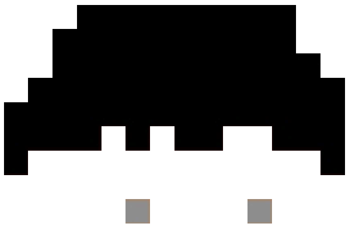
SHANGHAI JIAO TONG UNIVERSITY

Nemesis



Coach	**教练**
Yong Yu	俞勇
Siyu Sun	孙司宇
Boren Tan	谭博仁
Shangfei Yang	杨尚霏

Contestant	**队员**
Zonghan Yang	杨宗翰
Mingchi Zhang	张明驰
Jianjun Zhang	张建军

Compiled on November 6, 2023

Last Commit: 1fdfdd1

Contents

1. Geometry

1.1 二维几何基础操作

```
#define cp const point &
bool turn_left(cp a, cp b, cp c) {
  return sgn(det (b - a, c - a)) >= 0;} // strict: ... > 0
// 要求非退化凸包，或半平面交面积 0 时返回空集：用 strict
vector <point> convex_hull (vector <point> a) {
  int n = (int) a.size (), cnt = 0;
  sort (a.begin(), a.end()); // less<pair>
  if (n <= 2) return a; //未处理重点，非退化凸包：return {}
  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 ();
    ++cnt, ret.push_back (a[i]); }
  int fixed = cnt;
  for (int i = n - 2; i >= 0; --i) {
    while (cnt > fixed
    && turn_left (ret[cnt - 2], a[i], ret[cnt - 1]))
      --cnt, ret.pop_back ();
    ++cnt, ret.push_back (a[i]); }
  ret.pop_back (); return ret;
} // counter-clockwise, ret[0] = min(pair(x, y))
```

```
struct point {
  point rot(LD t) const { // 逆时针
    return {x*cos(t) - y*sin(t), x*sin(t) + y*cos(t)};}
  point rot90() const { return {-y, x}; }};
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; }
point line_inter(cl a, cl b) { // 直线交点
  LD s1 = det(a.t - a.s, b.s - a.s);
  LD s2 = det(a.t - a.s, b.t - a.s);
  return (b.s * s2 - b.t * s1) / (s2 - s1); }
vector <point> cut (const vector<point> &c, line p) {
  vector <point> ret; // 线切凸包，可直接实现半平面交
  int n = (int) c.size();
  if (!n) return ret;
  for (int i = 0; i < n; ++i) {
    int j = (i + 1) % n;
    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_inter (p, {c[i], c[j]})); }
  return ret; }
bool point_on_segment(cp a,cl b){ // 点在线段上
  return sgn (det(a - b.s, b.t - b.s)) == 0 // 在直线上
  && sgn (dot (b.s - a, b.t - a)) <= 0;}
bool inter_judge(cl a,cl b) { // 线段判非严格交
  if (point_on_segment (b.s, a)
   || point_on_segment (b.t, a)) return true;
  if (point_on_segment (a.s, b)
   || point_on_segment (a.t, b)) return true;
  return two_side (a.s, a.t, b)
      && two_side (b.s, b.t, a); }
point proj_to_line (cp a, cl b) { // 点在直线投影
  point st = b.t - b.s;
  return b.s + st * (dot(a - b.s, st) / dot(st, st));}
LD point_to_line (cp a, cl b) { // 点到直线距离
  return abs(det(b.t-b.s, a-b.s)) / dis(b.s, b.t); }
LD point_to_segment (cp a, cl b) { // 点到线段距离
  if (sgn (dot (b.s - a, b.t - b.s))
  * sgn (dot (b.t - a, b.t - b.s)) <= 0)
    return point_to_line(a, b);
  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 cnt = 0;
  for (int i = 0; i < n; ++i) {
    point a = po[i], b = po[(i + 1) % n];
    if (point_on_segment (p, {a, b})) return true;
    int x = sgn (det(p - a, b - a));
    int y = sgn (a.y - p.y);
    int z = sgn (b.y - p.y);
    if (x > 0 && y <= 0 && z > 0) ++cnt;
    if (x < 0 && z <= 0 && y > 0) --cnt; }
  return cnt != 0; }
bool point_on_ray (cp a, cl b) { // 点在射线上
  return sgn (det(a - b.s, b.t - b.s)) == 0
  && sgn (dot (a - b.s, b.t - b.s)) >= 0; }
bool ray_inter_judge(line a, line b) { // 射线判交
  LD s1, s2; // can be LL
  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); }
```

```
LD diameter(vector <point> p) { // 最远点对
  p = convex_hull(p);
  int n = (int) p.size(); LD ret = 0;
  for (int i = 0, j, k = 1; i < n; i++) {
    j = (i + 1) % n;
    if (k == j) ++k %= n;
    while (k != i) {
      int l = (k + 1) % n;
      if (sgn(det(p[j]-p[i], p[l]-p[k])) > 0) k = l;
      else break; }
    ret = max(ret, dis2(p[i], p[k]));
    ret = max(ret, dis2(p[j], p[k])); } return ret; }
```

```
int sgn(cp a){return a.y > 0||(a.y == 0 && a.x > 0)?1:-1;}
bool turn_left(cl l, cp p){return turn_left(l.s, l.t, p);}
vector <point> hpi(vector <line> h) { // 半平面交
  sort(h.begin(), h.end(), [](cl a, cl b) {
    int dir = sgn(a.t - a.s) - sgn(b.t -  b.s);
    if (dir) return dir > 0; // sign of direction
    dir = sgn(det(a.t - a.s, b.t - b.s));
    if (dir) return dir > 0; // direction
    dir = sgn(det(a.t - a.s, b.t - a.s));
    return dir < 0; // line position
    // start position: sgn(dot(a.t - a.s, b.s - a.s)) > 0
  });
  h.resize(unique(h.begin(), h.end(), [](cl a, cl b) {
    return !sgn(det(a.t - a.s, b.t - b.s));})-h.begin());
  vector <line> q; int l = 0, r = -1;
  for(auto &i : h) {
   while(l<r && !turn_left(i, line_inter(q[r - 1], q[r])))
     --r, q.pop_back();
   while(l<r && !turn_left(i, line_inter(q[l], q[l + 1])))
     ++l;
   ++r; q.push_back(i); }
  while(r-l>1&& !turn_left(q[l],line_inter(q[r - 1],q[r])))
    --r, q.pop_back();
  while(r-l>1&& !turn_left(q[r],line_inter(q[l],q[l + 1])))
    ++l;
  if(r - l < 2) return {};
  vector <point> ret(r - l + 1);
  for(int i = l; i <= r; i++)
    ret[i - l] = line_inter(q[i], q[i == r ? l : i + 1]);
  return ret; }
```

1.2 线段在多边形内 (Airport Construction)

```
bool in_polygon (cp u, cp v) {
  // assert : u, v in polygon, respectively
  for (int i = 0; i < n; i++) {
    int j = (i + 1) % n, k = (i + 2) % n;
    cp ii = p[i], jj = p[j], kk = p[k];
    if (inter_judge_strict({u, v}, {ii, jj})) return 0;
    if (point_on_segment (jj, {u, v})) {
      bool good = true, left = turn_left(ii, jj, kk);
      for (auto x : {u, v})
        if (left)
          good &=    turn_left(ii, jj, x)
                  && turn_left(jj, kk, x);
        else
          good &= !(   turn_left_strict(jj, x, kk)
                    && turn_left_strict(jj, ii, x));
      if (!good) return 0;
    } } return 1; }
LD get_far (int uid, int vid) {
  // u -> v in polygon, check the ray u -> polygon
  cp u = p[uid], v = p[vid];
  LD far = 1e9;
  for (int i = 0; i < n; i++) {
```

```
    int j = (i + 1) % n, k = (i + 2) % n;
    cp ii = p[i], jj = p[j], kk = p[k];
    if (two_side (ii, jj, {u, v})) {
      LD s1 = det(jj - ii, u - ii);
      LD s2 = det(jj - ii, v - ii);
      if (sgn(s1 - s2) && sgn(s1) != sgn(s2 - s1))
        far = min(far,
                  dis(u, line_inter({ii,jj},{u,v})));}
    if (j != uid && point_on_ray (jj, {u, v})) {
      bool good = !turn_left_strict(ii, jj, kk);
      for (auto x : {u - (jj - u), jj + (jj - u)})
        good &= !(   turn_left_strict(ii, jj, x)
                  && turn_left_strict(jj, kk, x));
      if (!good) far = min(far, dis(u, jj));
  } } return far; }
void work() {
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++) if (i != j) {
      if (!in_polygon(p[i], p[j])) continue;
      LD ret = get_far(i,j)+get_far(j,i)-dis(p[i],p[j]);
      ans = max(ans, ret); } }
```

1.3 凸包快速询问

```
/* INF 开到值域, point operator < > 为 pair(x, y)
除距离可改为整数: LD to LL
传入逆时针凸包要求无重点, 面积非空 */
struct Convex {
 int n, lz, uz;
 vector<point> a, upper, lower;
 Convex(vector<point> _a) {
   n = (int) _a.size(); int k = 0;
   a = _a; // if a[0] is lowest, otherwise:
 /*for(int i = 1; i < n; i++) if (_a[i] < _a[k]) k = i;
 for(int i = 0; i < n; i++) a.push_back(_a[(i + k) % n]);*/
   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]);
   lz = (int) lower.size(); uz = (int) upper.size(); }
 const point & at (int x) { return a[x % n]; }
 pair <LD, int> get_tan(vector <point> & con, cp vec) {
   int l = 0, r = (int) con.size() - 2;
   for ( ; l + 1 < r; ) {
     int m = (l + r) / 2;
     if (sgn(det(con[m + 1] - con[m], vec)) > 0) r = m;
     else l = m;
   }
   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; }
 // 判定点是否在凸包内, 在边界返回 true
 bool contain(cp p) {
   if (p.x < lower[0].x || p.x > lower.back().x) return 0;
   int id = int(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 0;
   } else if (det(lower[id - 1] - p, lower[id] - p) < 0)
     return 0;
   id = int(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 0;
   } else if (det(upper[id - 1] - p, upper[id] - p) < 0)
     return 0;
   return 1; }
 // 点关于凸包的切点, 在凸包外有序返回编号, 共线返回任意
 void search(int l, int r, cp p, int &i0, int &i1) {
   if (l == r) return;
   upd_tan(p, l % n, i0, i1);
   int sl = sgn(det(at(l) - p, at(l + 1) - p));
   for ( ; l + 1 < r; ) {
     int m = (l + r) / 2;
     int sm = sgn(det(at(m) - p, at(m + 1) - p));
     if (sm == sl) l = m;
     else r = m;
   }
   upd_tan(p, r % n, i0, i1); }
 bool get_tan(cp p, int &i0, int &i1) {
   // if (contain(p)) return 0;
   i0 = i1 = 0;
   int id = int(lower_bound(lower.begin(), lower.end(), p) - lower.begin());
   search(0, id, p, i0, i1);
   search(id, lz, p, i0, i1);
   id = int(lower_bound(upper.begin(), upper.end(), p, greater<point>()) - upper.begin());
   search(lz - 1, lz - 1 + id, p, i0, i1);
   search(lz - 1 + id, lz - 1 + uz, p, i0, i1);
   return 1; }
 // 求凸包外一点到凸包的最短距离, 如凸包内返回 0; 结果产生浮点
 LD search(int l, int r, cp p) {
   if (l == r) return dis (p, at(l));
   int sl = sgn(dot(a[l%n]-p, at(l + 1)-at(l)));
   for ( ; l + 1 < r; ) {
     int m = (l + r) / 2;
     int sm = sgn(dot(at(m)-p,at(m+1)-at(m)));
     if (sm == sl) l = m; else r = m;
   } return point_to_segment(p, {at(l), at(l+1)}); }
 LD get_dis(cp p) {
   if (contain(p)) return 0;
   int i0, i1;
   get_tan(p, i0, i1);
   return search(i0, i1 + (i0 > i1 ? n : 0), p); }
 // 求凸包上和向量 vec 叉积最大的点编号, 共线返回任意
 int get_tan(cp vec) {
   pair<LD, int> ret = get_tan(upper, vec);
   ret.second = (ret.second + lz - 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 0. 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
 int search(cp u, cp v, int l, int r) {
   int sl = sgn(det(v - u, at(l) - u));
   for ( ; l + 1 < r; ) {
     int m = (l + r) / 2;
     int sm = sgn(det(v - u, at(m) - u));
     if (sm == sl) l = m;
     else r = m; }
   return l % n; }
 bool get_inter(cp u, cp 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 0; }
};
```

1.4 闵可夫斯基和

```
// 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]);
```

1.5 圆

```
struct circle { point c; LD r;};
bool in_circle(cp a, const circle &b) {
  return (b.c - 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);
  LD 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
```

```
pair <point, point> line_circle_inter (cl a, cc c) {
 | LD d = point_to_line (c.c, a);
 | // 需要的话返回 vector <point>
 | /* if (sgn (d - R) >= 0) return {}; */
 | LD x = sqrt (sqr(c.r) - sqr(d)); // sqrt(max(0., ...))
 | return {
 |  | proj_to_line (c.c, a) + (a.s - a.t).unit() * x,
 |  | proj_to_line (c.c, a) - (a.s - a.t).unit() * x }; }
LD circle_inter_area (cc a, cc b) { // 圆面积交
 | LD 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) {
 |  | LD r = min (a.r, b.r);
 |  | return r * r * PI; }
 | LD x = (d * d + a.r * a.r - b.r * b.r) / (2 * d),
 |  |    t1 = acos (min (1., max (-1., x / a.r))),
 |  |    t2 = acos (min (1., max (-1., (d - x) / b.r)));
 | return sqr(a.r)*t1 + sqr(b.r)*t2 - d*a.r*sin(t1);}
vector <point> circle_inter (cc a, cc b) { // 圆交点
 | 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();
 | LD d = dis (a.c, b.c);
 | LD x = ((sqr (a.r) - sqr (b.r)) / d + d) / 2;
 | LD 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) {
 | return circle_inter (make_circle (a, b.c), 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.len()).rot90();
 |  | ret.push_back({a.c + dir, b.c + dir});
 |  | ret.push_back({a.c - dir, b.c - dir});
 | } else {
 |  | point p = (b.c * a.r - a.c * b.r) / (a.r - b.r);
 |  | vector u = tangent(p, a), v = tangent(p, b);
 |  | if (u.size() == 2 && v.size() == 2) {
 |  |  | if (sgn(a.r-b.r) < 0)
 |  |  |  | swap(u[0], u[1]), swap(v[0], v[1]);
 |  |  | ret.push_back({u[0], v[0]});
 |  |  | ret.push_back({u[1], v[1]}); } }
 | return ret; }
vector <line> intangent(cc a, cc b) {
 | vector <line> ret;
 | point p = (b.c * a.r + a.c * b.r) / (a.r + b.r);
 | vector u = tangent(p, a), v = tangent(p, b);
 | if (u.size() == 2 && v.size() == 2) {
 |  | ret.push_back({u[0], v[0]});
 |  | ret.push_back({u[1], v[1]}); } return ret; }
circle min_circle (vector <point> p) { // 最小覆盖圆
 circle ret({0, 0}, 0);
 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; }
```

1.6 圆并

```
int C; circle c[MAXN]; LD area[MAXN];
struct event {
 point p; LD ang; int delta;
 bool operator <(const event &a){return ang < a.ang;}};
void addevent(cc a, cc b, vector<event> &e, int &cnt) {
 LD d2=dis2(a.c, b.c),dw=((a.r-b.r)*(a.r+b.r)/d2+1)/2,pw=
sqrt(max(0.,-(d2-sqr(a.r-b.r))*(d2-sqr(a.r+b.r))/sqr(2*d2));
 point d = b.c - a.c, p = d.rot(PI / 2),
  q0 = a.c + d * dw + p * pw,
  q1 = a.c + d * dw - p * pw;
 LD ang0 = atan2((q0 - a.c).y, (q0 - a.c).x),
    ang1 = atan2((q1 - a.c).y, (q1 - a.c).x);
 e.push_back({q1,ang1,1}); e.push_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> e;
  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], e, cnt);
  if(e.empty()) area[cnt] += PI * c[i].r * c[i].r;
  else {
   sort(e.begin(), e.end());
   e.push_back(e.front());
   for(int j = 0; j + 1 <(int)e.size(); ++j) {
    cnt += e[j].delta;
    area[cnt] += det(e[j].p,e[j + 1].p) / 2;
    LD ang = e[j + 1].ang - e[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; } } } }
```

1.7 多边形与圆交

```
LD angle (cp u, cp v) {
 | return 2 * asin(dis(u.unit(), v.unit()) / 2); }
LD area(cp s, cp t, LD r) { // 2 * area
 | LD theta = angle(s, t);
 | LD dis = point_to_segment({0, 0}, {s, t});
 | if (sgn(dis - r) >= 0) return theta * r * r;
 | auto [u, v] = line_circle_inter({s, t}, {{0, 0}, r});
 | point lo = sgn(det(s, u)) >= 0 ? u : s;
 | point hi = sgn(det(v, t)) >= 0 ? v : t;
 | return det(lo, hi) + (theta - angle(lo, hi)) * r * r; }
LD solve(vector<point> &p, cc c) {
 | LD ret = 0;
 | for (int i = 0; i < (int) p.size (); ++i) {
 |  | auto u = p[i] - c.c;
 |  | auto v = p[(i + 1) % p.size()] - c.c;
 |  | int s = sgn(det(u, v));
 |  | if      (s > 0) ret += area (u, v, c.r);
 |  | else if (s < 0) ret -= area (v, u, c.r);
 | } return abs (ret) / 2; } //ret在p逆时针时为正
```

1.8 阿波罗尼茨圆

所有关于两点 A, B 满足 $PA/PB = k$ 且不等于 1 的点 P 的轨迹是一个圆.

硬币游戏

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 $r_4^{\pm} = \frac{r_1r_2r_3}{r_1r_2+r_1r_3+r_2r_3\pm 2\sqrt{r_1r_2r_3(r_1+r_2+r_3)}}$

1.9 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.

1.10 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: $S = ER^2$

1.11 经纬度球面距离

```cpp
// 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)); }
```

1.12 长方体表面两点最短距离

```cpp
int r;
void go(int i, int j, int x, int y, int z,int x0, int y0, int L, int W, int H) {
if (z==0) { int R = x*x+y*y; if (R<r) r=R;}
else {
if(i>=0&&i<2)go(i+1,j, x0+L+z, y, x0+L-x, x0+L, y0, H,W,L);
if(j>=0&&j<2)go(i,j+1, x, y0+W+z, y0+W-y, x0, y0+W, L,H,W);
if(i<=0&&i>-2)go(i-1, j, x0-z, y, x-x0, x0-H, y0, H, W, L);
if(j<=0&&j>-2)go(i, j-1, x, y0-z, y-y0, x0, y0-H, L, H, W);
} } int main(){
int L, H, W, x1, y1, z1, x2, y2, z2;
cin >> L >> W >> H >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;
if (z1!=0 && z1!=H) if (y1==0 || y1==W)
    swap(y1,z1), swap(y2,z2), swap(W,H);
else swap(x1,z1), swap(x2,z2), swap(L,H);
if (z1==H) z1=0, z2=H-z2;
r=0x3fffffff;
go(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
cout<<r<<endl; }
```

1.13 圆上整点

```cpp
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; }
```

1.14 三相之力

```cpp
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, {p.y, q.y}), det({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_inter ({a, b + (c - b).rot (sq3)}, {b, c + (a - c).rot (sq3)});
    else mid = line_inter ({a, c + (b - c).rot (sq3)}, {c, b + (a - b).rot (sq3)});
    return mid; } // minimize(|A-x|+|B-x|+|C-x|)
```

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

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

1.15.4 三角形外心

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

1.15.5 三角形垂心

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

1.15.6 三角形偏心

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

内角的平分线和对边的两个外角平分线交点, 外切圆圆心. 剩余两点的同理.

1.15.7 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 格点多边形面积

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

1.15.9 Euler's Formula 多面体与平面图的点、边、面

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

1.16 三角公式

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

1.16.1 超球坐标系

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

1.16.2 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

$$\phi : \text{二面角}$$
$$\Omega = (\phi_{ab} + \phi_{bc} + \phi_{ac})\ \text{rad} - \pi\ \text{sr}$$

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

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

1.16.4 常用体积公式

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

1.16.5 扇形与圆弧重心

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

1.16.6 高维球体积

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

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

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

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

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

1.17 三维几何基础操作

```cpp
/* 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

```cpp
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

```cpp
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);
```

```cpp
    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); }
```

1.20 高斯消元最小范数解

```cpp
typedef vector <LD> vec; /* sum a[i][0..d] = 0 */
pair<vec,vector<vec>> gauss(vector<vec> &a, int n, int d) {
  vector <int> pivot(d, -1);
  for (int i = 0, o = 0; i < d; i++) {
    int j = o; while (j < n && abs(a[j][i]) < eps) j++;
    if (j == n) continue;
    swap(a[j], a[o]); LD w = a[o][i];
    for (int k = 0; k <= d; k++) a[o][k] /= w;
    for (int x = 0; x < n; x++)
      if (x != o && abs(a[x][i]) > eps) {
        w = a[x][i];
        for (int k = 0; k <= d; k++)
          a[x][k] -= a[o][k] * w;   }
    pivot[i] = o++;
  } vec x0(d); vector <vec> t; int free = 0;
  for (int i = 0; i < d; i++)
    if (pivot[i] != -1) x0[i] = -a[pivot[i]][d];
    else free ++;
  for (int i = 0; i < d; i++) if (pivot[i] == -1) {
    vec x(d); x[i] = -1;
    for (int j = 0; j < d; j++)
      if (pivot[j] != -1) x[j] = a[pivot[j]][i];
    t.push_back(x);
  } if (t.size()) {
    vector <vec> f;
    for (int u = 0; u < free; u++) {
      vec x(free + 1);
      for (int i = 0; i < free; i++)
        for (int j = 0; j < d; j++)
          x[i] += t[u][j] * t[i][j];
      for (int j = 0; j < d; j++)
        x[free] += t[u][j] * x0[j];
      f.push_back(x);
    }
    auto [k, tt] = gauss(f, free, free);
    assert (tt.size() == 0);
    for (int x = 0; x < free; x++)
      for (int i = 0; i < d; i++)
        x0[i] += k[x] * t[x][i];
  } return {x0, t}; }
```

2. Tree & Graph

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

```cpp
// 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 Shuffle 一般图最大匹配 $O(VE)$

```cpp
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); } }
```

2.3 KM 最大权匹配 $O(V^3)$

```cpp
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;
```

2.4 极大团计数

```cpp
// 0下标，需删除自环（即确保 E_ii = 0，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

```cpp
ULL get(vector <ULL> ha) {
  sort(ha.begin(), ha.end());
```

```cpp
  ULL ret = 0xdeadbeef;
  for (auto i : ha) {
    ret = ret * P + i, ret ^= ret << 17;
  } return ret * 997; }
```

2.6 欧拉回路

```cpp
/* 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;}
```

2.7 2-SAT, 强连通分量

```cpp
int stp, comps, top; // N 开 **两倍**
int dfn[N], low[N], comp[N], stk[N], answer[N];
void add(int x, int a, int y, int b) {
//取 X_a 则必取 Y_b. 注意连边对称, 即必须 X_b → Y_a.
  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 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; }
```

2.8 Tarjan 点双, 边双

```cpp
/** 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 last) {
  dfn[now] = low[now] = ++stp;
  for (int i = head[now], d; i; i = h[i].next) {
    d = h[i].node;
    if (!dfn[d]) {
      tar(d, i);
      low[now] = min(low[now], low[d]);
      if (low[d] > dfn[now])
        bri[i] = bri[i ^ 1] = 1; }
    else if (dfn[d] < dfn[now] && ((i ^ 1) != last))
      low[now] = min(low[now], dfn[d]); } }
void DFS(int now) {
  vis[now] = 1;
  bcc_id[now] = bcc_cnt;
  bcc[bcc_cnt].push_back(now);
  for (int i = head[now], d; i; i = h[i].node) {
    d = h[i].node;
    if (bri[i]) continue;
    if (!vis[d]) DFS(d); } }
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); }
/** 建立圆方树+求割点 **/
int is_cut[N], DFN[N], low[N], cnt;
int stk[N], dep; // clear dfn low is_cut cnt, let pcnt=n
void tarjan(int x, int fa) {
  int child = 0;
  DFN[x] = low[x] = ++cnt; stk[++dep] = x;
  #define head org
  for (int i = head[x], d; i; i = h[i].next) {
    d = h[i].node;
    if (!DFN[d]) {
      ++child; tarjan(d, x);
      low[x] = min(low[x], low[d]);
      if (low[d] >= DFN[x]) {
        is_cut[x] = true;
        ++pcnt; // square node index
        int j = 0, sz = 1;
        do {
          j = stk[dep--];
          addedge(pcnt, j, tr);
          ++sz;
        } while (j != d);
        addedge(pcnt, x, tr);
      }
    } else if (DFN[d] < low[x]) low[x] = DFN[d]; }
  #undef head
  if (!fa && child == 1) is_cut[x] = false; }
```

2.9 Dominator Tree 支配树

```cpp
vector<int> G[maxn], R[maxn], son[maxn];
int ufs[maxn]; // R是反图, son存的是sdom树上的儿子
int idom[maxn], sdom[maxn], anc[maxn];
// anc: sdom的dfn最小的祖先
int p[maxn], dfn[maxn], id[maxn], tim;
int findufs(int x) { if (ufs[x] == x) return x;
  int t = ufs[x]; ufs[x] = findufs(ufs[x]);
  if (dfn[sdom[anc[x]]] > dfn[sdom[anc[t]]])
    anc[x] = anc[t];
  return ufs[x]; }
void dfs(int x) {
  dfn[x] = ++tim; id[tim] = x; sdom[x] = x;
  for (int y : G[x]) if (!dfn[y]) {
    p[y] = x; dfs(y); } }
void get_dominator(int n) {
  for (int i = 1; i <= n; i++) ufs[i] = anc[i] = i;
  dfs(1);
  for (int i = n; i > 1; i--) { int x = id[i];
    for (int y : R[x]) if (dfn[y]) { findufs(y);
      if (dfn[sdom[x]] > dfn[sdom[anc[y]]])
        sdom[x] = sdom[anc[y]]; }
    son[sdom[x]].push_back(x); ufs[x] = p[x];
    for (int u : son[p[x]]) { findufs(u);
      idom[u] = (sdom[u] == sdom[anc[u]] ? p[x] : anc[u]); }
    son[p[x]].clear(); }
  for (int i = 2; i <= n; i++) { int x = id[i];
    if (idom[x] != sdom[x]) idom[x] = idom[idom[x]];
    son[idom[x]].push_back(x); } }
```

2.10 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 $\mathrm{dist}(s,t)$ 的增广路集合 blocking flow, 增广后 $\mathrm{dist}(s,t)$ 将会增大. 因此只有 $O(V)$ 轮; 如果一轮增广是 $O(VE)$ 的, 总复杂度是 $O(V^2E)$. 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 $O(\min\{V^{2/3}, E^{1/2}\}E)$.
- $\mathrm{dist}(s,t) = d$, 残量网络上至多还存在 E/d 的流.
- 每个点只有一个入/出度时复杂度 $O(V^{1/2}E)$, 例如 Hopcroft-Karp.

```cpp
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; }
```

2.11 原始对偶费用流

```cpp
bool bfs() {
  for (int i = S; i <= T; i++) cur[i] = head[i];
  for (int i = S; i <= T; i++) dep[i] = INF_int; // S-T?
  dep[S] = 0; queue<int> q; q.push(S);
  while (!q.empty()) {
    int x = q.front(); q.pop();
    for (int i = head[x]; i; i = e[i].nxt) {
      int d = e[i].v;
      if (e[i].f > 0 && h[d] == h[x] + e[i].w
      && dep[d] > dep[x] + 1) {
        dep[d] = dep[x] + 1, q.push(d);
  } } } return dep[T] < INF_int; }
int dfs(int x, int lim) {
  if (x == T || !lim) return lim;
  int f = 0, flow = 0;
  for (int &i = cur[x]; i; i = e[i].nxt) {
    int d = e[i].v;
    if ((dep[d] == dep[x] + 1) && h[e[i].v] == e[i].w + h[x]
    && (f = dfs(d, min(lim, e[i].f)))) {
      e[i].f -= f; e[i ^ 1].f += f;
      flow += f; lim -= f;
      if (lim == 0) break;
  } } return flow; }
typedef pair <LL, int> pii; // NOTE: unusual!
pii solve() { // return {cost, flow}
  LL res = 0; int flow = 0;
  for (int i = 0; i <= T; ++i) h[i] = 0;
  int first = true;
  while (true) {
    priority_queue<pii, vector<pii>, greater<pii>> q;
    for (int i = S; i <= T; i++) dis[i] = INF_LL; // S-T?
    dis[S] = 0;
    if (first) {
      // TODO: SSSP, may Bellman-Ford or DP
      first = false;
    } else { 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) {
          LL w = dis[x] + e[o].w + h[x] - h[e[o].v];
          if (e[o].f > 0 && dis[e[o].v] > w) {
            dis[e[o].v] = w; q.push(pii(w, e[o].v));
    } } } }
    if (dis[T] >= INF_LL) break;
    // 所有点必须可达, 可以加 (i,T,0): min(dis[i], dis[T])
    for (int i = 0; i <= T; ++i) h[i] += dis[i];
    int fl = 0; while (bfs()) fl += dfs(S, INF_int);
    res += fl * h[T]; flow += fl;
  }
  return make_pair(res, flow); }
```

2.12 虚树

```cpp
int one = 0, top = 0;
for (auto i : q) one |= i == 1;
if (!one) q.push_back(1);
sort(q.begin(), q.end(), [](auto u, auto v) {
  return dfn[u] < dfn[v]; });
for (auto x : q) {
  used.push_back(x);
  if (top == 0) stk[++top] = x;
  else {
    int lca = LCA(stk[top], x);
    used.push_back(lca);
    while (top > 1 && dep[lca] < dep[stk[top - 1]]) {
      h[stk[top - 1]].push_back(stk[top]);
      --top; }
    if (dep[lca] < dep[stk[top]])
      h[lca].push_back(stk[top--]);
    if (stk[top] != lca)
      stk[++top] = lca;
    stk[++top] = x; } }
while (--top) // assert (top)
  h[stk[top]].push_back(stk[top + 1]);
LL ans = solve(1, 0);
for (auto i : used) h[i].clear();
```

2.13 网络流总结

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 $\mathrm{scc}_S = \mathrm{scc}_u$ 且 $\mathrm{scc}_T = \mathrm{scc}_v$, 则该边为必须边.

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 $\mathrm{deg}_i = \mathrm{in}_i - \mathrm{out}_i$ 为奇数, 肯定不存在欧拉回路. 对于 $\mathrm{deg}_i > 0$ 的点, 连接边 $(i, T, \mathrm{deg}_i/2)$; 对于 $\mathrm{deg}_i < 0$ 的点, 连接边 $(S, i, -\mathrm{deg}_i/2)$. 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 $\frac{x_1-x_2}{2}$.

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 $\mathrm{maxflow} == du_i > 0$) 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.

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).

2.14 Gomory-Hu 无向图最小割树 $O(V^3 E)$

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.

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

```cpp
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(vis, 0, sizeof(vis));
  memset(dis, 0, sizeof(dis));
  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()); }
```

2.16 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.

```cpp
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;
  } }
```

2.17 Minimum Mean Cycle 最小平均值环 $O(n^2)$

```cpp
// 点标号为 1,2,···,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); }
```

2.18 斯坦纳树

```cpp
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);}}
```

2.19 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 $\Sigma y_i, \Sigma x_i$ 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 Σd_i 为偶数, 且对于任意的 $1 \le k \le n - 1$ 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

曼哈顿转切比雪夫: $(x + y, x - y)$, 适用于一些每次只能向四联通的格子走一格的问题. 切比雪夫转曼哈顿: $(\frac{x+y}{2}, \frac{x-y}{2})$, 适用于统计距离.

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+2,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

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

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

找出无用边 将修改边标 ∞, 不在MST上的其余边为无用边, 删除之.

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

2.19.7 差分约束

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

2.19.8 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 $O(n \log n)$.

2.19.9 Segment Tree Beats

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

2.19.10 二分图

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

2.19.11 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 $O(n^2)$.

2.19.12 三元环

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

2.19.13 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.

2.19.14 竞赛图 Landau's Theorem

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

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

6 个人中存在 3 人相互认识或者相互不认识.

2.19.16 树的计数 Prufer序列

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

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

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

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

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

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

2.19.17 有根树计数 1,1,2,4,9,20,48,115,286,719,1842,4766

无标号 $a_{n+1} = 1/n \sum_{k=1}^{n} (\sum_{d|k} d \cdot a(d)) \cdot a(n - k + 1)$

2.19.18 无根树计数

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

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

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

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

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

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

有向图外向树计数: $c = |$ 去掉根所在的那阶得到的主子式 $|$

2.19.20 有向图欧拉回路计数 BEST Thoerem

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

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: $\mathrm{t_v}(G) = \mathrm{t_w}(G)$.

2.19.21 Tutte Matrix

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

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

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

2.19.22 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 有向图无环定向, 色多项式

图的色多项式 $P_G(q)$ 对图 G 的 q-染色计数.
Triangle $K_3 : x(x-1)(x-2)$
Complete graph $K_n : x(x-1)(x-2)\cdots(x-(n-1))$
Tree with n vertices : $x(x-1)^{n-1}$
Cycle $C_n : (x-1)^n + (-1)^n(x-1)$
acyclic orientations of an n-vertex graph G is $(-1)^n P_G(-1)$.

2.19.24 拟阵交问题

最大带权拟阵交问题: 全集 U 中每个元素都有权值 w_i. 设**同一个**全集 U 上有两个满足拟阵性质的集族 $\mathcal{F}_1, \mathcal{F}_2$. 对于 $k = 1..|U|$, 分别求出一个集合 S, 满足 $S \in \mathcal{F}_1 \cap \mathcal{F}_2$ 且 $|S|$ 恰好为 k 的前提下, S 中元素权值和最小.

设集合大小为 k 时已经求出了答案 S. 现在希望求出集合大小为 $k+1$ 的答案. U 中所有元素分为两个集合: 当前答案集合 S, 和剩余集合 $T = U \backslash S$. 考虑 T 中的某个元素 x_i. 记 $A = \{x_i | S \cup \{x_i\} \in \mathcal{F}_1\}$, $B = \{x_i | S \cup \{x_i\} \in \mathcal{F}_2\}$. 如果 T 中某个元素 $x_i \notin A$, 说明 x_i 加进 S 中形成了某个" 环", 从而不满足 $\mathcal{F}_1$ 的限制. 考虑这个" 环" 上每个元素 y_j, 满足 $S \backslash \{y_j\} \cup \{x_i\} \in \mathcal{F}_1$, 将 x_i 向每个 y_j 连边. 如果 T 中某个元素 $x_i \notin B$, 同理找出 S 中每一个元素 y_j 使得 $S \backslash \{y_j\} \cup \{x_i\} \in \mathcal{F}_2$, 将 y_j 向 x_i 连边. 现在求出从 A 到 B 的多源多汇最短路, 权值在点上, 若点属于 T 则权值为正, 否则属于 S, 权值为负. 最短路上每个 T 中的点放进 S, S 中的点放进 T, 则完成了一次增广. 由于每次增广路的起点和终点都在 T 中, 所以每次增广都会使得 $|S|$ 增加1.

最大拟阵交问题可以去掉权值直接求增广路.

2.19.25 双极定向

```
//双极定向：给定无向图和两个极点s,t，要求将每条边定向后成为DAG，使得s可达所有点，所有点均可达t
//topo为定向后DAG的拓扑序，边（u,v）定向为u->v当且仅当拓扑序中u在v的前面.
int n, dfn[N], low[N], stamp, p[N], preorder[N], topo[N];
bool fucked = 0, sign[N]; vector<int> G[N];
void dfs(int x, int fa, int s, int t){
    dfn[x] = low[x] = ++stamp;
    preorder[stamp] = x, p[x] = fa;
    if (x == s) dfs(t, x, s, t);
    for (int y : G[x]){
        if (x == s && y == t) continue;
        if (!dfn[y]){
            if (x == s) fucked = true;
            dfs(y, x, s, t);
            low[x] = min(low[x], low[y]); }
        else if (dfn[y] < dfn[x] && y != fa)
            low[x] = min(low[x], dfn[y]); } }
bool bipolar_orientation(int s, int t){
    G[s].push_back(t), G[t].push_back(s);
    stamp = fucked = 0, dfs(s, s, s, t);
    for (int i = 1; i <= n; i++)
        if (i != s && (!dfn[i] || low[i] >= dfn[i]))
            fucked = true;
    if (fucked) return false;
    sign[s] = 0;//memset sign[] is not necessary
    int pre[n + 5], suf[n + 5]; // list
    suf[0] = s; pre[s] = 0, suf[s] = t;
    pre[t] = s, suf[t] = n + 1; pre[n + 1] = t;
    for (int i = 3; i <= n; i++){
        int v = preorder[i];
        if (!sign[preorder[low[v]]]){ // insert before p[v]
            int P = pre[p[v]];
            pre[v] = P, suf[v] = p[v];
            suf[P] = pre[p[v]] = v; }
        else{ // insert after p[v]
            int S = suf[p[v]];
            pre[v] = p[v], suf[x] = S;
            suf[p[v]] = pre[S] = v; }
        sign[p[x]] = !sign[preorder[low[x]]]; }
    for (int x = s, cnt = 0; x != n + 1; x = suf[x])
        topo[++cnt] = x;
    return true; }
```

2.19.26 图中的环

没有奇环的图是**二分图**, 没有偶环的图是**仙人掌**. 判定没有奇环仅用深度奇偶性判即可; 判定没有偶环的图需要记录覆盖次数判定是否存在奇环有交.

3. Data Structure

3.1 非递归线段树

3.1.1 区间加，区间求最大值

```
void update(int l, int r, int d) {
    for (l += M-1, r += M+1; l^r^1; l >>= 1, r >>= 1) {
        if (l < M) {
            t[l] = max(t[l*2], t[l*2+1]) + mark[l];
            t[r] = max(t[r*2], t[r*2+1]) + mark[r]; }
        if (~l & 1) { t[l ^ 1] += d; mark[l ^ 1] += d; }
        if (r & 1) { t[r ^ 1] += d; mark[r ^ 1] += d; } }
    for (; l; l >>= 1, r >>= 1)
        if (l < M) t[l] = max(t[l*2], t[l*2+1]) + mark[l],
                t[r] = max(t[r*2], t[r*2+1]) + mark[r]; }
int query(int l, int r) {
    int maxl = -INF, maxr = -INF;
    for (l += M-1, r += M+1; l^r^1; l >>= 1, r >>= 1) {
        maxl += mark[l]; maxr += mark[r];
        if (~l & 1) maxl = max(maxl, t[l ^ 1]);
        if (r & 1) maxr = max(maxr, t[r ^ 1]); }
    while (l) { maxl += mark[l]; maxr += mark[r];
        l >>= 1; r >>= 1; }
    return max(maxl, maxr); }
```

3.2 点分治

```
vector<pair<int, int> > G[maxn];
int sz[maxn], son[maxn], q[maxn];
int pr[maxn], depth[maxn], rt[maxn][19], d[maxn][19];
int cnt_all[maxn],sum_all[maxn],cnt[maxn][],sum[maxn][];
bool vis[maxn], col[maxn];
int getcenter(int o, int s) {
    int head = 0, tail = 0; q[tail++] = o;
    while (head != tail) {
        int x = q[head++]; sz[x] = 1; son[x] = 0;
        for (auto [y, _] : G[x]) if (!vis[y] && y != pr[x]) {
            pr[y] = x; q[tail++] = y; } }
    for (int i = tail - 1; i; i--) {
        int x = q[i]; sz[pr[x]] += sz[x];
        if (sz[x] > sz[son[pr[x]]]) son[pr[x]] = x; }
    int x = q[0];
    while (son[x] && sz[son[x]] * 2 >= s) x = son[x];
    return x; }
void getdis(int o, int k) {
    int head = 0, tail = 0; q[tail++] = o;
    while (head != tail) {
        int x = q[head++]; sz[x] = 1; rt[x][k] = o;
        for (auto [y, w] : G[x]) if (!vis[y] && y != pr[x]) {
            pr[y]=x; d[y][k] = d[x][k] + w; q[tail++]=y; } }
    for (int i = tail - 1; i; i--)sz[pr[q[i]]] += sz[q[i]];}
void build(int o, int k, int s, int fa) {
    int x = getcenter(o, s);
    vis[x] = true; depth[x] = k; pr[x] = fa;
    for (auto [y, w] : G[x]) if (!vis[y]) {
        d[y][k] = w; pr[y] = x; getdis(y, k); }
    for (auto [y, w] : G[x]) if (!vis[y])
        build(y, k + 1, sz[y], x); }
void modify(int x) {
    int t = col[x] ? -1 : 1; cnt_all[x] += t;
    for (int u = pr[x], k = depth[x] - 1; u; u = pr[u],k--){
        sum_all[u] += t * d[x][k]; cnt_all[u] += t;
        sum[rt[x][k]][k] += t*d[x][k]; cnt[rt[x][k]][k] += t;
    } col[x] ^= true; }
int query(int x) { int ans = sum_all[x];
    for (int u = pr[x], k = depth[x] - 1; u; u = pr[u], k--)
        ans += sum_all[u] - sum[rt[x][k]][k]
            + d[x][k] * (cnt_all[u] - cnt[rt[x][k]][k]);
    return ans; }
```

3.3 KD 树

```
struct Node {
    int d[2], ma[2], mi[2], siz;
    Node *l, *r;
    void upd() {
        ma[0] = mi[0] = d[0]; ma[1] = mi[1] = d[1]; siz = 1;
        if (l) {
            Max(ma[0], l->ma[0]); Max(ma[1], l->ma[1]);
            Min(mi[0], l->mi[0]); Min(mi[1], l->mi[1]);
            siz += l->siz; }
        if (r) {
            Max(ma[0], r->ma[0]); Max(ma[1], r->ma[1]);
            Min(mi[0], r->mi[0]); Min(mi[1], r->mi[1]);
```

```
      siz += r->siz; } }
} mem[N], *ptr = mem, *rt;
int n, m, ans;
Node *tmp[N]; int top, D, Q[2];
Node *newNode(int x = Q[0], int y = Q[1]) {
  ptr->d[0] = ptr->ma[0] = ptr->mi[0] = x;
  ptr->d[1] = ptr->ma[1] = ptr->mi[1] = y;
  ptr->l = ptr->r = NULL; ptr->siz = 1;
  return ptr++; }
bool cmp(const Node* a, const Node* b) {
  return a->d[D] < b->d[D] || (a->d[D] == b->d[D] && a->d[!D] < b->d[!D]);}
Node *build(int l, int r, int d = 0) {
  int mid = (l + r) / 2; // chk if negative
  D = d; nth_element(tmp + l, tmp + mid, tmp + r + 1, cmp);
  Node *x = tmp[mid];
  if (l < mid) x->l = build(l, mid - 1, !d); else x->l = NULL;
  if (r > mid) x->r = build(mid + 1, r, !d); else x->r = NULL;
  x->upd(); return x; }
int dis(const Node *x) {
  return (abs(x->d[0] - Q[0]) + abs(x->d[1] - Q[1]));}
int g(Node *x) {
  return x ? (max(max(Q[0] - x->ma[0], x->mi[0] - Q[0]), 0) + max(max(Q[1] - x->ma[1], x->mi[1] - Q[1]), 0)) : INF;}
void dfs(Node *x) {
  if (x->l) dfs(x->l);
  tmp[++top] = x;
  if (x->r) dfs(x->r); }
Node *insert(Node *x, int d = 0) {
  if (!x) return newNode();
  if (x->d[d] > Q[d]) x->l = insert(x->l, !d);
  else x->r = insert(x->r, !d);
  x->upd();
  return x; }
Node *chk(Node *x, int d = 0) {
  if (!x) return 0;
  if (max(x->l ? x->l->siz : 0, x->r ? x->r->siz : 0) * 5 > x->siz * 4) {
    return top = 0, dfs(x), build(1, top, d); }
  if (x->d[d] > Q[d]) x->l = chk(x->l, !d);
  else if (x->d[0] == Q[0] && x->d[1] == Q[1]) return x;
  else x->r = chk(x->r, !d);
  return x; }
void query(Node *x) {
  if (!x) return;
  ans = min(ans, dis(x));
  int dl = g(x->l), dr = g(x->r);
  if (dl < dr) {
    if (dl < ans) query(x->l);
    if (dr < ans) query(x->r); }
  else {
    if (dr < ans) query(x->r);
    if (dl < ans) query(x->l); } }
int main() {
  read(n); read(m);
  for (int i = 1, x, y; i <= n; i++) read(x), read(y), tmp[i] = newNode(x, y);
  rt = build(1, n);
  for (int i = 1, op; i <= m; i++) {
    read(op); read(Q[0]); read(Q[1]);
    if (op == 1) rt = insert(rt), rt = chk(rt);
    else ans = INF, query(rt), printf("%d\n", ans); } }
```

3.4 LCT 动态树

```
// 记得初始化 mn
// 维护虚子树：access link cut pushup
#define lch(x) ch[x][0]
#define rch(x) ch[x][1]
int fa[MX], ch[MX][2], w[MX], mn[MX], mark[MX];
int get(int x) {return x == ch[fa[x]][1];}
int nrt(int x) {return get(x) || x == ch[fa[x]][0];}
void pushup(int x) {
  mn[x] = w[x];
  if (lch(x)) mn[x] = min(mn[x], mn[lch(x)]);
  if (rch(x)) mn[x] = min(mn[x], mn[rch(x)]); }
void rev(int x) {mark[x] ^= 1, swap(lch(x), rch(x));}
void pushdown(int x) {
  if (mark[x]) {
    if (lch(x)) rev(lch(x));
    if (rch(x)) rev(rch(x));
    mark[x] = false; } }
void rot(int x) {
  int f = fa[x], gf = fa[f];
  int which = get(x), W = ch[x][!which];
  if (nrt(f)) ch[gf][ch[gf][1] == f] = x;
  ch[x][!which] = f, ch[f][which] = W;
  if (W) fa[W] = f;
  fa[f] = x, fa[x] = gf;
  pushup(f); }
void splay(int x) {
  static int stk[MX];
  int f = x, dep = 0; stk[++dep] = f;
  while (nrt(f)) stk[++dep] = f = fa[f];
  while (dep) pushdown(stk[dep--]);
  while (nrt(x)) {
    if (nrt(f = fa[x])) rot(get(x) == get(f) ? f : x);
    rot(x);
  } pushup(x); }
void access(int x) {
  for(int y = 0 ; x ; x = fa[y = x])
    splay(x), rch(x) = y, pushup(x);
}
void makeroot(int x) {access(x), splay(x), rev(x);}
void split(int x, int y) {makeroot(x), access(y), splay(y);}
int findroot(int x) {
  access(x), splay(x);
  while (lch(x)) pushdown(x), x = lch(x);
  return splay(x), x;
}
void link(int x, int y) {
  makeroot(x);
  if (findroot(y) != x) fa[x] = y;
}
void cut(int x, int y) {
  makeroot(x);
  if (findroot(y) != x || fa[y] != x || lch(y)) return;
  rch(x) = fa[y] = 0, pushup(x);
}
```

3.5 可持久化平衡树

```
int Copy(int x){// 可持久化
  id++;sz[id]=sz[x];L[id]=L[x];R[id]=R[x];
  v[id]=v[x];return id;
}int merge(int x,int y){
  // 合并 x 和 y 两颗子树，可持久化到 z 中
  if(!x||!y)return x+y;int z;
  int o=rand()%(sz[x]+sz[y]);// 注意 rand 上限
  if(o<sz[x])z=Copy(y),L[z]=merge(x,L[y]);
  else z=Copy(x),R[z]=merge(R[x],y);
  ps(z);return z;
}void split(int x,int&y,int&z,int k){
  // 将 x 分成 y 和 z 两颗子树，y 的大小为 k
  y=z=0;if(!x)return;
  if(sz[L[x]]>=k)z=Copy(x),split(L[x],y,L[z],k),ps(z);
  else y=Copy(x),split(R[x],R[y],z,k-sz[L[x]]-1),ps(y);}
```

3.6 有旋 Treap

```
struct node { int key, size, p; node *ch[2];
  node(int key = 0) : key(key), size(1), p(rand()) {}
  void update(){size = ch[0] -> size + ch[1] -> size + 1;}
} null[maxn], *root = null, *ptr = null;
node *newnode(int x) { *++ptr = node(x);
  ptr -> ch[0] = ptr -> ch[1] = null; return ptr; }
void rot(node *&x, int d) { node *y = x -> ch[d ^ 1];
  x -> ch[d ^ 1] = y -> ch[d]; y -> ch[d] = x;
  x -> update(); (x = y) -> update(); }
void insert(int x, node *&o) {
  if (o == null) { o = newnode(x); return; }
  int d = x > o -> key; insert(x, o -> ch[d]); o->update();
  if (o -> ch[d] -> p < o -> p) rot(o, d ^ 1); }
void erase(int x, node *&o) {
  if (x == o -> key) {
    if (o -> ch[0] != null && o -> ch[1] != null) {
      int d = o -> ch[0] -> p < o -> ch[1] -> p;
      rot(o, d); erase(x, o -> ch[d]); }
    else o = o -> ch[o -> ch[0] == null]; }
```

```
  else erase(x, o -> ch[x > o -> key]);
  if (o != null) o -> update(); }
int rank(int x, node *o) {
  int ans = 1, d; while (o != null) {
    if ((d = x > o->key)) ans += o -> ch[0] -> size + 1;
    o = o -> ch[d]; } return ans; }
node *kth(int x, node *o) {
  int d; while (o != null) {
    if (x == o -> ch[0] -> size + 1) return o;
    if ((d = x > o -> ch[0] -> size))
      x -= o -> ch[0] -> size + 1;
    o = o -> ch[d]; } return o; }
node *pred(int x, node *o) {
  node *y = null; int d; while (o != null) {
    if ((d = x > o -> key)) y = o;
    o = o -> ch[d]; } return y; }
int main() { // null -> ch[0] = null -> ch[1] = null;
  null -> size = 0; return 0; }
```

4. String

4.1 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); }
```

4.2 Manacher

```
// n为串长，回文半径输出到p数组中，数组要开串长的两倍
void manacher(const char *t, int n) {
  static char s[maxn * 2];
  for (int i = n; i; i--) s[i * 2] = t[i];
  for (int i = 0; i <= n; i++) s[i * 2 + 1] = '#';
  s[0] = '$'; s[(n + 1) * 2] = '\0'; n = n * 2 + 1;
  int mx = 0, j = 0;
  for (int i = 1; i <= n; i++) {
    p[i] = (mx > i ? min(p[j * 2 - i], mx - i) : 1);
    while (s[i - p[i]] == s[i + p[i]]) p[i]++;
    if (i + p[i] > mx) { mx = i + p[i]; j = i; } } }
```

4.3 Multiple Hash

```
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; } };
```

4.4 KMP, exKMP

```
void kmp(const int *s, int n) {
  fail[0] = fail[1] = 0;
  for (int i = 1; i < n; i++) { int j = fail[i];
    while (j && s[i + 1] != s[j + 1]) j = fail[j];
    if (s[i + 1] == s[j + 1]) fail[i + 1] = j + 1;
    else fail[i + 1] = 0; } }
void exkmp(const char *s, int *a, int n) { // 0-based
  int l = 0, r = 0; a[0] = n;
  for (int i = 1; i <= n; i++) {
    a[i] = i > r ? 0 : min(r - i + 1, a[i - l]);
    while (i+a[i] < n && s[a[i]] == s[i+a[i]]) a[i]++;
    if (i + a[i] - 1 > r) {l = i; r = i + a[i] - 1;}}}
```

4.5 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]; } } }
```

4.6 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 = s1s2sk，其中 si 是Lyndon串，si ≥ si+1. 且这种分解方法是唯一的.
void mnsuf(char *s, int *mn, int n) { // 每个前缀的最小后缀
  for (int i = 0; i < n; ) {
    int j = i, k = i + 1; mn[i] = i;
    for (; k < n && s[j] <= s[k]; ++ k)
      if (s[j] == s[k]) mn[k] = mn[j] + k - j, ++j;
      else mn[k] = j = i;
    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) {} } }
```

4.7 后缀自动机

```
int last, val[maxn], par[maxn], go[maxn][26], sam_cnt;
void extend(int c) { // 结点数要开成串长的两倍
  int p = last, np = ++sam_cnt; val[np] = val[p] + 1;
  while (p && !go[p][c]) { go[p][c] = np; p = par[p]; }
  if (!p) par[np] = 1; else { int q = go[p][c];
    if (val[q] == val[p] + 1) par[np] = q;
    else { int nq = ++sam_cnt; val[nq] = val[p] + 1;
      memcpy(go[nq], go[q], sizeof(go[q]));
      par[nq] = par[q]; par[np] = par[q] = nq;
      while (p && go[p][c] == q) { go[p][c] = nq;
        p = par[p]; } } } last = np; }
int c[maxn], q[maxn];
void init() { last = sam_cnt = 1; }
void solve() {
  for (int i = 1; i <= sam_cnt; i++) c[val[i] + 1]++;
  for (int i = 1; i <= n; i++) c[i] += c[i - 1];
  for (int i = 1; i <= sam_cnt; i++) q[++c[val[i]]] = i;}
```

4.8 SAMSA & 后缀树

```
bool vis[maxn * 2]; char s[maxn];
int id[maxn * 2], ch[maxn * 2][26], height[maxn], tim = 0;
void dfs(int x) {
  if (id[x]) { height[tim++] = val[last];
    sa[tim] = id[x]; last = x; }
    for (int c = 0; c < 26; c++)
    if (ch[x][c]) dfs(ch[x][c]);
  last = par[x]; }
int main() { last = ++cnt; scanf("%s", s + 1);
    int n = strlen(s + 1); for (int i = n; i; i--) {
    expand(s[i] - 'a'); id[last] = i; }
  vis[1] = true; for (int i = 1; i <= cnt; i++) if (id[i])
      for (int x = i,pos = n; x && !vis[x]; x = par[x]){
        vis[x] = true; pos -= val[x] - val[par[x]];
        ch[par[x]][s[pos + 1] - 'a'] = x; }
  dfs(1); for (int i = 1; i <= n; i++)
    printf("%d%c", sa[i], i < n ? ' ' : '\n');
  for (int i = 1; i < n; i++) printf("%d%c", height[i],
      i < n ? ' ' : '\n'); return 0; }
```

4.9 后缀数组

```
// height[i] = lcp(sa[i], sa[i - 1])
void get_sa(char *s, int n, int *sa,
    int *rnk, int *height) { // 1-based
  static int buc[N], id[N], p[N], t[N * 2];
  int m = 300;
  for (int i = 1; i <= n; i++) buc[rnk[i] = s[i]]++;
  for (int i = 1; i <= m; i++) buc[i] += buc[i - 1];
  for (int i = n; i; i--) sa[buc[rnk[i]]--] = i;
  memset(buc, 0, sizeof(int) * (m + 1));
  for (int k = 1, cnt = 0; cnt != n; k *= 2, m = cnt) {
    cnt = 0;
    for (int i = n; i > n - k; i--) id[++cnt] = i;
    for (int i = 1; i <= n; i++)
      if (sa[i] > k) id[++cnt] = sa[i] - k;
    for (int i = 1; i <= n; i++) buc[p[i]=rnk[id[i]]]++;
    for (int i = 1; i <= m; i++) buc[i] += buc[i - 1];
    for (int i = n; i; i--) sa[buc[p[i]]--] = id[i];
    memset(buc, 0, sizeof(int) * (m + 1));
    memcpy(t, rnk, sizeof(int) * (max(n, m) + 1));
    cnt = 0; for (int i = 1; i <= n; i++) {
      if (t[sa[i]] != t[sa[i - 1]] ||
        t[sa[i] + k] != t[sa[i - 1] + k]) cnt++;
      rnk[sa[i]] = cnt; } }
  for (int i = 1; i <= n; i++) sa[rnk[i]] = i;
  for (int i = 1, k = 0; i <= n; i++) { if (k) k--;
    while (s[i + k] == s[sa[rnk[i] - 1] + k]) k++;
    height[rnk[i]] = k; } }
char s[N]; int sa[N], rnk[N], height[N];
int main() { cin >> (s + 1); int n = strlen(s + 1);
  get_sa(s, n, sa, rnk, height); }
```

4.10 Suffix Balanced Tree 后缀平衡树

```
// 后缀平衡树每次在字符串开头添加或删除字符，考虑在当前字符串 S 前插入一个字符 c，那么相当于在后缀平衡树中插入一个新的后缀 cS，简单的话可以使用预处理哈希二分 LCP 判断两个后缀的大小作 cmp，直接写 set，时间复杂度 O(nlg^2n). 为了方便可以把字符串反过来做
// 例题 : 加一个字符或删一个字符，同时询问不同子串个数
struct cmp{
  bool operator()(int a,int b){
    int p=lcp(a,b);//注意这里是后面加，lcp是反过来的
    if(a==p)return 0;if(b==p)return 1;
    return s[a-p]<s[b-p];}
};set<int,cmp>S;set<int,cmp>::iterator il,ir;
void del(){S.erase(L--);}//在后面删字符
void add(char ch){//在后面加字符
  s[++L]=ch;mx=0;il=ir=S.lower_bound(L);
  if(il!=S.begin())mx=max(mx,lcp(L,*--il));
  if(ir!=S.end())mx=max(mx,lcp(L,*ir));
  an[L]=an[L-1]+L-mx;S.insert(L); }
LL getan(){printf("%lld\n",an[L]);}//询问不同子串个数
```

4.11 广义 SAM 在线 BFS (zjj)

```
struct SAM{
int tot,fail[MM],len[MM],t[MM][26];
SAM(){tot=1;}
int insert(int c,int last){
  if(t[last][c]){
    int p=last,q=t[p][c];
    if(len[p]+1==len[q])return q;
    else{
      int nq=++tot;
      fail[nq]=fail[q];fail[q]=nq;
      len[nq]=len[p]+1;memcpy(t[nq],t[q],sizeof(t[q]));
      for(;p && t[p][c]==q;p=fail[p])t[p][c]=nq;
      //可以直接复制下面的代码。
      return nq;
    }
  }
  int p=last,np=++tot;
  len[np]=len[p]+1;
  for(;p && !t[p][c];p=fail[p])t[p][c]=np;
  if(!p)fail[np]=1;
  else{
    int q=t[p][c];
    if(len[q]==len[p]+1)fail[np]=q;
    else{
      int nq=++tot;
      fail[nq]=fail[q];fail[q]=nq;
      len[nq]=len[p]+1;memcpy(t[nq],t[q],sizeof(t[q]));
      for(;p && t[p][c]==q;p=fail[p])t[p][c]=nq;
      fail[np]=nq;
    }
  }
  return np;
}
}sam;
// scanf("%s",st+1);int slen=strlen(st+1);
// int last=1;
// for(int j=1;j<=slen;j++)last=sam.insert(st[j]-'a',last);
```

4.12 回文树

```
int len[N], par[N], go[N][26], last, cnt;
char s[N];
void extend(int n) { int p = last, c = s[n] - 'a';
  while (s[n - len[p] - 1] != s[n]) p = par[p];
  if (!go[p][c]) { int q = ++cnt, now = p;
    len[q] = len[p] + 2;
    do p = par[p];
    while (s[n - len[p] - 1] != s[n]);
    par[q] = go[p][c]; last = go[now][c] = q;
  } else last = go[p][c]; }
int main() { par[0] = cnt = 1; len[1] = -1; }
```

4.13 双端插入回文树 (zjj)

```
int t[N][26],fail[N],len[N];
int tot,lastl,lastr;
int ss[NN],L,R;
// slen 表示前端能够插入的最多字符数量
void init(int slen){
  L=slen+1;R=slen;
  tot=lastl=lastr=1;
   len[1]=-1;fail[0]=1; }
int Lfail(int pos,int x) {
  while(pos+len[x]+1>R || ss[pos+len[x]+1]!=ss[pos])x=fail[x];
  return x; }
int Rfail(int pos,int x) {
  while(pos-len[x]-1<L || ss[pos-len[x]-1]!=ss[pos])x=fail[x];
  return x; }
void add(int c,int type) {
  int x,pos;
  if(!type)ss[pos=--L]=c,x=Lfail(pos,lastl);
  else ss[pos=++R]=c,x=Rfail(pos,lastr);
  if(!t[x][c]){
    tot++;
    len[tot]=len[x]+2;
    if(!type)fail[tot]=t[Lfail(pos,fail[x])][c];
    else fail[tot]=t[Rfail(pos,fail[x])][c];
    t[x][c]=tot;
  }
  if(!type){
    lastl=t[x][c];
    if(len[lastl]==R-L+1)lastr=lastl;
  } else {
    lastr=t[x][c];
    if(len[lastr]==R-L+1)lastl=lastr; } }
```

4.14 String Conclusions

4.14.1 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.

4.14.2 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 $p + q \le |S| + gcd(p, q)$, 则 $gcd(p, q)$ 也是一个周期.

4.14.3 字符串匹配与Border

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

4.14.4 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.

4.14.5 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.

4.14.6 子串最小后缀

设 $s[p..n]$ 是 $s[i..n]$, $(l \le i \le r)$ 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)}, $(2^k < rl + 1 \le 2^{k+1})$.

4.14.7 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.

5. Math 数学

5.1 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
```

5.2 exgcd, 逆元

假设我们已经找到了一组解 (p_0, q_0) 满足 $ap_0 + bq_0 = \gcd(a, b)$, 那么其他的解都满足

$$p = p_0 + \frac{b}{\gcd(p, q)} \times t \qquad q = q_0 - \frac{a}{\gcd(p, q)} \times t$$

其中t为任意整数.

```
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; }
```

递推逆元: $\mathrm{inv}(i) \equiv (P - P/i) \cdot \mathrm{inv}(P \bmod i)$

5.3 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
```

5.4 Miller Rabin, Pollard Rho

```
mt19937 rng(123);
#define rand() LL(rng() & LLONG_MAX)
const int BASE[] = {2, 7, 61};//int(7,3e9)
//{2,325,9375,28178,450775,9780504,1795265022}LL(37)
struct miller_rabin {
bool check (const LL &M, const LL &base) {
    LL a = M - 1;
    while (~a & 1) a >>= 1;
    LL w = power (base, a, M); // power should use mul
    for (; a != M - 1 && w != 1 && w != M - 1; a <<= 1)
        w = mul (w, w, M);
    return w == M - 1 || (a & 1) == 1; }
bool solve (const LL &a) {//O((3 or 7)·log n·mul)
    if (a < 4) return a > 1;
    if (~a & 1) return false;
    for (int i = 0; i < sizeof(BASE)/4 && BASE[i] < a; ++i)
        if (!check (a, BASE[i])) return false;
    return true; } };
miller_rabin is_prime;
LL get_factor (LL a, LL seed) {//O(n^{1/4}·log n·mul)
    LL x = rand () % (a - 1) + 1, y = x;
    for (int head = 1, tail = 2; ; ) {
        x = mul (x, x, a); x = (x + seed) % a;
        if (x == y) return a;
        LL ans = gcd (abs (x - y), a);
        if (ans > 1 && ans < a) return ans;
        if (++head == tail) { y = x; tail <<= 1; } } }
void factor (LL a, vector<LL> &d) {
    if (a <= 1) return;
    if (is_prime.solve (a)) d.push_back (a);
    else {
        LL f = a;
        for (; f >= a; f = get_factor (a, rand() % (a - 1) + 1));
        factor (a / f, d);
        factor (f, d); } }
```

5.5 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), 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);}
```

5.6 阶乘取模

```
// n! mod p^q Time : O(pq^2 (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}; }
```

5.7 类欧几里得直线下格点统计

```
// \sum_{i=0}^{n-1} \lfloor (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); }
```

5.8 万能欧几里德

```
Val work(LL P, LL R, LL Q, LL n, Val VU, Val VR) {
//(Px+R)/Q, 1<=x<=i, 经过整点先U再R
  if(!(((__int128)n * P + R) / Q)) return ksm(VR, n);
  if(P>=Q) return work(P%Q,R,Q,n, VU, ksm(VU, P/Q) * VR);
  Val res; swap(VU,VR);
  res = ksm(VU, (Q-R-1)/P)*VR;
  LL m = ((__int128)n * P + R) / Q;
  res = res * work(Q, (Q-R-1)%P, P, m-1, VU, VR);
  return res * ksm(VU, n - ((__int128)m*Q - R - 1) / P); }
```

5.9 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; }
```

5.10 线性同余不等式

```
// Find the minimal non-negtive solutions for l ≤ d · x mod m ≤ r
// 0 ≤ d, l, r < m; l ≤ 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; }
```

5.11 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 $k|\delta_m(a)$.

5.12 FFT

```
const double pi = acos((double)-1.);
int fft_n; // 如果调用次数很多就改成分层预处理, cache友好
void FFT_init(int n) { fft_n = n;
  for (int i = 0; i < n; i++)
  | omega[i] = complex<double><double>(cos(2*pi/n*i), sin(2*pi/n*i));
  omega_inv[0] = omega[0];
  for (int i = 1; i < n; i++)omega_inv[i] = omega[n - i];}
void FFT(complex<double> *a, int n, int tp) {
  for (int i = 1, j = 0; i < n - 1; i++) { int k = n;
  | do j ^= (k >>= 1); while (j < k);
  | if (i < j) swap(a[i], a[j]); }
  for (int k = 2, m = fft_n / 2; k <= n; k *= 2, m /= 2)
  | for (int i = 0; i < n; i += k)
  | | for (int j = 0; j < k / 2; j++) {
  | | | auto u = a[i + j], v = (tp > 0 ? omega : omega_inv)[m * j] * a[i + j + k / 2];
  | | | a[i + j] = u + v; a[i + j + k / 2] = u - v;}
  if (tp < 0) for (int i = 0; i < n; i++) { a[i] /= n; } }
```

5.13 NTT

```
vector<int> omega[25];
void NTT_init(int n) {
  for (int k = 2, d = 0; k <= n; k *= 2, d++) {
  | omega[d].resize(k + 1);
  | int wn = qpow(3, (p - 1) / k), tmp = 1;
  | for (int i = 0; i <= k; i++) { omega[d][i] = tmp;
  | | tmp = (long long)tmp * wn % p; } } }
void NTT(int *c, int n, int tp) {
  static unsigned long long a[maxn];
  for (int i = 0; i < n; i++) a[i] = c[i];
  for (int i = 1, j = 0; i < n - 1; i++) {
  | int k = n; do j ^= (k >>= 1); while (j < k);
  | if (i < j) swap(a[i], a[j]); }
  for (int k = 1, d = 0; k < n; k *= 2, d++) {
  | if (d == 16) for (int i = 0; i < n; i++) a[i] %= p;
  | for (int i = 0; i < n; i += k * 2)
  | | for (int j = 0; j < k; j++) {
  | | | int w = omega[d][tp > 0 ? j : k * 2 - j];
  | | | unsigned long long u = a[i + j],
  | | | | v = w * a[i + j + k] % p;
  | | | a[i + j] = u + v;
  | | | a[i + j + k] = u - v + p; } }
  if (tp>0) {for (int i = 0; i < n; i++) c[i] = a[i] % p;}
  else { int inv = qpow(n, p - 2);
  | for (int i = 0; i < n; i++) c[i] = a[i] * inv % p;}}
```

5.14 MTT

```
const Complex ima = Complex(0, 1); int p, base;
void DFT(Complex *a, Complex *b, int n) {
  static Complex c[maxn];
  for (int i = 0; i < n; i++)
    c[i] = Complex(a[i].a, b[i].a);
  FFT(c, n, 1);
  for (int i = 0; i < n; i++) { int j = (n - i) & (n - 1);
    a[i] = (c[i] + c[j].conj()) * 0.5;
    b[i] = (c[i] - c[j].conj()) * -0.5 * ima; } }
void IDFT(Complex *a, Complex *b, int n) {
  static Complex c[maxn];
  for (int i = 0; i < n; i++) c[i] = a[i] + ima * b[i];
  FFT(c, n, -1);
  for (int i = 0; i < n; i++) {
    a[i].a = c[i].a;
    b[i].a = c[i].b; } }
Complex a[2][maxn], b[2][maxn], c[3][maxn]; int ans[maxn];
int main() { int n, m;
  scanf("%d%d%d", &n, &m, &p); n++; m++;
  base = (int)(sqrt(p) + 0.5);
  for (int i = 0; i < n; i++) {
    int x; scanf("%d", &x); x %= p;
    a[1][i].a = x / base; a[0][i].a = x % base; }
  for (int i = 0; i < m; i++) {
    int x; scanf("%d", &x); x %= p;
    b[1][i].a = x / base; b[0][i].a = x % base; }
  int N = 1; while (N < n + m - 1) N <<= 1; FFT_init(N);
  DFT(a[0], a[1], N); DFT(b[0], b[1], N);
  for (int i = 0; i < N; i++) c[0][i] = a[0][i] * b[0][i];
  for (int i = 0; i < N; i++)
    c[1][i] = a[0][i] * b[1][i] + a[1][i] * b[0][i];
  for (int i = 0; i < N; i++) c[2][i] = a[1][i] * b[1][i];
  FFT(c[1], N, -1); IDFT(c[0], c[2], N);
  for (int j = 2; ~j; j--)
    for (int i = 0; i < n + m - 1; i++)
      ans[i] = ((long long)ans[i] * base +
        (long long)(c[j][i].a + 0.5)) % p;
  // 实际上就是 c[2] * base ^ 2 + c[1] * base + c[0]
  return 0; }
```

5.15 多项式运算

```
void get_inv(int *a, int *c, int n) { // 求逆
  static int b[maxn];
  memset(c, 0, sizeof(int) * (n * 2));
  c[0] = qpow(a[0], p - 2);
  for (int k = 2; k <= n; k *= 2) {
    memcpy(b, a, sizeof(int) * k);
    memset(b + k, 0, sizeof(int) * k);
    NTT(b, k * 2, 1); NTT(c, k * 2, 1);
    for (int i = 0; i < k * 2; i++) {
      c[i] = (2-(long long)b[i]*c[i]) % p * c[i] % p;
      if (c[i] < 0) c[i] += p; }
    NTT(c, k * 2, -1);
    memset(c + k, 0, sizeof(int) * k); } }
void get_sqrt(int *a, int *c, int n) { // 开根
  static int b[maxn], d[maxn];
  memset(c, 0, sizeof(int) * (n * 2));
  c[0] = 1; // 如果不是1就要考虑二次剩余
  for (int k = 2; k <= n; k *= 2) {
    memcpy(b, a, sizeof(int) * k);
    memset(b + k, 0, sizeof(int) * k);
    get_inv(c, d, k);
    NTT(b, k * 2, 1); NTT(d, k * 2, 1);
    for (int i = 0; i < k * 2; i++)
      b[i] = (long long)b[i] * d[i] % p;
    NTT(b, k * 2, -1);
    for (int i = 0; i < k; i++)
      c[i] = (long long)(c[i] + b[i]) * inv_2 % p; } }
void get_derivative(int *a, int *c, int n) { // 求导
  for (int i = 1; i < n; i++)
    c[i - 1] = (long long)a[i] * i % p;
  c[n - 1] = 0; }
void get_integrate(int *a, int *c, int n) { // 不定积分
  for (int i = 1; i < n; i++)
    c[i] = (long long)a[i - 1] * qpow(i, p - 2) % p;
  c[0] = 0; } // 不定积分没有常数项
void get_ln(int *a, int *c, int n) { // 通常A常数项都是1
  static int b[maxn];
  get_derivative(a, b, n);
  memset(b + n, 0, sizeof(int) * n);
  get_inv(a, c, n);
  NTT(b, n * 2, 1); NTT(c, n * 2, 1);
  for (int i = 0;i < n * 2; i++)
    b[i] = (long long)b[i] * c[i] % p;
  NTT(b, n * 2, -1);
  get_integrate(b, c, n);
  memset(c + n, 0, sizeof(int) * n); }
// 多项式exp可以替换成分治FFT，依据：设 G(x) = exp F(x)，
// 则有 g_i = \sum_{k=1}^{i-1} f_{i-k} * k * g_k
void get_exp(int *a, int *c, int n) { // 要求A没有常数项
  static int b[maxn];
  memset(c, 0, sizeof(int) * (n * 2));
  c[0] = 1;
  for (int k = 2; k <= n; k *= 2) {
    get_ln(c, b, k);
    for (int i = 0; i < k; i++) {
      b[i] = a[i] - b[i]; if (b[i] < 0) b[i] += p; }
    (++b[0]) %= p;
    NTT(b, k * 2, 1); NTT(c, k * 2, 1);
    for (int i = 0; i < k * 2; i++)
      c[i] = (long long)c[i] * b[i] % p;
    NTT(c, k * 2, -1);
    memset(c + k, 0, sizeof(int) * k); } }
void get_pow(int *a, int *c, int n, int k) {
  static int b[maxn]; // A常数项不为1时需要转化
  get_ln(a, b, n);
  for (int i = 0; i < n; i++) b[i] = (long long)b[i]*k%p;
  get_exp(b, c, n); }
// 多项式除法, a / b, 结果输出在C
// A的次数为n, B的次数为m
void get_div(int *a, int *b, int *c, int n, int m) {
  static int f[maxn], g[maxn], gi[maxn];
  if (n < m) { memset(c, 0, sizeof(int) * m); return; }
  int N = 1; while (N < (n - m + 1)) N *= 2;
  memset(f, 0, sizeof(int) * N * 2);
  memset(g, 0, sizeof(int) * N * 2);
  // memset(gi, 0, sizeof(int) * N);
  for (int i = 0; i < n - m + 1; i++)
    f[i] = a[n - i - 1];
  for (int i = 0; i < m && i < n - m + 1; i++)
    g[i] = b[m - i - 1];
  get_inv(g, gi, N);
  for (int i = n - m + 1; i < N; i++) gi[i] = 0;
  NTT(f, N * 2, 1); NTT(gi, N * 2, 1);
  for (int i = 0; i < N * 2; i++)
    f[i] = (long long)f[i] * gi[i] % p;
  NTT(f, N * 2, -1);
  for (int i = 0; i < n - m + 1; i++)
    c[i] = f[n - m - i]; }
// 多项式取模, 余数输出到c, 商输出到d
void get_mod(int *a, int *b, int *c, int *d, int n, int m){
  static int u[maxn], v[maxn];
  if (n < m) { memcpy(c, a, sizeof(int) * n);
    if (d) memset(d, 0, sizeof(int) * m); return; }
  get_div(a, b, v, n, m); // d是商, 可以选择不要
  if (d){for (int i = 0; i < n - m + 1; i++) d[i] = v[i];}
  int N = 1; while (N < n) N *= 2;
  memcpy(u, b, sizeof(int) * m);
  NTT(u, N, 1); NTT(v, N, 1);
  for (int i = 0; i < N; i++)
    u[i] = (long long)v[i] * u[i] % p;
  NTT(u, N, -1);
  for (int i = 0; i < m - 1; i++)
    c[i] = (a[i] - u[i] + p) % p;
  memset(u, 0, sizeof(int) * N);
  memset(v, 0, sizeof(int) * N); }
```

5.15.1 多点求值

```
int q[maxn], ans[maxn]; // q是要代入的各个系数, ans是求出的值
int tg[25][maxn * 2], tf[25][maxn]; // tg是预处理乘积
// tf是项数越来越少的f, tf[0] 就是原来的函数
void pretreat(int l, int r, int k) { // 多点求值预处理
  static int a[maxn], b[maxn];
  int *g = tg[k] + l * 2;
  if (r - l + 1 <= 200) { g[0] = 1; // 小范围暴力
    for (int i = l; i <= r; i++) {
      for (int j = i - l + 1; j; j--) {
        g[j] = (g[j-1] - (long long)g[j]*q[i]) % p;
        if (g[j] < 0) g[j] += p; }
      g[0] = (long long)g[0] * (p-q[i]) % p;} return;}
  int mid = (l + r) / 2;
```

```cpp
    pretreat(l, mid, k + 1); pretreat(mid + 1, r, k + 1);
    if (!k) return;
    int N = 1; while (N <= r - l + 1) N *= 2;
    int *gl = tg[k+1] + l * 2, *gr = tg[k+1] + (mid+1) * 2;
    memset(a, 0, sizeof(int) * N);
    memset(b, 0, sizeof(int) * N);
    memcpy(a, gl, sizeof(int) * (mid - l + 2));
    memcpy(b, gr, sizeof(int) * (r - mid + 1));
    NTT(a, N, 1); NTT(b, N, 1);
    for (int i = 0; i < N; i++)
        a[i] = (long long)a[i] * b[i] % p;
    NTT(a, N, -1);
    for (int i = 0; i <= r - l + 1; i++) g[i] = a[i]; }
void solve(int l, int r, int k) { // 多点求值主过程
    int *f = tf[k];
    if (r - l + 1 <= 200) {
        for (int i = l; i <= r; i++) { int x = q[i];
            for (int j = r - l; ~j; j--)
                ans[i] = ((long long)ans[i]*x + f[j]) % p; }
        return; }
    int mid = (l + r) / 2, *ff = tf[k + 1],
        *gl = tg[k+1] + l * 2, *gr = tg[k+1] + (mid+1) * 2;
    get_mod(f, gl, ff, nullptr, r - l + 1, mid - l + 2);
    solve(l, mid, k + 1);
    memset(gl, 0, sizeof(int) * (mid - l + 2));
    memset(ff, 0, sizeof(int) * (mid - l + 1));
    get_mod(f, gr, ff, nullptr, r - l + 1, r - mid + 1);
    solve(mid + 1, r, k + 1);
    memset(gr, 0, sizeof(int) * (r - mid + 1));
    memset(ff, 0, sizeof(int) * (r - mid)); }
// f < x^n, m个询问, 询问是0-based, 当然改成1-based也很简单
void get_value(int *f, int *x, int *a, int n, int m) {
    if (m <= n) m = n + 1;
    if (n < m - 1) n = m - 1; // 补零方便处理
    memcpy(tf[0], f, sizeof(int) * n);
    memcpy(q, x, sizeof(int) * m);
    pretreat(0, m - 1, 0); solve(0, m - 1, 0);
    if (a) // 如果a是nullptr, 代表不复制答案, 直接用ans数组
        memcpy(a, ans, sizeof(int) * m); }
```

5.15.2 插值

牛顿插值 实现时可以用 k 次差分替代右边的式子.

$$f(n)=\sum_{i=0}^{k}\binom{n}{i}r_i,\quad r_i=\sum_{j=0}^{i}(-1)^{i-j}\binom{i}{j}f(j).$$

拉格朗日插值

$$f(x)=\sum_i f(x_i)\prod_{j\neq i}\frac{x-x_j}{x_i-x_j}$$

5.16 线性递推

$O(k^2\log n)$

```cpp
// 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; }};
```

$O(k\log k\log n)$

```cpp
// g < x^n, f是输出答案的数组
void pow_mod(long long k, int *g, int n, int *f) {
    static int a[maxn], t[maxn];
    memset(f, 0, sizeof(int) * (n * 2));
    f[0] = a[1] = 1;
    int N = 1; while (N < n * 2 - 1) N *= 2;
    while (k) { NTT(a, N, 1);
        if (k & 1) {
            memcpy(t, f, sizeof(int) * N);
            NTT(t, N, 1);
            for (int i = 0; i < N; i++)
                t[i] = (long long)t[i] * a[i] % p;
            NTT(t, N, -1);
            get_mod(t, g, f, nullptr, n * 2 - 1, n); }
        for (int i = 0; i < N; i++)
            a[i] = (long long)a[i] * a[i] % p;
        NTT(a, N, -1);
        memcpy(t, a, sizeof(int) * (n * 2 - 1));
        get_mod(t, g, a, nullptr, n * 2 - 1, n);
        fill(a + n - 1, a + N, 0); k >>= 1; }
    memset(a, 0, sizeof(int) * (n * 2)); }
// f_n = \sum_{i=1}^{m} f_{n-i} a_i
int linear_recurrence(long long n, int m, int *f, int *a) {
    static int g[maxn], c[maxn]; // f是0~m-1项的初值
    memset(g, 0, sizeof(int) * (m * 2 + 1));
    for (int i = 0; i < m; i++) g[i] = (p - a[m - i]) % p;
    g[m] = 1; pow_mod(n, g, m + 1, c);
    int ans = 0;
    for (int i = 0; i < m; i++)
        ans = (ans + (long long)c[i] * f[i]) % p;
    return ans; }
```

5.17 Berlekamp-Massey 最小多项式

如果要求出一个次数为 k 的递推式, 则输入的数列需要至少有 $2k$ 项.

返回的内容满足 $\sum_{j=0}^{m-1}a_{i-j}c_j=0$, 并且 $c_0=1$.

如果不加最后的处理的话, 代码返回的结果会变成 $a_i=\sum_{j=0}^{m-1}c_{j-1}a_{i-j}$, 有时候这样会方便接着跑递推, 需要的话就删掉最后的处理.

```cpp
vector<int> berlekamp_massey(const vector<int> &a) {
    vector<int> v, last; // v is the answer, 0-based
    int k = -1, delta = 0;
    for (int i = 0; i < (int)a.size(); i++) {
        int tmp = 0;
        for (int j = 0; j < (int)v.size(); j++)
            tmp = (tmp + (long long)a[i - j - 1] * v[j]) % p;
        if (a[i] == tmp) continue;
        if (k < 0) {
            k = i; delta = (a[i] - tmp + p) % p;
            v = vector<int>(i + 1); continue; }
        vector<int> u = v;
        int val = (long long)(a[i] - tmp + p) *
            qpow(delta, p - 2) % p;
        if (v.size() < last.size() + i - k)
            v.resize(last.size() + i - k);
        (v[i - k - 1] += val) %= p;
        for (int j = 0; j < (int)last.size(); j++) {
            v[i - k + j] = (v[i - k + j] -
                (long long)val * last[j]) % p;
            if (v[i - k + j] < 0) v[i - k + j] += p; }
        if ((int)u.size() - i < (int)last.size() - k) {
            last = u; k = i; delta = a[i] - tmp;
            if (delta < 0) delta += p; } }
    for (auto &x : v) x = (p - x) % p;
    v.insert(v.begin(), 1); //一般是需要最小递推式的, 处理一下
    return v; }
// \forall i, \sum_{j=0}^{m} a_{i-j} v_j = 0
```

如果要求向量序列的递推式, 就把每位乘一个随机权值 (或者说是乘一个随机行向量 v^T) 变成求数列递推式即可. 如果是矩阵序列的话就随机一个行向量 u^T 和列向量 v, 然后把矩阵变成 u^TAv 的数列.

优化矩阵快速幂DP 假设 f_i 有 n 维, 先暴力求出 $f_{0\sim 2n-1}$, 然后跑Berlekamp-Massey, 最后调用快速齐次线性递推即可.

求矩阵最小多项式 矩阵 A 的最小多项式是次数最小的并且 $f(A)=0$ 的多项式 f.

实际上最小多项式就是 $\{A^i\}$ 的最小递推式, 所以直接调用Berlekamp-Massey就好了, 并且显然它的次数不超过 n.

瓶颈在于求出 A^i, 实际上我们只要处理 $A^i v$ 就行了, 每次对向量做递推.

求稀疏矩阵的行列式 如果能求出特征多项式, 则常数项乘上 $(-1)^n$ 就是行列式, 但是最小多项式不一定就是特征多项式.

把 A 乘上一个随机对角阵 B, 则 AB 的最小多项式有很大概率就是特征多项式, 最后再除掉 $\det B$ 就行了.

求稀疏矩阵的秩 设 A 是一个 $n\times m$ 的矩阵, 首先随机一个 $n\times n$ 的对角阵 P 和一个 $m\times m$ 的对角阵 Q, 然后计算 $QAPA^TQ$ 的最小多项式即可.

实际上不用计算这个矩阵, 因为求最小多项式时要用它乘一个向量, 我们依次把这几个矩阵乘到向量里就行了. 答案就是最小多项式除掉所有 x 因子后剩下的次数.

解稀疏方程组 $Ax=b$, 其中 A 是一个 $n\times n$ 的满秩稀疏矩阵, b 和 x 是 $1\times n$ 的列向量, A,b 已知, 需要解出 x.

做法: 显然 $x=A^{-1}b$. 如果我们能求出 $\{A^ib\}(i\ge 0)$ 的最小递推式 $\{r_{0\sim m-1}\}(m\le n)$, 那么就有结论

$$A^{-1}b = -\frac{1}{r_{m-1}}\sum_{i=0}^{m-2} A^i b r_{m-2-i}$$

因为 A 是稀疏矩阵, 直接按定义递推出 $b\sim A^{2n-1}b$ 即可.

```cpp
vector<int> solve_sparse_equations(const vector<tuple<int, int, int> > &A, const vector<int> &b) {
  int n = (int)b.size(); // 0-based
  vector<vector<int> > f({b});
  for (int i = 1; i < 2 * n; i++) {
    vector<int> v(n); auto &u = f.back();
    for (auto [x, y, z] : A) // [x, y, value]
      v[x] = (v[x] + (long long)u[y] * z) % p;
    f.push_back(v); }
  vector<int> w(n); mt19937 gen;
  for (auto &x : w)
    x = uniform_int_distribution<int>(1, p - 1)(gen);
  vector<int> a(2 * n);
  for (int i = 0; i < 2 * n; i++)
    for (int j = 0; j < n; j++)
      a[i] = (a[i] + (long long)f[i][j] * w[j]) % p;
  auto c = berlekamp_massey(a); int m = (int)c.size();
  vector<int> ans(n);
  for (int i = 0; i < m - 1; i++)
    for (int j = 0; j < n; j++)
      ans[j] = (ans[j] +
        (long long)c[m - 2 - i] * f[i][j]) % p;
  int inv = qpow(p - c[m - 1], p - 2);
  for (int i = 0; i < n; i++)
    ans[i] = (long long)ans[i] * inv % p;
  return ans; }
```

5.18 FWT

```cpp
/*      FWT          IFWT
And:|  1  1  |    |  1 -1  |
    |  0  1  |    |  0  1  |
Or: |  1  0  |    |  1  0  |
    |  1  1  |    | -1  1  |
Xor:|  1  1  |    | 0.5  0.5|
    |  1 -1  |    | 0.5 -0.5|
IFWT的矩阵时FWT的逆，对于任意运算 ⊕，满足FWT的矩阵需要:
C[i][j] × C[i][k] = C[i][j ⊕ k]
对于不存在FWT矩阵的运算：通过映射01变成另外一个可行的运算。
*/
const LL AND[2][2] = {{1, 1}, {0, 1}};
const LL iAND[2][2] = {{1, M-1}, {0, 1}};
const LL OR[2][2] = {{1, 0}, {1, 1}};
const LL iOR[2][2] = {{1, 0}, {M-1, 1}};
const LL XOR[2][2] = {{1, 1}, {1, M-1}};
const LL iXOR[2][2] = {{(M+1)/2, (M+1)/2}, {(M+1)/2, M-(M+1)/2}};
void FWT(LL *f, LL C[2][2], int len) {/*点积相乘*/
  for(int t = 1;t<len;t<< = 1)
    for(int l = 0;l<len;l+ = t+t)
      for(int i = 0, r = l+t;i<t;i++){
        LL x = f[l+i], y = f[r+i];
        f[l+i] = (C[0][0]*x + C[0][1]*y)%M;
        f[r+i] = (C[1][0]*x + C[1][1]*y)%M;
      }  }
```

5.19 K 进制 FWT

```cpp
// 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);
  }
}
```

5.20 Simplex 单纯形

```cpp
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();}}
// 标准型: maximize cᵀx, subject to Ax ≤ b and x ≥ 0
// 对偶型: minimize bᵀy, subject to Aᵀx ≥ c and y ≥ 0
```

5.21 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]}; }}
```

5.22 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]; }
```

5.23 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));
}};
```

6. Appendix

6.1 Formulas 公式表

6.1.1 Mobius Inversion

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

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

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

6.2 杜教筛

$$S_\varphi(n) = \frac{n(n+1)}{2} - \sum_{d=2}^{n} S_\varphi\left(\left\lfloor \frac{n}{d} \right\rfloor\right)$$

$$S_\mu(n) = 1 - \sum_{d=2}^{n} S_\mu\left(\left\lfloor \frac{n}{d} \right\rfloor\right)$$

6.2.1 降幂公式

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

6.2.2 其他常用公式

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

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

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

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

6.2.3 单位根反演

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

引理

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

反演

$$\begin{aligned} Ans &= \sum_{i=0}^{n} C_n^i [k \mid i] \\ &= \sum_{i=0}^{n} C_n^i \left(\frac{1}{k} \sum_{j=0}^{k-1} \omega_k^{ij}\right) \\ &= \frac{1}{k} \sum_{i=0}^{n} C_n^i \sum_{j=0}^{k-1} \omega_k^{ij} \\ &= \frac{1}{k} \sum_{j=0}^{k-1} \left(\sum_{i=0}^{n} C_n^i (\omega_k^j)^i\right) \\ &= \frac{1}{k} \sum_{j=0}^{k-1} (1 + \omega_k^j)^n \end{aligned}$$

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.

6.2.4 Arithmetic Function

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

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

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

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

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

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

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

$J_k(n)$ 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

6.2.5 Binomial Coefficients

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

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

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

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

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

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

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

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

6.2.6 Fibonacci Numbers, Lucas Numbers

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

除了 $n=0,4,8,16$, L_n 是素数, 则 n 是素数.

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

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

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

6.2.7 Sum of Powers

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

n 个元素集合划分的方案数.

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

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

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

6.2.12 Stirling Numbers

第一类 n 个元素集合分作 k 个**非空轮换**方案数.

$${n+1\brack k}=n{n\brack k}+{n\brack k-1}$$

$$s(n,k)=(-1)^{n-k}{n\brack k}$$

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

$${n+1\brack 2}=n!H_n \text{ (see 6.2.14)}$$

$$x^{\underline{n}}=\sum_k{n\brack k}(-1)^{n-k}x^k$$

$$x^{\overline{n}}=\sum_k{n\brack k}x^k$$

For fixed k, EGF:

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

第二类 把 n 个元素集合分作 k 个**非空子集**方案数.

$${n+1\brace k}=k{n\brace k}+{n\brace k-1}$$

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

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

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

For fixed k, EGF and OGF:

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

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

6.2.13 Eulerian Numbers

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

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

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

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

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

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

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

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

6.2.15 卡迈克尔函数

卡迈克尔函数表示模 m 剩余系下最大的阶, 即 $\lambda(m) = \max_{a \perp m} \delta_m(a)$.

容易看出, 若 $\lambda(m) = \varphi(m)$, 则 m 存在原根.

该函数可由下述方法计算: 分解质因数 $m = p_1^{\alpha_1} p_2^{\alpha_2} ... p_t^{\alpha_t}$. 则 $\lambda(m) = \text{lcm}(\lambda(p_1^{\alpha_1}), p_2^{\alpha_2}, ..., p_t^{\alpha_t})$. 其中对奇质数 p, $\lambda(p^\alpha) = (p-1)p^{\alpha-1}$. 对2的幂, $\lambda(2^k) = 2^{k-2}, s.t. k \geq 3$. $\lambda(4) = \lambda(2) = 2$.

6.2.16 求拆分数

```python
def penta(k):
  return k*(3*k-1)//2
def compute_partition(goal):
  p = [1]
  for n in range(1,goal+1):
    p.append(0)
    for k in range(1,n+1):
      c = (-1)**(k+1)
      for t in [penta(k), penta(-k)]:
        if (n-t) >= 0:
          p[n] = p[n] + c*p[n-t]
  return p
```

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

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等于 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:

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

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

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

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

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

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

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

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

自然数幂次和关于次数的EGF:

$$F(x) = \sum_{k=0}^{\infty} \frac{\sum_{i=0}^{n} i^k}{k!} x^k$$

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

6.2.18 kMAX-MIN反演

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

代入 $k = 1$ 即为MAX-MIN反演:

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

6.2.19 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 $O(n^2)$.

6.2.20 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).

6.2.21 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.

6.2.22 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)

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

6.2.23 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.

6.2.24 常见博弈游戏

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

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

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

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

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

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

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

最大最小定理求纳什均衡点 在二人零和博弈中, 可以用以下方式求出一个纳什均衡点: 在博弈双方中**任选**一方, 求混合策略 **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.

6.2.25 概率相关

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

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

m 个数的方差:

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

6.2.26 邻接矩阵行列式的意义

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

6.2.27 Others (某些近似数值公式在这里)

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

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

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

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

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

$$\sum_{i=1}^{n} \frac{1}{n} \approx \ln\left(n + \frac{1}{2}\right) + \frac{1}{24(n+0.5)^2} + \Gamma, (\Gamma \approx 0.577215664901532860606065)$$

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

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

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

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

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

$n \bmod 2 = 1$:

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

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

6.3 Calculus Table 导数表

$(\frac{u}{v})' = \frac{u'v - uv'}{v^2}$
$(a^x)' = (\ln a)a^x$
$(\tan x)' = \sec^2 x$
$(\cot x)' = \csc^2 x$
$(\sec x)' = \tan x \sec x$
$(\csc x)' = -\cot x \csc x$
$(\arcsin x)' = \frac{1}{\sqrt{1-x^2}}$
$(\arccos x)' = -\frac{1}{\sqrt{1-x^2}}$

$(\arctan x)' = \frac{1}{1+x^2}$
$(\text{arccot}\, x)' = -\frac{1}{1+x^2}$
$(\text{arccsc}\, x)' = -\frac{1}{x\sqrt{1-x^2}}$
$(\text{arcsec}\, x)' = \frac{1}{x\sqrt{1-x^2}}$
$(\tanh x)' = \text{sech}^2 x$
$(\coth x)' = -\text{csch}^2 x$
$(\text{sech}\, x)' = -\text{sech}\, x \tanh x$
$(\text{csch}\, x)' = -\text{csch}\, x \coth x$

$(\text{arcsinh}\, x)' = \frac{1}{\sqrt{1+x^2}}$
$(\text{arccosh}\, x)' = \frac{1}{\sqrt{x^2-1}}$
$(\text{arctanh}\, x)' = \frac{1}{1-x^2}$
$(\text{arccoth}\, x)' = \frac{1}{x^2-1}$
$(\text{arccsch}\, x)' = -\frac{1}{|x|\sqrt{1+x^2}}$
$(\text{arcsech}\, x)' = -\frac{1}{x\sqrt{1-x^2}}$

6.4 Integration Table 积分表

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

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

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

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

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

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

三角函数的积分

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

反三角函数的积分 (其中 $a>0$)

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

指数函数的积分

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

对数函数的积分

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

6.5 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) {
// Decimals
BigDecimal ans, S, C;
MathContext fuckJava = new MathContext(100,
  ↪ RoundingMode.HALF_EVEN);
BigDecimal minx = BigDecimal.TEN.pow(18);
C = BigDecimal.ZERO;
C.compareTo(S);
S.divide(BigDecimal.valueOf(2), fuckJava);
ans.divide(S.multiply(S), fuckJava).multiply(new BigDecimal(2));
System.out.println(new DecimalFormat("0.000000000").format(ans));
// 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) {
```

```java
  /* 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
} } }
```

6.6 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())
```

7. Miscellany

7.1 Zeller 日期公式

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

7.2 DP 优化

7.2.1 四边形不等式

```cpp
// 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;
  } } }
```

7.2.2 树形背包优化

限制: 必须取与根节点相连的一个连通块.
转化: 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 $[i, i+\mathrm{size}_i-1]$, 转移为 $f(i,v)=\max\{f(i+1,V-v_i)+w_i, f(i+\mathrm{size}_i, v)\}$.
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.

7.2.3 $O(n \cdot \max a_i)$ Subset Sum

```cpp
int SubsetSum(vector<int> &a, int t) {
  int B = *max_element(a.begin(), a.end());
  int n = (int) a.size(), s = 0, i = 0;
  while (i < n && s + a[i] <= t) s += a[i++];
  vector<int> f(2*B+1, -1), pre(B+1, -1);
  f[s-(t-B)] = i;
  for ( ; i < n; i++) {
    s += a[i];
    for (int d = 0; d <= B; d++) pre[d] = max(0, f[d+B]);
    for (int d = B; d >= 0; d--)
      f[d+a[i]] = max(f[d+a[i]], f[d]);
    for (int d = 2*B; d > B; --d)
      for (int j = pre[d-B]; j < f[d]; j++)
        f[d-a[j]] = max(f[d-a[j]], j); }
  for (i = 0; i <= B; i++)
    if (f[B-i] >= 0) return max(t-i, s-(t-i)); }
```

7.3 Hash Table

```cpp
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; } };
```

7.4 基数排序

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

7.5 Hacks: O3, 读入优化, Bitset, builtin

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

7.6 试机赛与纪律文件

- 检查身份证件: 护照、学生证、胸牌以及现场所需通行证。
- 确认什么东西能带进场。特别注意: 智能手表、金属（钥匙）等等。
- 测试鼠标、键盘、显示器和座椅。如果有问题, 立刻联系工作人员。
- 确认比赛前能动什么, 不能动什么, 能存储什么配置文件。
- 测试本地栈大小: 如果 `ulimit -a` 是 unlimited, 那么在 bashrc 里加上 `ulimit -s 65536; ulimit -m 1048576`, 否则死递归会死机。
- 测试比赛提交方式。如果有 `submit` 命令, 确认如何使用。讨论是否应该不用以避免 `submit > a.cpp`。
- 如果可以的话, 设置定期备份文件。
- 测试 OJ 栈大小。如果不合常理, 发 clar 问一下。
- 测试编译器版本。C++20 不能用 `cin >> (s + 1)`; C++17 `auto [x, y]: a`; C++14 `[](auto x, auto y)`; C++11 `auto`; `bits/stdc++.h`; `pb_ds`。

```cpp
#include <ext/rope>
using namespace __gnu_cxx;
rope <int> R; R.insert(y, x); R[x]; R.erase(x, 1);
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
tree <int, null_type, less<int>, rb_tree_tag,
tree_order_statistics_node_update> s;
s.insert(1); s.find_by_order(0); s.order_of_key(5);
```

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

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

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

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

7.7 Constant Table 常数表

```
Random primes generated at Mon Nov 6 19:09:06 2023
5e2 367 373 401 439 449 461 467 487 491 499 509 521 563 587 631
1e3 857 859 881 911 937 953 971 977 1013 1033 1039 1049 1097 1103
3e4 28031 28163 28277 29437 29921 30467 31247 31327 31477 31687
1e5 93887 94207 96013 96443 96749 99559 100363 105691 106207
2e5 192749 195329 198719 201281 202679 205967 209701 211639
5e5 479593 499673 504617 509281 514177 514847 523333 526997
1e6 939613 948443 960569 970447 1025147 1045571 1056659 1068619
2e6 1906603 1922027 1942153 2065103 2078191 2134261 2134709
5e6 4678783 4971269 5097017 5166787 5170999 5194883 5205149
1e7 9441599 9460909 9462601 9526229 9839257 10142317 10389283
2e7 20606143 20875531 21170881 21197851 21236101 21301723
1e9 930113963 953503211 1018330333 1031492479 1063007761
NTT 976224257 r=3 (20) 985661441 r=3 (22) 998244353 r=3 (23)
1004535809 r=3 (21) 1007681537 r=3 (20) 1012924417 r=5 (21)
1045430273 r=3 (20) 1051721729 r=6 (20) 1053818881 r=7 (20)
```

n	$\log_{10} n$	$n!$	$C(n, n/2)$	$\mathrm{LCM}(1 \ldots n)$	P_n
2	0.30102999	2	2	2	2
3	0.47712125	6	3	6	3
4	0.60205999	24	6	12	5
5	0.69897000	120	10	60	7
6	0.77815125	720	20	60	11
7	0.84509804	5040	35	420	15
8	0.90308998	40320	70	840	22
9	0.95424251	362880	126	2520	30
10	1	3628800	252	2520	42
11	1.04139269	39916800	462	27720	56
12	1.07918125	479001600	924	27720	77
15	1.17609126	1.31e12	6435	360360	176
20	1.30103000	2.43e18	184756	232792560	627
25	1.39794001	1.55e25	5200300	26771144400	1958
30	1.47712125	2.65e32	155117520	1.444e14	5604
P_n	3733_{40}	204226_{50}	966467_{60}	190569292_{100}	$1e9_{114}$

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

Vimrc, Bashrc

```
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'
```

Delaunay 三角剖分

```cpp
/* Delaunay Triangulation 随机增量算法 :
节点数至少为点数的 6 倍, 空间消耗较大注意计算内存使用
建图的过程在 build 中, 注意初始化内存池和初始三角形 (LOTS)
Triangulation::find 返回包含某点的三角形
Triangulation::add_point 将某点加入三角剖分
某个 Tri 在三角剖分中当且仅当它的 has_ch 为 0
如果要找到三角形 u 的邻域, 则枚举它的所有 u.edge[i].tri, 该条边的两个点为 u.p[(i+1)%3], u.p[(i+2)%3] */
const int N = 100000 + 5, MAX_TRIS = N * 6;
bool in_circum(cp p1, cp p2, cp p3, cp p4) {
 LD u11 = p1.x-p4.x, u21 = p2.x-p4.x, u31 = p3.x-p4.x;
 LD u12 = p1.y-p4.y, u22 = p2.y-p4.y, u32 = p3.y-p4.y;
 LD u13 = sqr(p1.x)-sqr(p4.x) + sqr(p1.y) - sqr(p4.y);
 LD u23 = sqr(p2.x)-sqr(p4.x) + sqr(p2.y) - sqr(p4.y);
 LD u33 = sqr(p3.x)-sqr(p4.x) + sqr(p3.y) - sqr(p4.y);
 LD det = -u13*u22*u31 + u12*u23*u31 + u13*u21*u32 - u11*u23*u32 - u12*u21*u33 + u11*u22*u33;
 return det > eps; }
LD dir(cp a, cp b, cp p) { return det(b-a,p-a);}
typedef int SideRef; struct Tri; typedef Tri* TriRef;
struct Edge {
  TriRef tri; SideRef side; Edge() : tri(0), side(0) {}
  Edge(TriRef tri, SideRef side) : tri(tri), side(side) {} };
struct Tri { // Triangle
  point p[3];Edge edge[3];TriRef ch[3]; Tri(){}
  Tri(cp p0,cp p1,cp p2){
    p[0] = p0; p[1] = p1; p[2] = p2;
        ch[0] = ch[1] = ch[2] = 0; }
  bool has_ch() const { return ch[0] != 0; }
  int num_ch() const {
    return ch[0] == 0 ? 0
      : ch[1] == 0 ? 1
      : ch[2] == 0 ? 2 : 3; }
  bool contains(cp q) const {
    LD a=dir(p[0],p[1],q), b=dir(p[1],p[2],q), c=dir(p[2],p[0],q);
    return sgn(a) >= 0 && sgn(b) >= 0 && sgn(c) >= 0; }
} triange_pool[MAX_TRIS], *tot_tri;
void set_edge(Edge a, Edge b) {
  if (a.tri) a.tri->edge[a.side] = b;
  if (b.tri) b.tri->edge[b.side] = a; }
class Triangulation {
  public:
    Triangulation() {
      const LD LOTS = 1e6; // NOTE: below base triangle
      the_root = new(tot_tri++) Tri (point(-LOTS,-LOTS), point(+LOTS,-LOTS), point(0,+LOTS)); }
    TriRef find(point p) const { return find(the_root,p); }
    void add_point(cp p) { add_point(find(the_root,p),p); }
  private:
    TriRef the_root;
    static TriRef find(TriRef root,cp p){
      for( ; ; ) {
        if (!root->has_ch()) return root;
        else for (int i = 0; i < 3 && root->ch[i] ; ++i)
            if (root->ch[i]->contains(p))
              {root = root->ch[i]; break;} } }
    void add_point(TriRef root, cp p) {
      TriRef tab,tbc,tca;
      tab = new(tot_tri++) Tri(root->p[0], root->p[1], p);
      tbc = new(tot_tri++) Tri(root->p[1], root->p[2], p);
      tca = new(tot_tri++) Tri(root->p[2], root->p[0], p);
      set_edge(Edge(tab,0),Edge(tbc,1));
      set_edge(Edge(tbc,0),Edge(tca,1));
      set_edge(Edge(tca,0),Edge(tab,1));
      set_edge(Edge(tab,2),root->edge[2]);
      set_edge(Edge(tbc,2),root->edge[0]);
      set_edge(Edge(tca,2),root->edge[1]);
      root->ch[0]=tab;root->ch[1]=tbc;root->ch[2]=tca;
      flip(tab,2); flip(tbc,2); flip(tca,2); }
    void flip(TriRef tri, SideRef pi) {
      TriRef trj = tri->edge[pi].tri; int pj = tri->edge[pi].side;
      if(!trj || !in_circum(tri->p[0],tri->p[1],tri->p[2],trj->p[pj])) return;
      TriRef trk = new(tot_tri++) Tri(tri->p[(pi+1)%3], trj->p[pj], tri->p[pi]);
      TriRef trl = new(tot_tri++) Tri(trj->p[(pj+1)%3], tri->p[pi], trj->p[pj]);
      set_edge(Edge(trk,0), Edge(trl,0));
      set_edge(Edge(trk,1), tri->edge[(pi+2)%3]);
      set_edge(Edge(trk,2), trj->edge[(pj+1)%3]);
      set_edge(Edge(trl,1), trj->edge[(pj+2)%3]);
      set_edge(Edge(trl,2), tri->edge[(pi+1)%3]);
      tri->ch[0]=trk; tri->ch[1]=trl; tri->ch[2]=0;
      trj->ch[0]=trk; trj->ch[1]=trl; trj->ch[2]=0;
      flip(trk,1); flip(trk,2); flip(trl,1); flip(trl,2); }
};
int n; point ps[N];
void build(){
  tot_tri = triange_pool; cin >> n;
  for(int i = 0; i < n; ++ i) scanf("%lf%lf",&ps[i].x,&ps[i].y);
  random_shuffle(ps, ps + n); Triangulation tri;
  for(int i = 0; i < n; ++ i) tri.add_point(ps[i]); }
//The Euclidean minimum spanning tree of a set of points is a subset of the Delaunay triangulation of the same points.
//Connecting the centers of the circumcircles produces the Voronoi diagram.
//No point in P is inside the circumcircle of any triangle.
//Maximize the minimum angle of all the angles of the triangles.
```

动态凸包

```cpp
struct Upper_Hull {
struct Link {
  point p;
  Link *prev = nullptr, *next = nullptr;
  int id;
};
struct Node
{
  Link *chain, *chain_back;
  Link *tangent;
};
template<typename S, typename T>
pair<Link*, Link*> find_bridge(Link*l, Link*r, S next, T convex) {
  while (next(l) || next(r)) {
    if (!next(r) || (next(l) && convex(point{0, 0}, next(l)->p - l->p, next(r)->p - r->p))) {
      if (convex(l->p, next(l)->p, r->p)) {
        l = next(l);
      } else {
        break;
      }
    } else {
      if (!convex(l->p, r->p, next(r)->p)) {
        r = next(r);
      } else {
        break;
      }
    }
  }
  return {l, r};
}
template<bool rev = false>
void fix_chain(int u, Link*l, Link*r) {
  if (rev) { // l and r to the right of actual bridge
    tie(r, l) = find_bridge(r, l,
    [](Link * x) { return x->prev; },
    [](cp a, cp b, cp c) {
      return ccw(a, b, c) >= 0;
    });
  } else { // l and r to the left of actual bridge
    tie(l, r) = find_bridge(l, r,
    [](Link * x) { return x->next; },
    [](cp a, cp b, cp c) {
      return ccw(a, b, c) <= 0;
    });
  }
  tree[u].tangent = l;
  tree[u].chain = tree[2 * u].chain;
  tree[u].chain_back = tree[2 * u + 1].chain_back;
  tree[2 * u].chain = l->next;
  tree[2 * u + 1].chain_back = r->prev;
  if (l->next) {
    l->next->prev = nullptr;
  } else {
    tree[2 * u].chain_back = nullptr;
  }
  if (r->prev) {
    r->prev->next = nullptr;
  } else {
    tree[2 * u + 1].chain = nullptr;
  }
  l->next = r;
  r->prev = l;
}
void build(int u, int a, int b) {
  if (b - a == 1) {
    tree[u].chain = tree[u].chain_back = &lists[a];
    tree[u].tangent = nullptr;
    return;
  }
  const int m = a + (b - a) / 2;
  build(2 * u, a, m);
  build(2 * u + 1, m, b);
  auto l = tree[2 * u].chain, r = tree[2 * u + 1].chain;
  fix_chain(u, l, r);
```

```cpp
}
void rob(int u, int v) {
    tree[u].chain = tree[v].chain;
    tree[v].chain = nullptr;
    tree[u].chain_back = tree[v].chain_back;
    tree[v].chain_back = nullptr;
}
void remove(int u, int a, int b, int const&i) {
    if (i < a || i >= b) return;
    // we should never hit a leaf
    assert(b - a > 1);
    const int m = a + (b - a) / 2;
    // one child -> that child contains i
    if (!tree[u].tangent) {
        int v = i < m ? 2 * u : 2 * u + 1;
        tree[v].chain = tree[u].chain;
        tree[v].chain_back = tree[u].chain_back;
        if (i < m) {
            remove(2 * u, a, m, i);
        } else {
            remove(2 * u + 1, m, b, i);
        }
        rob(u, v);
        return;
    }
    // restore hull of children
    auto l = tree[u].tangent, r = l->next;
    l->next = tree[2 * u].chain;
    if (tree[2 * u].chain) {
        tree[2 * u].chain->prev = l;
    } else {
        tree[2 * u].chain_back = l;
    }
    tree[2 * u].chain = tree[u].chain;
    r->prev = tree[2 * u + 1].chain_back;
    if (tree[2 * u + 1].chain_back) {
        tree[2 * u + 1].chain_back->next = r;
    } else {
        tree[2 * u + 1].chain = r;
    }
    tree[2 * u + 1].chain_back = tree[u].chain_back;
    // delete i
    const int v = i < m ? 2 * u : 2 * u + 1;
    // only i
    if (tree[v].chain == tree[v].chain_back && tree[v].chain->id == i) {
        tree[v].chain = tree[v].chain_back = nullptr;
        rob(u, v ^ 1);
        tree[u].tangent = nullptr;
        return;
    }
    if (i < m) {
        if (l->id == i) {
            l = l->next;
        }
        remove(2 * u, a, m, i);
        if (!l) {
            l = tree[2 * u].chain_back;
        }
        fix_chain<true>(u, l, r);
    } else {
        if (r->id == i) {
            r = r->prev;
        }
        remove(2 * u + 1, m, b, i);
        if (!r) {
            r = tree[2 * u + 1].chain;
        }
        fix_chain<false>(u, l, r);
    }
}
void remove(int i) {
    // i is the only point
    if (tree[1].chain == tree[1].chain_back) {
        tree[1].chain = tree[1].chain_back = nullptr;
        return;
    }
    remove(1, 0, n, i);
}
Upper_Hull(vector<point> const&v) : n(v.size()), tree(4 * n), lists(n) {
    assert(is_sorted(v.begin(), v.end()));
    for (int i = 0; i < n; ++i) {
        lists[i].p = v[i];
        lists[i].id = i;
    }
    build(1, 0, n);
}
vector<int> get_hull() {
    vector<int> ret;
    for (Link* u = tree[1].chain; u; u = u->next) {
        ret.push_back(u->id);
    }
    return ret;
}
vector<point> get_hull_points() {
    vector<point> ret;
    for (Link* u = tree[1].chain; u; u = u->next) {
        ret.push_back(u->p);
    }
    return ret;
}
int n;
vector<Node> tree;
vector<Link> lists;
};
int main() {
    int n;
    cin >> n;
    vector<int> layer(n), ans(n);
    vector<point>ps;
    map<point, int>id;
    for (int i = 0; i < n; i++) {
        int x, y;
        cin >> x >> y;
        ps.push_back({x, y});
        id[ {x, y}] = i;
    }
    sort(ps.begin(), ps.end());
    Upper_Hull left(ps);
    reverse(ps.begin(), ps.end());
    for (auto &p : ps)p = -p;
    Upper_Hull right(ps);
    for (auto &p : ps)p = -p;
    reverse(ps.begin(), ps.end());
    for (int l = 1, cnt = 0; cnt < n; l++) {
        set<int>hull;
        for (int i : left.get_hull())hull.insert(i);
        for (int i : right.get_hull())hull.insert(n - 1 - i);
        for (int i : hull) {
            assert(!layer[i]);
            cnt++;
            layer[i] = l;
            left.remove(i);
            right.remove(n - 1 - i);
        }
    }
    for (int i = 0; i < n; i++)
        ans[id[ps[i]]] = layer[i];
    for (int i = 0; i < n; i++)cout << ans[i] << "\n";
    return 0;
}
```

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);
```

```cpp
  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

```cpp
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(){... }
  p3 () {x = y = z = 0;}
  p3 (D xx, D yy, D zz) {x = xx; y = yy; z = zz;}
  p3 operator + (cp a){return {x + a.x, y + a.y, z + a.z};}
  p3 operator - (cp a){return {x - a.x, y - a.y, z - a.z};}
  p3 operator * (D a){return {x * a, y * a, z * a};}
  p3 operator / (D a){return {x / a, y / a, z / a};}
  D &operator [] (int a){return a == 0 ? x : (a == 1 ? y : z);}
  D len2(){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 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];
  sort(a.begin(), a.end(), [](auto u, auto v) {
    return make_pair(u.x, v.x) < make_pair(u.y, v.y);
  });
  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) return puts("NO"), 0;
  from.normalize();
  p3 c = cross(from, to);
  if (abs(c.len2()) < eps) ;
  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] = {a[0].size() - 1, 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 < mnk.size(); i++) {
    p3 u = mnk[i], v = mnk[(i + 1) % int(mnk.size())];
    if (det(p - u, v - u) > eps)
      return puts("NO"), 0;}
  puts("YES");}
```

神谕

```
// PAM + log dp
char st[N],ss[N];
namespace PAM{
int trans[N][26],len[N],fail[N];
int dif[N],slink[N];
int tot,last;
int getfail(int pos,int x){
  while(pos-len[x]-1<1 || st[pos-len[x]-1]!=st[pos])
    x=fail[x];
  return x;
}
void init(){
  tot=last=1;len[1]=-1;fail[0]=1;
}
void add(int pos){
  int x=getfail(pos,last),c=st[pos]-'a';
  if(!trans[x][c]){
    tot++;
    len[tot]=len[x]+2;
    fail[tot]=trans[getfail(pos,fail[x])][c];
    trans[x][c]=tot;
    dif[tot]=len[tot]-len[fail[tot]];
    if(dif[tot]==dif[fail[tot]])
      slink[tot]=slink[fail[tot]];
    else slink[tot]=fail[tot];
  }
  last=trans[x][c];
}
LL g[N];
}
using PAM::dif;
using PAM::slink;
using PAM::g;
using PAM::len;
using PAM::fail;
int n;LL dp[N];
int main(){
  scanf("%s",ss+1);n=strlen(ss+1);
  for(int i=1;i<=n/2;i++)
    st[i*2-1]=ss[i],st[i*2]=ss[n-i+1];
  dp[0]=1;PAM::init();
  for(int i=1;i<=n;i++){
    PAM::add(i);
    for(int x=PAM::last;x>1;x=slink[x]){
      g[x]=dp[i-len[slink[x]]-dif[x]];
      if(dif[x]==dif[fail[x]])
        g[x]=(g[x]+g[fail[x]])%mod;
      if(i%2==0)
        dp[i]=(dp[i]+g[x])%mod;
    }
  }
  printf("%lld\n",dp[n]);
  return 0;
}
// exkmp
//(s1,n,val),(s2,m,exkmp)
int now=0,p=0;exkmp[1]=m;
for(int i=2;i<=m;i++){
  if(i<=p)exkmp[i]=min(p-i+1,exkmp[i-now+1]);
  if(exkmp[i]+i-1>=p){
    while(i+exkmp[i]<=m && s2[i+exkmp[i]]==s2[exkmp[i]+1])exkmp[i]++;
    now=i;p=i+exkmp[i]-1;
  }
}
now=p=0;
for(int i=1;i<=n;i++){
  if(i<=p)val[i]=min(p-i+1,exkmp[i-now+1]);
  if(val[i]+i-1>=p){
    while(i+val[i]<=n && val[i]+1<=m && s2[val[i]+1]==s1[i+val[i]])val[i]++;
    now=i;p=i+val[i]-1;
  }
}
// ACAM
namespace ACAM{
/*如果题目给的是多个字符串而不是一个现成的Trie。
那么last树的深度至多根号。*/
int tr[N][26],last[N],fail[N],tot;bool v[N];
int ins(char *st,int len){
  int now=0;
  for(int i=1;i<=len;i++){
    int x=st[i]-'a';
    if(!tr[now][x])tr[now][x]=++tot;
    now=tr[now][x];
  }
  v[now]=1;
  return now;
}
//lis还可以作为ACAM中任意一棵树的拓扑序。
//不过这个拓扑序里面没有0号点，也就是根。
int lis[N],head,tail;
void bfs(){
  head=1;
  for(int i=0;i<26;i++){
    if(tr[0][i])lis[++tail]=tr[0][i];}
  while(head<=tail){
    int x=lis[head++];
    if(!v[fail[x]])last[x]=last[fail[x]];
    else last[x]=fail[x];
    for(int i=0;i<26;i++){
      int y=tr[x][i];
      if(!y)tr[x][i]=tr[fail[x]][i];
      else fail[y]=tr[fail[x]][i],lis[++tail]=y;
    }
  }
}
}
// 广义 SAM, 离线 BFS
struct Trie{
  int tot,fa[M],tr[M][26],c[M];
  Trie(){tot=1;}
  void insert(char *ch,int slen){
    int now=1;
    for(int i=1;i<=slen;i++){
      int x=ch[i]-'a';
      if(!tr[now][x])tr[now][x]=++tot,c[tot]=x,fa[tot]=now;
      now=tr[now][x];
    }
  }
}trie;
struct SAM{
  int tot,pos[M],fail[MM],maxlen[MM],trans[MM][26];
  queue<int> Q;
  SAM(){tot=1;}
  int insert(int c,int last){
    int p=last,np=++tot;
    maxlen[np]=maxlen[p]+1;
    for(;p && !trans[p][c];p=fail[p])trans[p][c]=np;
    if(!p)fail[np]=1;
    else{
      int q=trans[p][c];
      if(maxlen[q]==maxlen[p]+1)fail[np]=q;
      else{
        int nq=++tot;
        fail[nq]=fail[q];fail[q]=nq;
        maxlen[nq]=maxlen[p]+1;
        memcpy(trans[nq],trans[q],sizeof(trans[q]));
```

```cpp
        for(;p && trans[p][c]==q;p=fail[p])trans[p][c]=nq;
        fail[np]=nq;
      }
    }
    return np;
  }
  void build(){
    pos[1]=1;
    for(int i=0;i<26;i++)
      if(trie.tr[1][i])Q.push(trie.tr[1][i]);
    while(!Q.empty()){
      int x=Q.front();Q.pop();
      pos[x]=insert(trie.c[x],pos[trie.fa[x]]);
      for(int i=0;i<26;i++){
        if(trie.tr[x][i])Q.push(trie.tr[x][i]);
      } } } } SAM;
// Runs(Hash)
const int N=1e6+5;const LL mod=998244335,base=29;
char st[N];int n;
struct Runs{int l,r,p;};vector<Runs> run;
bool cmp(Runs x,Runs y){return x.l!=y.l?x.l<y.l:x.r<y.r;}
bool operator==(Runs x,Runs y){return x.l==y.l && x.r==y.r;}
LL fc[N],ff[N];
void init(){
  fc[0]=1;
  for(int i=1;i<=n;i++)fc[i]=fc[i-1]*base%mod;
  for(int i=1;i<=n;i++)
    ff[i]=(ff[i-1]*base+st[i]-'A'+1)%mod;}
LL gv(int l,int r){
  return ((ff[r]-ff[l-1]*fc[r-l+1])%mod+mod)%mod;}
LL gethash(int l,int r){return gv(l,r);}
int gl(int x,int y){
  int ans=0,l=1,r=min(x,y);
  while(l<=r){
    int mid=(l+r)>>1;
    if(gethash(x-mid+1,x)==gethash(y-mid+1,y))
    ans=mid,l=mid+1; else r=mid-1;
  }
  return ans;}
int gr(int x,int y){
  int ans=0,l=1,r=min(n-x+1,n-y+1);
  while(l<=r){
    int mid=(l+r)>>1;
    if(gethash(x,x+mid-1)==gethash(y,y+mid-1))
      ans=mid,l=mid+1; else r=mid-1;
  }
  return ans; }
bool getcmp(int x,int y){
  int len=gr(x,y);
  return st[x+len]<st[y+len];}
int ly[N];
void lyndon(bool type){
  stack<PII> stk;stk.push({n,n});ly[n]=n;
  for(int i=n-1;i>=1;i--){
    int now=i;
    while(!stk.empty() && getcmp(i,stk.top().first)!=type)
      now=stk.top().second,stk.pop();
    ly[i]=now;
    stk.push({i,now});
  } }
void getrun(){
  for(int l=1;l<=n;l++){
    int r=ly[l],ll=l,rr=r;
    if(l!=1)ll-=gl(l-1,r);
    if(r!=n)rr+=gr(l,r+1);
    if(rr-ll+1>=2*(r-l+1))run.push_back({ll,rr,r-l+1});
  }
}
int main(){
  scanf("%s",st+1);n=strlen(st+1);
  init();
  for(int op=0;op<=1;op++){
    lyndon(op);
    getrun(); }
  sort(run.begin(),run.end(),cmp);
  run.erase(unique(run.begin(),run.end()),run.end());
  printf("%d\n",run.size());
  for(auto [l,r,p]:run)printf("%d %d %d\n",l,r,p);}
```

ZJJ: SAM处理手法

1. 基本子串结构: CLB 搞的那玩意。
2. 正反串 SAM 的基本联系: 一个子串出现的位置将会在两个SAM中同时得到映照。
3. SAM 上转成数点问题。
4. 线段树合并维护 endpos 集合。
5. 树剖保证到根的链上只涉及 log 次修改和查询。(区间 border)
6. LCT 保证到根的链只修改均摊 log 个不同的颜色段。(区间本质不同子串数量)