回路

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

2.13 斯坦纳树

```
LL d[1 << 10][N]; int c[15];
priority_queue < pair <LL, int> > q;
void dij(int S) {
|  for (int i = 1; i <= n; i++) q.push(mp(-d[S][i], i));
|  while (!q.empty()) {
|  |  pair <LL, int> o = q.top(); q.pop();
|  |  if (-o.x != d[S][o.y]) continue;
|  |  int x = o.y;
|  |  for (auto v : E[x]) if (d[S][v.v] > d[S][x] + v.w) {
|  |  |  d[S][v.v] = d[S][x] + v.w;
|  |  |  q.push(mp(-d[S][v.v], v.v));}}}
void solve() {
|  for (int i = 1; i < (1 << K); i++)
|  |  for (int j = 1; j <= n; j++) d[i][j] = INF;
|  for (int i = 0; i < K; i++) read(c[i]), d[1 << i][c[i]] = 0;
|  for (int S = 1; S < (1 << K); S++) {
|  |  for (int k = S; k > (S >> 1); k = (k - 1) & S) {
|  |  |  for (int i = 1; i <= n; i++) {
|  |  |  |  d[S][i] = min(d[S][i], d[k][i] + d[S ^ k][i]);
|  |  } }|  dij(S);}}
```

2.14 Dinic 最大流

复杂度证明思路 假设 dist 为残量网络上的距离. Dinic 一轮增广会找到一个极大的长度为 $\text{dist}(s,t)$ 的增广路集合, 称为 blocking flow. 增广 blocking flow 后 $\text{dist}(s,t)$ 将会增大. 因此, 最多只有 $O(V)$ 轮, 如果一轮增广是 $O(VE)$ 的, 那么总复杂度是 $O(V^2E)$.

注意, 没有当前弧优化的 Dinic 复杂度应是指数级别的.

单位流量网络 在 0-1 流量图上 Dinic 有更好的性质.

- 复杂度为 $O(\min\{V^{2/3}, E^{1/2}\}E)$.
- $\text{dist}(s,t) = d$, 残量网络上至多还存在 E/d 的流.
- 如果是每个点只有一个入/出度, 复杂度 $O(V^{1/2}E)$. 一个著名的特殊情况是 Hopcroft-Karp.

```
struct edge {
|  int v, nxt; LL f;
} e[M * 2];
int ecnt = 1, head[N], cur[N];
void add(int u, int v, LL f) {
|  e[++ecnt] = {v, head[u], f};   head[u] = ecnt;
|  e[++ecnt] = {u, head[v], 0ll}; head[v] = ecnt; }
int n, S, T;
int q[N], tag[N], he = 0, ta = 1;
bool bfs() {
|  for (int i = S; i<= T; i++) tag[i] = 0;
|  he = 0, ta = 1; q[0] = S;
|  tag[S] = 1;
|  while (he < ta) {
|  |  int x = q[he++];
|  |  for (int o = head[x]; o; o = e[o].nxt)
|  |  |  if (e[o].f && !tag[e[o].v])
|  |  |  |  tag[e[o].v] = tag[x] + 1, q[ta++] = e[o].v;
|  }
|  return !!tag[T]; }
LL dfs(int x, LL flow) {
|  if (x == T) return flow;
|  LL used = 0;
|  for (int &o = cur[x]; o; o = e[o].nxt) {
|  |  if (e[o].f && tag[x] < tag[e[o].v]) {
|  |  |  LL ret = dfs(e[o].v, min(flow - used, e[o].f));
|  |  |  if (ret) {
|  |  |  |  e[o].f -= ret; e[o ^ 1].f += ret;
|  |  |  |  used += ret;
|  |  |  |  if (used == flow) return flow;
|  |  |  } } }
|  return used; }
LL dinic() {
|  LL ans = 0;
|  while (bfs()) {
|  |  for (int i = S; i <= T; i++) cur[i] = head[i];
|  |  ans += dfs(S, INF);
|  } return ans; }
```

2.15 Dijkstra 费用流

```
pii solve() {
|  LL res = 0, flow = 0;
|  for (int i = S; i <= T; i++) h[i] = 0;
|  while (true) {// first time may SPFA
|  |  priority_queue <pii, vector<pii>, greater<pii>> q;
|  |  for (int i = S; i <= T; i++) dis[i] = INF;
|  |  dis[S] = 0; q.push(pii(0, S));
|  |  while (!q.empty()) {
|  |  |  pii now = q.top(); q.pop(); int x = now.second;
|  |  |  if (dis[x] < now.first) continue;
|  |  |  for (int o = head[x]; o; o = e[o].nxt) {
|  |  |  |  if (e[o].f > 0 && dis[e[o].v] > dis[x] + e[o].w + h[x] - h[e[o].v]) {
|  |  |  |  |  dis[e[o].v] = dis[x] + e[o].w + h[x] - h[e[o].v];
|  |  |  |  |  prevv[e[o].v] = x; pree[e[o].v] = o;
|  |  |  |  |  q.push(pii(dis[e[o].v], e[o].v)); } } }
|  |  if (dis[T] == INF) break;
|  |  for (int i = S; i <= T; i++) h[i] += dis[i];
|  |  int d = INF;
|  |  for (int v = t; v != S; v = prevv[v]) d = min(d, e[pree[v]].f);
|  |  flow += d; res += (LL)d * h[t];
|  |  for (int v = t; v != S; v = prevv[v]) {
|  |  |  e[pree[v]].f -= d; e[pree[v] ^ 1].f += d; } }
|  return make_pair(flow, res); }
```

2.16 Gomory-Hu 无向图最小割树 $O(V^3E)$

每次随便找两个点 s, t 求在原图的最小割, 在最小割树上连 (s, t, w_{cut}), 递归对由割集隔开的部分继续做. 在得到的树上, 两点最小割即为树上瓶颈路. 实现时, 由于是随意找点, 可以写为分治的形式.

2.17 Stoer-Wagner 无向图最小割 $O(VE + V^2 \log V)$

```
const int N = 601;
int f[N], siz[N], G[N][N];
int getf(int x) {return f[x] == x ? x : f[x] = getf(f[x]);}
int dis[N], vis[N], bin[N];
int n, m;
int contract(int &s, int &t) {  // Find s,t
|  memset(dis, 0, sizeof(dis));
|  memset(vis, 0, sizeof(vis));
|  int i, j, k, mincut, maxc;
|  for (i = 1; i <= n; i++) {
|  |  k = -1; maxc = -1;
|  |  for (j = 1; j <= n; j++)
|  |  |  if (!bin[j] && !vis[j] && dis[j] > maxc) {
|  |  |  |  k = j;
|  |  |  |  maxc = dis[j];
|  |  |  }
|  |  if (k == -1) return mincut;
|  |  s = t; t = k; mincut = maxc; vis[k] = true;
|  |  for (j = 1; j <= n; j++)
|  |  |  if (!bin[j] && !vis[j]) dis[j] += G[k][j];
|  } return mincut; }
const int inf = 0x3f3f3f3f;
int solve() {
|  int mincut, i, j, s, t, ans;
|  for (mincut = inf, i = 1; i < n; i++) {
|  |  ans = contract(s, t);
|  |  bin[t] = true;
|  |  if (mincut > ans) mincut = ans;
|  |  if (mincut == 0) return 0;
|  |  for (j = 1; j <= n; j++)
|  |  |  if (!bin[j]) G[s][j] = (G[j][s] += G[j][t]);
|  } return mincut; }
int main() {
|  cin >> n >> m;
|  for (int i = 1; i <= n; ++i) f[i] = i, siz[i] = 1;
|  for (int i = 1, u, v, w; i <= m; ++i) {
|  |  cin >> u >> v >> w;
|  |  int fu = getf(u), fv = getf(v);
|  |  if (fu != fv) {
|  |  |  if (siz[fu] > siz[fv]) swap(fu, fv);
|  |  |  f[fu] = fv, siz[fv] += siz[fu]; }
|  |  G[u][v] += w, G[v][u] += w; }
|  cout << (siz[getf(1)] != n ? 0 : solve()); }

```

2.18 网络流总结

2.18.1 最小割集，最小割必须边以及可行边

最小割集　从 S 出发，在残余网络中BFS所有权值非 0 的边（包括反向边），得到点集 $\{S\}$，另一集为 $\{V\} - \{S\}$.

最小割集必须点　残余网络中S直接连向的点必在S的割集中，直接连向T的点必在T的割集中；若这些点的并集为全集，则最小割方案唯一.

最小割可行边　在残余网络中求强联通分量，将强联通分量缩点后，剩余的边即为最小割可行边，同时这些边也必然满流.

最小割必须边　在残余网络中求强联通分量，若S出发可到u，T出发可到v，等价于 $scc_S = scc_u$ 且 $scc_T = scc_v$，则该边为必须边.

2.18.2 常见问题

最大权闭合子图　适用问题：每个点有点权，限制条件形如：选择A则必须选择B，选择B则必须选择C，D．建图方式：B向A连边，CD向B连边．求解：S向正权点连边，负权点向T连边，其余边容量 ∞，求最小割，答案为S所在最小割集.

二元关系　适用问题：有 n 个元素，每个元素可选A或者B，各有代价；有 m 个限制条件，若元素 i 与 j 的种类不同则产生额外的代价，求最小代价．求解：S向i连边 A_i，i向T连边 B_i，一组限制 (i, j) 代价为 z，则i与j之间连双向容量为 z 的边，求最小割.

混合图欧拉回路　把无向边随便定向，计算每个点的入度和出度，如果有某个点出入度之差 $deg_i = in_i - out_i$ 为奇数，肯定不存在欧拉回路．对于 $deg_i > 0$ 的点，连接边 $(i, T, deg_i/2)$；对于 $deg_i < 0$ 的点，连接边 $(S, i, -deg_i/2)$．最后检查是否满流即可.

二物流　水源 S_1，水汇 T_1，油源 S_2，油汇 T_2，每根管道流量共用．求流量和最大．建超级源 SS_1 汇 TT_1，连边 $SS_1 \to S_1$,$SS_1 \to S_2$,$T_1 \to TT_1$,$T_2 \to TT_1$，设最大流为 x_1．建超级源 SS_2 汇 TT_2，连边 $SS_2 \to S_1$，$SS_2 \to T_2$,$T_1 \to TT_2$，$S_2 \to TT_2$，设最大流为 x_2．则最大流中水流量 $\frac{x_1+x_2}{2}$，油流量 $\frac{x_1-x_2}{2}$.

2.18.3 一些网络流建图

无源汇有上下界可行流　每条边 (u, v) 有一个上界容量 $C_{u,v}$ 和下界容量 $B_{u,v}$，我们让下界变为 0，上界变为 $C_{u,v} - B_{u,v}$，但这样做流量不守恒．建立超级源点 SS 和超级汇点 TT，用 du_i 来记录每个节点的流量情况，$du_i = \sum B_{j,i} - \sum B_{i,j}$，添加一些附加弧．当 $du_i > 0$ 时，连边 (SS, i, du_i)；当 $du_i < 0$ 时，连边 $(i, TT, -du_i)$．最后对 (SS, TT) 求一次最大流即可，当所有附加边全部满流时（即 $maxflow == du_i > 0$）时有可行解.

有源汇有上下界最大可行流　建立超级源点 SS 和超级汇点 TT，首先判断是否存在可行流，用无源汇有上下界可行流的方法判断．增设一条从 T 到 S 没有下界容量为无穷的边，那么原图就变成了一个无源汇有上下界可行流问题．同样地建图后，对 (SS, TT) 进行一次最大流，判断是否有可行解．如果有可行解，删除超级源点 SS 和超级汇点 TT，并删去 T 到 S 的这条边，再对 (S, T) 进行一次最大流，此时得到的 $maxflow$ 即为有源汇有上下界最大可行流.

有源汇有上下界最小可行流　建立超级源点 SS 和超级汇点 TT，和无源汇有上下界可行流一样新增一些边，然后从SS到TT跑最大流．接着加上边 (T, S, ∞)，再从 SS 到 TT 跑一遍最大流．如果所有新增边都是满的，则存在可行流，此时 T 到 S 这条边的流量即为最小可行流.

有上下界费用流　如果求无源汇有上下界最小费用可行流或有源汇有上下界最小费用最大可行流，用1.6.3.1/1.6.3.2 的构图方法，给边加上费用即可．求有源汇有上下界最小费用最小可行流，要先用1.6.3.3的方法建图，先求出一个保证必要边满流情况下的最小费用．如果费用全部非负，那么这时的费用就是答案．如果费用有负数，那么流多了可能更好，继续做从 S 到 T 的流量任意的最小费用流，加上原来的费用就是答案.

费用流消负环　新建超级源SS汇TT，对于所有流量非空的负权边e，先流满（`ans+=e.f*e.c, e.rev.f+=e.f, e.f=0`），再连边SS→e.to，e.from→TT，流量均为e.f(>0)，费用均为0．再连边T→S流量 ∞ 费用0．此时没有负环了．做一遍SS到TT的最小费用最大流，将费用累加ans，拆掉T→S的那条边（此边的流量为残量网络中S→T的流量）．此时负环已消，再继续跑最小费用最大流.

2.18.4 二分图最小点覆盖和最大独立集

最小点覆盖：求出一个最大匹配，从左部开始每次寻找一个未匹配点，从该点出发可以得到“未匹配-匹配-未匹配...”形式的交错树，标记所有这些点．则最小点覆盖方案为右部未标记点与左部标记点的并集．显然最小点覆盖集合大小 = 最大匹配.

最大独立集 = 全集 - 最小点覆盖.

2.18.5 整数线性规划转费用流

首先将约束关系转化为所有变量下界为 0，上界没有要求，并满足一些等式，每个变量在均在等式左边且出现恰好两次，系数为 +1 和 -1，优化目标为 $\max \sum v_i x_i$ 的形式．将等式看做点，等式i右边的值 b_i 若为正，则 S 向 i 连边 $(b_i, 0)$，否则i向T连边 $(-b_i, 0)$．将变量看做边，记变量 x_i 的上界为 m_i(无上界则 $m_i = inf$)，将 x_i 系数为 +1 的那个等式 u 向系数为 -1 的等式 v 连边 (m_i, v_i).

2.19 图论结论

2.19.1 最小乘积问题原理

每个元素有两个权值 $\{x_i\}$ 和 $\{y_i\}$，要求在某个限制下（例如生成树，二分图匹配）使得 $\Sigma x \Sigma y$ 最小．对于任意一种符合限制的选取方法，记 $X = \Sigma x_i, Y = \Sigma y_i$，可看做平面内一点 (X, Y)．答案必在下凸壳上，找出该下凸壳所有点，即可枚举获得最优答案．可以递归求出此下凸壳所有点，分别找出距 x, y 轴最近的两点 A, B，分别对应于 Σy_i，Σx_i 最小．找出距离线段最远的点 C，则 C 也在下凸壳上，C 点满足 $AB \times AC$ 最小，也即

$$(X_B - X_A)Y_C + (Y_A - Y_B)X_C - (X_B - X_A)Y_A - (Y_B - Y_A)X_A$$

最小，后两项均为常数，因此将所以权值改成 $(X_B - X_A)y_i + (Y_B - Y_A)x_i$，求同样问题（例如最小生成树，最小权匹配）即可．求出 C 点以后，递归 AC，BC.

2.19.2 最小环

无向图最小环：每次floyd到 k 时，判断 1 到 $k - 1$ 的每一个 i, j:

$$\text{ans} = \min\{\text{ans}, d(i, j) + G(i, k) + G(k, j)\}.$$

有向图最小环：做完floyd后，$d(i, i)$ 即为经过 i 的最小环.

2.19.3 度序列的可图性

判断一个度序列是否可转化为简单图，除了一种贪心构造的方法外，下列方法更快速．EG定理：将度序列从大到小排序得到 $\{d_i\}$，此序列可转化为简单图当且仅当 (1)Σd_i 为偶数．(2) 对于任意的 $1 \le k \le n - 1$ 满足 $\sum_{i=1}^{k} d_i \le k(k-1) + \sum_{i=k+1}^{n} min(k, d_i)$.

2.19.4 切比雪夫距离与曼哈顿距离转化

曼哈顿距离转切比雪夫距离：$(x + y, x - y)$，适用于一些每次只能向四联通的格子走一格的问题.

切比雪夫距离转曼哈顿距离：$(\frac{x+y}{2}, \frac{x-y}{2})$，适用于一些统计距离的问题.

2.19.5 树链的交

```
bool cmp(int a,int b){return dep[a]<dep[b];}
path merge(path u, path v){
|  int d[4], c[2];
|  if (!u.x||!v.x) return path(0, 0);
|  d[0]=lca(u.x,v.x); d[1]=lca(u.x,v.y);
|  d[2]=lca(u.y,v.x); d[3]=lca(u.y,v.y);
|  c[0]=lca(u.x,u.y); c[1]=lca(v.x,v.y);
|  sort(d,d+4,cmp); sort(c,c+1,cmp);
|  if (dep[c[0]] <= dep[d[0]] && dep[c[1]] <= dep[d[2]])
|  |  return path(d[2],d[3]);
|   else return path(0, 0); }
```

2.19.6 带修改MST

维护一个基于一个图并对少量边进行修改的最小生成树，可以考虑缩点缩边来使得每次暴力求最小生成树复杂度变低．(银川 21：求有 16 个 ‘某两条边中至少选一条’ 的限制条件的最小生成树)

找出必须边　将需要修改的边标记为 $-\infty$，然后跑MST，这时在MST上且不为 $-\infty$ 的边为必须边，将这些边连接的点合并，缩小点集．

找出无用边　将需要修改的边标记为 ∞，然后跑MST，这时不在MST上的且值不为 ∞ 边必为无用边，删除这些边，减少边数．假设当前区间内需要修改的边数为 k，进行删去无用边和找出必须边操作后，图中最多剩下 $k+1$ 个点和 $2k$ 条边．

2.19.7 LCT常见应用

动态维护边双　可以通过LCT来解决一类动态边双连通分量问题．即静态的询问可以用边双连通分量来解决，而树有加边等操作的问题．

把一个边双连通分量缩到LCT的一个点中，然后在LCT上求出答案．缩点的方法为加边时判断两点的连通性，如果已经联通则把两点在目前LCT路径上的点都缩成一个点．

2.19.8 差分约束

若要使得所有量两两的值最接近，则将如果将源点到各点的距离初始化为 0．若要使得某一变量与其余变量的差最大，则将源点到各点的距离初始化为 ，其中之一为 0．若求最小方案则跑最长路，否则跑最短路．

2.19.9 李超线段树

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

2.19.10 吉如一线段树

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

2.19.11 二分图最大匹配

最大独立集最小覆盖点集最小路径覆盖　最大独立集指求一个二分图中最大的一个点集，使得该点集内的点互不相连．最大独立集=总顶点数-最大匹配数．最小覆盖点集指用最少的点，使所有的边至少和一个点有关联．最小覆盖点集=最大匹配数．最小路径覆盖指一个DAG图 G 中用最少的路径使得所有点都被经过．最小路径覆盖=总点数-最大匹配数 (拆点构图)．最大独立集 S 与最小覆盖集 T 互补．构造方法：1. 做最大匹配，没有匹配的空闲点 $u \in S$ 2. 如果 $u \in S$ 那么 u 的邻点必然属于 T 3. 如果一对匹配的点中有一个属于 T 那么另外一个属于 S 4. 还不能确定的，把左子图的放入 S，右子图放入 T．

关键点　一定在最大匹配中的点．由于二分图左右两侧是对称的，我们只考虑找左侧的关键点．先求任意一个最大匹配，然后给二分图定向：匹配边从右到左，非匹配边从左到右，从左侧每个不在最大匹配中的点出发DFS，给到达的那些点打上标记，最终左侧每个没有标记的匹配点即为关键点．时间复杂度 $O(n+m)$．

Hall定理　$G = (X, Y, E), |M| = |X| \Leftrightarrow \forall S \subseteq X, |S| \le |A(S)|$．

2.19.12 稳定婚姻问题

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

2.19.13 三元环

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

2.19.14 图同构

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

2.19.15 竞赛图存在 Landau’ s Theorem

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

2.19.16 Ramsey Theorem

6 个人中至少存在 3 人相互认识或者相互不认识．$R(3, 3) = 6, R(4, 4) = 18$

2.19.17 树的计数 Prufer序列

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

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

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

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

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

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

2.19.18 有根树的计数

首先，令 $S_{n,j} = \sum_{1 \le j \le n/j}$；于是 $n+1$ 个结点的有根树的总数为 $a_{n+1} = \frac{\sum_{j=1}^{n} j a_j S_{n-j}}{n}$．注：$a_1 = 1, a_2 = 1, a_3 = 2, a_4 = 4, a_5 = 9, a_6 = 20, a_9 = 286, a_11 = 1842$．

2.19.19 无根树的计数

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

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

2.19.20 生成树计数 Kirchhoff’s Matrix-Tree Thoerem

Kirchhoff Matrix $T = Deg - A$，Deg 是度数对角阵，A 是邻接矩阵．
无向图度数矩阵是每个点度数；有向图度数矩阵是每个点入度．
邻接矩阵 $A[u][v]$ 表示 $u \to v$ 边个数，重边按照边数计算，自环不计入度数．
无向图生成树计数：$c = |K$ 的任意1个 $n1$ 阶主子式 $|$
有向图外向树计数：$c = |$ 去掉根所在的那阶得到的主子式 $|$

2.19.21 有向图欧拉回路计数 BEST Thoerem

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

其中 deg 为入度 (欧拉图中等于出度)，$t_w(G)$ 为以 w 为根的外向树的个数．相关计算参考生成树计数．
欧拉连通图中任意两点外向树个数相同：$\mathrm{t_v}(G) = \mathrm{t_w}(G)$．

2.19.22 Edmonds Matrix

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

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

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

2.19.23 Count Acyclic Orientations

The chromatic polynomial is a function $P_G(q)$ that counts the number of q-colorings of G.
Triangle K_3 : $t(t-1)(t-2)$
Complete graph K_n : $t(t-1)(t-2)\cdots(t-(n-1))$
Tree with n vertices : $t(t-1)^{n-1}$
Cycle C_n : $(t-1)^n + (-1)^n(t-1)$
The number of acyclic orientations of an n-vertex graph G is $(1)^n P_G(1)$, where $P_G(q)$ is the chromatic polynomial of the graph G.

3. String

3.1 最小表示法

```
int min_pos(vector <int> a) {
|   int n = a.size(), i = 0, j = 1, k = 0;
|   while (i < n && j < n && k < n) {
|   |   auto u = a[(i + k) % n];
|   |   auto v = a[(j + k) % n];
|   |   int t = u > v ? 1 : (u < v ? -1 : 0);
|   |   if (t == 0) k++;
|   |   else {
|   |   |   if (t > 0) i += k + 1; else j += k + 1;
|   |   |   if (i == j) j++;
|   |   |   k = 0; } }
|   return min(i, j); }

```

3.2 Manacher

```
// 这段代码仅仅处理奇回文，使用时请往字符串中间加入 # 来使用
for(int i = 1, j = 0; i != (n << 1) - 1; ++i){
|   int p=i>>1, q = i - p, r = ((j + 1) >> 1) + l[j] - 1;
|   l[i] = r < q ? 0 : min(r - q + 1, l[(j << 1) - i]);
|   while (p - l[i] != -1 && q + l[i] != n
|   |   && s[p - l[i]] == s[q + l[i]]) l[i]++;
|   if(q + l[i] - 1 > r) j=i;
|  a += l[i];
}
```

3.3 Multiple Hash

Primes available at Page 20.

```
const int HA = 2;
const int PP[] = {318255569, 66604919, 19260817},
|   |    QQ[] = {1010451419, 1011111133, 1033111117};
int pw[HA][N];
struct hashInit { hashInit () {
|  for (int h = 0; h < HA; h++) {
|  |  pw[h][0] = 1;
|  |  for (int i = 1; i < N; i++)
|  |  |  pw[h][i] = (LL)pw[h][i - 1] * PP[h] % QQ[h];
|  } } } __init_hash;
struct Hash {
int v[HA], len;
Hash () {memset(v, 0, sizeof v); len = 0;}
Hash (int x) { for (int h = 0; h < HA; h++) v[h] = x; len = 1; }
friend Hash operator + (const Hash &a, const int &b) {
|  Hash ret; ret.len = a.len + 1;
|  for (int h = 0; h < HA; h++)
|  |  ret.v[h] = ((LL)a.v[h] * PP[h] + b) % QQ[h];
|  return ret; }
friend Hash operator - (const Hash &a, const Hash &b) {
|  Hash ret; ret.len = a.len - b.len;
|  for (int h = 0; h < HA; h++) {
|  |  ret.v[h] = (a.v[h] - (LL)pw[h][ret.len] * b.v[h]) % QQ[h];
|  |  if (ret.v[h] < 0) ret.v[h] += QQ[h];
|  } return ret; }
friend bool operator == (const Hash &a, const Hash &b) {
|  for (int h = 0; h < HA; h++)
|  |  if (a.v[h] != b.v[h]) return false;
|  return a.len == b.len; }
// below : not that frequently used
friend Hash operator + (const Hash &a, const Hash &b) {
|  Hash ret; ret.len = a.len + b.len;
|  for (int h = 0; h < HA; h++)
|  |  ret.v[h] = ((LL)a.v[h] * pw[h][b.len] + b.v[h]) % QQ[h];
|  return ret; }
friend Hash operator + (const int &a, const Hash &b) {
|  Hash ret; ret.len = b.len + 1;
|  for (int h = 0; h < HA; h++)
|  |  ret.v[h] = ((LL)a * pw[h][b.len] + b.v[h]) % QQ[h];
|  return ret; }
int to_int() {
|  unsigned ret = 114;
|  for (int h = 0; h < HA; h++) ret = ret * 997 + v[h];
|  return ret & 0x7fffffff; }|  };
```

3.4 KMP exKMP

```
void kmp(string A,int*p){//A 为模式串, p 为失配数组
|  int n=A.length(),i=1,j=0;
|  for(CL(p);i<n;i++){
|  |  for(;j&&A[j]^A[i];j=p[j-1]);
|  |  if(A[i]==A[j])j++;
|  |  p[i]=j;}
}int gans(string A,string B,int*p){
//B 为标准串, A 为待匹配串, p 为失配数组
|  int n=B.length(),m=A.length(),j=0;
|  fr(i,0,m-1){
|  |  for(;j&&B[j]^A[i];j=p[j-1]);
|  |  if(B[j]==A[i])j++;
|  |  if(j==n)an++,j=p[j-1];
|  } return an; }
void exkmp(char *s, int *a, int n) {
// 如果想求一个字符串相对另外一个字符串的最长公共前缀, 可以把他们拼接起来从而求得
|  a[0] = n; int p = 0, r = 0;
|  for (int i = 1; i < n; ++i) {
|  |  a[i] = (r > i) ? min(r - i, a[i - p]) : 0;
|  |  while (i + a[i] < n && s[i + a[i]] == s[a[i]]) ++a[i];
|  |  if (r < i + a[i]) r = i + a[i], p = i; }}
```

3.5 AC 自动机

```
int ch[maxn][26], fail[maxn], q[maxn], sum[maxn], cnt = 0;
int insert(const char *c) { int x = 0; while (*c) {
|  if (!ch[x][*c - 'a']) ch[x][*c - 'a'] = ++cnt;
|  x = ch[x][*c++ - 'a']; } return x; }
void getfail() { int x, head = 0, tail = 0;
|  for (int c = 0; c < 26; c++) if (ch[0][c])
|  |  |  q[tail++] = ch[0][c];
|  while (head != tail) { x = q[head++];
|  |  for (int c = 0; c < 26; c++) { if (ch[x][c]) {
|  |  |  |  fail[ch[x][c]] = ch[fail[x]][c];
|  |  |  |  q[tail++] = ch[x][c];
|  |  |  } else ch[x][c] = ch[fail[x]][c]; } } }
```

3.6 Lydon Word Decomposition

```
//满足s的最小后缀等于s本身的串s称为Lyndon串.
//等价于: s是它自己的所有循环移位中唯一最小的一个.
//任意字符串s可以分解为 $s = s_1s_2s_k$, 其中 $s_i$ 是Lyndon串, $s_i \geq s_{i+1}$. 且这种分解方法是唯一的.
void mnsuf(char *s, int *mn, int n) { // 每个前缀的最小后缀
|  for (int i = 0; i < n; ) {
|  |  int j = i, k = i + 1; mn[i] = i;
|  |  for (; k < n && s[j] <= s[k]; ++ k)
|  |  |  if (s[j] == s[k]) mn[k] = mn[j] + k - j, ++j;
|  |  |  |  else mn[k] = j = i;
|  |  for (; i <= j; i += k - j) {} } } //lyn+=s[i..i+k-j-1]
void mxsuf(char *s, int *mx, int n) { // 每个前缀的最大后缀
|  fill(mx, mx + n, -1);
|  for (int i = 0; i < n; ) {
|  |  int j = i, k = i + 1; if (mx[i] == -1) mx[i] = i;
|  |  for (; k < n && s[j] >= s[k]; ++k) {
|  |  |  j = s[j] == s[k] ? j + 1 : i;
|  |  |  if (mx[k] == -1) mx[k] = i; }
|  |  for (; i <= j; i += k - j) {} } }
```

3.7 后缀数组

```
void Sort(int in[], int out[], int p[], int n, int m) {
|  static int P[N];
|  for (int i = 1; i <= m; i++) P[i] = 0;
|  for (int i = 1; i <= n; i++) P[in[i]]++;
|  for (int i = 2; i <= m; i++) P[i] += P[i - 1];
|  for (int i = n; i; i--) out[P[in[p[i]]]--] = p[i]; }
int n; char s[N]; int sa[N], rk[N], h[N];
void getsa() {
|  static int t1[N], t2[N], *x = t1, *y = t2; //clear n + 1
|  int m = 127;
|  for (int i = 1; i <= n; i++) x[i] = s[i], y[i] = i;
|  Sort(x, sa, y, n, m);
|  for (int j = 1, i, k = 0; k < n; m = k, j <<= 1) {
|  |  for (i = n - j + 1, k = 0; i <= n; i++) y[++k] = i;
|  |  for (i = 1; i <= n; i++)
|  |  |  if (sa[i] > j) y[++k] = sa[i] - j;
|  |  Sort(x, sa, y, n, m);
|  |  for(swap(x, y), i = 2, x[sa[1]] = k = 1; i <= n; i++){
|  |  |  x[sa[i]] = (y[sa[i - 1]] == y[sa[i]] &&
|  |  |  y[sa[i - 1] + j] == y[sa[i] + j]) ? k : ++k; } }
|  for (int i = 1; i <= n; i++) rk[sa[i]] = i;
|  for (int i = 1, k = 0; i <= n; h[rk[i++]] = k) {
|  |  k -= !!k;
|  |  for(int j = sa[rk[i] - 1];s[i + k]==s[j + k];k++);}}

```

3.8 String Conclusions

3.8.1 双回文串

如果 $s = x_1x_2 = y_1y_2 = z_1z_2, |x_1| < |y_1| < |z_1|, x_2, y_1, y_2, z_1$ 是回文串, 则 x_1 和 z_2 也是回文串.

3.8.2 Border 和周期

如果 r 是 S 的一个border, 则 $|S| - r$ 是 S 的一个周期.

如果 p 和 q 都是 S 的周期, 且满足 $p+q \leq |S|+gcd(p,q)$, 则 $gcd(p,q)$ 也是一个周期.

3.8.3 字符串匹配与Border

若字符串 S, T 满足 $2|S| \geq |T|$, 则 S 在 T 中所有匹配位置成一个等差数列.

进一步地, 若 S 的匹配次数大于2, 则等差数列的周期恰好等于 S 的最小周期.

3.8.4 Border 的结构

字符串 S 的所有不小于 $|S|/2$ 的border长度组成一个等差数列.

字符串 S 的所有 border 按长度排序后可分成 $O(\log |S|)$ 段, 每段是一个等差数列.

3.8.5 回文串Border

回文串长度为 t 的后缀是一个回文后缀，等价于 t 是该串的border．因此回文后缀的长度也可以划分成 $O(\log |S|)$ 段.

3.8.6 子串最小后缀

设 $s[p..n]$ 是 $s[i..n]$，$(l \leq i \leq r)$ 中最小者，则minsuf(l, r) 等于 $s[p..r]$ 的最短非空 border．minsuf(l, r) $= \min\{s[p..r]$, minsuf(r $- 2^k + 1$, r)$\}$, $(2^k < rl + 1 \leq 2^{k+1})$.

3.8.7 子串最大后缀

从左往右扫，用set维护后缀的字典序递减的单调队列，并在对应时刻添加”小于事件”点以便在之后修改队列；查询直接在set里lower_bound.

4. Math 数学

4.1 exgcd

```
LL exgcd(LL a, LL b, LL &x, LL &y) {
|  if (b == 0) return x = 1, y = 0, a;
|  LL t = exgcd(b, a % b, y, x);
|  y -= a / b * x; return t;}
LL inv(LL x, LL m) {
|  LL a, b; exgcd(x, m, a, b); return (a % m + m) % m; }

```

4.2 CRT 中国剩余定理

```
bool crt_merge(LL a1, LL m1, LL a2, LL m2, LL &A, LL &M) {
LL c = a2 - a1, d = __gcd(m1, m2); //合并两个模方程
if(c % d) return 0; // gcd(m1, m2) | (a2 - a1) 时才有解
c = (c % m2 + m2) % m2; c /= d; m1 /= d; m2 /= d;
c = c * inv(m1 % m2, m2) % m2; //0逆元可任意值
M = m1*m2*d; A = (c *m1 %M *d %M +a1) % M; return 1;}//有解
```

4.3 扩展卢卡斯

```
int l,a[33],p[33],P[33];
U fac(int k,LL n){// 求 n! mod pk^tk, 返回值 U{ 不包含 pk 的值 ,pk 出现的次数 }
|  if (!n)return U{1,0};LL x=n/p[k],y=n/P[k],ans=1;int i;
|  if(y){// 求出循环节的答案
|  |  for(i=2;i<P[k];i++)if(i%p[k])ans=ans*i%P[k];
|  |  ans=Pw(ans,y,P[k]);
|  }for(i=y*P[k];i<=n;i++) if(i%p[k])ans=ans*i%M;// 求零散部分
|  U z=fac(k,x);return U{ans*z.x%M,x+z.z};
}LL get(int k,LL n,LL m){// 求 C(n,m) mod pk^tk
|  U a=fac(k,n),b=fac(k,m),c=fac(k,n-m);// 分三部分求解
|  return Pw(p[k],a.z-b.z-c.z,P[k])*a.x%P[k]*inv(b.x,P[k])%P[k]*inv(c.x,P[k])%P[k];
}LL CRT(){// CRT 合并答案
|  LL d,w,y,x,ans=0;
|  fr(i,1,l)w=M/P[i],exgcd(w,P[i],x,y),ans=(ans+w*x%M*a[i])%M;
|  return (ans+M)%M;
}LL C(LL n,LL m){// 求 C(n,m)
|  fr(i,1,l)a[i]=get(i,n,m);
|  return CRT();
}LL exLucas(LL n,LL m,int M){
|  int jj=M,i; // 求 C(n,m)mod M,M=prod(pi^ki), 时间O(pi^kilg^2n)
|  for(i=2;i*i<=jj;i++)if(jj%i==0) for(p[++l]=i,P[l]=1;jj%i==0;P[l]*=p[l])jj/=i;
|  if(jj>1)l++,p[l]=P[l]=jj;
|  return C(n,m);}
```

4.4 阶乘取模

```
// n! mod p^q Time : $O(pq^2 \frac{\log^2 n}{\log p})$
// Output : {a, b} means a*p^b
using Val=unsigned long long; //Val 需要 mod p^q 意义下 + *
typedef vector<Val> poly;
poly polymul(const poly &a,const poly &b){
|  int n = (int) a.size(); poly c (n, Val(0));
|  for (int i = 0; i < n; ++ i) {
|  |  for (int j = 0; i + j < n; ++ j) {
|  |  |  c[i + j] = c[i + j] + a[i] * b[j]; } }
|  return c; } Val choo[70][70];
poly polyshift(const poly &a, Val delta) {
|  int n = (int) a.size(); poly res (n, Val(0));
|  for (int i = 0; i < n; ++ i) { Val d = 1;
|  |  for (int j = 0; j <= i; ++ j) {
|  |  |  res[i - j] = res[i - j]+a[i]*choo[i][j]*d;
|  |  |  d = d * delta; } } return res; }
void prepare(int q) {
|  for (int i = 0; i < q; ++ i) { choo[i][0] = Val(1);
|  |  for (int j = 1; j <= i; ++ j)
|  |  |  choo[i][j]=choo[i-1][j-1]+choo[i-1][j]; } }
pair<Val, LL> fact(LL n, LL p, LL q) { Val ans = 1;
|  for (int r = 1; r < p; ++ r) {
|  |  poly x (q, Val(0)), res (q, Val(0));
|  |  res[0] = 1; LL _res = 0; x[0] = r; LL _x = 0;
|  |  if (q > 1) x[1] = p, _x = 1; LL m = (n - r + p) / p;
|  |  while (m) { if (m & 1) {
|  |  |  |  res=polymul(res,polyshift(x,_res)); _res+=_x; }
|  |  |  m >>= 1; x = polymul(x, polyshift(x, _x)); _x+=_x; }
|  |  ans = ans * res[0]; }
|  LL cnt = n / p; if (n >= p) { auto tmp=fact(n / p, p, q);
|  |  ans = ans * tmp.first; cnt += tmp.second; }
|  return {ans, cnt}; }
```

4.5 类欧几里得直线下格点统计

```
// $\sum_{i=0}^{n-1} \lfloor \frac{a+bi}{m} \rfloor,\ n, m, a, b > 0$
ll solve(ll n, ll a, ll b, ll m){
|  if (b == 0) return n * (a / m);
|  if (a >= m) return n * (a / m) + solve(n, a % m, b, m);
|  if (b >= m) return (n-1)*n/2*(b/m) + solve(n,a,b%m,m);
|  return solve((a + b * n) / m, (a + b * n) % m, m, b); }
```

4.6 平方剩余

```
// x^2=a (mod p),0 <=a<p, 返回 true or false 代表是否存在解
// p必须是质数，若是多个单次质数的乘积，可以分别求解再用CRT合并
// 复杂度为 O(log n)
void multiply(ll &c, ll &d, ll a, ll b, ll w) {
|  int cc = (a * c + b * d % MOD * w) % MOD;
|  int dd = (a * d + b * c) % MOD; c = cc, d = dd; }
bool solve(int n, int &x) {
|  if (n==0) return x=0,true; if (MOD==2) return x=1,true;
|  if (power(n, MOD / 2, MOD) == MOD - 1) return false;
|  ll c = 1, d = 0, b = 1, a, w;
|  // finding a such that a^2 - n is not a square
|  do { a = rand() % MOD; w = (a * a - n + MOD) % MOD;
|  |  if (w == 0) return x = a, true;
|  } while (power(w, MOD / 2, MOD) != MOD - 1);
|  for (int times = (MOD + 1) / 2; times; times >>= 1) {
|  |  if (times & 1) multiply(c, d, a, b, w);
|  |  multiply(a, b, a, b, w); }
|  // x = (a + sqrt(w)) ^ ((p + 1) / 2)
|  return x = c, true; }
```

4.7 线性同余不等式

```
// Find the minimal non-negtive solutions for $l \leq d \cdot x \bmod m \leq r$
// $0 \leq d, l, r < m; l \leq r, O(\log n)$
LL cal(LL m, LL d, LL l, LL r) {
|  if (l==0) return 0; if (d==0) return MXL; // 无解
|  if (d * 2 > m) return cal(m, m - d, m - r, m - l);
|  if ((l - 1) / d < r / d) return (l - 1) / d + 1;
|  LL k = cal(d, (-m % d + d) % d, l % d, r % d);
|  return k==MXL ? MXL : (k*m + l - 1)/d+1;}// 无解 2
// return all x satisfying l1<=x<=r1 and l2<=(x*mul+add)%LIM<=r2
// here LIM = 2^32 so we use UI instead of "%".
// $O(\log p + \#solutions)$
struct Jump { UI val, step;
|  Jump(UI val, UI step) : val(val), step(step) { }
|  Jump operator + (const Jump & b) const {
|  |  return Jump(val + b.val, step + b.step); }
|  Jump operator - (const Jump & b) const {
|  |  return Jump(val - b.val, step + b.step); }};
inline Jump operator * (UI x, const Jump & a) {
|  return Jump(x * a.val, x * a.step); }
vector<UI> solve(UI l1, UI r1, UI l2, UI r2, pair<UI,UI> muladd) {
|  UI mul = muladd.first, add = muladd.second, w = r2 - l2;
|  Jump up(mul, 1), dn(-mul, 1); UI s(l1 * mul + add);
|  Jump lo(r2 - s, 0), hi(s - l2, 0);
|  function<void(Jump&, Jump&)> sub=[&](Jump& a, Jump& b){
|  |  if (a.val > w) {
|  |  |  UI t(((LL)a.val-max(0LL, w+1LL-b.val)) / b.val);
|  |  |  a = a - t * b; } };
|  sub(lo, up), sub(hi, dn);
|  while (up.val > w || dn.val > w) {
```

```
|  |  sub(up, dn); sub(lo, up);
|  |  sub(dn, up); sub(hi, dn); }
|  assert(up.val + dn.val > w); vector<UI> res;
|  Jump bg(s + mul * min(lo.step, hi.step), min(lo.step,
   ↪ hi.step));
|  while (bg.step <= r1 - l1) {
|  |  if (l2 <= bg.val && bg.val <= r2)
|  |  |  res.push_back(bg.step + l1);
|  |  if (l2 <= bg.val-dn.val && bg.val-dn.val <= r2) {
|  |  |  bg = bg - dn;
|  |  } else bg = bg + up; }
|  return res; }
```

4.8 Miller Rabin, Pollard Rho

```
// Miller Rabin : bool miller_rabin::solve (const LL &) :
   ↪ tests whether a certain integer is prime.
typedef long long LL; struct miller_rabin {
int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31,
   ↪ 37};
bool check (const LL &prime, const LL &base) {
|  LL number = prime - 1;
|  for (; ~number & 1; number >>= 1);
|  LL result = llfpm (base, number, prime);
|  for (; number != prime - 1 && result != 1 && result !=
   ↪ prime - 1; number <<= 1)
|  |  result = mul_mod (result, result, prime);
|  return result == prime - 1 || (number & 1) == 1; }
bool solve (const LL &number) { // is prime
|  if (number < 2) return false;
|  if (number < 4) return true;
|  if (~number & 1) return false;
|  for (int i = 0; i < 12 && BASE[i] < number; ++i)
|  |  if (!check (number, BASE[i])) return false;
|  return true; } };
miller_rabin is_prime; const LL threshold = 13E9;
LL factorize (LL number, LL seed) {
|  LL x = rand() % (number - 1) + 1, y = x;
|  for (int head = 1, tail = 2; ; ) {
|  |  x = mul_mod (x, x, number); x = (x + seed) % number;
|  |  if (x == y) return number;
|  |  LL answer = gcd (abs (x - y), number);
|  |  if (answer > 1 && answer < number) return answer;
|  |  if (++head == tail) { y = x; tail <<= 1; } } }
void search (LL number, vector<LL> &divisor) {
|  if (number <= 1) return;
|  if (is_prime.solve (number)) divisor.push_back (number);
|  else {
|  |  LL factor = number;
|  |  for (; factor >= number; factor = factorize (number,
   ↪ rand () % (number - 1) + 1));
|  |  search (number / factor, divisor);
|  |  search (factor, divisor); } }
```

4.9 原根

定义 使得 $a^x \bmod m = 1$ 的最小的x, 记作 $\delta_m(a)$. 若 $a \equiv g^s \bmod m$, 其中 g 为 m 的一个原根. 则虽然 s 随 g 的不同取值有所不同, 但是必然满足 $\delta_m(a) = \gcd(s, \varphi(m))$.

性质 $\delta_m(a^k) = \frac{\delta_m(a)}{\gcd(\delta_m(a),k)}$

k 次剩余 给定方程 $x^k \equiv a \mod m$, 求所有解. 若 k 与 $\varphi(m)$ 互质, 则可以直接求出 k 对 $\varphi(m)$ 的逆元. 否则, 将 k 拆成两部分, $k = uv$, 其中 $u \perp \varphi(m)$, $v | \varphi(m)$, 先求 $x^v \equiv a \mod m$, 则 $ans = x^{u^{-1}}$. 下面讨论 $k|\varphi(m)$ 的问题. 任取一原根 g, 对两侧取离散对数, 设 $x = g^s$, $a = g^t$, 其中 t 可以用BSGS求出, 则问题转化为求出所有的 s 满足 $ks \equiv t \mod \varphi(m)$, exgcd 即可求解, 显然有解的条件是 $k|\delta_m(a)$.

4.10 FFT NTT FWT 多项式操作

```
// double 精度对 10^9 + 7 取模最多可以做到 2^20
// FFT(Reverse(FFT(a,N)),N)=Na, Reverse是a_0不变,1到N-1反过
   ↪ 来
const int MOD = 1e9 + 7; const double PI = acos(-1);
typedef complex<double> Complex;
const int MAXN = 262144, L = 15, MASK = (1<<L) - 1;
Complex w[MAXN];
void FFTInit(int N) {
|  for (int i = 0; i < N; ++i)
|  |  w[i] = Complex(cos(2*i*PI/N), sin(2*i*PI/N));}
void FFT(Complex p[], int n) { FFTInit(N);
|  for (int i = 1, j = 0; i < n - 1; ++i) {
|  |  for (int s = n; j ^= s >>= 1, ~j & s;);
|  |  if (i < j) swap(p[i], p[j]); }
|  for (int d = 0; (1 << d) < n; ++d) {
|  |  int m = 1 << d, m2 = m * 2, rm = n >> (d+1);
|  |  for (int i = 0; i < n; i += m2) {
|  |  |  for (int j = 0; j < m; ++j) {
|  |  |  |  Complex &p1 = p[i+j+m], &p2 = p[i+j];
|  |  |  |  Complex t = w[rm * j] * p1;
|  |  |  |  p1 = p2 - t, p2 = p2 + t; } } } }
void FFT_inv(Complex p[],int n) {
|  FFT(p,n); reverse(p + 1, p + n);
|  for (int i = 0; i < n; ++ i) p[i] /= n; }
Complex A[MAXN], B[MAXN], C[MAXN], D[MAXN];
void mul(int *a, int *b, int N) {
|  for (int i = 0; i < N; ++i) {
|  |  A[i] = Complex(a[i] >> L, a[i] & MASK);
|  |  B[i] = Complex(b[i] >> L, b[i] & MASK); }
|  FFT(A, N), FFT(B, N);
|  for (int i = 0; i < N; ++i) { int j = (N - i) % N;
|  |  Complex da = (A[i]-conj(A[j]))*Complex(0, -0.5),
|  |  |  |  db = (A[i]+conj(A[j]))*Complex(0.5, 0),
|  |  |  |  dc = (B[i]-conj(B[j]))*Complex(0, -0.5),
|  |  |  |  dd = (B[i]+conj(B[j]))*Complex(0.5, 0);
|  |  C[j] = da * dd + da * dc * Complex(0, 1);
|  |  D[j] = db * dd + db * dc * Complex(0, 1); }
|  FFT(C, N), FFT(D, N);
|  for (int i = 0; i < N; ++i) {
|  |  LL  da = (LL)(C[i].imag() / N + 0.5) % MOD,
|  |  |  db = (LL)(C[i].real() / N + 0.5) % MOD,
|  |  |  dc = (LL)(D[i].imag() / N + 0.5) % MOD,
|  |  |  dd = (LL)(D[i].real() / N + 0.5) % MOD;
|  |  a[i]=((dd << (L*2)) + ((db+dc) << L) + da) %MOD;}}
// 4179340454199820289LL (4e18) 原根=3 两倍不会爆 LL
// 2013265921 原根=31 两倍平方不会爆 LL
// 998244353 原根=3 // 1004535809 原根=3 // 469762049 原根=3
void NTT(int *a,int n,int f=1){
|  int i,j,k,m,u,v,w,wm;
|  for(i=n>>1,j=1,k;j<n-1;j++){
|  |  if(i>j)swap(a[i],a[j]);
|  |  for(k=n>>1;k<=i;k>>=1)i^=k;i^=k;
|  }for(m=2;m<=n;m<<=1)
|  |  for(i=0,wm=Pw(G,f==1?(M-1)/m:(M-1)/m*(m-1),M);i<n;i+=m)
|  |  |  for(j=i,w=1;j<i+(m>>1);j++){
|  |  |  |  u=a[j],v=1ll*w*a[j+(m>>1)]%M;
|  |  |  |  if((a[j]=u+v)>=M)a[j]-=M;
|  |  |  |  if((a[j+(m>>1)]=u-v)<0) a[j+(m>>1)]+=M;
|  |  |  |  w=1ll*w*wm%M;
|  |  |  }
|  if(f==-1)for(w=Pw(n,M-2,M),i=0;i<n;i+
   ↪ +)a[i]=1ll*a[i]*w%M;
}
void FWT(int w){//w=1为正运算, w=0为逆运算
|  int i,j,k,x,y;
|  for(i=1;i<N;i*=2)for(j=0;j<N;j+=i*2){
|  |  for(k=0;k<i;k++) {
|  |  x=a[j+k],y=a[i+j+k];
|  |  if(w){
|  |  |  //xor a[j+k]=x+y a[i+j+k]=x-y
|  |  |  //and a[j+k]=x+y
|  |  |  //or  a[i+j+k]=x+y
|  |  }else{
|  |  |  //xor a[j+k]=(x+y)/2 a[i+j+k]=(x-y)/2
|  |  |  //and a[j+k]=x-y
|  |  |  //or  a[i+j+k]=y-x
|  |  } } } }
```

4.11 K 进制 FWT

```
// n : power of k, omega[i] : (primitive kth root) ^ i
void fwt(int* a, int k, int type) {
|  static int tmp[K];
|  for (int i = 1; i < n; i *= k)
|  |  for (int j = 0, len = i * k; j < n; j += len)
|  |  |  for (int low = 0; low < i; low++) {
|  |  |  |  for (int t = 0; t < k; t++)
|  |  |  |  |  tmp[t] = a[j + t * i + low];
|  |  |  |  for (int t = 0; t < k; t++){
|  |  |  |  |  int x = j + t * i + low;
|  |  |  |  |  a[x] = 0;
|  |  |  |  |  for (int y = 0; y < k; y++)
|  |  |  |  |  |  a[x] = int(a[x] + 1ll * tmp[y] * omega[(k
   ↪ + type) * t * y % k] % MOD);
|  |  |  |  |  }
|   |   |   }
|   if (type == -1) for (int i = 0, invn = inv(n); i < n; i++) a[i] = int(1ll * a[i] * invn % MOD);
}
```

4.11.1 多项式牛顿法

已知函数 $G(x)$，求多项式 $F(x) \mod x^n$ 满足方程 $G(F(x)) \equiv 0 \mod x^n$.

当 $n = 1$ 时，有 $G(F(x)) \equiv 0 \mod x$，根据实际情况（逆元，二次剩余）求解．假设已经求出了 $G(F_0(x)) \equiv 0 \mod x^n$，考虑扩展到 $\mod x^{2n}$ 下：将 $G(F(x))$ 在 $F_0(x)$ 点泰勒展开，有

$$G(F(x)) = G(F_0(x)) + \frac{G'(F_0(x))}{1!}(F(x) - F_0(x)) + \cdots$$

因为 $F(x)$ 和 $F_0(x)$ 的最后 n 项相同，所以 $(F(x) - F_0(x))^2$ 的最低的非 0 项次数大于 $2n$，经过推导可以得到

$$F(x) \equiv F_0(x) - \frac{G(F_0(x))}{G'(F_0(x))} \mod x^{2n}$$

应用（以下复杂度均为 $O(n\log n)$）：

多项式求逆　给定 $A(x)$，求 $A(x)B(x) \equiv 1 \mod x^n$ ：构造方程 $A(x)B(x) - 1 \equiv 0 \mod x^n$，初始解 $G_{invA}(B(x)) \equiv A[0]^{-1} \mod x$，递推式 $F(x) \equiv 2F_0(x) - A(x)F_0^2(x) \mod x^{2n}$

多项式开方　给定 $A(x)$，求 $B^2(x) \equiv A(x) \mod x^n$ ：初始解 $G_{sqrtA}(B(x)) \equiv \sqrt{A[0]} \mod x$，递推式 $F(x) \equiv \frac{F_0^2(x)+A(x)}{2F_0(x)} \mod x^{2n}$

多项式对数　给定常数项为 1 的 $A(x), B(x) \equiv \ln A(x)$ ：对 x 求导得 $(\ln A(x))' = \frac{A'(x)}{A(x)}$，使用多项式求逆，再积分回去 $\ln A(x) \equiv \int \frac{A'(x)}{A(x)}$

多项式指数　给定常数项为 0 的 $A(x)$，求 $B(x) \equiv e^{A(X)}$ ：初始解 $G_{expA}(B(x)) \equiv 1$，递推式 $F(x) \equiv F_0(x)(1 - \ln F_0(x) + A(x))$

多项式任意幂次　给定 $A(x)$，求 $B(x) \equiv A^k(x), k \in Q$ ：$A^k(x) \equiv e^{k\ln A(x)}$

```
void Inv(int*A,int*B,int n){ //注意数组大小2n
//多项式求逆，$B = A^{-1}$,n需为2的幂次
|   static int C[N];B[0]=Pw(A[0],M-2,M);B[1]=0; //n=1时 $B[0] = A[0]^{-1}$
|   for(int m=2,i;m<=n;m<<=1){//递归转递推
|   |   for(i=0;i<m;i++)C[i]=A[i];
|   |   for(i=m;i<2*m;i++)C[i]=B[i]=0; //在模 $x^m$ 意义下超过m次均为0
|   |   NTT(C,m*2);NTT(B,m*2);
                //$g(x) = g_0(x)(2 - f(x)g_0(x))(\mod x^n)$
|   |   for(i=0;i<m*2;i++)
                B[i]=1ll*B[i]*(2-1ll*B[i]*C[i]%M+M)%M;
|   |   NTT(B,m*2,-1);for(i=m;i<m*2;i++)B[i]=0;}
}
void Sqrt(int*A,int*B,int n){//多项式开根，B=sqrt(A)，n为2的幂次
|   static int D[N],IB[N];
|   B[0]=1;B[1]=0;//n=1时根据题意或二次剩余求解
|   int I2=Pw(2,M-2,M),m,i;
|   for(m=2;m<=n;m<<=1){//递归转递推
|   |   for(i=0;i<m;i++)D[i]=A[i];
|   |   for(i=m;i<2*m;i++)D[i]=B[i]=0;
|   |   NTT(D,m*2);Inv(B,IB,m);NTT(IB,m*2);NTT(B,m*2);
|   |   for(i=0;i<m*2;i++)
                B[i]=(1ll*B[i]*I2+1ll*I2*D[i]%M*IB[i])%M;
|   |   NTT(B,m*2,-1);for(i=m;i<m*2;i++)B[i]=0;
|   }
}
// 多项式除法：给定 $n$ 次多项式 $A(x)$ 和 $m \le n$ 次多项式 $B(x)$，求出 $D(x)$，$R(x)$ 满足 $A(x) = D(x)B(x) + R(x)$，并且 $degD \le n - m$，$degR < m$，复杂度 $O(n\log n)$，常用于线性递推将 $2k$ 项系数拍回 $k$ 项时的优化：本质是将 $2k$ 项的多项式除以 $k$ 项零化多项式得到的余数
void Div(int *a, int n, int *b, int m, int *d, int *r) {
|   // 注意这里 $n$ 和 $m$ 为多项式长度，注意需要4倍空间
|   static int A[MAXN], B[MAXN]; while (!b[m - 1]) m --;
|   int p = 1, t = n - m + 1; while (p < t << 1) p <<= 1;
|   fill(A, A+p, 0); reverse_copy(b, b+m, A); Inv(A, B, p);
|   fill(B+t, B+p, 0); NTT(B, p); reverse_copy(a, a+n, A);
|   fill(A + t, A + p, 0); NTT(A, p);
|   for (int i = 0; i < p; ++i) A[i] = 1LL*A[i]*B[i] % M;
|   NTT(A, p, -1); reverse(A,A+t); copy(A,A+t,d); //lenD<=t
|   for (p = 1; p < n; p <<= 1);
|   fill(A + t, A + p, 0); NTT(A, p); copy(b, b + m, B);
|   fill(B + m, B + p, 0); NTT(B, p);
|   for (int i = 0; i < p; ++i) A[i] = 1LL*A[i]*B[i] % M;
|   NTT(A, p, -1);
|   for (int i = 0; i < m; ++i) r[i] = (a[i]-A[i]+M) % M;
|   fill(r+m, r+p, 0); assert(r[m-1] == 0); } //lenR < m
```

4.11.2 单位根反演

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

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

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

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

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

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

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

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

4.12 Simplex 单纯形

```
// 标准型: maximize $\mathbf{c}^T\mathbf{x}$, subject to $\mathbf{Ax} \le \mathbf{b}$ and $\mathbf{x} \ge \mathbf{0}$
// 对偶型: minimize $\mathbf{b}^T\mathbf{y}$, subject to $\mathbf{A}^T\mathbf{x} \ge \mathbf{c}$ and $\mathbf{y} \ge \mathbf{0}$
const LD eps = 1e-9, INF = 1e9; const int N = 105;
namespace Simplex {
int n, m, id[N], tp[N]; LD a[N][N];
void pivot(int r, int c) {
|   swap(id[r + n], id[c]);
|   LD t = -a[r][c]; a[r][c] = -1;
|   for (int i = 0; i <= n; i++) a[r][i] /= t;
|   for (int i = 0; i <= m; i++) if (a[i][c] && r != i) {
|   |   t = a[i][c]; a[i][c] = 0;
|   |   for (int j = 0; j <= n; j++) a[i][j] += t * a[r][j];}}
bool solve() {
|   for (int i = 1; i <= n; i++) id[i] = i;
|   for ( ; ; ) {
|   |   int i = 0, j = 0; LD w = -eps;
|   |   for (int k = 1; k <= m; k++)
|   |   |   if (a[k][0] < w || (a[k][0] < -eps && rand() & 1))
|   |   |   |   w = a[i = k][0];
|   |   if (!i) break;
|   |   for (int k = 1; k <= n; k++)
|   |   |   if (a[i][k] > eps) {j = k; break;}
|   |   if (!j) { printf("Infeasible"); return 0;}
|   |   pivot(i, j);}
|   for ( ; ; ) {
|   |   int i = 0, j = 0; LD w = eps, t;
|   |   for (int k = 1; k <= n; k++)
|   |   |   if (a[0][k] > w) w = a[0][j = k];
|   |   if (!j) break;
|   |   w = INF;
|   |   for (int k = 1; k <= m; k++)
|   |   |   if (a[k][j] < -eps && (t = -a[k][0] / a[k][j]) < w)
|   |   |   |   w = t, i = k;
|   |   if (!i) { printf("Unbounded"); return 0;}
|   |   pivot(i, j);}
|   return 1;}
LD ans() {return a[0][0];}
void output() {
|   for (int i = n + 1; i <= n + m; i++) tp[id[i]] = i - n;
|   for (int i = 1; i <=n; i++) printf("%.9lf ", tp[i] ? a[tp[i]][0] : 0);}
}using namespace Simplex;
int main() { int K; read(n); read(m); read(K);
for (int i = 1; i <= n; i++) {LD x; scanf("%lf", &x); a[0][i] = x;}
for (int i = 1; i <= m; i++) {LD x;
|   for (int j = 1; j <= n; j++) scanf("%lf", &x), a[i][j] = -x;
|   scanf("%lf", &x); a[i][0] = x;}
if (solve()) { printf("%.9lf\n", (LD)ans()); if (K) output();}}
```

4.13 线性递推

```
// Complexity: init O(n^2log) query O(n^2logk)
// Requirement: const LOG const MOD
// Example: In: {1, 3} {2, 1} an = 2an-1 + an-2
//          Out: calc(3) = 7
```

```
typedef vector<int> poly;
struct LinearRec {
|  int n; poly first, trans; vector<poly> bin;
poly add(poly &a, poly &b) {
|  poly res(n * 2 + 1, 0);
|  // 不要每次新开 vector，可以使用矩阵乘法优化
|  for (int i = 0; i <= n; ++i) {
|  |  for (int j = 0; j <= n; ++j) {
|  |  |  (res[i+j]+=(LL)a[i] * b[j] % MOD) %= MOD;
|  for (int i = 2 * n; i > n; --i) {
|  |  for (int j = 0; j < n; ++j) {
|  |  |  (res[i-1-j]+=(LL)res[i]*trans[j]%MOD) %=MOD;}
|  |  res[i] = 0; }
|  res.erase(res.begin() + n + 1, res.end());
|  return res; }
LinearRec(poly &first, poly &trans): first(first), trans(trans) {
|  n = first.size(); poly a(n + 1, 0); a[1] = 1;
|  bin.push_back(a); for (int i = 1; i < LOG; ++i)
|  |  bin.push_back(add(bin[i - 1], bin[i - 1])); }
int calc(int k) { poly a(n + 1, 0); a[0] = 1;
|  for (int i = 0; i < LOG; ++i)
|  |  if (k >> i & 1) a = add(a, bin[i]);
|  int ret = 0; for (int i = 0; i < n; ++i)
|  |  if ((ret += (LL)a[i + 1] * first[i] % MOD) >= MOD)
|  |  |  ret -= MOD;
|  return ret; }};
```

4.14 Berlekamp-Massey 最小多项式

```
// Complexity: O(n^2) Requirement: const MOD, inverse(int)
// Input: the first elements of the sequence
// Output: the recursive equation of the given sequence
// Example In: {1, 1, 2, 3}
// Example Out: {1, 1000000006, 1000000006} (MOD = 1e9+7)
struct Poly { vector<int> a; Poly() { a.clear(); }
|  Poly(vector<int> &a): a(a) {}
|  int length() const { return a.size(); }
|  Poly move(int d) { vector<int> na(d, 0);
|  |  na.insert(na.end(), a.begin(), a.end());
|  |  return Poly(na); }
|  int calc(vector<int> &d, int pos) { int ret = 0;
|  |  for (int i = 0; i < (int)a.size(); ++i) {
|  |  |  if ((ret+=(LL)d[pos - i]*a[i]%MOD) >= MOD) {
|  |  |  |  ret -= MOD; }}
|  |  return ret; }
|  Poly operator - (const Poly &b) {
|  |  vector<int> na(max(this->length(), b.length()));
|  |  for (int i = 0; i < (int)na.size(); ++i) {
|  |  |  int aa = i < this->length() ? this->a[i] : 0,
|  |  |  |  bb = i < b.length() ? b.a[i] : 0;
|  |  |  na[i] = (aa + MOD - bb) % MOD; }
|  |  return Poly(na); } };
Poly operator * (const int &c, const Poly &p) {
|  vector<int> na(p.length());
|  for (int i = 0; i < (int)na.size(); ++i) {
|  |  na[i] = (LL)c * p.a[i] % MOD; }
|  return na; }
vector<int> solve(vector<int> a) {
|  int n = a.size(); Poly s, b;
|  s.a.push_back(1), b.a.push_back(1);
|  for (int i = 0, j = -1, ld = 1; i < n; ++i) {
|  |  int d = s.calc(a, i); if (d) {
|  |  |  if ((s.length() - 1) * 2 <= i) {
|  |  |  |  Poly ob = b; b = s;
|  |  |  |  s=s-(LL)d*inverse(ld)%MOD*ob.move(i - j);
|  |  |  |  j = i; ld = d;
|  |  |  } else {
|  |  |  |  s=s-(LL)d*inverse(ld)%MOD*b.move(i-j);}}}
|  //Caution: s.a might be shorter than expected
|  return s.a; }
/* 求行列式 -> 求特征多项式 : det(A)=(-1)^nPA(0)
 求矩阵或向量列最小多项式 : 随机投影成数列
 如果最小多项式里面有 x 的因子，那么行列式为 0，否则
 随机乘上对角阵 D, det(A)=det(AD)/det(D)*/
```

4.15 Pell 方程

```
// $x^2 - n * y^2 = 1$ 最小正整数根，n 为完全平方数时无解
// $x_{k+1} = x_0 x_k + n y_0 y_k$
// $y_{k+1} = x_0 y_k + y_0 x_k$
pair<LL, LL> peLL(LL n) {
|  static LL p[N], q[N], g[N], h[N], a[N];
|  p[1] = q[0] = h[1] = 1; p[0] = q[1] = g[1] = 0;
|  a[2] = (LL)(floor(sqrtl(n) + 1e-7L));
|  for(int i = 2; ; i ++) {
|  |  g[i] = -g[i - 1] + a[i] * h[i - 1];
|  |  h[i] = (n - g[i] * g[i]) / h[i - 1];
|  |  a[i + 1] = (g[i] + a[2]) / h[i];
|  |  p[i] = a[i] * p[i - 1] + p[i - 2];
|  |  q[i] = a[i] * q[i - 1] + q[i - 2];
|  |  if(p[i] * p[i] - n * q[i] * q[i] == 1)
|  |  |  return {p[i], q[i]}; }}
```

4.16 解一元三次方程

```
double a(p[3]), b(p[2]), c(p[1]), d(p[0]);
double k(b / a), m(c / a), n(d / a);
double p(-k * k / 3. + m);
double q(2. * k * k * k / 27 - k * m / 3. + n);
Complex omega[3] = {Complex(1, 0), Complex(-0.5, 0.5 * sqrt(3)), Complex(-0.5, -0.5 * sqrt(3))};
Complex r1, r2; double delta(q * q / 4 + p * p * p / 27);
if (delta > 0) {
|  r1 = cubrt(-q / 2. + sqrt(delta));
|  r2 = cubrt(-q / 2. - sqrt(delta));
} else {
|  r1 = pow(-q / 2. + pow(Complex(delta), 0.5), 1. / 3);
|  r2 = pow(-q / 2. - pow(Complex(delta), 0.5), 1. / 3); }
for(int _(0); _ < 3; _++) {
|  Complex x = -k/3. + r1*omega[_] + r2*omega[_* 2 % 3]; }
```

4.17 自适应 Simpson

```
// Adaptive Simpson's method : double simpson::solve (double (*f) (double), double l, double r, double eps) : integrates f over (l, r) with error eps.
struct simpson {
double area (double (*f) (double), double l, double r) {
|  double m = l + (r - l) / 2;
|  return (f (l) + 4 * f (m) + f (r)) * (r - l) / 6;
}
double solve (double (*f) (double), double l, double r, double eps, double a) {
|  double m = l + (r - l) / 2;
|  double left = area (f, l, m), right = area (f, m, r);
|  if (fabs (left + right - a) <= 15 * eps) return left + right + (left + right - a) / 15.0;
|  return solve (f, l, m, eps / 2, left) + solve (f, m, r, eps / 2, right);
}
double solve (double (*f) (double), double l, double r, double eps) {
|  return solve (f, l, r, eps, area (f, l, r));
}};
```

5. Appendix

5.1 Formulas 公式表

5.1.1 Mobius Inversion

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

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

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

5.1.2 Gcd Inversion

$$\begin{aligned}
\sum_{a=1}^{n}\sum_{b=1}^{n} gcd^2(a,b) &= \sum_{d=1}^{n} d^2 \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{d} \rfloor} [gcd(i,j) = 1] \\
&= \sum_{d=1}^{n} d^2 \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{t|gcd(i,j)} \mu(t) \\
&= \sum_{d=1}^{n} d^2 \sum_{t=1}^{\lfloor \frac{n}{d} \rfloor} \mu(t) \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} [t|i] \sum_{j=1}^{\lfloor \frac{n}{d} \rfloor} [t|j] \\
&= \sum_{d=1}^{n} d^2 \sum_{t=1}^{\lfloor \frac{n}{d} \rfloor} \mu(t) \lfloor \frac{n}{dt} \rfloor^2
\end{aligned}$$

The formula can be calculated in $O(nlogn)$ complexity. Moreover, let $l = dt$, then

$$\sum_{d=1}^{n} d^2 \sum_{t=1}^{\lfloor \frac{n}{d} \rfloor} \mu(t) \lfloor \frac{n}{dt} \rfloor^2 = \sum_{l=1}^{n} \lfloor \frac{n}{l} \rfloor^2 \sum_{d|l} d^2 \mu(\frac{l}{d})$$

Let $f(l) = \sum_{d|l} d^2 \mu(\frac{l}{d})$. It can be proven that $f(l)$ is multiplicative. Besides, $f(p^k) = p^{2k} - p^{2k-2}$. Therefore, with linear sieve the formula can be solved in $O(n)$ complexity.

5.1.3 Arithmetic Function

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.1.4 Binomial Coefficients

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

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

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

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

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

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

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

5.1.5 Fibonacci Numbers

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

$$f_n = \frac{\phi^n - \hat{\phi}^n}{\sqrt{5}}, \phi = \frac{1+\sqrt{5}}{2}, \hat{\phi} = \frac{1-\sqrt{5}}{2}$$

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

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

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

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

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

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

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

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

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

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

5.1.6 Lucas Numbers

$$L_0 = 2, L_1 = 1, L_n = L_{n-1} + L_{n-2} = (\frac{1+\sqrt{5}}{2})^n + (\frac{1-\sqrt{5}}{2})^n$$

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

5.1.7 Catlan Numbers

$$c_1 = 1, c_n = \sum_{i=0}^{n-1} c_i c_{n-1-i} = c_{n-1} \frac{4n-2}{n+1} = \frac{\binom{2n}{n}}{n+1} = \binom{2n}{n} - \binom{2n}{n-1}$$

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

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

5.1.8 Stirling Cycle Numbers

把 n 个元素集合分作 k 个非空环方案数.

$$s(n,0) = 0, s(n,n) = 1, s(n+1,k) = s(n,k-1) - nS(n,k)$$

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

$$\begin{bmatrix} n+1 \\ k \end{bmatrix} = n\begin{bmatrix} n \\ k \end{bmatrix} + \begin{bmatrix} n \\ k-1 \end{bmatrix}, \ \begin{bmatrix} n+1 \\ 2 \end{bmatrix} = n!H_n \text{ (see 5.1.12)}$$

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

5.1.9 Stirling Subset Numbers

把 n 个元素集合分作 k 个非空子集方案数.

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

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

$$m!\begin{Bmatrix} n \\ m \end{Bmatrix} = \sum_k \binom{m}{k} k^n (-1)^{m-k}$$

For fixed k, generating functions :

$$\sum_{n=0}^{\infty} \begin{Bmatrix} n \\ k \end{Bmatrix} x^{n-k} = \prod_{r=1}^{k} \frac{1}{1-rx}$$

5.1.10 Motzkin Numbers

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

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

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

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

5.1.11 Eulerian Numbers

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

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

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

5.1.12 Harmonic Numbers

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

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

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

5.1.13 五边形数定理求拆分数

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

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

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

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

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

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

5.1.14 Bell Numbers

n 个元素集合划分的方案数.

$$B_n = \sum_{k=1}^{n} \begin{Bmatrix} n \\ k \end{Bmatrix}, \ B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$$

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

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

5.1.15 Bernoulli Numbers

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

$$G(x) = \sum_{k=0}^{\infty} \frac{B_k}{k!} x^k = \frac{1}{\sum_{k=0}^{\infty} \frac{x^k}{(k+1)!}}$$

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

5.1.16 Sum of Powers

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

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

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

5.1.17 kMAX-MIN反演

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

代入 $k = 1$ 即为MAX-MIN反演:

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

5.1.18 伍德伯里矩阵不等式

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

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

5.1.19 Sum of Squares

$r_k(n)$ 表示用 k 个平方数组成 n 的方案数. 假设:

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

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

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

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

5.1.20 枚举勾股数 Pythagorean Triple

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

5.1.21 四面体体积 Tetrahedron Volume

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

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

5.1.22 杨氏矩阵与钩子公式

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

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

钩子公式: 对于给定形状 λ, 不同杨氏矩阵的个数为:

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

$h_\lambda(i, j)$ 表示该格子右边和上边的格子数量加1.

5.1.23 重心

半径为 r , 圆心角为 θ 的扇形重心与圆心的距离为 $\frac{4r\sin(\theta/2)}{3\theta}$ 半径为 r , 圆心角为 θ 的圆弧重心与圆心的距离为 $\frac{4r\sin^3(\theta/2)}{3(\theta - \sin(\theta))}$

5.1.24 常见游戏

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

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

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

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

约瑟夫环 令 n 个人标号为 $0, 1, 2, ..., n-1$，令 $f_{i,m}$ 表示 i 个人报 m 胜利者的编号，则 $f_{1,m} = 0, f_{i,m} = (f_{i-1,m} + m) \bmod i$.

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

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

5.1.25 最大最小定理求纳什均衡点

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

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

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

5.1.26 错排公式

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

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

5.1.27 概率相关

对于随机变量 X，期望用 $E(X)$ 表示，方差用 $D(X)$ 表示，则 $D(X) = E(X - E(X))^2 = E(X^2) - (E(X))^2, D(X+Y) = D(X) + D(Y)D(aX) = a^2 D(X)$

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

5.1.28 常用泰勒展开

$$f(x) = \frac{f(x_0)}{0!} + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots$$

$$\frac{1}{(1-x)^n} = 1 + \binom{n}{1}x + \binom{n+1}{2}x^2 + \binom{n+2}{3}x^3 + \cdots$$

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^i}{i!}$$

5.1.29 类卡特兰数

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

5.1.30 邻接矩阵行列式的意义

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

5.1.31 Others (某些近似数值公式在这里)

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

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

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

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

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

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

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

$$\max\{x_a - x_b, y_a - y_b, z_a - z_b\} - \min\{x_a - x_b, y_a - y_b, z_a - z_b\}$$

$$= \frac{1}{2} \sum_{cyc} |(x_a - y_a) - (x_b - y_b)|$$

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

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

$n \bmod 2 = 1:$

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

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

5.2 Calculus Table 导数表

$(\frac{u}{v})' = \frac{u'v - uv'}{v^2}$

$(a^x)' = (\ln a)a^x$

$(\tan x)' = \sec^2 x$

$(\cot x)' = \csc^2 x$

$(\sec x)' = \tan x \sec x$

$(\csc x)' = -\cot x \csc x$

$(\arcsin x)' = \frac{1}{\sqrt{1-x^2}}$

$(\arccos x)' = -\frac{1}{\sqrt{1-x^2}}$

$(\arctan x)' = \frac{1}{1+x^2}$

$(\operatorname{arccot} x)' = -\frac{1}{1+x^2}$

$(\operatorname{arccsc} x)' = -\frac{1}{x\sqrt{1-x^2}}$

$(\operatorname{arcsec} x)' = \frac{1}{x\sqrt{1-x^2}}$

$(\tanh x)' = \operatorname{sech}^2 x$

$(\coth x)' = -\operatorname{csch}^2 x$

$(\operatorname{sech} x)' = -\operatorname{sech} x \tanh x$

$(\operatorname{csch} x)' = -\operatorname{csch} x \coth x$

$(\operatorname{arcsinh} x)' = \frac{1}{\sqrt{1+x^2}}$

$(\operatorname{arccosh} x)' = \frac{1}{\sqrt{x^2-1}}$

$(\operatorname{arctanh} x)' = \frac{1}{1-x^2}$

$(\operatorname{arccoth} x)' = \frac{1}{x^2-1}$

$(\operatorname{arccsch} x)' = -\frac{1}{|x|\sqrt{1+x^2}}$

$(\operatorname{arcsech} x)' = -\frac{1}{x\sqrt{1-x^2}}$

5.3 Integration Table 积分表

5.3.1 $ax^2 + bx + c(a > 0)$

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

5.3.2 $\sqrt{\pm ax^2 + bx + c}(a > 0)$

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

5.3.3 $\sqrt{\pm\frac{x-a}{x-b}}$ 或 $\sqrt{(x-a)(x-b)}$

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

5.3.4 三角函数的积分

1. $\int \tan x \mathrm{d}x = -\ln|\cos x| + C$
2. $\int \cot x \mathrm{d}x = \ln|\sin x| + C$
3. $\int \sec x \mathrm{d}x = \ln\left|\tan\left(\frac{\pi}{4} + \frac{x}{2}\right)\right| + C = \ln|\sec x + \tan x| + C$
4. $\int \csc x \mathrm{d}x = \ln\left|\tan\frac{x}{2}\right| + C = \ln|\csc x - \cot x| + C$
5. $\int \sec^2 x \mathrm{d}x = \tan x + C$
6. $\int \csc^2 x \mathrm{d}x = -\cot x + C$
7. $\int \sec x \tan x \mathrm{d}x = \sec x + C$
8. $\int \csc x \cot x \mathrm{d}x = -\csc x + C$
9. $\int \sin^2 x \mathrm{d}x = \frac{x}{2} - \frac{1}{4}\sin 2x + C$

10. $\int \cos^2 x\mathrm{d}x = \frac{x}{2} + \frac{1}{4}\sin 2x + C$

11. $\int \sin^n x\mathrm{d}x = -\frac{1}{n}\sin^{n-1} x\cos x + \frac{n-1}{n}\int \sin^{n-2} x\mathrm{d}x$

12. $\int \cos^n x\mathrm{d}x = \frac{1}{n}\cos^{n-1} x\sin x + \frac{n-1}{n}\int \cos^{n-2} x\mathrm{d}x$

13. $\int \frac{\mathrm{d}x}{\sin^n x} = -\frac{1}{n-1}\frac{\cos x}{\sin^{n-1} x} + \frac{n-2}{n-1}\int \frac{\mathrm{d}x}{\sin^{n-2} x}$

14. $\int \frac{\mathrm{d}x}{\cos^n x} = \frac{1}{n-1}\frac{\sin x}{\cos^{n-1} x} + \frac{n-2}{n-1}\int \frac{\mathrm{d}x}{\cos^{n-2} x}$

15.

$$\begin{aligned}&\int \cos^m x\sin^n x\mathrm{d}x\\ =&\frac{1}{m+n}\cos^{m-1} x\sin^{n+1} x + \frac{m-1}{m+n}\int \cos^{m-2} x\sin^n x\mathrm{d}x\\ =&-\frac{1}{m+n}\cos^{m+1} x\sin^{n-1} x + \frac{n-1}{m+1}\int \cos^m x\sin^{n-2} x\mathrm{d}x\end{aligned}$$

16. $\int \frac{\mathrm{d}x}{a+b\sin x} = \begin{cases}\frac{2}{\sqrt{a^2-b^2}}\arctan\frac{a\tan\frac{x}{2}+b}{\sqrt{a^2-b^2}} + C & (a^2 > b^2)\\ \frac{1}{\sqrt{b^2-a^2}}\ln\left|\frac{a\tan\frac{x}{2}+b-\sqrt{b^2-a^2}}{a\tan\frac{x}{2}+b+\sqrt{b^2-a^2}}\right| + C & (a^2 < b^2)\end{cases}$

17. $\int \frac{\mathrm{d}x}{a+b\cos x} = \begin{cases}\frac{2}{a+b}\sqrt{\frac{a+b}{a-b}}\arctan\left(\sqrt{\frac{a-b}{a+b}}\tan\frac{x}{2}\right) + C & (a^2 > b^2)\\ \frac{1}{a+b}\sqrt{\frac{a+b}{a-b}}\ln\left|\frac{\tan\frac{x}{2}+\sqrt{\frac{a+b}{b-a}}}{\tan\frac{x}{2}-\sqrt{\frac{a+b}{b-a}}}\right| + C & (a^2 < b^2)\end{cases}$

18. $\int \frac{\mathrm{d}x}{a^2\cos^2 x+b^2\sin^2 x} = \frac{1}{ab}\arctan\left(\frac{b}{a}\tan x\right) + C$

19. $\int \frac{\mathrm{d}x}{a^2\cos^2 x-b^2\sin^2 x} = \frac{1}{2ab}\ln\left|\frac{b\tan x+a}{b\tan x-a}\right| + C$

20. $\int x\sin ax\mathrm{d}x = \frac{1}{a^2}\sin ax - \frac{1}{a}x\cos ax + C$

21. $\int x^2\sin ax\mathrm{d}x = -\frac{1}{a}x^2\cos ax + \frac{2}{a^2}x\sin ax + \frac{2}{a^3}\cos ax + C$

22. $\int x\cos ax\mathrm{d}x = \frac{1}{a^2}\cos ax + \frac{1}{a}x\sin ax + C$

23. $\int x^2\cos ax\mathrm{d}x = \frac{1}{a}x^2\sin ax + \frac{2}{a^2}x\cos ax - \frac{2}{a^3}\sin ax + C$

5.3.5 反三角函数的积分（其中 $a > 0$ ）

1. $\int \arcsin\frac{x}{a}\mathrm{d}x = x\arcsin\frac{x}{a} + \sqrt{a^2-x^2} + C$

2. $\int x\arcsin\frac{x}{a}\mathrm{d}x = (\frac{x^2}{2} - \frac{a^2}{4})\arcsin\frac{x}{a} + \frac{x}{4}\sqrt{x^2-x^2} + C$

3. $\int x^2\arcsin\frac{x}{a}\mathrm{d}x = \frac{x^3}{3}\arcsin\frac{x}{a} + \frac{1}{9}(x^2+2a^2)\sqrt{a^2-x^2} + C$

4. $\int \arccos\frac{x}{a}\mathrm{d}x = x\ arccos\frac{x}{a} - \sqrt{a^2-x^2} + C$

5. $\int x\arccos\frac{x}{a}\mathrm{d}x = (\frac{x^2}{2} - \frac{a^2}{4})\arccos\frac{x}{a} - \frac{x}{4}\sqrt{a^2-x^2} + C$

6. $\int x^2\arccos\frac{x}{a}\mathrm{d}x = \frac{x^3}{3}\arccos\frac{x}{a} - \frac{1}{9}(x^2+2a^2)\sqrt{a^2-x^2} + C$

7. $\int \arctan\frac{x}{a}\mathrm{d}x = x\arctan\frac{x}{a} - \frac{a}{2}\ln(a^2+x^2) + C$

8. $\int x\arctan\frac{x}{a}\mathrm{d}x = \frac{1}{2}(a^2+x^2)\arctan\frac{x}{a} - \frac{a}{2}x + C$

9. $\int x^2\arctan\frac{x}{a}\mathrm{d}x = \frac{x^3}{3}\arctan\frac{x}{a} - \frac{a}{6}x^2 + \frac{a^3}{6}\ln(a^2+x^2) + C$

5.3.6 指数函数的积分

1. $\int a^x\mathrm{d}x = \frac{1}{\ln a}a^x + C$

2. $\int \mathrm{e}^{ax}\mathrm{d}x = \frac{1}{a}a^{ax} + C$

3. $\int x\mathrm{e}^{ax}\mathrm{d}x = \frac{1}{a^2}(ax-1)a^{ax} + C$

4. $\int x^n\mathrm{e}^{ax}\mathrm{d}x = \frac{1}{a}x^n\mathrm{e}^{ax} - \frac{n}{a}\int x^{n-1}\mathrm{e}^{ax}\mathrm{d}x$

5. $\int xa^x\mathrm{d}x = \frac{x}{\ln a}a^x - \frac{1}{(\ln a)^2}a^x + C$

6. $\int x^n a^x\mathrm{d}x = \frac{1}{\ln a}x^n a^x - \frac{n}{\ln a}\int x^{n-1}a^x\mathrm{d}x$

7. $\int \mathrm{e}^{ax}\sin bx\mathrm{d}x = \frac{1}{a^2+b^2}\mathrm{e}^{ax}(a\sin bx - b\cos bx) + C$

8. $\int \mathrm{e}^{ax}\cos bx\mathrm{d}x = \frac{1}{a^2+b^2}\mathrm{e}^{ax}(b\sin bx + a\cos bx) + C$

9. $\int \mathrm{e}^{ax}\sin^n bx\mathrm{d}x = \frac{1}{a^2+b^2n^2}\mathrm{e}^{ax}\sin^{n-1} bx(a\sin bx - nb\cos bx) + \frac{n(n-1)b^2}{a^2+b^2n^2}\int \mathrm{e}^{ax}\sin^{n-2} bx\mathrm{d}x$

10. $\int \mathrm{e}^{ax}\cos^n bx\mathrm{d}x = \frac{1}{a^2+b^2n^2}\mathrm{e}^{ax}\cos^{n-1} bx(a\cos bx + nb\sin bx) + \frac{n(n-1)b^2}{a^2+b^2n^2}\int \mathrm{e}^{ax}\cos^{n-2} bx\mathrm{d}x$

5.3.7 对数函数的积分

1. $\int \ln x\mathrm{d}x = x\ln x - x + C$

2. $\int \frac{\mathrm{d}x}{x\ln x} = \ln\left|\ln x\right| + C$

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

4. $\int (\ln x)^n\mathrm{d}x = x(\ln x)^n - n\int (\ln x)^{n-1}\mathrm{d}x$

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

5.4 Java Template

```java
import java.io.*; import java.util.*; import java.math.*;
public class Main {
static class solver { public void run(Scanner cin,
  ↪PrintStream out) {} }
public static void main(String[] args) {
// Fast Reader & Big Numbers
InputReader in = new InputReader(System.in);
PrintWriter out = new PrintWriter(System.out);
BigInteger c = in.nextInt();
out.println(c.toString(8)); out.close(); // as Oct
BigDecimal d = new BigDecimal(10.0);
// d=d.divide(num, length, BigDecimal.ROUND_HALF_UP)
d.setScale(2, BigDecimal.ROUND_FLOOR); // 用于输出
System.out.printf("%.6f\n", 1.23); // C 格式
BigInteger num = BigInteger.valueOf(6);
num.isProbablePrime(10); // 1 - 1 / 2 ^ certainty
BigInteger rev = num.modPow(BigInteger.valueOf(-1),
  ↪BigInteger.valueOf(25));  // rev=6^-1mod25 要互质
num = num.nextProbablePrime(); num.intValue();
Scanner cin=new Scanner(System.in);//SimpleReader
int n = cin.nextInt();
int [] a = new int [n]; // 初值未定义
// Random rand.nextInt(N) [0,N)
Arrays.sort(a, 5, 10, cmp); // sort(a+5, a+10)
ArrayList<Long> list = new ArrayList(); // vector
// .add(val) .add(pos, val) .remove(pos)
Comparator cmp=new Comparator<Long>(){ // 自定义逆序
  @Override public int compare(Long o1, Long o2) {
  /* o1 < o2 ? 1 :( o1 > o2  ? -1 : 0) */ } };
// Collections. shuffle(list) sort(list, cmp)
Long [] tmp = list.toArray(new Long [0]);
// list.get(pos) list.size()
Map<Integer,String> m = new HashMap<Integer,String>();
//m.put(key,val) get(key) containsKey(key) remove(key)
for (Map.Entry<Integer,String> entry:m.entrySet()); //
  ↪entry.getKey() getValue()
Set<String> s = new HashSet<String>(); // TreeSet
//s.add(val)contains(val)remove(val);for(var : s)
solver Task=new solver();Task.run(cin,System.out);
PriorityQueue<Integer> Q=new PriorityQueue<Integer>();
// Q. offer(val) poll() peek() size()
// Read / Write a file : FileWriter FileReader PrintStream
} static class InputReader { // Fast Reader
public BufferedReader reader;
public StringTokenizer tokenizer;
public InputReader(InputStream stream) {
|  reader = new BufferedReader(new
  ↪InputStreamReader(stream), 32768);
|  tokenizer = null; }
public String next() {
|  while (tokenizer==null||!tokenizer.hasMoreTokens()) {
|  |  try { String line = reader.readLine();
|  |  |  /*line == null ? end of file*/
|  |  |  tokenizer = new StringTokenizer(line);
|  |  } catch (IOException e) {
|  |  |  throw new RuntimeException(e); }
|  } return tokenizer.nextToken(); }
public BigInteger nextInt() {
|  //return Long.parseLong(next()); Double Integer
|  return new BigInteger(next(), 16); // as Hex
} } }
```

5.5 Python Hint

```python
def IO_and_Exceptions():
|  try:
|  |  with open("a.in", mode="r") as fin:
|  |  |  for line in fin:
|  |  |  |  a = list(map(int, line.split()))
|  |  |  |  print(a, end = "\n")
|  except:
|  |  exit()
|  assert False, '17 cards can\'t kill me'
def Random():
|  import random as rand
|  rand.normalvariate(0.5, 0.1)
|  l = [str(i) for i in range(9)]
|  sorted(l), min(l), max(l), len(l)
|  rand.shuffle(l)
|  l.sort(key=lambda x:x ^ 1,reverse=True)
|  import functools as ft
|  l.sort(key=ft.cmp_to_key(lambda x, y:(y^1)-(x^1)))
def FractionOperation():
|  from fractions import Fraction
|  a = Fraction(0.233).limit_denominator()
|  a == Fraction("0.233") #True
|  a.numerator, a.denominator, str(a)
def DecimalOperation():
|  import decimal
|  from decimal import Decimal, getcontext
|  getcontext().prec = 100
|  getcontext().rounding = getattr(decimal,
     ↪ 'ROUND_HALF_EVEN')
|  # default; other: FLOOR, CELILING, DOWN, ...
|  getcontext().traps[decimal.FloatOperation] = True
|  Decimal((0, (1, 4, 1, 4), -3)) # 1.414
|  a = Decimal(1<<31) / Decimal(100000)
|  print(round(a, 5)) # total digits
|  print(a.quantize(Decimal("0.00000")))
|  # 21474.83648
|  print(a.sqrt(), a.ln(), a.log10(), a.exp(), a ** 2)
def Complex():
|  a = 1-2j
|  print(a.real, a.imag, a.conjugate())
def FastIO():
|  import atexit
|  import io
|  import sys
|  _INPUT_LINES = sys.stdin.read().splitlines()
|  input = iter(_INPUT_LINES).__next__
|  _OUTPUT_BUFFER = io.StringIO()
|  sys.stdout = _OUTPUT_BUFFER
|  @atexit.register
|  def write():
|  |  sys.__stdout__.write(_OUTPUT_BUFFER.getvalue())
```

6. Miscellany

6.1 Zeller 日期公式

```
// weekday=(id+1)%7;{Sun=0,Mon=1,...}   getId(1, 1, 1) = 0
int getId(int y, int m, int d) {
|  if (m < 3) { y --; m += 12; }
|  return 365 * y + y / 4 - y / 100 + y / 400 + (153 * (m -
  ↪ 3) + 2) / 5 + d - 307; }
//  y<0: 统一加400的倍数年
auto date(int id) {
|  int x=id+1789995, n, i, j, y, m, d;
|  n = 4 * x / 146097; x -= (146097 * n + 3) / 4;
|  i = (4000 * (x + 1)) / 1461001; x -= 1461 * i / 4 - 31;
|  j = 80 * x / 2447; d = x - 2447 * j / 80; x = j / 11;
|  m = j + 2 - 12 * x; y = 100 * (n - 49) + i + x;
|  return make_tuple(y, m, d); }
```

6.2 DP 优化

6.2.1 四边形不等式

```
// a ≤ b ≤ c ≤ d : w(b,c) ≤ w(a,d),
  ↪ w(a,c) + w(b,d) ≤ w(a,d) + w(b,c)
for (int len = 2; len <= n; ++len) {
|  for (int l = 1, r = len; r <= n; ++l, ++r) {
|  |  f[l][r] = INF;
|  |  for (int k = m[l][r - 1]; k <= m[l + 1][r]; ++k) {
|  |  |  if (f[l][r] > f[l][k] + f[k + 1][r] + w(l, r)) {
|  |  |  |  f[l][r] = f[l][k] + f[k + 1][r] + w(l, r);
|  |  |  |  m[l][r] = k;
|  } } }
```

6.2.2 一类树形背包优化

一类树形背包限制：若父节点不取，则子节点必不取，也即最后必须取与根节点相连的一个连通块.
转化：考虑此树的任意DFS序，一个点的子树对应于DFS序中的一个区间. 则每个点的决策为，取该点，或者舍弃该点对应的区间. 从后往前dp，设$f(i,v)$ 表示从后往前考虑到i号点，总体积为V的最优价值，设i号点对应的区间为 $[i, i+size_i-1]$，转移为 $f(i,v) = max\{f(i+1, V-v_i)+w_i, f(i+size_i, v)\}$.
如果要求任意连通块，则点分治后转为指定根的连通块问题即可.

6.3 Long Long $O(1)$ 乘

```
// kactl,   M ≤ 7.2 · 10^18
ULL modmul(ULL a, ULL b, LL M) {
|  LL ret = a * b - M * ULL(1.L / M * a * b);
|  return ret + M * (ret < 0) - M * (ret >= (LL)M); }
// orz@CF, M in 63 bit
ULL modmul(ULL a, ULL b, LL M) {
|  ULL c = (long double)a * b / M;
|  LL ret = LL(x * y - c * M) % M; // must be signed
|  return ret < 0 ? ret + M : ret;}
// use int128 instead if c > 63 bit
```

6.4 Hash Table

```
template <class T,int P = 314159/
  ↪ *,451411,1141109,2119969*/>
struct hashmap {
ULL id[P]; T val[P];
int rec[P]; // del: no many clears
hashmap() {memset(id, -1, sizeof id);}
T get(const ULL &x) const {
|  for (int i = int(x % P), j = 1; ~id[i]; i = (i + j) % P,
  ↪ j = (j + 2) % P /*unroll if needed*/) {
|  |  if (id[i] == x) return val[i]; }
|  return 0; }
T& operator [] (const ULL &x) {
|  for (int i = int(x % P), j = 1;       ; i = (i + j) % P,
  ↪ j = (j + 2) % P) {
|  |  if (id[i] == x) return val[i];
|  |  else if (id[i] == -1llu) {
|  |  |  id[i] = x;
|  |  |  rec[++rec[0]] = i;  // del: no many clears
|  |  |  return val[i]; } } }
void clear() { // del
|  while(rec[0]) id[rec[rec[0]]] = -1, val[rec[rec[0]]] =
  ↪ 0, --rec[0]; }
void fullclear() {
|  memset(id, -1, sizeof id); rec[0] = 0; } };

```

6.5 基数排序

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

6.6 O3 与读入优化

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

6.7 Vimrc, Bashrc

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

```
export CXXFLAGS='-g -Wall -Wextra -Wconversion -Wshadow -std=c++14 -fsanitize=undefined -fsanitize=address'
```

6.8 I used to roll the dice

```
1e3 967 971 991 1013 1019 1031 1033 1039 1049 1051
1e5 97651 99689 99761 100501 101009 102059 102079 102191
    103177 105319
5e5 487247 492511 494191 503599 505511 512849 521471 523021
    526829 529241
1e6 999007 1002073 1006463 1009037 1030021 1032341 1033037
    1042759 1044779 1051151
2e6 1920917 1927249 1950367 1963469 1988023 1990243 2012849
    2062759 2063731 2111729
1e7 9657773 9897931 10018753 10080751 10235507 10306243
    10326529 10371029 10442293 10571299
1e9 972797339 976378307 982409653 998074327 1002294037
    1004886599 1013610911 1016251069 1029340099 1049855029
```

6.9 Constant Table 常数表

n	log10 n	n!	C(n,n/2)	LCM(1...n)
2	0.30	2	2	2
3	0.48	6	3	6
4	0.60	24	6	12
5	0.70	120	10	60
6	0.78	720	20	60
7	0.85	5040	35	420
8	0.90	40320	70	840
9	0.95	362880	126	2520
10	1	3628800	252	2520
11		39916800	462	27720
12		479001600	924	27720
15			6435	360360
20			184756	232792560
25			5200300	
30			155117520	

Java Example

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

// Byte Camp 2021 Final : F
// Calulate a, b ~ Polygon, E[|a.x - b.x| + |a.y - b.y|]
public class Main {
    static class point {
        BigDecimal x, y;
        point () {}
        point (int xx, int yy) {x = new BigDecimal(xx); y = new BigDecimal(yy);}
        point (BigDecimal xx, BigDecimal yy) {x = xx; y = yy;}

        point add(point a) {
            return new point(x.add(a.x), y.add(a.y));
        }
        point sub(point a) {
            return new point(x.subtract(a.x), y.subtract(a.y));
        }
    }

    static BigDecimal det(point a, point b) {return a.x.multiply(b.y).subtract(b.x.multiply(a.y));}

    static BigDecimal ans, S, C;
    static MathContext fuckJava = new MathContext(100, RoundingMode.HALF_EVEN);

    static BigDecimal getlen(point a1, point a2, point b1, point b2, BigDecimal x) {
        BigDecimal k1 = a2.y.subtract(a1.y).divide(a2.x.subtract(a1.x), fuckJava);
        BigDecimal k2 = b2.y.subtract(b1.y).divide(b2.x.subtract(b1.x), fuckJava);
        BigDecimal p1 = x.subtract(a1.x).multiply(k1).add(a1.y), p2 = x.subtract(b1.x).multiply(k2).add(b1.y);
        return p2.subtract(p1).abs();
    }
    /* ... */

    static int n;
    static point[] a = new point[31111];
    static List<point> p = new ArrayList<>();

    static void work() {
        BigDecimal minx = BigDecimal.TEN.pow(18);
        int id = 0;
        for (int i = 0; i < n; i++) {
            if (a[i].x.compareTo(minx) < 0) {
                minx = a[i].x;
                id = i;
            }
        }
        p.clear();
        for (int i = id; i < n; i++) p.add(a[i]);
        for (int i = 0; i < id; i++) p.add(a[i]);
        p.add(p.get(0));

        C = BigDecimal.ZERO;
        int l = 1, r = n - 1;
        for ( ; ; ) {
            solve(p.get(l - 1), p.get(l), p.get(r + 1), p.get(r));
            if (l == r) break;
            if (p.get(l).x.compareTo(p.get(r).x) < 0) {
                l++;
            } else {
                r--;
            } } }
    static Scanner scanner;
    public static void main(String[] args) {
        scanner = new Scanner(new BufferedInputStream(System.in));
        n = scanner.nextInt();
        ans = BigDecimal.ZERO;
        for (int i = 0; i < n; i++) {
            int x = scanner.nextInt();
            a[i] = new point();
            a[i].x = new BigDecimal(x);
        }
        for (int i = 0; i < n; i++) {
            int x = scanner.nextInt();
            a[i].y = new BigDecimal(x);
        }
        S = BigDecimal.ZERO;
        for (int i = 1; i < n - 1; i++) {
            S = S.add(det(a[i].sub(a[0]), a[i + 1].sub(a[0])));
        }
        S = S.divide(BigDecimal.valueOf(2), fuckJava);
        work();
        for (int i = 0; i < n; i++) {
            BigDecimal t = a[i].x;
            a[i].x = a[i].y;
            a[i].y = t;
        }
        work();
        ans = ans.divide(S.multiply(S), fuckJava).multiply(new BigDecimal(2));

        System.out.println(new DecimalFormat("0.0000000000000").format(ans))

//        ans = ans.setScale(20, RoundingMode.HALF_EVEN);
//        System.out.println(ans.stripTrailingZeros().toPlainString());

//        System.out.printf("%.9f\n", ans.doubleValue());
    }
}


// Closest Pair of Points

public class Main {

    public static final int N = 111111;
    public static final long INF = (1l << 63) - 1;

    static class point implements Comparable<point> {
        long x, y;
        public point () {}
        public point (long xx, long yy) {x = xx; y = yy;}
        public int compareTo(point b) {
            return y > b.y ? 1 : (y < b.y ? -1 : 0);
        }
    }
    static point[] p;

    static long sqr(long x) {
        return x * x;
    }
    static long solve(int l, int r) {
        if (l + 1 >= r) return INF;
        int m = (l + r) / 2;
        long mx = p[m].x;
        List <point> v = new ArrayList<>();
        long ret = Long.min(solve(l, m), solve(m, r));
        for (int i = l; i < r; i++) {
            if (sqr(p[i].x - mx) < ret) v.add(p[i]);
        }
        Collections.sort(v);
        for (int i = 0; i < v.size(); i++) {
            for (int j = i + 1; j < v.size(); j++) {
                if (sqr(v.get(i).y - v.get(j).y) > ret) break;
                ret = Long.min(ret, sqr(v.get(i).y - v.get(j).y) + sqr(v.get(i).x - v.get(j).x));
            }
        }
        v.clear();
        return ret;
    }

    static class InputReader { // Fast Reader
        public BufferedReader reader;
        public StringTokenizer tokenizer;
    public InputReader(InputStream stream) {
      reader = new BufferedReader(new InputStreamReader(stream), 32768);
      tokenizer = null; }
    public String next() {
      while (tokenizer==null||!tokenizer.hasMoreTokens()) {
        try { String line = reader.readLine();
          tokenizer = new StringTokenizer(line);
        } catch (IOException e) {
          throw new RuntimeException(e); }
      } return tokenizer.nextToken(); }
    public int nextInt() {
      return Integer.valueOf(next());
    }
  }

  public static void main(String[] args) {
    InputReader in = new InputReader(System.in);
    int n = in.nextInt();
    p = new point[n];
    for (int i = 0; i < n; i++) {
      int x = in.nextInt();
      int y = in.nextInt();
      p[i] = new point(x, y);|
    }
    Arrays.sort(p, new Comparator <point>() {
      public int compare(point a, point b) { return a.x > b.x ? 1 : (a.x < b.x ? -1 : 0); }
    });
    System.out.printf("%.4f\n", Math.sqrt(solve(0, n)));
  }
}
// Yinchuan 2019, I. Base62
public class Main {
  public static void main(String[] args) {
    Scanner in = new Scanner(new BufferedInputStream(System.in));
    int x = in.nextInt();
    int y = in.nextInt();
    String z = in.next();
    int n = z.length();
    BigInteger val = BigInteger.ZERO;
    for (int i = 0; i < n; i++) {
      char ch = z.charAt(i);
      int v;
      if (ch >= '0' && ch <= '9') v = ch - '0';
      else if (ch >= 'A' && ch <= 'Z') v = ch - 'A' + 10;
      else v = ch - 'a' + 36;
      val = val.multiply(BigInteger.valueOf(x)).add(BigInteger.valueOf(v));
    }
    StringBuilder ans = new StringBuilder();
    while (val.compareTo(BigInteger.ZERO) > 0) {
      int last = val.mod(BigInteger.valueOf(y)).intValue();
      if (last < 10) ans.append((char)('0' + last));
      else if (last < 36) ans.append((char)('A' + last - 10));
      else ans.append((char)('a' + last - 36));
      val = val.divide(BigInteger.valueOf(y));
    }
    String s = ans.reverse().toString();
    System.out.println(s.length() == 0 ? "0" : s);
  }
}

```

Python Example

```python
import math
import sys
def inp():
  while True:
    _INPUT = input().split() # sys.stdin.read().split()
    for j in _INPUT:
      yield j
read = inp().__next__
a = []
def solve(l, r):
  if l + 1 >= r:
    return 2 ** 64
  m = (l + r) // 2
  mx = a[m][0]
  v = []
  ret = min(solve(l, m), solve(m, r))
  for i in range(l, r):
    if (a[i][0] - mx) ** 2 < ret:
      v.append(a[i])
  v.sort(key=lambda x : x[1])
  for i in range(len(v)):
    for j in range(i + 1, len(v)):
      if (v[i][1] - v[j][1]) ** 2 > ret:
        break
      ret = min(ret, (v[i][0] - v[j][0]) ** 2 + (v[i][1] - v[j][1]) ** 2)
  return ret

n = int(read())
for i in range(n):
  x = int(read())
  y = int(read())
  a.append((x, y))
a.sort(key=lambda x : x[0])
print("%.4f" % math.sqrt(solve(0, n)))
```

Blossom

```cpp
#define DIST(e) (lab[e.u]+lab[e.v]-g[e.u][e.v].w*2)
struct Edge{ int u,v,w; } g[N][N];
int n,m,n_x,lab[N],match[N],slack[N],
st[N],pa[N],fl_from[N][N],S[N],vis[N];
vector<int> fl[N];
deque<int> q;
void update_slack(int u,int x){
  if(!slack[x]||DIST(g[u][x])<DIST(g[slack[x]][x])) slack[x]=u; }
void set_slack(int x){
  slack[x]=0;
  for(int u=1; u<=n; ++u)
    if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)update_slack(u,x);
}
void q_push(int x){
  if(x<=n)return q.push_back(x);
  for(int i=0; i<fl[x].size(); i++)q_push(fl[x][i]);
}
void set_st(int x,int b){
  st[x]=b;
  if(x<=n)return;
  for(int i=0; i<fl[x].size(); ++i)set_st(fl[x][i],b);
}
int get_pr(int b,int xr){
  int pr=find(fl[b].begin(),fl[b].end(),xr)-fl[b].begin();
  if(pr%2==1){
    reverse(fl[b].begin()+1,fl[b].end());
    return (int)fl[b].size()-pr;
  }
  else return pr;
}
void set_match(int u,int v){
  match[u]=g[u][v].v;
  if(u<=n)return;
  Edge e=g[u][v];
  int xr=fl_from[u][e.u],pr=get_pr(u,xr);
  for(int i=0; i<pr; ++i)set_match(fl[u][i],fl[u][i^1]);
  set_match(xr,v);
  rotate(fl[u].begin(),fl[u].begin()+pr,fl[u].end());
}
void augment(int u,int v){
  int xnv=st[match[u]];
  set_match(u,v);
  if(!xnv)return;
  set_match(xnv,st[pa[xnv]]);
  augment(st[pa[xnv]],xnv);
}
int get_lca(int u,int v){
  static int t=0;
  for(++t; u||v; swap(u,v)){
    if(u==0)continue;
    if(vis[u]==t)return u;
    vis[u]=t;
    u=st[match[u]];
    if(u)u=st[pa[u]];
  }
  return 0;
}
void add_blossom(int u,int lca,int v){
  int b=n+1;
  while(b<=n_x&&st[b])++b;
  if(b>n_x)++n_x;
  lab[b]=0,S[b]=0;
  match[b]=match[lca];
  fl[b].clear();
  fl[b].push_back(lca);
  for(int x=u,y; x!=lca; x=st[pa[y]])
    fl[b].push_back(x),
    fl[b].push_back(y=st[match[x]]),q_push(y);
  reverse(fl[b].begin()+1,fl[b].end());
  for(int x=v,y; x!=lca; x=st[pa[y]])
    fl[b].push_back(x),
    fl[b].push_back(y=st[match[x]]),q_push(y);
  set_st(b,b);
  for(int x=1; x<=n_x; ++x)g[b][x].w=g[x][b].w=0;
  for(int x=1; x<=n; ++x)fl_from[b][x]=0;
  for(int i=0; i<fl[b].size(); ++i){
    int xs=fl[b][i];
    for(int x=1; x<=n_x; ++x)
      if(g[b][x].w==0||DIST(g[xs][x])<DIST(g[b][x]))
        g[b][x]=g[xs][x],g[x][b]=g[x][xs];
    for(int x=1; x<=n; ++x)
      if(fl_from[xs][x])fl_from[b][x]=xs;
  }
  set_slack(b);
}
void expand_blossom(int b){
  for(int i=0; i<fl[b].size(); ++i)
    set_st(fl[b][i],fl[b][i]);
  int xr=fl_from[b][g[b][pa[b]].u],pr=get_pr(b,xr);
  for(int i=0; i<pr; i+=2){
    int xs=fl[b][i],xns=fl[b][i+1];
    pa[xs]=g[xns][xs].u;
    S[xs]=1,S[xns]=0;
    slack[xs]=0,set_slack(xns);
    q_push(xns);
  }
  S[xr]=1,pa[xr]=pa[b];
  for(int i=pr+1; i<fl[b].size(); ++i){
    int xs=fl[b][i];
    S[xs]=-1,set_slack(xs);
  }
  st[b]=0;
}
bool on_found_Edge(const Edge &e){
  int u=st[e.u],v=st[e.v];
  if(S[v]==-1){
    pa[v]=e.u,S[v]=1;
    int nu=st[match[v]];
    slack[v]=slack[nu]=0;
    S[nu]=0,q_push(nu);
  }
  else if(S[v]==0){
    int lca=get_lca(u,v);
    if(!lca)return augment(u,v),augment(v,u),1;
    else add_blossom(u,lca,v);
  }
  return 0;
}
bool matching(){
  fill(S,S+n_x+1,-1),fill(slack,slack+n_x+1,0);
  q.clear();
  for(int x=1; x<=n_x; ++x)
    if(st[x]==x&&!match[x])pa[x]=0,S[x]=0,q_push(x);
  if(q.empty())return 0;
  for(;;){
    while(q.size()){
      int u=q.front();
      q.pop_front();
      if(S[st[u]]==1)continue;
      for(int v=1; v<=n; ++v)
        if(g[u][v].w>0&&st[u]!=st[v]){
          if(DIST(g[u][v])==0){
            if(on_found_Edge(g[u][v]))return 1;
          }
          else update_slack(u,st[v]);
        }
    }
    int d=INF;
    for(int b=n+1; b<=n_x; ++b)
      if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2);
    for(int x=1; x<=n_x; ++x)
      if(st[x]==x&&slack[x]){
        if(S[x]==-1)d=min(d,DIST(g[slack[x]][x]));
        else if(S[x]==0)d=min(d,DIST(g[slack[x]][x])/2);
      }
    for(int u=1; u<=n; ++u){
      if(S[st[u]]==0){
        if(lab[u]<=d)return 0;
        lab[u]-=d;
      }
      else if(S[st[u]]==1)lab[u]+=d;
    }
    for(int b=n+1; b<=n_x; ++b)
      if(st[b]==b){
        if(S[st[b]]==0)lab[b]+=d*2;
        else if(S[st[b]]==1)lab[b]-=d*2;
      }
    q.clear();
    for(int x=1; x<=n_x; ++x)
      if(st[x]==x&&slack[x]&&st[slack[x]]!=x&&DIST(g[slack[x]][x])==0)
        if(on_found_Edge(g[slack[x]][x]))return 1;
    for(int b=n+1; b<=n_x; ++b)
      if(st[b]==b&&S[b]==1&&lab[b]==0)expand_blossom(b); }
  return 0; }
pair<ll,int> weight_blossom(){
  fill(match,match+n+1,0);
  n_x=n;
  int n_matches=0;
  ll tot_weight=0;
  for(int u=0; u<=n; ++u)st[u]=u,fl[u].clear();
  int w_max=0;
  for(int u=1; u<=n; ++u)
    for(int v=1; v<=n; ++v){
      fl_from[u][v]=(u==v?u:0);
      w_max=max(w_max,g[u][v].w); }
  for(int u=1; u<=n; ++u)lab[u]=w_max;
  while(matching())++n_matches;
  for(int u=1; u<=n; ++u)
    if(match[u]&&match[u]<u)
      tot_weight+=g[u][match[u]].w;
  return make_pair(tot_weight,n_matches); }
int main(){
  cin>>n>>m;
  for(int u=1; u<=n; ++u)
    for(int v=1; v<=n; ++v)
      g[u][v]=Edge {u,v,0};
  for(int i=0,u,v,w; i<m; ++i){
    cin>>u>>v>>w;
    g[u][v].w=g[v][u].w=w; }
  cout<<weight_blossom().first<<'\n';
  for(int u=1; u<=n; ++u)cout<<match[u]<<' '; }
```

Chu-liu

```
struct UnionFind {
  int fa[N * 2];
  UnionFind() { memset(fa, 0, sizeof(fa)); }
  void clear(int n) { memset(fa + 1, 0, sizeof(int) * n); }
  int find(int x) { return fa[x] ? fa[x] = find(fa[x]) : x; }
  int operator[](int x) { return find(x); } };
struct Edge { int u, v, w, w0; };
struct Heap {
  Edge *e;
  int rk, constant;
  Heap *lch, *rch;
  Heap(Edge *_e) : e(_e), rk(1), constant(0), lch(NULL), rch(NULL) {}
  void push() {
    if (lch) lch->constant += constant;
    if (rch) rch->constant += constant;
    e->w += constant;
    constant = 0; } };
Heap *merge(Heap *x, Heap *y) {
  if (!x) return y;
```

```cpp
  if (!y) return x;
  if (x->e->w + x->constant > y->e->w + y->constant) swap(x, y);
  x->push();
  x->rch = merge(x->rch, y);
  if (!x->lch || x->lch->rk < x->rch->rk) swap(x->lch, x->rch);
  if (x->rch) x->rk = x->rch->rk + 1;
  else x->rk = 1;
  return x;
}
Edge *extract(Heap *&x) {
  Edge *r = x->e; x->push();
  x = merge(x->lch, x->rch);
  return r;
}
vector<Edge> in[N];
int n, m, fa[N * 2], nxt[N * 2];
Edge *ed[N * 2];
Heap *Q[N * 2];
UnionFind id;
void contract() {
  bool mark[N * 2];
  /* 将图上的每一个结点与其相连的那些结点进行记录 */
  for (int i = 1; i <= n; i++) {
    queue<Heap *> q;
    for (int j = 0; j < in[i].size(); j++) q.push(new Heap(&in[i][j]));
    while (q.size() > 1) {
      auto u = q.front(); q.pop();
      auto v = q.front(); q.pop();
      q.push(merge(u, v)); }
    Q[i] = q.front(); }
  mark[1] = true;
  for (int a = 1, b = 1, p; Q[a]; b = a, mark[b] = true) {
    /* 找最小入边及其端点，保证无环 */
    do {
      ed[a] = extract(Q[a]);
      a = id[ed[a]->u];
    } while (a == b && Q[a]);
    if (a == b) break;
    if (!mark[a]) continue;
    /* 收缩环，环内的结点重编号，总权值更新 */
    for (a = b, n++; a != n; a = p) {
      id.fa[a] = fa[a] = n;
      if (Q[a]) Q[a]->constant -= ed[a]->w;
      Q[n] = merge(Q[n], Q[a]);
      p = id[ed[a]->u];
      nxt[p == n ? b : p] = a;
    } } }

LL expand(int x, int r);
LL expand_iter(int x) {
  LL r = 0;
  for (int u = nxt[x]; u != x; u = nxt[u]) {
    if (ed[u]->w0 >= INF) return INF;
    else r += expand(ed[u]->v, u) + ed[u]->w0; }
  return r;
}
LL expand(int x, int t) {
  LL r = 0;
  for (; x != t; x = fa[x]) {
    r += expand_iter(x);
    if (r >= INF) return INF; }
  return r;
}
void adde(int u, int v, int w){
  in[v].push_back({u, v, w, w}); }

int main() {
  int rt;
  scanf("%d %d %d", &n, &m, &rt);
  for (int i = 0; i < m; i++) {
    int u, v, w;
    scanf("%d %d %d", &u, &v, &w);
    adde(u, v, w); }
  /* 保证强连通 */
  for (int i = 1; i <= n; i++)
    adde(i > 1 ? i - 1 : n, i, INF);
  contract();
  LL ans = expand(rt, n);
  if (ans >= INF) puts("-1");
  else printf("%lld\n", ans); }
```

天动万象

```cpp
typedef double D;
#define cp const p3 &
struct p3 {
  D x, y, z;
  void read() {
    int xx, yy, zz;
    scanf("%d%d%d", &xx, &yy, &zz);
    x = xx, y = yy, z = zz;
  }
  p3 () {x = y = z = 0;}
  p3 (D xx, D yy, D zz) {x = xx; y = yy; z = zz;}
  p3 operator + (cp a) const {return {x + a.x, y + a.y, z + a.z};}
  p3 operator - (cp a) const {return {x - a.x, y - a.y, z - a.z};}
  p3 operator * (const D &a) const {return {x * a, y * a, z * a};}
  p3 operator / (const D &a) const {return {x / a, y / a, z / a};}
  D &operator [] (const int a) { return a == 0 ? x : (a == 1 ? y : z); }
  const D &operator [] (const int a) const { return a == 0 ? x : (a == 1 ? y : z); }
  D len2() const { return x * x + y * y + z * z; }
  void normalize() {
    D l = sqrt(len2());
    x /= l; y /= l; z /= l; }
};
const D pi = acos(-1);
D A[3][3];
void calc(p3 n, D cosw) {
  D sinw = sqrt(1 - cosw * cosw);
  n.normalize();
  for (int i = 0; i < 3; i++) {
    int j = (i + 1) % 3, k = (j + 1) % 3;
    D x = n[i], y = n[j], z = n[k];
    A[i][i] = (y * y + z * z) * cosw + x * x;
    A[i][j] = x * y * (1 - cosw) + z * sinw;
    A[i][k] = x * z * (1 - cosw) - y * sinw;
  }
}
p3 turn(p3 x) {
  p3 y;
  for (int i = 0; i < 3; i++)
    for (int j = 0; j < 3; j++)
      y[i] += x[j] * A[j][i];
  return y;
}
p3 cross(cp a, cp b) { return p3(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x); }
D dot(cp a, cp b) {
  D ret = 0;
  for (int i = 0; i < 3; i++)
    ret += a[i] * b[i];
  return ret;
}
const int N = 5e4 + 5;
const D eps = 1e-5;
int sgn(D x) { return (x > eps ? 1 : (x < -eps ? -1 : 0)); }
D det(cp a, cp b) { return a.x * b.y - b.x * a.y; }
p3 base;
bool cmp(cp a, cp b) {
  int d = sgn(det(a - base, b - base));
  if (d) return d > 0;
  else return (a - base).len2() < (b - base).len2();
}
bool turn_left(cp a, cp b, cp c) { return sgn(det(b - a, c - a)) >= 0; }
vector <p3> convex_hull (vector <p3> a) {
  int n = (int) a.size(), cnt = 0;
  base = a[0];
  for (int i = 1; i < n; i++) {
    int s = sgn(a[i].x - base.x);
    if (s == -1 || (s == 0 && a[i].y < base.y))
      base = a[i];
  }
  sort(a.begin(), a.end(), cmp);
  vector <p3> ret;
|  for (int i = 0; i < n; i++) {
|  |  while (cnt > 1 && turn_left(ret[cnt - 2], a[i], ret[cnt - 1])) {
|  |  |  --cnt; ret.pop_back();
|  |  }
|  |  ret.push_back(a[i]); ++cnt;
|  }
|  int fixed = cnt;
|  for (int i = n - 2; i >= 0; i--) {
|  |  while (cnt > fixed && turn_left(ret[cnt - 2], a[i], ret[cnt - 1])) {
|  |  |  --cnt; ret.pop_back();
|  |  }
|  |  ret.push_back(a[i]); ++cnt;
|  }
|  ret.pop_back();
|  return ret;
}
int n, m;
p3 ap[N], bp[N];
int main() {
|  scanf("%d", &n);
|  for (int i = 1; i <= n; i++) ap[i].read();
|  ap[0].read();
|  scanf("%d", &m);
|  for (int i = 1; i <= m; i++) bp[i].read();
|  bp[0].read();
|  p3 from = ap[0] - bp[0], to = {0, 0, 1};
|  if (from.len2() < eps) {
|  |  puts("NO");
|  |  return 0;
|  }
|  from.normalize();
|  p3 c = cross(from, to);
|  if (abs(c.len2()) < eps) {
|  |  // ok
|  }
|  else {
|  |  D cosw = dot(from, to);
|  |  calc(c, cosw);
|  |  for (int i = 1; i <= n; i++) {
|  |  |  ap[i] = turn(ap[i]);
|  |  |  ap[i].z = 0;
|  |  }
|  |  for (int i = 1; i <= m; i++) bp[i] = turn(bp[i]), bp[i].z = 0;
|  }
|  vector <p3> a[2];|
|  for (int i = 1; i <= n; i++) a[0].push_back(ap[i]);
|  for (int i = 1; i <= m; i++) a[1].push_back(p3()-bp[i]);
|  a[0] = convex_hull (a[0]);
|  a[1] = convex_hull (a[1]);

|  vector <p3> mnk;
|  {
|  |  a[0].push_back(a[0].front()); a[1].push_back(a[1].front());
|  |  int i[2] = {0, 0};
|  |  int len[2] = {(int)a[0].size() - 1, (int)a[1].size() - 1};
|  |  mnk.push_back(a[0][0] + a[1][0]);
|  |  do {
|  |  |  int d = sgn(det(a[1][i[1] + 1] - a[1][i[1]],
|  |  |  |  |  |  a[0][i[0] + 1] - a[0][i[0]])) >= 0;
|  |  |  mnk.push_back(a[d][i[d] + 1] - a[d][i[d]] + mnk.back());
|  |  |  i[d] = (i[d] + 1) % len[d];
|  |  } while(i[0] || i[1]);
|  }
|  //mnk = convex_hull(mnk);
|  p3 p; // 0
|  for (int i = 0; i < (int)mnk.size(); i++) {
|  |  p3 u = mnk[i], v = mnk[(i + 1) % int(mnk.size())];
|  |  if (det(p - u, v - u) > eps) {
|  |  |  puts("NO");
|  |  |  return 0;
|  |  }
|  }
|  puts("YES");
}
```