Shanghai Jiaotong University

Metis

Member:
Sishan Long
Yutong Xie
Jingyi Cai

Coach:
Yunqi Li
Xueyuan Zhao

Contents

0.0.1 开栈

```
#pragma comment(linker, "/STACK:16777216")//大小随便定
```

0.0.2 运行命令

```
g++ A.cpp -o A -Wall -O2
```

1 计算几何

1.1 精度

```
const double eps = 1e-8, pi = acos(-1.0);
inline int sign(double x) {return x < -eps ? -1 : x > eps;}
inline double Acos(double x) {
  if (sign(x + 1) == 0) return acos(-1.0);
  if (sign(x - 1) == 0) return acos(1.0);
  return acos(x);
}
inline double Asin(double x) {
  if (sign(x + 1) == 0) return asin(-1.0);
  if (sign(x - 1) == 0) return asin(1.0);
  return asin(x);
}
inline double Sqrt(double x) {
  if (sign(x) == 0) return 0;
  return sqrt(x);
}
```

1.2 点类 (向量类)

```
struct point
{
  double x,y;
  point(){}
  point(double x,double y) : x(x), y(y) {}
  double len() const {return(sqrt(x * x + y * y));}
  point unit() const {double t = len(); return(point(x / t, y /
      t));}
  point rotate() const {return(point(-y, x));}
  point rotate(double t) const
      {return(point(x*cos(t)-y*sin(t), x*sin(t)+y*cos(t)));}
};
point operator +(const point &a, const point &b)
    {return(point(a.x + b.x, a.y + b.y));}
point operator -(const point &a, const point &b)
    {return(point(a.x - b.x, a.y - b.y));}
point operator *(const point &a, double b)
    {return(point(a.x * b, a.y * b));}
point operator /(const point &a, double b)
    {return(point(a.x / b, a.y / b));}
bool operator <(const point &a, const point &b)
    {return(sign(a.x - b.x)<0||sign(a.x - b.x)==0&&sign(a.y - b
        .y)<0);}
double dot(const point &a, const point &b)
    {return(a.x * b.x + a.y * b.y);}
double det(const point &a, const point &b)
    {return(a.x * b.y - a.y * b.x);}
double mix(const point &a, const point &b, const point &c)
    {return dot(det(a, b), c);}//混合积,它等于四面体有向体积的
      六倍
double dist(const point &a, const point &b)
    {return((a - b).len());}
```

1.3 直线

```
//点在直线的哪一侧
int side(const point &p, const point &a, const point &b)
    {return(sign(det(b - a, p - a)));}
//点是否在线段上
bool online(const point&p,const point&a,const point&b)
    {return(sign(dot(p - a, p - b ))<=0 && sign(det(p - a, p - b))==0);}
//点关于直线垂线交点
point project(const point &p,const point &a, const point &b){
    double t = dot(p - a, b - a) / dot(b - a, b - a);
    return(a + (b - a) * t);}
//点到直线距离
double ptoline(const point &p, const point &a, const point &b)
    {return(fabs(det(p - a, p - b)) / dist(a, b));}
//点关于直线的对称点
point reflect(const point &p, const point &a, const point &b)
    {return(project(p, a, b) * 2 - p);}
//判断两直线是否平行
bool parallel(const point &a,const point &b,const point &c, const point &d)
    {return(sign(det(b - a, d - c)) == 0);}
//判断两直线是否垂直
bool orthogonal(const point&a,const point&b,const point&c,const point&d)
    {return(sign(dot(b - a, d - c)) == 0);}
//判断两线段是否相交
bool cross(const point&a,const point&b,const point&c,const point&d)
    {return(side(a, c, d) * side(b, c, d) == -1 && side(c, a, b) * side(d, a, b) == -1);}
//求两线段的交点
point intersect(const point&a,const point&b,const point&c,const point&d){
    double s1 = det(b - a, c - a), s2 = det(b - a, d - a);
    return((c * s2 - d * s1) / (s2 - s1));}
//两点求线ax+by+c=0
line point_make_line(point a, point b) {
     line h; h.a = b.y - a.y; h.b = -(b.x - a.x); h.c = -a.x * b.y + a.y * b.x;
     return h;
}
//线段平移D的长度
line move_d(line a, const double d) {
  return line(a.a, a.b, a.c + d * sqrt(a.a * a.a + a.b * a.b);
}
```

1.4 圆

```
//直线与圆交点
pair <point,point> intersect(const point &a,const point &b, const point &o,double r){
    point tmp = project(o, a, b); double d = dist(tmp, o);
    double l = Sqrt(sqr(r) - sqr(d));
    point dir = (b - a).unit() * l;
    return(make_pair(tmp + dir, tmp - dir));}
//两圆交点
pair <point, point> intersect(const point &o1, double r1,const point &o2, double r2){
    double d = dist(o1, o2), x = (sqr(r1) - sqr(r2)) / (2 * d) + d / 2;
    double l = Sqrt(sqr(r1) - sqr(x)); point dir = (o2 - o1).unit();
    return(make_pair(o1 + dir * x + dir.rotate() * l,
                     o1 + dir * x - dir.rotate() * l));}
//点与圆切线与圆交点
point tangent(const point &p, const point &o, double r)
    {return(intersect((p + o) / 2, dist(p, o) / 2, o, r).first);}
//两圆内公切线
pair <point,point> intangent(const point &o1,double r1,const point &o2,double r2){
    double t = r1 / (r1 + r2); point tmp = o1 + (o2 - o1) * t;
    point P = tangent(tmp, o1, r1), Q = tangent(tmp, o2, r2);
    return(make_pair(P, Q));}
//两圆外公切线
pair <point,point> extangent(const point &a, double r1,const point &b, double r2){
    if (sign(r1 - r2) == 0) {
        point dir = (b - a).rotate().unit();
        return(make_pair(a + dir * r1, b + dir * r2));}
    if (sign(r1 - r2) > 0) {
    pair <point, point> tmp = extangent(b, r2, a, r1);
        return(make_pair(tmp.second, tmp.first));}
    point p = tangent(a, b, r2 - r1), dir = (p - b).unit();
    return(make_pair(a + dir * r1, b + dir * r2));}
//两圆交线|P - P1| = r1 and |P - P2| = r2 of the ax + by + c = 0 form
void CommonAxis(point p1, double r1, point p2, double r2, double &a, double &b, double &c) {
  double sx = p2.x + p1.x, mx = p2.x - p1.x;
  double sy = p2.y + p1.y, my = p2.y - p1.y;
  a = 2 * mx; b = 2 * my; c = -sx * mx - sy * my - (r1 + r2) * (r1 - r2);
}
//两圆交点，两个圆不能共圆心，请特判
int CircleCrossCircle(point p1, double r1, point p2, double r2, point &cp1, point &cp2) {
  double mx = p2.x - p1.x, sx = p2.x + p1.x, mx2 = mx * mx;
  double my = p2.y - p1.y, sy = p2.y + p1.y, my2 = my * my;
  double sq = mx2 + my2, d = -(sq - sqr(r1 - r2)) * (sq - sqr(r1 + r2));
  if (d + eps < 0) return 0; if (d < eps) d = 0; else d = sqrt(d);
  double x = mx * ((r1 + r2) * (r1 - r2) + mx * sx) + sx * my2;
  double y = my * ((r1 + r2) * (r1 - r2) + my * sy) + sy * mx2;
  double dx = mx * d, dy = my * d; sq *= 2;
  cp1.x = (x - dy) / sq; cp1.y = (y + dx) / sq;
  cp2.x = (x + dy) / sq; cp2.y = (y - dx) / sq;
  if (d > eps) return 2; else return 1;
}
//两圆面积交:dist是距离，dis是平方
double twoCircleAreaUnion(point a, point b, double r1, double r2) {
  if (r1 + r2 <= (a - b).dist()) return 0;
  if (r1 + (a - b).dist() <= r2) return pi * r1 * r1;
  if (r2 + (a - b).dist() <= r1) return pi * r2 * r2;
  double c1, c2, ans = 0;
  c1 = (r1 * r1 - r2 * r2 + (a - b).dis()) / (a - b).dist() / r1 / 2.0;
  c2 = (r2 * r2 - r1 * r1 + (a - b).dis()) / (a - b).dist() / r2 / 2.0;
  double s1, s2; s1 = acos(c1); s2 = acos(c2);
  ans += s1 * r1 * r1 - r1 * r1 * sin(s1) * cos(s1);
  ans += s2 * r2 * r2 - r2 * r2 * sin(s2) * cos(s2);
  return ans;
}
```

1.4.1 最小覆盖球

```
double eps(1e-8);
int sign(const double & x) { return (x > eps) - (x + eps < 0);}
bool equal(const double & x, const double & y) {return x + eps > y and y + eps > x;}
struct Point {
  double x, y, z;
  Point() {}
  Point(const double & x, const double & y, const double & z) : x(x), y(y), z(z){}
  void scan() {scanf("%lf%lf%lf", &x, &y, &z);}
  double sqrlen() const {return x * x + y * y + z * z;}
  double len() const {return sqrt(sqrlen());}
  void print() const {printf("(%lf %lf %lf)\n", x, y, z);}
} a[33];
Point operator + (const Point & a, const Point & b) {return Point(a.x + b.x, a.y + b.y, a.z + b.z);}
Point operator - (const Point & a, const Point & b) {return Point(a.x - b.x, a.y - b.y, a.z - b.z);}
Point operator * (const double & x, const Point & a) {return Point(x * a.x, x * a.y, x * a.z);}
double operator % (const Point & a, const Point & b) {return a.x * b.x + a.y * b.y + a.z * b.z;}
Point operator * (const Point & a, const Point & b) {return Point(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);}
struct Circle {
  double r; Point o;
  Circle() {o.x = o.y = o.z = r = 0;}
  Circle(const Point & o, const double & r) : o(o), r(r) {}
  void scan() {o.scan();scanf("%lf", &r);}
  void print() const {o.print();printf("%lf\n", r);}
};
struct Plane {
  Point nor; double m;
  Plane(const Point & nor, const Point & a) : nor(nor){m = nor % a;}
};
Point intersect(const Plane & a, const Plane & b, const Plane & c) {
  Point c1(a.nor.x, b.nor.x, c.nor.x), c2(a.nor.y, b.nor.y, c.nor.y), c3(a.nor.z, b.nor.z, c.nor.z), c4(a.m, b.m, c.m);
  return 1 / ((c1 * c2) % c3) * Point((c4 * c2) % c3, (c1 * c4) % c3, (c1 * c2) % c4);
}
bool in(const Point & a, const Circle & b) {return sign((a - b.o).len() - b.r) <= 0;}
bool operator < (const Point & a, const Point & b) {
  if(!equal(a.x, b.x)) {return a.x < b.x;}
  if(!equal(a.y, b.y)) {return a.y < b.y;}
  if(!equal(a.z, b.z)) {return a.z < b.z;}
  return false;
}
bool operator == (const Point & a, const Point & b) {
  return equal(a.x, b.x) and equal(a.y, b.y) and equal(a.z, b.z);
}
vector<Point> vec;
Circle calc() {
  if(vec.empty()) {return Circle(Point(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(((vec[0] - vec[2]) * (vec[1] - vec[2])).len()));
    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((vec[1] - vec[0]) * (vec[2] - vec[0]), vec[0])), r);
  }else {
    Point 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(a[i], res)) {
      vec.push_back(a[i]); res = miniBall(i); vec.pop_back();
      if(i) {
        Point tmp(a[i]); memmove(a + 1, a, sizeof(Point) * i);
            a[0] = tmp;
      }
    }
  return res;
}
int main() {
  int n;
    for(int i(0); i < n; i++) a[i].scan();
    sort(a, a + n); n = unique(a, a + n) - a; vec.clear();
    printf("%.10f\n", miniBall(n).r);
  }
}
```

1.4.2 最小覆盖圆

```
const double eps=1e-6;
struct couple {
  double x, y;
  couple(){}
  couple(const double &xx, const double &yy){x = xx; y = yy;}
} a[100001];
int n;
bool operator < (const couple & a, const couple & b){return a.x < b.x - eps or (abs(a.x - b.x) < eps and a.y < b.y - eps);}
bool operator == (const couple & a, const couple & b){return !(a < b) and !(b < a);}
couple operator - (const couple &a, const couple &b){return couple(a.x-b.x, a.y-b.y);}
couple operator + (const couple &a, const couple &b){return couple(a.x+b.x, a.y+b.y);}
couple operator * (const couple &a, const double &b){return couple(a.x*b, a.y*b);}
couple operator / (const couple &a, const double &b){return a*(1/b);}
double operator * (const couple &a, const couple &b){return a.x*b.y-a.y*b.x;}
double len(const couple &a){return a.x*a.x+a.y*a.y;}
double di2(const couple &a, const couple &b){return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);}
double dis(const couple &a, const couple &b){return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));}
```

```
struct circle{
  double r; couple c;
} cir;
bool inside(const couple & x){return di2(x, cir.c) < cir.r*cir.r+eps;}
void p2c(int x, int y){
  cir.c.x = (a[x].x+a[y].x)/2; cir.c.y = (a[x].y+a[y].y)/2; cir.r = dis(cir.c, a[x]);
}
inline void p3c(int i, int j, int k){
  couple x = a[i], y = a[j], z = a[k];
  cir.r = sqrt(di2(x,y)*di2(y,z)*di2(z,x))/fabs(x*y+y*z+z*x)/2;
  couple t1((x-y).x, (y-z).x), t2((x-y).y, (y-z).y), t3((len(x)-len(y))/2, (len(y)-len(z))/2);
  cir.c = couple(t3*t2, t1*t3)/(t1*t2);
}
inline circle mi(){
  sort(a + 1, a + 1 + n); n = unique(a + 1, a + 1 + n) - a - 1;
  if(n == 1){
    cir.c = a[1]; cir.r = 0; return cir;
  }
  random_shuffle(a + 1, a + 1 + n);
  p2c(1, 2);
  for(int i = 3; i <= n; i++)
    if(!inside(a[i])){
      p2c(1, i);
      for(int j = 2; j < i; j++)
        if(!inside(a[j])){
          p2c(i, j);
          for(int k = 1; k < j; k++)
            if(!inside(a[k])) p3c(i,j, k);
        }
    }
  return cir;
}
```

1.5 多边形

```
水平序凸包
void convex(int &n, point a[]) {
    static point b[100010]; int m = 0;
    sort(a + 1, a + n + 1);
    for (int i = 1; i <= n; i++) {
        while (m >= 2 && sign(det(b[m] - b[m - 1], a[i] - b[m])) <= 0) m--;
        b[++m] = a[i];}
    int rev = m;
    for (int i = n - 1; i; i--) {
        while (m > rev && sign(det(b[m] - b[m - 1], a[i] - b[m])) <= 0) m--;
        b[++m] = a[i];}
    n = m - 1;
    for (int i = 1; i <= n; i++) a[i] = b[i];}
判断点与多边形关系 0外 1边 2内
int inPolygon(const point &p, int n, point a[]) {
    int res = 0; a[0] = a[n];
    for (int i = 1; i <= n; i++) {
        point A = a[i - 1], B = a[i];
        if (online(p, A, B)) return 2;
        if (sign(A.y - B.y) <= 0) swap(A,B);
        if (sign(p.y - A.y) > 0 || sign(p.y - B.y) <= 0)
            continue;
        res += sign(det(B - p, A - p)) > 0;}
    return(res & 1);}
多边形求重心
point center(const point &a, const point &b, const point &c)
    {return((a + b + c) / 3);}
point center(int n, point a[]) {
    point ret(0, 0); double area = 0;
    for (int i = 1; i <= n; i++) {
        ret += center(point(0, 0), a[i - 1], a[i]) * det(a[i - 1], a[i]);
        area += det(a[i - 1], a[i]);}
    return(ret / area);}
```

1.5.1 动态凸包

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

1.5.2 对踵点对

```
// 返回点集直径的平方
int diameter2(vector<Point>& points) {
  vector<Point> p = ConvexHull(points); int n = p.size();
  if(n == 1) return 0; if(n == 2) return Dist2(p[0], p[1]);
  p.push_back(p[0]); // 免得取模
  int ans = 0;
  for(int u = 0, v = 1; u < n; u++) {
    // 一条直线贴住边p[u]-p[u+1]
    for(;;) {
      // 当Area(p[u], p[u+1], p[v+1]) <= Area(p[u], p[u+1], p[v])时停止旋转
      // 即Cross(p[u+1]-p[u], p[v+1]-p[u]) - Cross(p[u+1]-p[u], p[v]-p[u]) <= 0
      // 根据Cross(A,B) - Cross(A,C) = Cross(A,B-C)
      // 化简得Cross(p[u+1]-p[u], p[v+1]-p[v]) <= 0
      int diff = Cross(p[u+1]-p[u], p[v+1]-p[v]);
      if(diff <= 0) {
        ans = max(ans, Dist2(p[u], p[v])); // u 和v是对踵点
        if(diff == 0) ans = max(ans, Dist2(p[u], p[v+1])); // diff == 0时u和v+1也是对踵点
        break;
      }
      v = (v + 1) % n;
    }
  }
  return ans;
}
```

1.5.3 凸多面体的重心

```
质量均匀的三棱锥重心坐标为四个定点坐标的平均数
对于凸多面体，可以先随便找一个位于凸多面体内部的点，得到若干个三棱锥和他们的重心，按照质量加权平均
```

1.5.4 圆与多边形交

```
转化为圆与各个三角形有向面积的交
（一）三角形的两条边全部短于半径。
（二）三角形的两条边全部长于半径，且另一条边与圆心的距离也长于半径。
（三）三角形的两条边全部长于半径，但另一条边与圆心的距离短于半径，并且垂足落在这条边上。
（四）三角形的两条边全部长于半径，但另一条边与圆心的距离短于半径，且垂足未落在这条边上。
（五）三角形的两条边一条长于半径，另外一条短于半径。
```

1.5.5 nlogn 半平面交

```
typedef long long LL;
const double eps = 1e-10, inf = 10000;
const int N = 20005;
#define zero(a) fabs(a) < eps
struct Point{
    double x, y;
} p[N * 2];
struct Segment {
  Point s, e;
  double angle;
  void get_angle() {angle = atan2(e.y - s.y, e.x - s.x);}
}seg[N];
int m;    //叉积为正说明，p2在p0-p1的左侧
double xmul(Point p0, Point p1, Point p2) {
  return (p1.x - p0.x) * (p2.y - p0.y) - (p2.x - p0.x) * (p1.y - p0.y);
}
Point Get_Intersect(Segment s1, Segment s2) {
  double u = xmul(s1.s, s1.e, s2.s), v = xmul(s1.e, s1.s, s2.e);
  Point t;
  t.x = (s2.s.x * v + s2.e.x * u) / (u + v);
  t.y = (s2.s.y * v + s2.e.y * u) / (u + v);
  return t;
}
bool cmp(Segment s1, Segment s2) {
  if(s1.angle > s2.angle) return true;
  else if(zero(s1.angle - s2.angle) && xmul(s2.s, s2.e, s1.e) > -eps) return true;
    return false;
}
void HalfPlaneIntersect(Segment seg[], int n){
    sort(seg, seg + n, cmp);
    int tmp = 1;
   for(int i = 1; i <  n; i++)
  if(!zero(seg[i].angle - seg[tmp - 1].angle)) seg[tmp++] = seg[i];
    n = tmp;
    Segment deq[N];
    deq[0] = seg[0]; deq[1] = seg[1];
    int head = 0, tail = 1;
    for(int i = 2; i < n; i++) {
    while(head < tail && xmul(seg[i].s, seg[i].e, Get_Intersect(deq[tail], deq[tail - 1])) < -eps)
      tail--;
    while(head < tail && xmul(seg[i].s, seg[i].e, Get_Intersect(deq[head], deq[head + 1])) < -eps) head++;
    deq[++tail]=seg[i];
    }
   while(head < tail && xmul(deq[head].s, deq[head].e, Get_Intersect(deq[tail], deq[tail - 1])) < -eps) tail--;
    while(head < tail && xmul(deq[tail].s, deq[tail].e, Get_Intersect(deq[head], deq[head + 1])) < -eps) head++;
    if(head == tail) return;
    m=0;
    for(int i = head;i<tail;i++)
        p[m++]=Get_Intersect(deq[i],deq[i+1]);
    if(tail>head+1)
        p[m++]=Get_Intersect(deq[head],deq[tail]);
}
double Get_area(Point p[],int &n){
    double area=0;
    for(int i = 1; i < n - 1; i++) area += xmul(p[0], p[i], p[i + 1]);
    return fabs(area) / 2.0;
}
int main(){
    int n;
    while(scanf("%d", &n) != EOF) {
       seg[0].s.x = 0; seg[0].s.y = 0; seg[0].e.x = 10000; seg[0].e.y = 0;
       seg[0].get_angle();
       seg[1].s.x = 10000; seg[1].s.y = 0; seg[1].e.x = 10000; seg[1].e.y=10000;
       seg[1].get_angle();
       seg[2].s.x = 10000; seg[2].s.y = 10000; seg[2].e.x = 0; seg[2].e.y=10000;
       seg[2].get_angle();
       seg[3].s.x=0; seg[3].s.y=10000; seg[3].e.x=0; seg[3].e.y=0;
       seg[3].get_angle();
       for(int i=0; i<n; i++){
        scanf("%lf%lf%lf%lf", &seg[i+4].s.x, &seg[i+4].s.y, &seg[i+4].e.x, &seg[i+4].e.y);
        seg[i+4].get_angle();
        }
        HalfPlaneIntersect(seg, n+4);
        printf("%.1f\n", Get_area(p,m)); //m<3 表示无解
    }
    return 0;
}
```

1.5.6 直线和凸包交点（返回最近和最远点）

```
double calc(point a, point b){
  double k=atan2(b.y-a.y , b.x-a.x);  if (k<0) k+=2*pi;return k
      ;
}//= the convex must compare y, then x£ ?a[0] is the lower-
      right point
//======= three is no 3 points in line. a[] is convex 0~n-1
void prepare(point a[] ,double w[],int &n) {
  int i;  rep(i,n) a[i+n]=a[i]; a[2*n]=a[0];
  rep(i,n) {  w[i]=calc(a[i],a[i+1]);w[i+n]=w[i];}
}
int find(double k,int n , double w[]){
  if (k<=w[0] || k>w[n-1]) return 0;  int l,r,mid; l=0; r=n-1;
  while (l<=r) {  mid=(l+r)/2;if (w[mid]>=k) r=mid-1; else l=
      mid+1;
  }return r+1;
}
int dic(const point &a, const point &b , int l ,int r , point c
      []) {
  int s;  if (area(a,b,c[l])<0) s=-1; else s=1; int mid;
  while (l<=r) {
    mid=(l+r)/2; if (area(a,b,c[mid])*s <= 0) r=mid-1;  else l=
        mid+1;
  }return r+1;
}
point get(const point &a, const point &b, point s1, point s2) {
  double k1,k2; point tmp; k1=area(a,b,s1); k2=area(a,b,s2);
  if (cmp(k1)==0) return s1; if (cmp(k2)==0) return s2;
  tmp=(s1*k2 ¨C s2*k1) / (k2-k1);  return tmp;
}
bool line_cross_convex(point a, point b ,point c[] , int n,
     point &cp1, point &cp2 , double w[]) {
  int i,j;
  i=find(calc(a,b),n,w);
  j=find(calc(b,a),n,w);
  double k1,k2;
  k1=area(a,b,c[i]); k2=area(a,b,c[j]);
  if (cmp(k1)*cmp(k2)>0) return false; //no cross
  if (cmp(k1)==0 || cmp(k2)==0) { //cross a point or a line in
      the convex
    if (cmp(k1)==0) {
      if (cmp(area(a,b,c[i+1]))==0) {cp1=c[i]; cp2=c[i+1];}
      else cp1=cp2=c[i];  return true;
    }
    if (cmp(k2)==0) {
      if (cmp(area(a,b,c[j+1]))==0) {cp1=c[j];cp2=c[j+1];}
      else cp1=cp2=c[j];
    }return true;
  }
  if (i>j) swap(i,j); int x,y;  x=dic(a,b,i,j,c); y=dic(a,b,j,i
      +n,c);
  cp1=get(a,b,c[x-1],c[x]); cp2=get(a,b,c[y-1],c[y]);
  return true;
}
```

1.5.7 Farmland

```
const int mx = 210;
const double eps = 1e-8;
struct TPoint {  double x, y;} p[mx];
struct TNode { int n, e[mx];} a[mx];
bool visit[mx][mx], valid[mx];
int l[mx][2], n, m, tp, ans, now, test;
double area;
int dcmp(double x) { return x < eps ? -1 : x > eps; }
int cmp(int a, int b){
    return dcmp(atan2(p[a].y - p[now].y, p[a].x - p[now].x) -
        atan2(p[b].y - p[now].y, p[b].x - p[now].x)) < 0;
}
double cross(const TPoint&a, const TPoint&b){    return a.x * b
    .y - b.x * a.y;}
void init();
void work();
bool check(int, int);
int main() {
    scanf("%d", &test);
    while(test--) {
        init();work();
    }
    return 0;
}
void init(){
    memset(visit, 0, sizeof(visit));
    memset(p, 0, sizeof(p));
    memset(a, 0, sizeof(a));
    scanf("%d", &n);
    for(int i = 0; i < n; i++) {
        scanf("%d", &a[i].n);  scanf("%lf%lf", &p[i].x, &p[i].y
            );
        scanf("%d", &a[i].n);
        for(int j = 0; j < a[i].n; j++) {
            scanf("%d", &a[i].e[j]);  a[i].e[j]--;
        }
    }
    scanf("%d", &m);
    for(now = 0; now < n; now++) sort(a[now].e, a[now].e + a[
        now].n, cmp);
}
void work() {
    ans = 0;
    for(int i = 0; i < n; i++)
        for(int j = 0; j < a[i].n; j++) if(!visit[i][a[i].e[j
            ]])
            if(check(i, a[i].e[j])) ans++;
    printf("%d\n", ans);
}
bool check(int b1, int b2) {
    area = 0;  l[0][0] = b1; l[0][1] = b2;
    for(tp = 1; ; tp++) {
        visit[l[tp - 1][0]][l[tp - 1][1]] = 1;
        area += cross(p[l[tp - 1][0]], p[l[tp - 1][1]]);
        int k, r(l[tp][0] = l[tp - 1][1]);
        for(k = 0; k < a[r].n; k++) if(a[r].e[k] == l[tp -
            1][0]) break;
        l[tp][1] = a[r].e[(k + a[r].n - 1) % a[r].n];
        if(l[tp][0] == b1 && l[tp][1] == b2) break;
    }
    if(dcmp(area) < 0 || tp < 3 || tp != m) return 0;
    fill_n(valid, n, 0);
    for(int i = 0; i < tp; i++) {
        if(valid[l[i][0]]) return 0;  valid[l[i][0]] = 1;
    }
    return 1;
}
```

1.6 三维操作

```
//平面法向量
double norm(const point &a, const point &b, const point &c)
    {return(det(b - a, c - a));}
//判断点在平面的哪一边
double side(const point &p,const point &a,const point &b,const
    point &c)
    {return(sign(dot(p - a, norm(a, b, c))));}
//点到平面距离
double ptoplane(const point&p,const point&a,const point&b,const
     point&c) {
    return(fabs(dot(p - a, norm(a, b, c).unit())));}
//点在平面投影
point project(const point&p,const point&a,const point&b,const
    point&c) {
    point dir = norm(a, b, c).unit();
    return(p - dir * (dot(p - a, dir)));}
//直线与平面交点
point intersect(const point &a,const point &b,const point &u,
    const point &v,const point &w) {
    double t = dot(norm(u,v,w),u-a)/dot(norm(u,v,w),b-a);
    return(a + (b - a) * t);}
//两平面交线
pair <point,point> intersect(const point &a,const point &b,
    const point &c,const point &u, const point &v, const point
     &w) {
    point p = parallel(a, b, u, v, w) ? intersect(a, c, u, v, w
        ) : intersect(a, b, u, v, w);
    point q = p + det(norm(a, b, c), norm(u, v, w));
    return(make_pair(p, q));}
```

1.6.1 经纬度（角度）转化为空间坐标

```
//角度转为弧度
double torad(double deg) {return deg / 180 * acos(-1);}
void get_coord(double R, double lat, double lng, double &x,
    double &y, double &z) {
    lat = torad(lat); lng = torag(lng);
    x = R * cos(lat) * cos(lng); y = R * cos(lat) * sin(lng); z
        = R * sin(lat);
}
```

1.6.2 多面体的体积

类似平面多边形面积的求法，不过需要首先规定好多面体的存储方式。一种简单的表示方法是点-面，即一个顶点数组 V 和面数组 F。其中 V 里保存着各个顶点的空间坐标，而 F 数组保存着各个面的 3 个顶点在 V 数组中的索引。简单起见，假设各个面都是三角形，且这三个点由右手定则确定的方向指向多边形的外部（即从外部看，3 个顶点呈逆时针排列），所以这些面上 3 个点的排列顺序并不是任意的。

1.6.3 三维凸包（加扰动）

```
double rand01() { return rand() / (double)RAND_MAX; }
double randeps() { return (rand01() - 0.5) * eps; }
Point3 add_noise(const Point3& p) {
  return Point3(p.x + randeps(), p.y + randeps(), p.z + randeps
      ());
}
struct Face {
  int v[3];
  Face(int a, int b, int c) { v[0] = a; v[1] = b; v[2] = c; }
  Vector3 Normal(const vector<Point3>& P) const {
    return Cross(P[v[1]]-P[v[0]], P[v[2]]-P[v[0]]);
  }
  // f是否能看见P[i]
  int CanSee(const vector<Point3>& P, int i) const {
    return Dot(P[i]-P[v[0]], Normal(P)) > 0;
  }
};
// 增量法求三维凸包
// 注意：没有考虑各种特殊情况（如四点共面）。实践中，请在调用前
    对输入点进行微小扰动
vector<Face> CH3D(const vector<Point3>& P) {
  int n = P.size();
  vector<vector<int> > vis(n);
  for(int i = 0; i < n; i++) vis[i].resize(n);
  vector<Face> cur;
  cur.push_back(Face(0, 1, 2)); // 由于已经进行扰动，前三个点不
      共线
  cur.push_back(Face(2, 1, 0));
  for(int i = 3; i < n; i++) {
    vector<Face> next;
    // 计算每条边的“左面”的可见性
    for(int j = 0; j < cur.size(); j++) {
      Face& f = cur[j];
      int res = f.CanSee(P, i);
      if(!res) next.push_back(f);
      for(int k = 0; k < 3; k++) vis[f.v[k]][f.v[(k+1)%3]] =
          res;
    }
    for(int j = 0; j < cur.size(); j++)
      for(int k = 0; k < 3; k++) {
        int a = cur[j].v[k], b = cur[j].v[(k+1)%3];
        if(vis[a][b] != vis[b][a] && vis[a][b]) // (a,b)是分界
            线，左边对P[i]可见
          next.push_back(Face(a, b, i));
      }
    cur = next;
  }
  return cur;
}
```

1.6.4 长方体表面最近距离

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

1.6.5 三维向量操作矩阵

- 绕单位向量 $u=(u_x,u_y,u_z)$ 右手方向旋转 θ 度的矩阵：
$$\begin{bmatrix}\cos\theta+u_x^2(1-\cos\theta) & u_xu_y(1-\cos\theta)-u_z\sin\theta & u_xu_z(1-\cos\theta)+u_y\sin\theta\\ u_yu_x(1-\cos\theta)+u_z\sin\theta & \cos\theta+u_y^2(1-\cos\theta) & u_yu_z(1-\cos\theta)-u_x\sin\theta\\ u_zu_x(1-\cos\theta)-u_y\sin\theta & u_zu_y(1-\cos\theta)+u_x\sin\theta & \cos\theta+u_z^2(1-\cos\theta)\end{bmatrix}$$
$$=\cos\theta I+\sin\theta\begin{bmatrix}0 & -u_z & u_y\\ u_z & 0 & -u_x\\ -u_y & u_x & 0\end{bmatrix}+(1-\cos\theta)\begin{bmatrix}u_x^2 & u_xu_y & u_xu_z\\ u_yu_x & u_y^2 & u_yu_z\\ u_zu_x & u_zu_y & u_z^2\end{bmatrix}$$

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

- 关于向量 v 作对称变换的矩阵 $H=I-2\frac{vv^T}{v^Tv}$,

- 点 a 对称点：$a'=a-2\frac{v^Ta}{v^Tv}\cdot v$

1.6.6 立体角

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

1.7 向量旋转

```
void rotate(double theta){
  double coss = cos(theta), sinn = sin(theta);
  double tx = x * coss - y * sinn;
  double ty = x * sinn + y * coss;
  x = tx, y = ty;
}
```

1.8 计算几何杂

```
bool pit_on_seg(pit a, pit b, pit c){ // 点在线段上
  if(dcmp(det(b - a, c - a)) != 0) return false;
  if(dcmp(dot(a - b, a - c)) > 0) return false;
  return true;
}
bool pit_in_polygon(pit q){ // 点在多边形内
  int cnt = 0;
  for(int i = 1; i <= n; ++i){
    pit p1 = p[i];
    pit p2 = p[suc[i]];
    if(pit_on_seg(q, p1, p2)) return true;
    int k = dcmp(det(p2 - p1, q - p1));
    int d1 = dcmp(p1.y - q.y);
    int d2 = dcmp(p2.y - q.y);
    if(k > 0 && d1 <= 0 && d2 > 0) ++cnt;
    if(k < 0 && d2 <= 0 && d1 > 0) --cnt;
  }
  if(cnt != 0) return true;
  else return false;
}
bool seg_in_polygon(pit a, pit b){ // 线段在多边形内 撒点
  vec v = b - a;
  for(int t = 1; t <= 1000; ++t){
    pit c = a + v * (1.00 * (rand() % 10000) / 10000);
    if(pit_in_polygon(c)) continue;
    else return false;
  }
  return true;
}
```

1.9 三维变换

```
struct Matrix{
    double a[4][4];
    int n,m;
    Matrix(int n = 4):n(n),m(n){
    for(int i = 0; i < n; ++i)
    a[i][i] = 1;
  }
    Matrix(int n, int m):n(n),m(m){}
    Matrix(Point A){
        n = 4;
        m = 1;
        a[0][0] = A.x;
        a[1][0] = A.y;
        a[2][0] = A.z;
        a[3][0] = 1;
    }
    //略
    Matrix operator *(const Matrix &b)const{
        Matrix ans(n,b.m);
        for (int i = 0; i < n; ++i)
        for (int j = 0; j < b.m; ++j)
        {
            ans.a[i][j] = 0;
            for (int k = 0; k < m; ++k)
            ans.a[i][j] += a[i][k] * b.a[k][j];
        }
        return ans;
    }
    Matrix operator * (double k)const{
        Matrix ans(n,m);
        for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j)
        ans.a[i][j] = a[i][j] * k;
        return ans;
    }
};
Matrix cur(4), I(4);
Point get(int i){//以下三个是变换矩阵，get是使用方法
    Matrix ori(p[i]);
    ori = cur * ori;
    return Point(ori.a[0][0],ori.a[1][0],ori.a[2][0]);
}
void trans(){//平移
    int l,r;
    Point vec;
    vec.read();
    cur = I;
    cur.a[0][3] = vec.x;
    cur.a[1][3] = vec.y;
    cur.a[2][3] = vec.z;
}
void scale(){//以base为原点放大k倍
    Point base;
    base.read();
    scanf("%lf",&k);
    cur = I;
    cur.a[0][0] = cur.a[1][1] = cur.a[2][2] = k;
    cur.a[0][3] = (1.0 - k) * base.x;
    cur.a[1][3] = (1.0 - k) * base.y;
    cur.a[2][3] = (1.0 - k) * base.z;
}
void rotate(){//绕以base为起点vec为方向向量的轴逆时针旋转theta
    Point base,vec;
    base.read();
    vec.read();
    double theta;
    scanf("%lf",&theta);
    if (dcmp(vec.x)==0&&dcmp(vec.y)==0&&dcmp(vec.z)==0)return;
    double C = cos(theta), S = sin(theta);
    vec = vec / len(vec);
    Matrix T1,T2;
    T1 = T2 = I;
    T1.a[0][3] = base.x;
    T1.a[1][3] = base.y;
    T1.a[2][3] = base.z;
    T2.a[0][3] = -base.x;
    T2.a[1][3] = -base.y;
    T2.a[2][3] = -base.z;
    cur = I;
    cur.a[0][0] = sqr(vec.x) * (1 - C) + C;
    cur.a[0][1] = vec.x * vec.y * (1-C) - vec.z * S;
    cur.a[0][2] = vec.x * vec.z * (1-C) + vec.y * S;
    cur.a[1][0] = vec.x * vec.y * (1-C) + vec.z * S;
    cur.a[1][1] = sqr(vec.y) * (1-C) + C;
    cur.a[1][2] = vec.y * vec.z * (1-C) - vec.x * S;
    cur.a[2][0] = vec.x * vec.z * (1-C) - vec.y * S;
    cur.a[2][1] = vec.y * vec.z * (1-C) + vec.x * S;
    cur.a[2][2] = vec.z * vec.z * (1-C) + C;
    cur = T1 * cur * T2;
}
```

1.10 三维凸包的重心（输入为凸包）

```
struct Point {
  double x, y, z;
  Point (double x = 0, double y = 0, double z = 0):x(x),y(y),z(
      z){}
  bool operator < (const Point &b)const{
    if (dcmp(x - b.x)==0)return y < b.y;
    else return x < b.x;
  }
};
inline double dot(const Point &a, const Point &b){return a.x*b.
    x + a.y * b.y + a.z * b.z;}
inline double Length(const Point &a){return sqrt(dot(a,a));}
inline Point cross(const Point &a, const Point &b){
  return Point(a.y*b.z - a.z*b.y, a.z*b.x - a.x*b.z, a.x*b.y -
      a.y*b.x);
}
inline double det(const Point &A, const Point &B, const Point &
    C){//前两维的平面情况！！！！！！
  Point a = B - A;
  Point b = C - A;
  return a.x * b.y - a.y * b.x;
}
double Volume(const Point &a,const Point &b, const Point &c,
    const Point &d){
  return fabs(dot(d-a, cross(b-a,c-a)));
}
double dis(const Point & p, const vector<Point> &v) {
  Point n = cross(v[1] - v[0],v[2] - v[0]);
  return fabs(dot(p - v[0], n))/Length(n);
}
int n;
Point p[100],Zero,basee,vec;
vector<Point>v[300];
bool cmp(const Point &A, const Point &B) {
  Point a = A - basee;
  Point b = B - basee;
  return  dot(vec, cross(a,b)) <= 0;
}
void caltri(const Point &A, Point B, Point C, double &w, Point
    &p) {
  double vol = Volume(Zero,A,B,C);
  w += vol;
  p = p + (Zero + A + B + C)/4*vol ;
}
pair<double, Point> cal(vector<Point> &v){
  basee = v[0];
  vec = cross(v[1] - v[0], v[2] - v[0]);
  double w = 0;
  Point centre;
  sort(v.begin(), v.end(),cmp);
  for (int i = 1; i < v.size() - 1; ++i)
  {
    caltri(v[0],v[i],v[i+1],w,centre);
  }
  return make_pair(w,centre);
}
bool vis[70][70][70];
double work(){
  scanf("%d",&n);
  for (int i = 0; i < n; ++i)p[i].read();
  Zero = p[0];
  for (int i = 0; i < 200; ++i)
  v[i].clear();
  memset(vis,0,sizeof(vis));
  int rear = -1;
  Point centre;
  double w = 0;
  for (int a = 0; a < n; ++a)
  for (int b = a + 1; b < n; ++b)
  for (int c = b + 1; c < n; ++c)
  if (!vis[a][b][c])
  {
    Point A = p[b] - p[a];
    Point B = p[c] - p[a];
    Point N = cross(A,B);
    int flag[3] = {0};
    for (int i = 0; i < n; ++i)
    if (i != a && i != b && i != c)flag[dcmp(dot(N, p[i] - p[a
        ]))+1] = 1;
    int cnt = 0;
    for (int i = 0; i < 3; ++i)
    if (flag[i])cnt++;
    if (!((cnt==2 && flag[1]==1) || cnt==1))continue;
    ++rear;
    vector<int>num;
    v[rear].push_back(p[a]);
    v[rear].push_back(p[b]);
    v[rear].push_back(p[c]);
    num.push_back(a);
    num.push_back(b);
    num.push_back(c);
    for (int i = c+1; i < n; ++i)
    if (dcmp(dot(N, p[i] - p[a]))==0) {
      v[rear].push_back(p[i]);
      num.push_back(i);
    }
    for (int x = 0; x < num.size(); ++x)
    for (int y = 0; y < num.size(); ++y)
    for (int z = 0; z < num.size(); ++z)
    vis[num[x]][num[y]][num[z]] = 1;
    pair<double, Point>tmp = cal(v[rear]);
```

```
    centre = centre + tmp.second;
    w += tmp.first;
  }
  centre = centre / w;
  double minn = 1e10;
  for (int i = 0; i <= rear; ++i)
  minn = min(minn, dis(centre, v[i]));
  return minn;
}
```

1.11 点在多边形内判断

```
bool point_on_line(const Point &p, const Point &a, const Point &b) {
  return sgn(det(p, a, b)) == 0 && sgn(dot(a-p, b-p)) <= 0;
}
bool point_in_polygon(const Point &p, const vector<Point> &polygon) {
  int counter = 0;
  for (int i = 0; i < (int)polygon.size(); ++i) {
    Point a = polygon[i], b = polygon[(i + 1) % (int)polygon.size()];
    if (point_on_line(p, a, b)) {
      // Point on the boundary are excluded.
      return false;
    }
    int x = sgn(det(a, p, b));
    int y = sgn(a.y - p.y);
    int z = sgn(b.y - p.y);
    counter += (x > 0 && y <= 0 && z > 0);
    counter -= (x < 0 && z <= 0 && y > 0);
  }
  return counter;//内: 1; 外: 0
}
```

1.12 圆交面积及重心

时间复杂度：n^2logn

```
struct Event {
  Point p;
  double ang;
  int delta;
  Event (Point p = Point(0, 0), double ang = 0, double delta = 0) : p(p), ang(ang), delta(delta) {}
};
bool operator < (const Event &a, const Event &b) {
  return a.ang < b.ang;
}
void addEvent(const Circle &a, const Circle &b, vector<Event> &evt, int &cnt) {
  double d2 = (a.o - b.o).len2(),
       dRatio = ((a.r - b.r) * (a.r + b.r) / d2 + 1) / 2,
       pRatio = sqrt(-(d2 - sqr(a.r - b.r)) * (d2 - sqr(a.r + b.r)) / (d2 * d2 * 4));
  Point d = b.o - a.o, p = d.rotate(PI / 2),
      q0 = a.o + d * dRatio + p * pRatio,
      q1 = a.o + d * dRatio - p * pRatio;
  double ang0 = (q0 - a.o).ang(),
       ang1 = (q1 - a.o).ang();
  evt.push_back(Event(q1, ang1, 1));
  evt.push_back(Event(q0, ang0, -1));
  cnt += ang1 > ang0;
}
bool issame(const Circle &a, const Circle &b) { return sign((a.o - b.o).len()) == 0 && sign(a.r - b.r) == 0; }
bool overlap(const Circle &a, const Circle &b) { return sign(a.r - b.r - (a.o - b.o).len()) >= 0; }
bool intersect(const Circle &a, const Circle &b) { return sign((a.o - b.o).len() - a.r - b.r) < 0; }
Circle c[N];
double area[N];  // area[k] -> area of intersections >= k.
Point centroid[N];
bool keep[N];
void add(int cnt, DB a, Point c) {
  area[cnt] += a;
  centroid[cnt] = centroid[cnt] + c * a;
}
void solve(int C) {
  for (int i = 1; i <= C; ++ i) {
      area[i] = 0;
      centroid[i] = Point(0, 0);
  }
  for (int i = 0; i < C; ++i) {
    int cnt = 1;
    vector<Event> evt;
    for (int j = 0; j < i; ++j) if (issame(c[i], c[j])) ++cnt;
    for (int j = 0; j < C; ++j) {
      if (j != i && !issame(c[i], c[j]) && overlap(c[j], c[i]))
        {
        ++cnt;
      }
    }
    for (int j = 0; j < C; ++j) {
      if (j != i && !overlap(c[j], c[i]) && !overlap(c[i], c[j]) && intersect(c[i], c[j])) {
        addEvent(c[i], c[j], evt, cnt);
      }
    }
    if (evt.size() == 0u) {
      add(cnt, PI * c[i].r * c[i].r, c[i].o);
    } else {
      sort(evt.begin(), evt.end());
      evt.push_back(evt.front());
      for (int j = 0; j + 1 < (int)evt.size(); ++j) {
        cnt += evt[j].delta;
        add(cnt, det(evt[j].p, evt[j + 1].p) / 2, (evt[j].p + evt[j + 1].p) / 3);
        double ang = evt[j + 1].ang - evt[j].ang;
        if (ang < 0) {
          ang += PI * 2;
        }
            if (sign(ang) == 0) continue;
            add(cnt, ang * c[i].r * c[i].r / 2, c[i].o +
              Point(sin(ang1) - sin(ang0), -cos(ang1) + cos(ang0)) * (2 / (3 * ang) * c[i].r));
        add(cnt, -sin(ang) * c[i].r * c[i].r / 2, (c[i].o + evt[j].p + evt[j + 1].p) / 3);
      }
    }
  }
  for (int i = 1; i <= C; ++ i)
    if (sign(area[i])) {
      centroid[i] = centroid[i] / area[i];
    }
}
```

2 数据结构

2.1 KD Tree

```
long long norm(const long long &x) {
    //    For manhattan distance
    return std::abs(x);
    //    For euclid distance
    return x * x;
}
struct Point {
    int x, y, id;
    const int& operator [] (int index) const {
        if (index == 0) {
            return x;
        } else {
            return y;
        }
    }
    friend long long dist(const Point &a, const Point &b) {
        long long result = 0;
        for (int i = 0; i < 2; ++i) {
            result += norm(a[i] - b[i]);
        }
        return result;
    }
} point[N];
struct Rectangle {
    int min[2], max[2];
    Rectangle() {
        min[0] = min[1] = INT_MAX;
        max[0] = max[1] = INT_MIN;
    }
    void add(const Point &p) {
        for (int i = 0; i < 2; ++i) {
            min[i] = std::min(min[i], p[i]);
            max[i] = std::max(max[i], p[i]);
        }
    }
    long long dist(const Point &p) {
        long long result = 0;
        for (int i = 0; i < 2; ++i) {
            //    For minimum distance
            result += norm(std::min(std::max(p[i], min[i]), max[i]) - p[i]);
            //    For maximum distance
            result += std::max(norm(max[i] - p[i]), norm(min[i] - p[i]));
        }
        return result;
    }
};
struct Node {
    Point seperator;
    Rectangle rectangle;
    int child[2];
    void reset(const Point &p) {
        seperator = p;
        rectangle = Rectangle();
        rectangle.add(p);
        child[0] = child[1] = 0;
    }
} tree[N << 1];
int size, pivot;
bool compare(const Point &a, const Point &b) {
    if (a[pivot] != b[pivot]) {
        return a[pivot] < b[pivot];
    }
    return a.id < b.id;
}
int build(int l, int r, int type = 1) {
    pivot = type;
    if (l >= r) {
        return 0;
    }
    int x = ++size;
    int mid = l + r >> 1;
    std::nth_element(point + l, point + mid, point + r, compare);
    tree[x].reset(point[mid]);
    for (int i = l; i < r; ++i) {
        tree[x].rectangle.add(point[i]);
    }
    tree[x].child[0] = build(l, mid, type ^ 1);
    tree[x].child[1] = build(mid + 1, r, type ^ 1);
    return x;
}
int insert(int x, const Point &p, int type = 1) {
    pivot = type;
    if (x == 0) {
        tree[++size].reset(p);
        return size;
    }
    tree[x].rectangle.add(p);
    if (compare(p, tree[x].seperator)) {
        tree[x].child[0] = insert(tree[x].child[0], p, type ^ 1);
    } else {
        tree[x].child[1] = insert(tree[x].child[1], p, type ^ 1);
    }
    return x;
}
//    For minimum distance
void query(int x, const Point &p, std::pair<long long, int> &answer, int type = 1) {
    pivot = type;
    if (x == 0 || tree[x].rectangle.dist(p) > answer.first) {
        return;
    }
    answer = std::min(answer,
             std::make_pair(dist(tree[x].seperator, p), tree[x].seperator.id));
    if (compare(p, tree[x].seperator)) {
        query(tree[x].child[0], p, answer, type ^ 1);
        query(tree[x].child[1], p, answer, type ^ 1);
    } else {
        query(tree[x].child[1], p, answer, type ^ 1);
        query(tree[x].child[0], p, answer, type ^ 1);
    }
}
std::priority_queue<std::pair<long long, int> > answer;
void query(int x, const Point &p, int k, int type = 1) {
    pivot = type;
    if (x == 0 ||
        (int)answer.size() == k && tree[x].rectangle.dist(p) > answer.top().first) {
        return;
    }
    answer.push(std::make_pair(dist(tree[x].seperator, p), tree[x].seperator.id));
    if ((int)answer.size() > k) {
        answer.pop();
    }
    if (compare(p, tree[x].seperator)) {
        query(tree[x].child[0], p, k, type ^ 1);
        query(tree[x].child[1], p, k, type ^ 1);
    } else {
        query(tree[x].child[1], p, k, type ^ 1);
        query(tree[x].child[0], p, k, type ^ 1);
    }
}
```

2.2 Splay

```
struct Splay{
  int tot, rt;
  struct Node{
    int lson, rson, fath, sz;
    int data;
    bool lazy;
  };
  Node nd[MAXN];
  void reverse(int i){
    if(!i) return;
    swap(nd[i].lson, nd[i].rson);
    nd[i].lazy = true;
  }
  void push_down(int i){
    if(!i || !nd[i].lazy) return;
    reverse(nd[i].lson);
    reverse(nd[i].rson);
    nd[i].lazy = false;
  }
  void zig(int i){
    int j = nd[i].fath;
    int k = nd[j].fath;
    if(k && j == nd[k].lson) nd[k].lson = i;
    else if(k) nd[k].rson = i;
    nd[i].fath = k;
    nd[j].fath = i;
    nd[nd[i].rson].fath = j;
    nd[j].lson = nd[i].rson;
    nd[i].rson = j;
    nd[i].sz = nd[j].sz;
    nd[j].sz = nd[nd[j].lson].sz + nd[nd[j].rson].sz + 1;
  }
  void zag(int i){
    int j = nd[i].fath;
    int k = nd[j].fath;
    if(k && j == nd[k].lson) nd[k].lson = i;
    else if(k) nd[k].rson = i;
    nd[i].fath = k;
    nd[j].fath = i;
    nd[nd[i].lson].fath = j;
    nd[j].rson = nd[i].lson;
    nd[i].lson = j;
    nd[i].sz = nd[j].sz;
    nd[j].sz = nd[nd[j].lson].sz + nd[nd[j].rson].sz + 1;
  }
  void down_path(int i){
    if(nd[i].fath) down_path(nd[i].fath);
    push_down(i);
  }
  void splay(int i){
    down_path(i);
    while(nd[i].fath){
      int j = nd[i].fath;
      if(nd[j].fath == 0){
        if(i == nd[j].lson) zig(i);
        else zag(i);
      }else{
            int k = nd[j].fath;
            if(j == nd[k].lson){
                if(i == nd[j].lson) zig(j), zig(i);
                else zag(i), zig(i);
            }else{
                if(i == nd[j].rson) zag(j), zag(i);
                else zig(i), zag(i);
            }
      }
    }
    rt = i;
  }
  int insert(int stat){ // 插入信息
    int i = rt;
    ++tot;
    nd[tot].data = stat;
    nd[tot].sz = 1;
    if(!nd[i].sz){
      nd[tot].fath = 0;
      rt = tot;
      return tot;
    }
    while(i){
      ++nd[i].sz;
      if(stat < nd[i].data){
            if(nd[i].lson) i = nd[i].lson;
            else{
            nd[i].lson = tot;
            break;
            }
      }else{
            if(nd[i].rson) i = nd[i].rson;
            else{
            nd[i].rson = tot;
            break;
            }
      }
    }
    nd[tot].fath = i;
    splay(tot);
    return tot;
  }
  void delet(int i){ // 删除信息
    if(!i) return;
    splay(i);
    int ls = nd[i].lson;
    int rs = nd[i].rson;
    nd[ls].fath = nd[rs].fath = 0;
    nd[i].lson = nd[i].rson = 0;
    if(ls == 0){
      rt = rs;
      nd[rs].fath = 0;
    }else{
      rt = ls;
      while(nd[ls].rson) ls = nd[ls].rson;
      splay(ls);
      nd[ls].fath = 0;
      nd[rs].fath = ls;
      nd[ls].rson = rs;
    }
    nd[rt].sz += nd[nd[rt].rson].sz;
  }
  int get_rank(int i){ // 查询节点编号为 i 的 rank
    splay(i);
    return nd[nd[i].rson].sz + 1;
  }
  int find(int stat){ // 查询信息为 stat 的节点编号
    int i = rt;
    while(i){
      if(stat < nd[i].data) i = nd[i].lson;
      else if(stat > nd[i].data) i = nd[i].rson;
         else return i;
    }
    return i;
  }
  int get_kth_max(int k){ // 查询第 k 大 返回其节点编号
    int i = rt;
    while(i){
      if(k <= nd[nd[i].rson].sz) i = nd[i].rson;
      else if(k > nd[nd[i].rson].sz + 1) k -= nd[nd[i].rson].sz
           + 1, i = nd[i].lson;
           else return i;
    }
    return i;
  }
}sp;
```

2.3 主席树

```
const int N = 1e5 + 5;
const int inf = 1e9 + 1;
struct segtree{
  int tot, rt[N];
  struct node{int ls, rs, size;}nd[N*40];
  void insert(int &i, int lf, int rg, int x){
    int j = ++tot;
    nd[j] = nd[i]; nd[j].size++; i = j;
    if(lf == rg) return;
    int mid = (lf + rg) >> 1;
    if(x <= mid) insert(nd[j].ls, lf, mid, x);
    else insert(nd[j].rs, mid + 1, rg, x);
    }
  int query(int i, int j, int lf, int rg, int k){
    if(lf == rg) return lf;
    int mid = (lf + rg) >> 1;
    if(nd[nd[j].ls].size - nd[nd[i].ls].size >= k)
      return query(nd[i].ls, nd[j].ls, lf, mid, k);
    else return query(nd[i].rs, nd[j].rs, mid + 1, rg,
      k - (nd[nd[j].ls].size - nd[nd[i].ls].size));
  }
}st;
int n, m, a[N], b[N], rnk[N], mp[N];
bool cmp(int i, int j){return a[i] < a[j];}
int main(){
  scanf("%d%d", &n, &m);
  for(int i = 1; i <= n; ++i) scanf("%d", &a[i]);
  for(int i = 1; i <= n; ++i) rnk[i] = i;
  sort(rnk + 1, rnk + 1 + n, cmp);
  a[0] = inf;
  for(int i = 1, j = 0; i <= n; ++i){
    int k = rnk[i], kk = rnk[i-1];
    if(a[k] != a[kk]) b[k] = ++j;
    else b[k] = j;
    mp[b[k]] = a[k];
  }
  for(int i = 1; i <= n; ++i)
    st.insert(st.rt[i] = st.rt[i-1], 1, n, b[i]);
  for(int i = 1, x, y, k; i <= m; ++i){
    scanf("%d%d%d", &x, &y, &k);
    printf("%d\n", mp[st.query(st.rt[x-1], st.rt[y], 1, n, k)])
        ;
  }
  return 0;
}
```

2.4 树链剖分 by cjy

```
const int N = 800005;
int n, m, Max, b[N], edge_pos[N], path[N];
int tot, id[N * 2], nxt[N * 2], lst[N], val[N * 2];
int fa[N], siz[N], dep[N], hvy[N], top[N], pos[N];
struct Tree {
  int l, r;
  int mn, mx, sgn;
} h[N * 4];
void Add(int x, int y, int z) {
  id[++tot] = y; nxt[tot] = lst[x]; lst[x] = tot; val[tot] = z;
}
void dfs1(int x, int Fa) {
  fa[x] = Fa;
  siz[x] = 1;
  dep[x] = dep[Fa] + 1;
  int max_size = 0;
  for (int i = lst[x]; i; i = nxt[i]) {
    int y = id[i];
    if (y != Fa) {
      path[y] = i; //------------------
      dfs1(y, x);
      if (siz[y] > max_size) {
        max_size = siz[y];
        hvy[x] = y;
      }
      siz[x] += siz[y];
    }
  }
}
void dfs2(int x, int Top) {
  top[x] = Top;
  pos[x] = ++m;
  b[m] = val[path[x]]; //b[m] = val[x];
  edge_pos[path[x] / 2] = m; //when change only one edge's
       value
  if (hvy[x]) dfs2(hvy[x], Top); //heavy son need to be visited
       first
  for (int i = lst[x]; i; i = nxt[i]) {
    int y = id[i];
    if (y == fa[x] || y == hvy[x]) continue;
    dfs2(y, y);
  }
}
void work(int x, int y) {
  int X = top[x], Y = top[y];
  if (X == Y) {
    if (dep[x] < dep[y]) Negate(1, pos[x] + 1, pos[y]);
    else if (dep[x] > dep[y]) Negate(1, pos[y] + 1, pos[x]);
    //if (dep[x] <= dep[y]) Negate(1, pos[x], pos[y]);
    //else Negate(1, pos[y], pos[x]);
    return ;
  }
  if (dep[X] >= dep[Y]) {
    Negate(1, pos[X], pos[x]);
    work(fa[X], y);
  }
  else {
    Negate(1, pos[Y], pos[y]);
    work(x, fa[Y]);
  }
}
int main() {
  tot = 1; memset(lst, 0, sizeof(lst));
  memset(hvy, 0, sizeof(hvy));
  (Add_edge)
  dep[0] = 0; dfs1(1, 0); //the root is 1
  m = 0; dfs2(1, 1);
  build(1, 1, n);
  Change(1, edge_pos[x], y); //change one edge's valve directly
       in Tree
  work(x, y); //change value of a chain
  return 0;
}
```

2.5 点分治

```
// POJ 1741
/*询问树上有多少对pair距离不超过k
  每次找重心 经过一些容斥
  求经过重心与不经过重心pair数*/
typedef pair<int, int> pii;
const int maxn = 1e4 + 5;
vector<pii> mp[maxn];
void add_edge(int u, int v, int d){
  mp[u].push_back(make_pair(v, d));
  mp[v].push_back(make_pair(u, d));
}
int n, ans, limit, gra, min_maxx;
int sz[maxn];
bool flag[maxn];
vector<int> vec;
void get_gra(int u, int fa, int nowsize){
  sz[u] = 1;
  int maxx = 0;
  for(int l = 0; l < mp[u].size(); ++l){
    int v = mp[u][l].first;
    if(v == fa || flag[v]) continue;
    get_gra(v, u, nowsize);
    sz[u] += sz[v];
    maxx = max(maxx, sz[v]);
  }
  maxx = max(maxx, nowsize - sz[u]);
  if(maxx < min_maxx) min_maxx = maxx, gra = u;
}
void get_dist(int u, int fa, int d){
  vec.push_back(d);
  for(int l = 0; l < mp[u].size(); ++l){
    int v = mp[u][l].first;
    if(v == fa || flag[v]) continue;
    get_dist(v, u, d + mp[u][l].second);
  }
}
int calc(int u, int delta){
  int rtn = 0;
  vec.clear();
  get_dist(u, 0, 0);
  sort(vec.begin(), vec.end());
  int m = vec.size();
  for(int i = 0, j = m - 1; i < j; ++i){
    while(i < j && vec[i] + vec[j] + delta > limit) --j;
    rtn += j - i;
  }
  return rtn;
}
void devide(int u, int nowsize){
  min_maxx = maxn;
  get_gra(u, 0, nowsize);
  flag[u=gra] = true;
  ans += calc(u, 0); // 加上经过重心的答案
  for(int l = 0; l < mp[u].size(); ++l){ // 容斥掉同一棵子树中
      经过重心的答案
    int v = mp[u][l].first;
    if(flag[v]) continue;
    ans -= calc(v, mp[u][l].second * 2);
    devide(v, sz[v] > sz[u] ? nowsize - sz[u] : sz[v]);
  }
}
void init(){
  ans = 0;
  for(int i = 1; i <= n; ++i) mp[i].clear();
  memset(flag, 0, sizeof flag);
}
void work(){
  init();
  for(int i = 1; i < n; ++i){
    int u, v, d;
    scanf("%d%d%d", &u, &v, &d);
    add_edge(u, v, d);
  }
  devide(1, n);
  printf("%d\n", ans);
}
int main(){
  while(true){
    scanf("%d%d", &n, &limit);
    if(n == 0) break;
    work();
  }
  return 0;
}
```

2.6 LCT

```
struct LinkCutTree {
    struct Node {
        int value, max, inc, father, child[2];
        bool rev;
        Node() {}
    }node[N];
    const Node EMPTY;
    void clear() {std::fill(node + 1, node + n + 1, EMPTY);}
    void __inc(int x, int delta) {
        if (x == 0) return;node[x].inc += delta;
        node[x].value += delta;node[x].max += delta;
    }
    void update(int x) {
        if (x == 0) return;
        if (node[x].inc != 0) {
            __inc(node[x].child[0], node[x].inc);
            __inc(node[x].child[1], node[x].inc);
            node[x].inc = 0;
        }
        if (node[x].rev == true) {
            std::swap(node[x].child[0], node[x].child[1]);
            node[node[x].child[0]].rev ^= true;
            node[node[x].child[1]].rev ^= true;
            node[x].rev = false;
        }
    }
    void renew(int x) {
        update(x); update(node[x].child[0]); update(node[x].
            child[1]);
        node[x].max = std::max(node[x].value, std::max(node[
            node[x].child[0]].max, node[node[x].child[1]].max)
            );
    }
    void change_value(int x, int value) {
        splay(x); node[x].value = node[x].max = value; renew(x)
            ;
    }
    bool is_splay_father(int y, int x) {
        return (y != 0) && (node[y].child[0] == x || node[y].
            child[1] == x);
    }
    void rotate(int x, int c) {
        int y = node[x].father;
        node[y].child[c ^ 1] = node[x].child[c];
        if (node[x].child[c] != 0) node[node[x].child[c]].
            father = y;
        node[x].father = node[y].father;
        if (node[node[y].father].child[0] == y) node[node[x].
            father].child[0] = x;
        else if(node[node[y].father].child[1]==py)node[node[x].
            father].child[1] = x;
        node[x].child[c] = y; node[y].father = x; renew(y);
    }
    void splay(int x) {
        if (x == 0) return; update(x);
        while (is_splay_father(node[x].father, x)) {
            int y = node[x].father, z = node[y].father;
            if (is_splay_father(z, y)) {
                update(z);update(y);update(x);
                int c = (y == node[z].child[0]);
                if (x == node[y].child[c]) rotate(x, c ^ 1),
                    rotate(x, c);
                else rotate(y, c);rotate(x, c);
            } else {
                update(y);update(x); rotate(x, x == node[y].
                    child[0]); break;
            }
        }
        renew(x);
    }
    int access(int x) {
        int y = 0;
        for ( ; x != 0; x = node[x].father) {
            splay(x); node[x].child[1] = y; renew(y = x);
        }
        return y;
    }
    int get_root(int x) {
        x = access(x);
        while (true) {
            update(x); if (node[x].child[0] == 0) break; x =
                node[x].child[0];
        }
        return x;
    }
    void make_root(int x) {node[access(x)].rev ^= true;splay(x)
        ;}
    void link(int x, int y) {
        make_root(x);node[x].father = y; access(x);
    }
    void cut(int x, int y) {
        make_root(x); access(y); splay(y);
        node[node[y].child[0]].father = 0; node[y].child[0] =
            0;
        renew(y);
    }
    void modify(int x, int y, int delta) {
        make_root(x); access(y); splay(y);  __inc(y, delta);
    }
    int get_max(int x, int y) {
        make_root(x); access(y); splay(y);p return node[y].max;
    }
};
```

3 字符串

3.1 串最小表示

```
int solve(char *text, int length) {
    int i = 0, j = 1, delta = 0;
    while (i < length && j < length && delta < length) {
        char tokeni = text[(i + delta) % length];
        char tokenj = text[(j + delta) % length];
        if (tokeni == tokenj) {
            delta++;
        } else {
            if (tokeni > tokenj) {
                i += delta + 1;
            } else {
                j += delta + 1;
            }
            if (i == j) {
                j++;
            }
            delta = 0;
        }
    }
    return std::min(i, j);
}
```

3.2 Manacher

```
// manacher
// 0-base
// odd s[i] len[i*2]
// even s[i],s[i+1] len[i*2+1]
void manacher(char *s) {
  int l = strlen(s);
  len[0] = 1;
  for (int i = 1, j = 0; i < n * 2 - 1; ++i) {
    int p = i / 2, q = i - p;
    int mx = (j + 1) / 2 + len[j] - 1;
    len[i] = mx < q ? 0 : min(mx - q + 1, len[j * 2 - i]);
    while (p - len[i] >= 0 && q + len[i] < l && s[p - len[i]]
         == s[q + len[i]]) len[i]++;
    if (q + len[i] - 1 > mx) mx = q + len[i] - 1;
  }
}
// 1-base
// only even s[i],s[i+1] len[i]
void manacher(char *s) {
  int l = strlen(s + 1);
  int mx = 0, id;
  for (int i = 1; i <= l; ++i) {
    if (mx >= i) len[i] = min(mx - i, len[id * 2 - i]); else
        len[i] = 0;
    for (; s[i - len[i]] == s[i + len[i] + 1]; len[i]++);
    if (i + len[i] > mx) mx = len[i] + i, id = i;
  }
}
```

3.3 AC 自动机

```
struct trie{
    int size, indx[maxs][26], word[maxs], fail[maxs];
    bool jump[maxs];
    int idx(char ff){return ff - 'a';}
    void insert(char s[]){
        int u = 0;
        for(int i = 0; s[i]; ++i){
            int k = idx(s[i]);
            if(!indx[u][k]) indx[u][k] = ++size;
            u = indx[u][k];
        }
        word[u] = 1;
        jump[u] = true;
    }
    void get_fail(){
        queue<int> que;
        int head = 0, tail = 0;
```

```
        que.push(0);
        while(!que.empty()){
            int u = que.front();
            que.pop();
            for(int k = 0; k < 26; ++k){
                if(!indx[u][k]) continue;
                int v = indx[u][k];
                int p = fail[u];
                while(p && !indx[p][k]) p = fail[p];
                if(indx[p][k] && indx[p][k] != v) p = indx[p][k
                    ];
                fail[v] = p;
                jump[v] |= jump[p];
                que.push(v);
            }
        }
    }
    int query(char s[]){
        int rtn = 0, p = 0;
        int flag[maxs];
        memcpy(flag, word, sizeof flag);
        for(int i = 0; s[i]; ++i){
            int k = idx(s[i]);
            while(p && !indx[p][k]) p = fail[p];
            p = indx[p][k];
            int v = p;
            while(jump[v]){
                rtn += flag[v];
                flag[v] = 0;
                v = fail[v];
            }
        }
        return rtn;
    }
} dict;
```

3.4 后缀数组

```
void calsa(int n, int m) {
  for (int i = 1; i <= n; i++) Rank[i] = num[i];
  for (int i = 1; i <= m; i++) c[i] = 0;
  for (int i = 1; i <= n; i++) c[Rank[i]]++;
  for (int i = 2; i <= m; i++) c[i] += c[i - 1];
  for (int i = n; i >= 1; i--) sa[c[Rank[i]]--] = i;
  for (int k = 1; k < n; k <<= 1) {
    int t = 0, j = n - k + 1;
    for (int i = j; i <= n; i++) sb[++t] = i;
    for (int i = 1; i <= n; i++) if (sa[i] > k) sb[++t] = sa[i]
        - k;
    for (int i = 1; i <= m; i++) c[i] = 0;
    for (int i = 1; i <= n; i++) c[Rank[i]]++;
    for (int i = 2; i <= m; i++) c[i] += c[i - 1];
    for (int i = n; i >= 1; i--) sa[c[Rank[sb[i]]]--] = sb[i];
    a[sa[1]] = t = 1;
    for (int i = 2; i <= n; i++)
      if (Rank[sa[i]] == Rank[sa[i - 1]] && Rank[sa[i] + k] ==
          Rank[sa[i - 1] + k])
        a[sa[i]] = t; else a[sa[i]] = ++t;
    for (int i = 1; i <= n; i++) Rank[i] = a[i];
    if (t == n) break; m = t;
  }
}
void calheight(int n) {
  int t = 0;
  for (int i = 1; i <= n; i++) {
    if (Rank[i] == 1) height[1] = t = 0;
    else {
      if (t > 0) t--;
      int j = sa[Rank[i] - 1];
      while (i + t <= n && j + t <= n && num[i + t] == num[j +
          t]) t++;
      height[Rank[i]] = t;
    }
  }
}
```

3.5 扩展 KMP

```
// (1-base) next[i] = lcp(text[1..n], text[i..n]), text[1..next
     [i]] = text[i..(i + next[i] - 1)]
void build(char *pattern) {
  int len = strlen(pattern + 1);
  int j = 1, k = 2;
  for (; j + 1 <= len && pattern[j] == pattern[j + 1]; j++);
  next[1] = len;
  next[2] = j - 1;
  for (int i = 3; i <= len; i++) {
    int far = k + next[k] - 1;
    if (next[i - k + 1] < far - i + 1) {
      next[i] = next[i - k + 1];
    }
    else {
      j = max(far - i + 1, 0);
      for (; i + j <= len && pattern[1 + j] == pattern[i + j];
          j++);
      next[i] = j;
      k = i;
    }
  }
}
void solve(char *text, char *pattern) {
  int len = strlen(text + 1);
  int lenp = strlen(pattern + 1);
  int j = 1, k = 1;
  for (; j <= len && j <= lenp && pattern[j] == text[j]; j++);
  extend[1] = j - 1;
  for (int i = 2; i <= len; i++) {
    int far = k + extend[k] - 1;
    if (next[i - k + 1] < far - i + 1) {
      extend[i] = next[i - k + 1];
    }
    else {
      j = max(far - i + 1, 0);
      for (; i + j <= len && 1 + j <= lenp && pattern[1 + j] ==
          text[i + j]; j++);
      extend[i] = j;
      k = i;
    }
  }
}
```

3.6 回文树

```
/*len[i]节点i的回文串的长度（一个节点表示一个回文串）
  nxt[i][c]节点i的回文串在两边添加字符c以后变成的回文串的编号
  fail[i]节点i失配以后跳转不等于自身的节点i表示的回文串的最长后
      缀回文串
  cnt[i]节点i表示的本质不同的串的个数（count()函数统计fail树上
      该节点及其子树的cnt和）
  num[i]以节点i表示的最长回文串的最右端点为回文串结尾的回文串个
      数
  lst指向新添加一个字母后所形成的最长回文串表示的节点
  s[i]表示第i次添加的字符（s[0]是任意一个在串s中不会出现的字
      符）
  n表示添加的字符个数
  一开始回文树有两个节点，0表示偶数长度串的根和1表示奇数长度串
      的根*/
const int N = 100005;
const int M = 30;
int n, ans[1005][1005];
char s[1005];
struct Palindromic_Tree {
  int nxt[N][M], fail[N];
  int cnt[N], num[N], len[N];
  int s[N], lst, n, m;
  int newnode (int l) {
    m++;
    for (int i = 1; i <= 26; i++) nxt[m][i] = 0; //-----------
    /*fail[m] = */cnt[m] = num[m] = 0;
    len[m] = l;
    return m;
  }
  void init() {
    m = -1;
    newnode(0);
    newnode(-1);
    lst = 0;
    n = 0; s[n] = 0;
    fail[0] = 1;
  }
  int get_fail(int x) {
    while (s[n - len[x] - 1] != s[n]) x = fail[x];
    return x;
  }
  void Insert(char c) {
    int t = c - 'a' + 1;
    s[++n] = t;
    int now = get_fail(lst);
    if (nxt[now][t] == 0) {
      int tmp = newnode(len[now] + 2);
      fail[tmp] = nxt[get_fail(fail[now])][t];
      nxt[now][t] = tmp;
      num[tmp] = num[fail[tmp]] + 1;
    }
    lst = nxt[now][t];
    cnt[lst]++; //位置不同的相同串算多次
  }
  void Count() {
    for (int i = m; i >= 0; i--) cnt[fail[i]] += cnt[i];
  }
} st;
int main() {
  st.init();
  for (int i = 1; i <= n; i++)
    st.Insert(s[i]);
  st.Count();
  ans = st.m - 1;
}
```

3.7 后缀自动机

```
const int L = 600005;//n * 2 开大一点，只开n会挂
struct Node
{
  Node *nx[26], *fail;
  int l, num;
};
Node *root, *last, sam[L], *b[L];
int sum[L], f[L];
int cnt;
char s[L];
int l;
void add(int x)
{
  ++cnt;
  Node *p = &sam[cnt];
  Node *pp = last;
  p->l = pp->l + 1;
  last = p;
  for(; pp && !pp->nx[x]; pp = pp->fail) pp->nx[x] = p;
  if(!pp) p->fail = root;
  else{
    if(pp->l + 1 == pp->nx[x]->l) p->fail = pp->nx[x];
    else{
      ++cnt;
      Node *r = &sam[cnt], *q = pp->nx[x];
      *r = *q;
      r->l = pp->l + 1;
      q->fail = p->fail = r;
      for(; pp && pp->nx[x] == q; pp = pp->fail) pp->nx[x] = r;
    }
  }
}
int main()
{
  scanf("%s", s);
  l = strlen(s);
  root = last = &sam[0];
  for(int i = 0; i < l; ++i) add(s[i] - 'a');
  for(int i = 0; i <= cnt; ++i) ++sum[sam[i].l];
  for(int i = 1; i <= l; ++i) sum[i] += sum[i - 1];
  for(int i = 0; i <= cnt; ++i) b[--sum[sam[i].l]] = &sam[i];
  Node *now = root;
  for(int i = 0; i < l; ++i){
    now = now->nx[s[i] - 'a'];
    ++now->num;
  }
  for(int i = cnt; i > 0; --i){
    int len = b[i]->l;
    //cerr<<"num="<<b[i]->num<<endl;
    f[len] = max(f[len], b[i]->num);
    //cerr<<b[i]->num<<" "<<b[i]->fail->num<<" ..."<<endl;
    b[i]->fail->num += b[i]->num;
    //cerr<<b[i]->num<<" "<<b[i]->fail->num<<" ..."<<endl;
  }
  for(int i = l - 1; i >= 1; --i) f[i] = max(f[i], f[i + 1]);
  for(int i = 1; i <= l; ++i) printf("%d\n", f[i]);
  return 0;
}
```

4 图论

4.1 图论相关

1. 差分约束系统

(1) 以 x[i] - x[j] <= c 为约束条件，j -> i : c，求最短路得到的是 x[i] <= x[s] 的最大解，存在负权回路无解

(2) 以 x[i] - x[j] >= c 为约束条件，j -> i : c，求最长路得到的时 x[i] >= x[s] 的最小解，存在正权回路无解 // 若有 x[i] = x[j] 则 i <-0-> j

2. 最大闭合权子图

s 向正权点连边，负权点向 t 连边，边权为点权绝对值，再按原图连边，边权为 INF

3. 最大密度子图：$\max\frac{|E'|}{|V'|}$

(1) 猜测答案 g 若最大流大于 EPS 则 g 合法 (2) s -> v: INF, u -> t : INF + g - deg[u], u -> v : 1.00

4. 2-SAT

如果 Ai 与 Aj 不相容，那么如果选择了 Ai，必须选择 Aj’；同样，如果选择了

Aj，就必须选择 Ai′：Ai => Aj′，Aj => Ai′（这样的两条边对称）输出方案：求图的极大强连通子图 => 缩点并根据原图关系构造一个 DAG => 拓扑排 => 自底（被指向的点）向上进行选择删除 （选择当前 $id[k][t]$ 及其后代结点并删除 $id[k][t^1]$ 及其前代结点）

5. 最小割

(1)二分图最小点权覆盖集:s -> u : w[u], u -> v : INF, v -> t : w[v]

4.2 欧拉回路

```
void dfs(int x)
{
    int y;
    for (int p=hd[x]; p != -1; p=ed[p].next) if (!ed[p].vst)
     {
        y = ed[p].b;
        ed[p].vst = 1;
        ed[p ^ 1].vst = 1;      //如果是有向图则不要这句
        dfs(y);
        res[v--] = y + 1;p
    }
}
```

4.3 斯坦纳树（网格图连接一些确定点的最小生成树）

```
//N点数，M边数，P关键点数
const int inf = 0x3f3f3f3f;
int n, m, p, status, idx[P], f[1 << P][N];
priority_queue<pair<int, int> > q; //int top, h[N];
void dijkstra(int dis[]) {}
void Steiner_Tree() {
  for (int i = 1; i < status; i++) {
    while (!q.empty()) q.pop(); //top = 0;
    memset(vis, 0, sizeof(vis));
    for (int j = 1; j <= n; j++) {
      for (int k = i & (i - 1); k; (--k) &= i)
        f[i][j] = min(f[i][j], f[k][j] + f[i ^ k][j]);
      if (f[i][j] != inf)
        q.push(make_pair(-f[i][j], j)); //h[++top] = j, vis[j]
             = 1;
    }
    dijkstra(f[i]); //SPFA(f[i]);
  }
}
int main() {
  scanf("%d%d%d", &n, &m, &p);
  status = 1 << p;
  tot = 0; memset(lst, 0, sizeof(lst));
  /*求最小生成森林
    每棵生成树中至少选择一个点，点权为代价
    新开一个空白关键点0作为源
    Add(0, i, val[i]); Add(i, 0, val[i]);*/
  for (int i = 1; i <= p; i++) scanf("%d", &idx[i]);
  memset(f, 0x3f, sizeof(f));
  for (int i = 1; i <= n; i++) f[0][i] = 0;
  for (int i = 1; i <= p; i++) f[1 << (i - 1)][idx[i]] = 0;
  Steiner_Tree();
  int ans = inf;
  for (int i = 1; i <= n; i++) ans = min(ans, f[status - 1][i])
      ;
}
```

4.4 Tarjan

```
// 针对无向图：求双联通分量（按割边缩点），求割点和桥
vector<pii> edge[N]; // pii => pair<int, int>
bool vist[M]; // 去掉vist判定及加单向边就是求强连通分量
void add_edge(int u, int v, int id){
  edge[u].push_back(make_pair(v, id));
  edge[v].push_back(make_pair(u, id));
}
int top, cnt, scc;
int dfn[N], low[N], stck[N], bel[N];
bool brg[M], inst[N], cut[N]; // brg => bridge
void tarjan(int u, int rt){
  dfn[u] = low[u] = ++cnt;
  stck[++top] = u;
  inst[u] = true;
  int son = 0, good_son = 0; //
  for(int l = 0; l < edge[u].size(); ++l){
    int id = edge[u][l].second;
    if(vist[id]) continue;
    vist[id] = true; ++son; //
    int v = edge[u][l].first;
    if(!dfn[v]){
      tarjan(v, rt);
      low[u] = min(low[u], low[v]);
      if(dfn[u] < low[v]) brg[id] = true;
    }else if(inst[v]) low[u] = min(low[u], dfn[v]);
    if(dfn[u] <= low[v]) ++good_son; //
  }
  if(u == rt){if(son >= 2) cut[u] = true;}
  else if(good_son > 0) cut[u] = true;
  if(dfn[u] == low[u]){
    ++scc; int v;
    do{
      v = stck[top--];
      bel[v] = scc;
      inst[v] = false;
    }while(v != u);
  }
}
// 针对无向图：求双联通分量（按割点缩点并建出森林）
int totedge, hd[N], th[M], nx[M];
void addedge(int x, int y){
  th[++totedge] = y; nx[totedge] = hd[x]; hd[x] = totedge;
  th[++totedge] = x; nx[totedge] = hd[y]; hd[y] = totedge;
}
int tottree, thd[N * 2], tth[M * 2], tnx[M * 2];
void addtree(int x, int y){
  tth[++tottree] = y; tnx[tottree] = thd[x]; thd[x] = tottree;
  tth[++tottree] = x; tnx[tottree] = thd[y]; thd[y] = tottree;
}
bool mark[M];
int part, ind, top;
int dfn[N], low[N], st[N], root[N];
void tarjan(int x, int cur){
  dfn[x] = low[x] = ++ind;
  for(int i = hd[x]; i; i = nx[i]){
    if(mark[i]) continue;
    mark[i] = mark[i ^ 1] = true;
    st[++top] = i;
    int v = th[i];
    if(dfn[v]){
      low[x] = min(low[x], low[v]);
      continue;
    }
    tarjan(v, cur);
    low[x] = min(low[x], low[v]);
    if(low[v] >= dfn[x]){
      ++part; int k;
      do{ //cur:=联通块里点双联通分量标号最小值
        k = st[top--];
        root[th[k]] = root[th[k ^ 1]] = cur;
        addtree(part, th[k]); //part为点双联通分量的标号
        addtree(part, th[k ^ 1]);
      }while(th[k ^ 1] != x);
    }
  }
}
int main(){
  part = n;
  for(int i = 1; i <= n; ++i) if(!dfn[i]) tarjan(i, part + 1);
}
```

4.5 LCA

```
int maxbit, dpth[maxn], ance[maxn][maxb];
void dfs(int u, int fath){
    dpth[u] = dpth[fath] + 1; ance[u][0] = fath;
    for(int i = 1; i <= maxbit; ++i) ance[u][i] = ance[ance[u][
         i-1]][i-1];
    for(int l = last[u]; l; l = next[l]){
        int v = dstn[l];
        if(v == fath) continue;
        dfs(v, u);
    }
}
int lca(int u, int v){
    if(dpth[u] < dpth[v]) swap(u, v);
    int p = dpth[u] - dpth[v];
    for(int i = 0; i <= maxbit; ++i)
        if(p & (1 << i)) u = ance[u][i];
    if(u == v) return u;
    for(int i = maxbit; i >= 0; --i){
        if(ance[u][i] == ance[v][i]) continue;
        u = ance[u][i]; v = ance[v][i];
    }
    return ance[u][0];
}
```

4.6 KM

```
int weight[M][M], lx[M], ly[M];
bool sx[M], sy[M];
int match[M];
bool search_path(int u){
  sx[u] = true;
  for (int v = 0; v < n; v++){
    if (!sy[v] && lx[u] + ly[v] == weight[u][v]){
      sy[v] = true;
      if (match[v] == -1 || search_path(match[v])){
        match[v] = u;
        return true;
      }
    }
  }
  return false;
}
int KM()
{
  for (int i = 0; i < n; i++){
    lx[i] = ly[i] = 0;
    for (int j = 0; j < n; j++)
      if (weight[i][j] > lx[i])
        lx[i] = weight[i][j];
  }
  memset(match, -1, sizeof(match));
  for (int u = 0; u < n; u++){
    while (1){
      memset(sx, 0, sizeof(sx));
      memset(sy, 0, sizeof(sy));
      if (search_path(u)) break;
      int inc = len * len;
      for (int i = 0; i < n; i++)
           if (sx[i])
           for (int j = 0; j < n; j++)
           if (!sy[j] && ((lx[i] + ly[j] - weight[i][j]) < inc
               ))
               inc = lx[i] + ly[j] - weight[i][j];
      for (int i = 0; i < n; i++){
        if (sx[i]) lx[i] -= inc;
        if (sy[i]) ly[i] += inc;
      }
    }
  }
  int sum = 0;
  for (int i = 0; i < n; i++)
    if (match[i] >= 0) sum += weight[match[i]][i];
  return sum;
}
int main()
{
  memset(weight, 0, sizeof(weight));
  for (int i = 1; i <= len; i++)
    weight[a[i]][b[i]]++;
  cout<<KM()<<endl;
  return 0;
}
```

4.7 KM 三次方

```
const int N=1010;
const int INF = 1e9;
int n;
struct KM{
int w[N][N];
int lx[N], ly[N], match[N],way[N], slack[N];
bool used[N];
void initialization(){
    for(int i = 1; i <= n; i++){
        match[i] = 0;
        lx[i] = 0;
        ly[i] = 0;
        way[i] = 0;
    }
}
void hungary(int x){//for i(1 -> n) : hungary(i);
    match[0] = x;
    int j0 = 0;
    for(int j = 0; j <= n; j++){
        slack[j] = INF;
        used[j] = false;
    }
    do{
        used[j0] = true;
        int i0 = match[j0], delta = INF, j1;
        for(int j = 1; j <= n; j++){
            if(used[j] == false){
                int cur = -w[i0][j] - lx[i0] - ly[j];
                if(cur < slack[j]){
                    slack[j] = cur;
                    way[j] = j0;
                }
```

```
                if(slack[j] < delta){
                    delta = slack[j];
                    j1 = j;
                }
            }
        }
        for(int j = 0; j <= n; j++){
            if(used[j]){
                lx[match[j]] += delta;
                ly[j] -= delta;
            }
            else slack[j] -= delta;
        }
        j0 = j1;
    }while (match[j0] != 0);
    do{
        int j1 = way[j0];
        match[j0] = match[j1];
        j0 = j1;
    }while(j0);
}
    int get_ans(){//maximum ans
    int sum = 0;
    for(int i = 1; i<= n; i++)
        if(match[i] > 0) sum += -w[match[i]][i];
    return sum;
}
}KM_solver;
```

4.8 网络流

```
// sap
struct edge{
    int v, r, flow;
    edge(int v, int flow, int r) : v(v), flow(flow), r(r) {}
};
vector<edge> mp[maxn];
void add_edge(int u, int v, int flow){
    mp[u].push_back(edge(v, flow, mp[v].size()));
    mp[v].push_back(edge(u, 0, mp[u].size() - 1));
}
int maxflow, disq[maxn], dist[maxn];
int sap(int u, int nowflow){
    if(nowflow == 0 || u == T) return nowflow;
    int tempflow, deltaflow = 0;
    for(int l = 0; l < mp[u].size(); ++l){
        int v = mp[u][l].v;
        if(mp[u][l].flow > 0 && dist[u] == dist[v] + 1){
            tempflow = sap(v, min(nowflow - deltaflow, mp[u][l].flow));
            mp[u][l].flow -= tempflow;
            mp[v][mp[u][l].r].flow += tempflow;
            deltaflow += tempflow;
            if(deltaflow == nowflow || dist[S] >= T) return deltaflow;
        }
    }
    disq[dist[u]]--;
    if(disq[dist[u]] == 0) dist[S] = T;
    dist[u]++;
    disq[dist[u]]++;
    return deltaflow;
}
int main(){
    while(dist[S] < T) maxflow += sap(S, inf);
}
// 费用流
struct edge{
    int v, r, cost, flow;
    edge(int v, int flow, int cost, int r) : v(v), flow(flow), cost(cost), r(r) {}
};
vector<edge> mp[maxn];
void add_edge(int u, int v, int flow, int cost){
    mp[u].push_back(edge(v, flow, cost, mp[v].size()));
    mp[v].push_back(edge(u, 0, -cost, mp[u].size() - 1));
}
int S, T, maxflow, mincost;
int dist[maxn], pth[maxn], lnk[maxn];
bool inq[maxn];
queue<int> que;
bool find_path(){
    for(int i = 1; i <= T; ++i) dist[i] = inf;
    dist[S] = 0;
    que.push(S);
    while(!que.empty()){
        int u = que.front();
        que.pop();
        inq[u] = false;
        for(int l = 0; l < mp[u].size(); ++l){
            int v = mp[u][l].v;
            if(mp[u][l].flow > 0 && dist[v] > dist[u] + mp[u][l].cost){
                dist[v] = dist[u] + mp[u][l].cost;
                pth[v] = u;
                lnk[v] = l;
                if(!inq[v]){
                    inq[v] = true;
                    que.push(v);
                }
            }
        }
    }
    if(dist[T] < inf) return true;
    else return false;
}
void adjust(){
    int deltaflow = inf, deltacost = 0;
    for(int v = T; v != S; v = pth[v]){
        deltaflow = min(deltaflow, mp[pth[v]][lnk[v]].flow);
        deltacost += mp[pth[v]][lnk[v]].cost;
    }
    maxflow += deltaflow;
    mincost += deltaflow * deltacost;
    for(int v = T; v != S; v = pth[v]){
        mp[pth[v]][lnk[v]].flow -= deltaflow;
        mp[mp[pth[v]][lnk[v]].v][mp[pth[v]][lnk[v]].r].flow += deltaflow;
    }
}
int main(){while(find_path()) adjust();}
```

4.9 ZKW 费用流

使用条件: 费用非负

```
#include <bits/stdc++.h>
using namespace std;
const int N = 4e3 + 5;
const int M = 2e6 + 5;
const long long INF = 1e18;
struct eglist{
  int tot_edge;
  int dstn[M], nxt[M], lst[N];
  long long cap[M], cost[M];
  void clear(){
    memset(lst, -1, sizeof lst);
    tot_edge = 0;
  }
  void _addEdge(int a, int b, long long c, long long d){
    dstn[tot_edge] = b;
    nxt[tot_edge] = lst[a];
    lst[a] = tot_edge;
    cost[tot_edge] = d;
    cap[tot_edge++] = c;
  }
  void add_edge(int a, int b, long long c, long long d){
    _addEdge(a, b, c, d);
    _addEdge(b, a, 0, -d);
  }
}e;
int st, ed, vist[N], cur[N];
long long tot_flow, tot_cost, dist[N], slack[N];
int modlable(){
  long long delta = INF;
  for(int i = 1; i <= ed; ++i){
    if(!vist[i] && slack[i] < delta)
      delta = slack[i];
    slack[i] = INF;
    cur[i] = e.lst[i];
  }
  if(delta == INF) return 1;
  for(int i = 1; i <= ed; ++i)
    if(vist[i])
      dist[i] += delta;
  return 0;
}
long long dfs(int x, long long flow){
  if(x == ed){
    tot_flow += flow;
    tot_cost += flow * (dist[st] - dist[ed]);
    return flow;
  }
  vist[x] = 1;
  long long left = flow;
  for(int i = cur[x]; ~i; i = e.nxt[i])
    if(e.cap[i] > 0 && !vist[e.dstn[i]]){
      int y = e.dstn[i];
      if(dist[y] + e.cost[i] == dist[x]){
        long long delta = dfs(y, min(left, e.cap[i]));
        e.cap[i] -= delta;
        e.cap[i ^ 1] += delta;
        left -= delta;
        if(!left) return flow;
      }else slack[y] = min(slack[y], dist[y] + e.cost[i] - dist[x]);
    }
  return flow - left;
}
void minCost(){
  tot_flow = 0, tot_cost = 0;
  fill(dist + 1, dist + 1 + ed, 0);
  for(int i = 1; i <= ed; ++i) cur[i] = e.lst[i];
  do{
    do{
      fill(vist + 1, vist + 1 + ed, 0);
    }while(dfs(st, INF));
  }while(!modlable());
}
int main(){
  e.clear(); minCost();
}
```

4.10 最大密度子图

```
const int maxn = 1e2 + 5;
const double eps = 1e-10;
const double d = 1e2;
const double inf = 1e9;
struct edge{
  int r, v;
  double flow;
  edge(int v, int r, double flow) : v(v), r(r), flow(flow) {}
};
vector<edge> mp[maxn];
void add_edge(int u, int v, double flow){
  mp[u].push_back(edge(v, mp[v].size(), flow));
  mp[v].push_back(edge(u, mp[u].size() - 1, 0.00));
}
int n, m, S, T, a[maxn], deg[maxn];
int dist[maxn], disq[maxn];
double sap(int u, double nowflow){
}
double value(){
  double maxflow = 0.00;
  while(dist[S] <= T) maxflow += sap(S, inf);
  return -0.50 * (maxflow - d * n);
}
void build(double g){
  g *= 2.00;
  for(int i = 1; i <= n; ++i) add_edge(S, i, d); // s -> v : INF
  for(int i = 1; i <= n; ++i) add_edge(i, T, d + g - deg[i]);
      // u -> t : INF + g - deg[u] 其中 deg[u] 为点 u 的度数(双向边)
  for(int i = 1; i <= n; ++i)
    for(int j = 1; j < i; ++j){
      if(a[i] >= a[j]) continue;
      add_edge(i, j, 1.00); // u -> v : 1.00
      add_edge(j, i, 1.00);
    }
}
void clear(){
  memset(dist, 0, sizeof dist);
  memset(disq, 0, sizeof disq);
  for(int i = 1; i <= T; ++i) mp[i].clear();
}
double binary(double left, double rght){ // 猜测答案 g [1 / n, m / 1]
  int step = 0;
  while(left + eps < rght && step <= 50){
    ++step;
    double mid = (left + rght) / 2;
    clear();
    build(mid);
    double h = value();
    if(h > eps) left = mid;
    else rght = mid;
  }
  return left;
}
void work(){
  m = 0;
  scanf("%d", &n);
  S = n + 1, T = n + 2;
  for(int i = 1; i <= n; ++i) scanf("%d", &a[i]);
  for(int i = 1; i <= n; ++i) deg[i] = 0;
  for(int i = 1; i <= n; ++i)
    for(int j = 1; j < i; ++j){
```

```
        if(a[i] >= a[j]) continue;
        ++m;
        ++deg[i];
        ++deg[j];
      }
  printf("%.12f\n", binary(0.00, m));
}
int main(){
  int case_number;
  scanf("%d", &case_number);
  for(int cs = 1; cs <= case_number; ++cs){
    printf("Case #%d: ", cs);
    work();
  }
  return 0;
}
```

4.11 上下界网络流

原图中边流量限制为 (a,b)，增加一个新的源点 S'，汇点 T'，对于每个顶点，
向 S' 连容量为所有流入它的边的下界和的边，向 T' 连容量为所有它流出的下界和的边，
T' 向 S' 连容量为无穷大的边，第一次跑 S' 到 T' 的网络流，判断 S' 流出的边是否满流，
即可判断是否有可行解，然后再跑 S 到 T 的网络流，总流量为两次之和。

$B(u,v)$ 表示边 (u,v) 流量的下界，$C(u,v)$ 表示边 (u,v) 流量的上界，$F(u,v)$ 表示边 (u,v) 的流量。设 $G(u,v)=F(u,v)-B(u,v)$，显然有

$$0 \leq G(u,v) \leq C(u,v)-B(u,v)$$

4.11.1 无源汇的上下界可行流

建立超级源点 S^* 和超级汇点 T^*，对于原图每条边 (u,v) 在新网络中连如下三条边：$S^* \to v$，容量为 $B(u,v)$；$u \to T^*$，容量为 $B(u,v)$；$u \to v$，容量为 $C(u,v)-B(u,v)$。最后求新网络的最大流，判断从超级源点 S^* 出发的边是否都满流即可，边 (u,v) 的最终解中的实际流量为 $G(u,v)+B(u,v)$。

4.11.2 有源汇的上下界可行流

从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边。按照**无源汇的上下界可行流**一样做即可，流量即为 $T \to S$ 边上的流量。

4.11.3 有源汇的上下界最大流

1. 在**有源汇的上下界可行流**中，从汇点 T 到源点 S 的边改为连一条上界为 ∞，下届为 x 的边。x 满足二分性质，找到最大的 x 使得新网络存在**无源汇的上下界可行流**即为原图的最大流。

2. 从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边，变成无源汇的网络。按照**无源汇的上下界可行流**的方法，建立超级源点 S^* 和超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，再将从汇点 T 到源点 S 的这条边拆掉，求一次 $S \to T$ 的最大流即可。

4.11.4 有源汇的上下界最小流

1. 在**有源汇的上下界可行流**中，从汇点 T 到源点 S 的边改为连一条上界为 x，下界为 0 的边。x 满足二分性质，找到最小的 x 使得新网络存在**无源汇的上下界可行流**即为原图的最小流。

2. 按照**无源汇的上下界可行流**的方法，建立超级源点 S^* 与超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，但是注意这一次不加上汇点 T 到源点 S 的这条边，即不使之改为无源汇的网络去求解。求完后，再加上那条汇点 T 到源点 S 上界 ∞ 的边。因为这条边下界为 0，所以 S^*，T^* 无影响，再直接求一次 $S^* \to T^*$ 的最大流。若超级源点 S^* 出发的边全部满流，则 $T \to S$ 边上的流量即为原图的最小流，否则无解。

4.12 无向图全局最小割

注意事项：处理重边时，应该对边权累加

```
int node[N], dist[N];
bool visit[N];
int solve(int n) {
    int answer = INT_MAX;
    for (int i = 0; i < n; ++i) {
        node[i] = i;
    }
    while (n > 1) {
        int max = 1;
        for (int i = 0; i < n; ++i) {
            dist[node[i]] = graph[node[0]][node[i]];
            if (dist[node[i]] > dist[node[max]]) {
                max = i;
            }
        }
        int prev = 0;
        memset(visit, 0, sizeof(visit));
        visit[node[0]] = true;
        for (int i = 1; i < n; ++i) {
            if (i == n - 1) {
                answer = std::min(answer, dist[node[max]]);
                for (int k = 0; k < n; ++k) {
                    graph[node[k]][node[prev]] =
                        (graph[node[prev]][node[k]] += graph[
                            node[k]][node[max]]);
                }
                node[max] = node[--n];
            }
            visit[node[max]] = true;
            prev = max;
            max = -1;
            for (int j = 1; j < n; ++j) {
                if (!visit[node[j]]) {
                    dist[node[j]] += graph[node[prev]][node[j
                        ]];
                    if (max == -1 || dist[node[max]] < dist[
                        node[j]]) {
                        max = j;
                    }
                }
            }
        }
    }
    return answer;
}
```

4.13 K 短路

4.13.1 可重复

```
// POJ 2449
/*********************************************
K短路 用dijsktra+A*启发式搜索
当点v第K次出堆的时候，这时候求得的路径是k短路。
A*算法有一个启发式函数f(p)=g(p)+h(p), 即评估函数=当前值+当前位
       置到终点的最短距离
g(p):当前从s到p点所走的路径长度，h(p)就是点p到目的点t的最短距
       离。
f(p)就是当前路径从s走到p在从p到t的所走距离。
步骤:
1>求出h(p)。将有向边反向，求出目的点t到所有点的最短距离，用
       dijkstra算法
2>将原点s加入优先队列中
3>优先队列取出f(p)最小的一个点p
如果p==t,并且出来的次数恰好是k次，那么算法结束
否则，如果p出来的次数多余k次，就不用再进入队列中
否则遍历p相邻的边，加入优先队列中
注意:如果s==t，那么求得k短路应该变成k++;
*********************************************/
#define MAXN 1005
#define MAXM 200100
struct Node{
        int v,c,nxt;
}Edge[MAXM];
int head[MAXN], tail[MAXN], h[MAXN];
struct Statement{
        int v,d,h;
        bool operator <( Statement a )const
        {    return a.d+a.h<d+h;   }
};
void addEdge( int u,int v,int c,int e ){
    Edge[e<<1].v=v; Edge[e<<1].c=c; Edge[e<<1].nxt=head[u];
        head[u]=e<<1;
    Edge[e<<1|1].v=u; Edge[e<<1|1].c=c; Edge[e<<1|1].nxt=tail[
        v]; tail[v]=e<<1|1;
}
void Dijstra( int n,int s,int t ){
    bool vis[MAXN];
    memset( vis,0,sizeof(vis) );
    memset( h,0x7F,sizeof(h) );
    h[t]=0;
    for( int i=1;i<=n;i++ ){
        int min=0x7FFF;
        int k=-1;
        for( int j=1;j<=n;j++ ){
            if( vis[j]==false && min>h[j] )
                min=h[j],k=j;
        }
        if( k==-1 )break;
        vis[k]=true;
        for( int temp=tail[k];temp!=-1;temp=Edge[temp].nxt ){
            int v=Edge[temp].v;
            if( h[v]>h[k]+Edge[temp].c )
                h[v]=h[k]+Edge[temp].c;
        }
    }
}
int Astar_Kth( int n,int s,int t,int K ){
    Statement cur,nxt;
    //priority_queue<Q>q;
    priority_queue<Statement>FstQ;
    int cnt[MAXN];
    memset( cnt,0,sizeof(cnt) );
    cur.v=s; cur.d=0; cur.h=h[s];
    FstQ.push(cur);
    while( !FstQ.empty() ){
        cur=FstQ.top();
        FstQ.pop();
        cnt[cur.v]++;
        if( cnt[cur.v]>K ) continue;
        if( cnt[t]==K )return cur.d;
        for( int temp=head[cur.v];temp!=-1;temp=Edge[temp].
            nxt ){
            int v=Edge[temp].v;
            nxt.d=cur.d+Edge[temp].c;
            nxt.v=v;
            nxt.h=h[v];
            FstQ.push(nxt);
        }
    }
    return -1;
}
int main()
{
    int n,m;
    while( scanf( "%d %d",&n,&m )!=EOF ){
        int u,v,c;
        memset( head,0xFF,sizeof(head) );
        memset( tail,0xFF,sizeof(tail) );
        for( int i=0;i<m;i++ ){
            scanf( "%d %d %d",&u,&v,&c );
            addEdge( u,v,c,i );
        }
        int s,t,k;
        scanf( "%d %d %d",&s,&t,&k );
        if( s==t ) k++;
        Dijstra( n,s,t );
        printf( "%d\n",Astar_Kth( n,s,t,k ) );
    }
    return 0;
}
```

4.13.2 不可重复

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

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

4.14 匈牙利

```
int n, m, stamp;
int match[N], visit[N];
bool dfs(int x) {
    for (int i = 0; i < (int)edge[x].size(); ++i) {
        int y = edge[x][i];
        if (visit[y] != stamp) {
            visit[y] = stamp;
            if (match[y] == -1 || dfs(match[y])) {
                match[y] = x;
                return true;
            }
        }
    }
    return false;
}
int solve() {
    std::fill(match, match + m, -1);
    int answer = 0;
    for (int i = 0; i < n; ++i) {
        stamp++;
        answer += dfs(i);
    }
    return answer;
}
```

4.15 hopcroft-karp

```
//O(n^0.5*m)
/*最小点覆盖数 = 最大匹配数
最大独立数 = 最大匹配数 (选取最多的点，使任意所选两点均不相连)
(DAG)最小路径覆盖数 = 顶点数 - 最大匹配数*/
int matchx[N], matchy[N], d[N];
bool dfs(int x) {
  for (int i = lst[x], y; i; i = nxt[i]) {
    y = id[i];
    int t = matchy[y];
    if (t == -1 || d[x] + 1 == d[t] && dfs(t)) {
      matchx[x] = y; matchy[y] = x;
      return true;
    }
  }
  d[x] = -1;
  return false;
}
int solve() {
  memset(matchx, -1, sizeof(matchx));
  memset(matchy, -1, sizeof(matchy));
  for (int ans = 0; ; ) {
    while (!Q.empty()) Q.pop();
    for (int i = 1; i <= n; i++)
      if (matchx[i] == -1) {
        d[i] = 0;
        Q.push(i);
      } else d[i] = -1;
    while (!Q.empty()) {
      int x = Q.front(); Q.pop();
      for (int i = lst[x], y; i; i = nxt[i]) {
        y = id[i];
        int t = matchy[y];
        if (t != -1 && d[t] == -1) {
          d[t] = d[x] + 1;
          Q.push(t);
        }
      }
    }
    int delta = 0;
    for (int i = 1; i <= n; i++)
      if (matchx[i] == -1 && dfs(i)) delta++;
    if (delta == 0) return ans;
    ans += delta;
  }
}
```

4.16 带花树 (任意图最大匹配)

```
//n全局变量, ans是匹配的点数, 即匹配数两倍
const int N = 240;
int n, Next[N], f[N], mark[N], visited [N], Link[N], Q[N], head
     , tail;
vector <int > E[N];
int getf(int x) {return f[x] == x ? x : f[x] = getf(f[x]);}
void merge(int x, int y) {x = getf(x); y = getf(y); if (x != y)
     f[x] = y;}
int LCA(int x, int y) {
    static int flag = 0;
    flag ++;
    for (; ; swap(x, y)) if (x != -1) {
        x = getf(x);
        if (visited [x] == flag) return x;
        visited [x] = flag;
        if (Link[x] != -1) x = Next[Link[x]];
        else x = -1;
    }
}
void go(int a, int p) {
    while (a != p) {
        int b = Link[a], c = Next[b];
        if (getf(c) != p) Next[c] = b;
        if (mark[b] == 2) mark[Q[tail ++] = b] = 1;
        if (mark[c] == 2) mark[Q[tail ++] = c] = 1;
        merge(a, b); merge (b, c); a = c;
    }
}
void find(int s) {
    for (int i = 0; i < n; i++) {
        Next[i] = -1; f[i] = i;
        mark[i] = 0; visited [i] = -1;
    }
    head = tail = 0; Q[tail ++] = s; mark[s] = 1;
    for (; head < tail && Link[s] == -1; )
    for (int i = 0, x = Q[head ++]; i < (int) E[x]. size (); i
         ++)
    if (Link[x]!=E[x][i]&&getf(x)!=getf(E[x][i])&&mark[E[x][i
         ]]!=2) {
        int y = E[x][i];
        if (mark[y] == 1) {
            int p = LCA(x, y);
            if (getf(x) != p) Next[x] = y;
            if (getf(y) != p) Next[y] = x;
            go(x, p);
            go(y, p);
        } else if (Link[y] == -1) {
            Next[y] = x;
            for (int j = y; j != -1; ) {
                int k = Next[j];
                int tmp = Link[k];
                Link[j] = k;
                Link[k] = j;
                j = tmp;
            }
            break;
        } else {
            Next[y] = x;
            mark[Q[tail ++] = Link[y]] = 1;
            mark[y] = 2;
        }
    }
}
int main () {
    for (int i = 0; i < n; i++) Link[i] = -1;
    for (int i = 0; i < n; i++) if (Link[i] == -1) find(i);
    int ans = 0;
    for (int i = 0; i < n; i++) ans += Link[i] != -1;
}
```

4.17 仙人掌图判定

条件是：1. 是强连通图；2. 每条边在仙人掌图中只属于一个强连通分量。// 仙人掌图的三个性质：1. 仙人掌 dfs 图中不能有横向边，简单的理解为每个点只能出现在一个强联通分量中；// 2.low[v]<dfn[u]，其中 u 为 v 的父节点；// 3.a[u]+b[u]<2，a[u] 为 u 节点的儿子节点中有 a[u] 个 low 值小于 u 的 dfn 值，b[u] 为 u 的逆向边条数。//

```
bool tarjan(int x) {
    dfn[x] = low[x] = ++cnt;
    stack[++top] = x; ins[x] = 1;
    int num = 0;
    for (int now = g[x]; now; now = pre[now]) {
        int y = nex[now];
        if (!dfn[y]) {
            if (!tarjan(y)) return 0;
            if (low[y] > dfn[x]) return 0;
            if (low[y] < dfn[x]) num++;
            low[x] = min(low[x], low[y]);
        } else if (ins[y]) {
            num++;
            low[x] = min(low[x], dfn[y]);
        } else return 0;
    }
    if (num >= 2) return 0;
    if (low[x] == dfn[x]) {
        while (stack[top] != x) {
            int y = stack[top];
            ins[y] = 0;
            stack[top--] = 0;
        }
        ins[x] = 0;
        stack[top--] = 0;
    }
    return 1;
}
```

4.18 最小树形图

```
const int maxn=1100;
int n,m , g[maxn][maxn] , used[maxn] , pass[maxn] , eg[maxn] ,
     more , queue[maxn];
void combine (int id , int &sum ) {
  int tot = 0 , from , i , j , k ;
  for ( ; id!=0 && !pass[ id ] ; id=eg[id] ) {
    queue[tot++]=id ; pass[id]=1;
  }
  for ( from=0; from<tot && queue[from]!=id ; from++);
  if  ( from==tot ) return ;
  more = 1 ;
  for ( i=from ; i<tot ; i++) {
    sum+=g[eg[queue[i]]][queue[i]] ;
    if ( i!=from ) {
      used[queue[i]]=1;
      for ( j = 1 ; j <= n ; j++) if ( !used[j] )
        if ( g[queue[i]][j]<g[id][j] ) g[id][j]=g[queue[i]][j]
             ;
    }
  }
  for ( i=1; i<=n ; i++) if ( !used[i] && i!=id ) {
    for ( j=from ; j<tot ; j++){
      k=queue[j];
      if ( g[i][id]>g[i][k]-g[eg[k]][k] ) g[i][id]=g[i][k]-g[eg
           [k]][k];
    }
  }
}
int mdst( int root ) { // return the total length of MDST
  int i , j , k , sum = 0 ;
  memset ( used , 0 , sizeof ( used ) ) ;
  for ( more =1; more ; ) {
```

```
    more = 0 ;
    memset (eg,0,sizeof(eg)) ;
    for ( i=1 ; i <= n ; i ++) if ( !used[i] && i!=root ) {
      for ( j=1 , k=0 ; j <= n ; j ++) if ( !used[j] && i!=j )
        if ( k==0 || g[j][i] < g[k][i] ) k=j ;
      eg[i] = k ;
    }
    memset(pass,0,sizeof(pass));
    for ( i=1; i<=n ; i++) if ( !used[i] && !pass[i] && i!=
        root ) combine ( i , sum ) ;
  }
  for ( i =1; i<=n ; i ++) if ( used[i] && i!= root ) sum+=g[
      eg[i]][i];
  return sum ;
}
```

4.19 有根树的同构

```
const unsigned long long MAGIC = 4423;
unsigned long long magic[N];
std::pair<unsigned long long, int> hash[N];
void solve(int root) {
    magic[0] = 1;
    for (int i = 1; i <= n; ++i) {
        magic[i] = magic[i - 1] * MAGIC;
    }
    std::vector<int> queue;
    queue.push_back(root);
    for (int head = 0; head < (int)queue.size(); ++head) {
        int x = queue[head];
        for (int i = 0; i < (int)son[x].size(); ++i) {
            int y = son[x][i];
            queue.push_back(y);
        }
    }
    for (int index = n - 1; index >= 0; --index) {
        int x = queue[index];
        hash[x] = std::make_pair(0, 0);

        std::vector<std::pair<unsigned long long, int> > value;
        for (int i = 0; i < (int)son[x].size(); ++i) {
            int y = son[x][i];
            value.push_back(hash[y]);
        }
        std::sort(value.begin(), value.end());

        hash[x].first = hash[x].first * magic[1] + 37;
        hash[x].second++;
        for (int i = 0; i < (int)value.size(); ++i) {
            hash[x].first = hash[x].first * magic[value[i].
                second] + value[i].first;
            hash[x].second += value[i].second;
        }
        hash[x].first = hash[x].first * magic[1] + 41;
        hash[x].second++;
    }
}
```

4.20 弦图

- 任何一个弦图都至少有一个单纯点，不是完全图的弦图至少有两个不相邻的单纯点.
- 设第 i 个点在弦图的完美消除序列第 $p(i)$ 个. 令 $N(v) = \{w|w与v相邻且p(w) > p(v)\}$ 弦图的极大团一定是 $v \cup N(v)$ 的形式.
- 弦图最多有 n 个极大团.
- 设 $next(v)$ 表示 $N(v)$ 中最前的点. 令 $w*$ 表示所有满足 $A \in B$ 的 w 中最后的一个点. 判断 $v \cup N(v)$ 是否为极大团，只需判断是否存在一个 w，满足 $Next(w) = v$ 且 $|N(v)| + 1 \leq |N(w)|$ 即可.
- 最小染色：完美消除序列从后往前依次给每个点染色，给每个点染上可以染的最小的颜色. (团数 = 色数)
- 最大独立集：完美消除序列从前往后能选就选.
- 最小团覆盖：设最大独立集为 $\{p_1, p_2, \ldots, p_t\}$，则 $\{p_1 \cup N(p_1), \ldots, p_t \cup N(p_t)\}$ 为最小团覆盖. (最大独立集数 = 最小团覆盖数)

```
//O(mlogn) 可以做到 O(n+m)
#define maxn 1005
#define maxm 2000005
int head[maxn],heap[maxn],l[maxn],hz,Link[maxn];
int vtx[maxm],next[maxm],tot,n,m,A[maxn];
bool map[maxn][maxn];
inline void Add(int a,int b){vtx[tot]=b; next[tot]=head[a];
     head[a]=tot++;}
inline void sink(int x){
  int mid=x*2;
  while (mid<=hz) {
    if (mid+1<=hz && l[heap[mid+1]]>l[heap[mid]]) ++mid;
    if (l[heap[x]]<l[heap[mid]]) {
      swap(Link[heap[x]],Link[heap[mid]]);swap(heap[x],heap[mid
          ]);
    }else break;
    x=mid;mid=x*2;
  }
}
inline void up(int x) {
  for (int mid=x/2;mid>0;mid=x/2) {
    if (l[heap[mid]]<l[heap[x]]) {
      swap(Link[heap[x]],Link[heap[mid]]);swap(heap[x],heap[mid
          ]);
    } else break;
    x=mid;
  }
}
int main() {
  for (;scanf("%d%d",&n,&m) && (m+n);) {
    tot=2;memset(map,false,sizeof(map));memset(head,0,sizeof(
        head));
    for (int i=0;i<m;++i) {
      int a,b;scanf("%d%d",&a,&b);--a;--b;
      map[a][b]=map[b][a]=true;Add(a,b);Add(b,a);
    }
    memset(l,0,sizeof(l));hz=0;
    for (int i=0;i<n;++i) {Link[i]=++hz;heap[hz]=i;}
    for (int i=n;i>0;--i) {
      int v=-1;int u=heap[1];
      //序列的第i项就是u
      Link[u]=-1;Link[heap[hz]]=1;
      heap[1]=heap[hz--];sink(1);
      for (int p=head[u];p;p=next[p])
      if (Link[vtx[p]]!=-1) {++l[vtx[p]];up(Link[vtx[p]]);
      } else {
        if (v==-1) v=vtx[p];
        else {
          if (!map[v][vtx[p]]) {
            printf("Imperfect\n");
            //判定不是弦图
            goto answer;
          }
        }
      }
    }
  }
  return 0;
}
```

4.21 哈密尔顿回路（ORE 性质的图）

ORE 性质：$\forall x, y \in V \wedge (x, y) \notin E \quad s.t. \quad deg_x + deg_y \geq n$ 返回结果：从顶点 1 出发的一个哈密尔顿回路. 使用条件：$n \geq 3$

```
int left[N], right[N], next[N], last[N];
void cover(int x) {
    left[right[x]] = left[x];
    right[left[x]] = right[x];
}
int adjacent(int x) {
    for (int i = right[0]; i <= n; i = right[i]) {
        if (graph[x][i]) {
            return i;
        }
    }
    return 0;
}
std::vector<int> solve() {
    for (int i = 1; i <= n; ++i) {
        left[i] = i - 1;
        right[i] = i + 1;
    }
    int head, tail;
    for (int i = 2; i <= n; ++i) {
        if (graph[1][i]) {
            head = 1;
            tail = i;
            cover(head);
            cover(tail);
            next[head] = tail;
            break;
        }
    }
    while (true) {
        int x;
        while (x = adjacent(head)) {
            next[x] = head;
            head = x;
            cover(head);
        }
        while (x = adjacent(tail)) {
            next[tail] = x;
            tail = x;
            cover(tail);
        }
        if (!graph[head][tail]) {
            for (int i = head, j; i != tail; i = next[i]) {
                if (graph[head][next[i]] && graph[tail][i]) {
                    for (j = head; j != i; j = next[j]) {
                        last[next[j]] = j;
                    }
                    j = next[head];
                    next[head] = next[i];
                    next[tail] = i;
                    tail = j;
                    for (j = i; j != head; j = last[j]) {
                        next[j] = last[j];
                    }
                    break;
                }
            }
        }
        next[tail] = head;
        if (right[0] > n) {
            break;
        }
        for (int i = head; i != tail; i = next[i]) {
            if (adjacent(i)) {
                head = next[i];
                tail = i;
                next[tail] = 0;
                break;
            }
        }
    }
    std::vector<int> answer;
    for (int i = head; ; i = next[i]) {
        if (i == 1) {
            answer.push_back(i);
            for (int j = next[i]; j != i; j = next[j]) {
                answer.push_back(j);
            }
            answer.push_back(i);
            break;
        }
        if (i == tail) {
            break;
        }
    }
    return answer;
}
```

4.22 度限制生成树

```
const int N = 55, M = 1010, INF = 1e8;
int n, m, S, K, ans, cnt, Best[N], fa[N], FE[N];
int f[N], p[M], t[M], c[M], o, Cost[N];
bool u[M], d[M];
pair<int, int> MinCost[N];
struct Edge {
  int a, b, c;
  bool operator < (const Edge & E) const { return c < E.c; }
}E[M];
vector<int> SE;
inline int F(int x) { return fa[x] == x ? x : fa[x] = F(fa[x]);
    }
inline void AddEdge(int a, int b, int C) {
  p[++o] = b; c[o] = C;
  t[o] = f[a]; f[a] = o;
}
void dfs(int i, int father) {
  fa[i] = father;
  if (father == S) Best[i] = -1;
  else {
    Best[i] = i;
    if (Cost[Best[father]] > Cost[i]) Best[i] = Best[father];
  }
  for (int j = f[i]; j; j = t[j])
  if (!d[j] && p[j] != father) {
    Cost[p[j]] = c[j];
    FE[p[j]] = j;
```

```
    dfs(p[j], i);
  }
}
inline void Kruskal() {
  cnt = n - 1; ans = 0; o = 1;
  for (int i = 1; i <= n; i++) fa[i] = i, f[i] = 0;
  sort(E + 1, E + m + 1);
  for (int i = 1; i <= m; i++) {
    if (E[i].b == S) swap(E[i].a, E[i].b);
    if (E[i].a != S && F(E[i].a) != F(E[i].b)) {
      fa[F(E[i].a)] = F(E[i].b);
      ans += E[i].c;
      cnt--;
      u[i] = true;
      AddEdge(E[i].a, E[i].b, E[i].c);
      AddEdge(E[i].b, E[i].a, E[i].c);
    }
  }
  for (int i = 1; i <= n; i++) MinCost[i] = make_pair(INF, INF);
  for (int i = 1; i <= m; i++)
  if (E[i].a == S) {
    SE.push_back(i);
    MinCost[F(E[i].b)] = min(MinCost[F(E[i].b)], make_pair(E[i].c, i));
  }
  for (int i = 1; i <= n; i++)
  if (i != S && fa[i] == i) {
    dfs(E[MinCost[i].second].b, S);
    u[MinCost[i].second] = true;
    ans += MinCost[i].first;
  }
}
bool Solve() {
  Kruskal();
  for (int i = cnt + 1; i <= K && i <= n; i++) {
    int MinD = INF, MinID = -1;
    for (int j = (int) SE.size() - 1; j >= 0; j--)
    if (u[SE[j]])
      SE.erase(SE.begin() + j);
    for (int j = 0; j < (int) SE.size(); j++) {
      int tmp = E[SE[j]].c - Cost[Best[E[SE[j]].b]];
      if (tmp < MinD) {
        MinD = tmp;
        MinID= SE[j];
      }
    }
    if (MinID == -1) return false;
    if (MinD >= 0) break;
    ans += MinD;
    u[MinID] = true;
    d[FE[Best[E[MinID].b]]] = d[FE[Best[E[MinID].b]] ^ 1] = true;
    dfs(E[MinID].b, S);
  }
  return true;
}
```

5 数学

5.1 FFT

```
// 复数 递归
const int maxn = 1e6 + 5;
typedef complex<long double> cpb;
int N;cpb a[maxn], aa[maxn], b[maxn], bb[maxn], c[maxn], cc[maxn];
typedef complex<double> cpb;
void fft(cpb x[], cpb xx[], int n, int step, int type){
    if(n == 1){xx[0] = x[0]; return;}
    int m = n >> 1;
    fft(x, xx, m, step << 1, type); // A[0]
    fft(x + step, xx + m, m, step << 1, type); // A[1]
    cpb w = exp(cpb(0, type * pi / m)); // 求原根 pi / m 其实就是 2 * pi / n
    cpb t = 1;
    for(int i = 0; i < m; ++i){
        cpb t0 = xx[i]; // 这个里面是A[0]的内容
        cpb t1 = xx[i+m]; // 这个里面是A[1]的内容
        xx[i] = t0 + t * t1;
        xx[i+m] = t0 - t * t1;
        t *= w;
    }
}
int main(){
    A = a.length(); B = b.length();
    for(N = 1; N < A + B; N <<= 1);
    fft(a, aa, N, 1, 1);
    fft(b, bb, N, 1, 1);
    for(int i = 0; i < N; ++i) cc[i] = aa[i] * bb[i];
    fft(cc, c, N, 1, -1);
    for(int i = 0; i < N; ++i) c[i] /= N;
}
// 原根 蝶型
const int p = 7340033, g = 3;
void fft(int xx[], int n, int type){
    for(int i = 0; i < n; ++i){ // i枚举每一个下表
        int j = 0; // j为n位二进制下i的对称
        for(int k = i, m = n - 1; m != 0; j = (j << 1) | (k & 1), k >>= 1, m >>= 1);
        if(i < j) swap(xx[i], xx[j]); // 为了防止换了之后又换回来于是只在 i < j 时交换
    }
    for(int m = 1; m < n; m <<= 1){ // m为当前讨论区间长度的一半
        int w = powmod(g, (1LL * type * (p - 1) / (m << 1) + p - 1) % (p - 1));
        for(int j = 0; j < n; j += (m << 1)){ // j为当前讨论区间起始位
            int t = 1;
            for(int i = 0; i < m; ++i){
                int t0 = xx[i+j];
                int t1 = 1LL * xx[i+j+m] * t % p;
                xx[i+j] = (t0 + t1) % p;
                xx[i+j+m] = (t0 - t1 + p) % p;
                t = 1LL * t * w % p;
            }
        }
    }
}
int main(){
    for(N = 1; N < A + B; N <<= 1);
    fft(a, N, 1);
    fft(b, N, 1);
    for(int i = 0; i < N; ++i) c[i] = 1LL * a[i] * b[i] % p;
    fft(c, N, -1);
    int inv_N = powmod(N, p - 2);
    for(int i = 0; i < N; ++i) c[i] = 1LL * c[i] * inv_N % p;
}
```

5.2 NTT

```
void solve(long long number[], int length, int type) {
    for (int i = 1, j = 0; i < length - 1; ++i) {
        for (int k = length; j ^= k >>= 1, ~j & k; );
        if (i < j) {
            std::swap(number[i], number[j]);
        }
    }
    long long unit_p0;
    for (int turn = 0; (1 << turn) < length; ++turn) {
        int step = 1 << turn, step2 = step << 1;
        if (type == 1) {
            unit_p0 = power_mod(MAGIC, (MOD - 1) / step2, MOD);
        } else {
            unit_p0 = power_mod(MAGIC, MOD - 1 - (MOD - 1) / step2, MOD);
        }
        for (int i = 0; i < length; i += step2) {
            long long unit = 1;
            for (int j = 0; j < step; ++j) {
                long long &number1 = number[i + j + step];
                long long &number2 = number[i + j];
                long long delta = unit * number1 % MOD;
                number1 = (number2 - delta + MOD) % MOD;
                number2 = (number2 + delta) % MOD;
                unit = unit * unit_p0 % MOD;
            }
        }
    }
}
void multiply() {
    for (; lowbit(length) != length; ++length);
    solve(number1, length, 1);
    solve(number2, length, 1);
    for (int i = 0; i < length; ++i) {
        number[i] = number1[i] * number2[i] % MOD;
    }
    solve(number, length, -1);
    for (int i = 0; i < length; ++i) {
        answer[i] = number[i] * power_mod(length, MOD - 2, MOD) % MOD;
    }
}
```

5.3 中国剩余定理 (含 exgcd)

```
long long extended_Euclid(long long a, long long b, long long &x, long long &y) { //return gcd(a, b)
  if (b == 0) {
    x = 1;
    y = 0;
    return a;
  }
  else {
    long long tmp = extended_Euclid(b, a % b, x, y);
    long long t = x;
    x = y;
    y = t - a / b * y;
    return tmp;
  }
}
long long China_Remainder(long long a[], long long b[], int n, long long &cir) { //a[]存放两两互质的除数  b[]存放余数
  long long x, y, ans;
  ans = 0; cir = 1;
  for (int i = 1; i <= n; i++) cir *= a[i];
  for (int i = 1; i <= n; i++) {
    long long tmp = cir / a[i];
    extended_Euclid(a[i], tmp, x, y);
    ans = (ans + y * tmp * b[i]) % cir; //可能会爆long long  用快速乘法
  }
  return (cir + ans % cir) % cir;
}
bool merge(long long &a1, long long &b1, long long a2, long long b2) { //num = b1(mod a1), num = b2(mod a2)
  long long x, y;
  long long d = extended_Euclid(a1, a2, x, y);
  long long c = b2 - b1;
  if (c % d) return false;
  long long p = a2 / d;
  x = (c / d * x % p + p) % p;
  b1 += a1 * x;
  a1 *= a2 / d;
  return true;
}
long long China_Remainder2(long long a[], long long b[], int n) { //a[]存放除数(不一定两两互质)  b[]存放余数
  long long x, y, ans, cir;
  cir = a[1]; ans = b[1];
  for (int i = 2; i <= n; i++) {
    if (!merge(cir, ans, a[i], b[i])) return -1;
  }
  return (cir + ans % cir) % cir;
}
```

6 数值

6.1 行列式取模

```
inline long long solve(int n, long long p) {
  for(int i = 1; i <= n; ++i)
    for(int j = 1; j <= n; ++j)
      a[i][j] %= p;
  long long ans(1);
  long long sgn(1);
  for(int i = 1; i <= n; ++i) {
    for(int j = i + 1; j <= n; ++j) {
      while(a[j][i]) {
        long long t = a[i][i] / a[j][i];
        for(int k = 1; k <= n; ++k) {
          a[i][k] = (a[i][k] - t * a[j][k]) % p;
          swap(a[i][k], a[j][k]);
        }
        sgn = -sgn;
      }
    }
    if(a[i][i] == 0)
      return 0;
    ans = ans * a[i][i] % p;
  }
  ans = ans * sgn;
  return (ans % p + p) % p;
}
```

6.2 最小二乘法

```
// calculate argmin ||AX - B||
solution least_squares(vector<vector<double> > a, vector<double> b) {
  int n = (int)a.size(), m = (int)a[0].size();
  vector<vector<double> > p(m, vector<double>(m, 0));
  vector<double> q(m, 0);
  for (int i = 0; i < m; ++i)
    for (int j = 0; j < m; ++j)
```

```
    for (int k = 0; k < n; ++k)
      p[i][j] += a[k][i] * a[k][j];
  for (int i = 0; i < m; ++i)
    for (int j = 0; j < n; ++j)
      q[i] += a[j][i] * b[j];
  return gauss_elimination(p, q);
}
```

6.3 多项式求根

```
const double eps=1e-12;
double a[10][10];
typedef vector<double> vd;
int sgn(double x) { return x < -eps ? -1 : x > eps; }
double mypow(double x,int num){
  double ans=1.0;
  for(int i=1;i<=num;++i)ans*=x;
  return ans;
}
double f(int n,double x){
  double ans=0;
  for(int i=n;i>=0;--i)ans+=a[n][i]*mypow(x,i);
  return ans;
}
double getRoot(int n,double l,double r){
  if(sgn(f(n,l))==0)return l;
  if(sgn(f(n,r))==0)return r;
  double temp;
  if(sgn(f(n,l))>0)temp=-1;else temp=1;
  double m;
  for(int i=1;i<=10000;++i){
    m=(l+r)/2;
    double mid=f(n,m);
    if(sgn(mid)==0){
      return m;
    }
    if(mid*temp<0)l=m;else r=m;
  }
  return (l+r)/2;
}
vd did(int n){
  vd ret;
  if(n==1){
    ret.push_back(-1e10);
    ret.push_back(-a[n][0]/a[n][1]);
    ret.push_back(1e10);
    return ret;
  }
  vd mid=did(n-1);
  ret.push_back(-1e10);
  for(int i=0;i+1<mid.size();++i){
    int t1=sgn(f(n,mid[i])),t2=sgn(f(n,mid[i+1]));
    if(t1*t2>0)continue;
    ret.push_back(getRoot(n,mid[i],mid[i+1]));
  }
  ret.push_back(1e10);
  return ret;
}
int main(){
  int n; scanf("%d",&n);
  for(int i=n;i>=0;--i){
    scanf("%lf",&a[n][i]);
  }
  for(int i=n-1;i>=0;--i)
    for(int j=0;j<=i;++j)a[i][j]=a[i+1][j+1]*(j+1);
  vd ans=did(n);
  sort(ans.begin(),ans.end());
  for(int i=1;i+1<ans.size();++i)printf("%.10f\n",ans[i]);
  return 0;
}
```

6.4 线性规划

```
//N[0]代表N中的元素个数，B[0]代表B中的元素个数。
//读入格式：首先两个数n，m，表示未知数的数量和约束的数量。接下
     来一行n个数，为目标函数的系数。然后m行，每行m+1个数，表示
     一个约束。前m个数是系数，最后一个是常数项。
//输出格式：如果无解，只有一行"Infeasible"。如果解可以无穷大，
     只有一行"Unbounded"。否则，第一行为最大的目标函数值，接下
     来是每个未知数的值。
const double eps = 1e-10;
const int MAXSIZE = 2000, oo = 19890709;
double v, A[MAXSIZE+1][MAXSIZE+1], tA[MAXSIZE+1][MAXSIZE+1];
double b[MAXSIZE+1], tb[MAXSIZE+1], c[MAXSIZE+1], tc[MAXSIZE
     +1];
int n, m, N[MAXSIZE+1+1], B[MAXSIZE+1+1];
class LinearProgramming {
  void read() {
    scanf("%d%d", &n, &m);
    for(int i=1; i<=n; i++) scanf("%lf", &c[i]);
    for(int i=1; i<=m; i++) {
      for(int j=1; j<=n; j++) scanf("%lf", &A[n+i][j]);
      scanf("%lf", &b[n+i]);
    }
  }
  void pivot(int l, int e) {
    tb[e] = b[l]/A[l][e]; tA[e][l] = 1/A[l][e];
    for(int i=1; i<=N[0]; i++) if (N[i] != e) tA[e][N[i]] = A[l
         ][N[i]]/A[l][e];
    for(int i=1; i<=B[0]; i++) {
      tb[B[i]] = b[B[i]]-A[B[i]][e]*tb[e]; tA[B[i]][l] = -A[B[i
           ]][e]*tA[e][l];
      for(int j=1; j<=N[0]; j++)
        if (N[j] != e) tA[B[i]][N[j]] = A[B[i]][N[j]]-tA[e][N[j
             ]]*A[B[i]][e];
    }
    v += tb[e]*c[e]; tc[l] = -tA[e][l]*c[e];
    for(int i=1; i<=N[0]; i++) if (N[i] != e) tc[N[i]] = c[N[i
         ]]-tA[e][N[i]]*c[e];
    for(int i=1; i<=N[0]; i++) if (N[i] == e) N[i] = l;
    for(int i=1; i<=B[0]; i++) if (B[i] == l) B[i] = e;
    for(int i=1; i<=B[0]; i++) {
      for(int j=1; j<=N[0]; j++) A[B[i]][N[j]] = tA[B[i]][N[j
           ]];
      b[B[i]] = tb[B[i]];
    }
    for(int i=1; i<=N[0]; i++) c[N[i]] = tc[N[i]];
  }
  bool opt() { //false stands for unbounded
    while (true) {
      int l, e; double maxUp = -1;//不能是0!
      for(int ie=1; ie<=N[0]; ie++) {
        int te = N[ie]; if (c[te] <= eps) continue;  //eps or 0
        double delta = oo; int tl = MAXSIZE+1;
        for(int i=1; i<=B[0]; i++)
          if (A[B[i]][te] > eps) {  //eps or 0
            double temp = b[B[i]]/A[B[i]][te];
            if (delta == oo || temp < delta || temp == delta &&
                 B[i] < tl) {
              delta = temp; tl = B[i];
            }
          }
        if (tl == MAXSIZE+1) return false;
        if (delta*c[te] > maxUp) {
          maxUp = delta*c[te]; l = tl; e = te;
        }
      }
      if (maxUp == -1) break; pivot(l, e);
    }
    return true;
  }
  void delete0() {
    int p;
    for(p=1; p<=B[0]; p++) if (B[p] == 0) break;
    if (p <= B[0]) pivot(0, N[1]);
    for(p=1; p<=N[0]; p++) if (N[p] == 0) break;
    for(int i=p; i<N[0]; i++) N[i] = N[i+1];
    N[0]--;
  }
  bool initialize() {
    N[0] = B[0] = 0;
    for(int i=1; i<=n; i++) N[++N[0]] = i;
    for(int i=1; i<=m; i++) B[++B[0]] = n+i;
    v = 0; int l = B[1];
    for(int i=2; i<=B[0]; i++) if (b[B[i]] < b[l]) l = B[i];
    if (b[l] >= 0) return true;
    double origC[MAXSIZE+1];
    memcpy(origC, c, sizeof(double)*(n+m+1));
    N[++N[0]] = 0;
    for(int i=1; i<=B[0]; i++) A[B[i]][0] = -1;
    memset(c, 0, sizeof(double)*(n+m+1));
    c[0] = -1; pivot(l, 0);
    opt();//unbounded????
    if (v < -eps) return false;//eps
    delete0();
    memcpy(c, origC, sizeof(double)*(n+m+1));
    bool inB[MAXSIZE+1];
    memset(inB, false, sizeof(bool)*(n+m+1));
    for(int i=1; i<=B[0]; i++) inB[B[i]] = true;
    for(int i=1; i<=n+m; i++)
      if (inB[i] && c[i] != 0) {
        v += c[i]*b[i];
        for(int j=1; j<=N[0]; j++) c[N[j]] -= A[i][N[j]]*c[i];
        c[i] = 0;
      }
    return true;
  }
public: void simplex(string inputName, string outputName) {
    freopen(inputName.c_str(), "r", stdin);
    freopen(outputName.c_str(), "w", stdout);
    read();
    if (!initialize()) {
      printf("Infeasible\n");
      return;
    }
    if (!opt()) {
      printf("Unbounded\n");
      return
    } else printf("Max value is %lf\n", v);
    bool inN[MAXSIZE+1];
    memset(inN, false, sizeof(bool)*(n+m+1));
    for(int i=1; i<=N[0]; i++) inN[N[i]] = true;
    for(int i=1; i<=n; i++)
      if (inN[i]) printf("x%d = %lf\n", i, 0.0);
      else printf("x%d = %lf\n", i, b[i]);
  }
};
int main() {
  LinearProgramming test;
  test.simplex("a.in", "a.out");
}
```

7 数论

7.1 离散对数

```
struct hash_table {
  static const int Mn = 100003;
  int hd[Mn], key[Mn], val[Mn], nxt[Mn], tot;
  hash_table() : tot(0) {
    memset(hd, -1, sizeof hd);
  }
  void clear() {
    memset(hd, -1, sizeof hd);
    tot = 0;
  }
  int &operator[] (const int &cur) {
    int pos = cur % Mn;
    for(int i = hd[pos]; ~i; i = nxt[i]) {
      if(key[i] == cur) {
        return val[i];
      }
    }
    nxt[tot] = hd[pos];
    hd[pos] = tot;
    key[tot] = cur;
    return val[tot++];
  }
  bool find(const int &cur) {
    int pos = cur % Mn;
    for(int i = hd[pos]; ~i; i = nxt[i]) {
      if(key[i] == cur)
        return true;
    }
    return false;
  }
};
// base ^ res = n % mod
inline int discrete_log(int base, int n, int mod) {
  int size = int(sqrt(mod)) + 1;
  hash_table hsh;
  int val = 1;
  for (int i = 0; i < size; ++i) {
    if(hsh.find(val) == 0)
      hsh[val] = i;
    val = (long long) val * base % mod;
  }
  int inv = inverse(val, mod);
  val = 1;
  for(int i = 0; i < size; ++i) {
    if(hsh.find((long long) val * n % mod))
      return i * size + hsh[(long long)val * n % mod];
    val = (long long) inv * val % mod;
  }
  return -1;
}
```

7.2 原根

x 为 p 的原根当且仅当对 $p-1$ 任意质因子 k 有 $x^k \neq 1 (\mathrm{mod}\ p)$。

7.3 Miller Rabin and Rho

```
const int bas[12]={2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
bool check(const long long &prime,const long long &base){
  long long number = prime - 1;
  for (; ~number & 1; number>>=1);
  long long result= power_mod(base, number, prime);
  for (; number != prime - 1 && result != 1 && result != prime
       - 1; number<<=1){
    result = multiply_mod(result, result, prime);
```

```
  }
  return result == prime - 1 || (number & 1) == 1 ;
}
bool miller_rabin(const long long &number){
  if  (number < 2)  return  0;
  if  (number < 4)  return  1;
  if  (~number & 1) return  0;
  for (int i = 0; i < 12 && bas[i] < number; ++i)
    if  (!check(number, bas[i])) return 0;
  return  1;
}
long long pollard_rho(const long long &number, const long long
     &seed){
  long long x = rand() % (number - 1) + 1, y = x;
  for (int head = 1, tail = 2; ; ){
    x = multiply_mod(x, x, number);
    x = add_mod(x, seed, number);
    if  (x == y)  return  number;
    long long ans = gcd(myabs(x - y), number);
    if  (ans > 1 && ans < number) return  ans;
    if  (++head == tail){
      y = x;
      tail <<= 1;
    }
  }
}
void factorize(const long long &number, vector<long long> &
     divisor){
  if  (number > 1)
    if  (miller_rabin(number))
      divisor.push_back(number);
    else{
      long long factor = number;
      for (; factor >= number; factor = pollard_rho(number,
           rand() % (number - 1) + 1));
      factorize(number / factor, divisor);
      factorize(factor, divisor);
    }
}
```

7.4 离散平方根

```
inline bool quad_resi(long long x,long long p){
  return power_mod(x, (p - 1) / 2, p) == 1;
}
struct quad_poly {
  long long zero, one, val, mod;
  quad_poly(long long zero,long long one,long long val,long
       long mod):\
    zero(zero),one(one),val(val),mod(mod) {}
  quad_poly multiply(quad_poly o){
    long long z0 = (zero * o.zero + one * o.one % mod * val %
         mod) % mod;
    long long z1 = (zero * o.one + one * o.zero) % mod;
    return quad_poly(z0, z1, val ,mod);
  }
  quad_poly pow(long long x){
    if (x == 1) return *this;
    quad_poly ret = this -> pow(x / 2);
    ret = ret.multiply(ret);
    if (x & 1) ret = ret.multiply(*this);
    return ret;
  }
};
inline long long calc_root(long long a,long long p){
  a %= p;
  if (a < 2) return a;
  if (!quad_resi(a, p)) return p;
  if (p % 4 == 3) return power_mod(a, (p + 1) / 4, p);
  long long b = 0;
  while (quad_resi((my_sqr(b, p) - a + p) % p, p)) b = rand() %
       p;
  quad_poly ret = quad_poly(b, 1, (my_sqr(b, p) - a + p) % p, p
       );
  ret = ret.pow((p + 1) / 2);
  return ret.zero;
}
void exgcd(long long a,long long b,long long &d,long long &x,
     long long &y){
  if (b == 0){
    d = a; x = 1; y = 0;
  }
  else{
    exgcd(b, a%b, d, y, x);
    y -= a / b * x;
  }
}
void solve_sqrt(long long c,long long a,long long b,long long r
     ,long long mod,vector<long long> &ans){
  long long x, y, d;
  exgcd(a, b, d, x, y);
  long long n = 2 * r;
  if (n % d == 0){
    x *= n / d;
    x = (x % (b / d) + (b / d)) % (b / d);
    long long m = x * a - r;
    while (m < mod){
      if (m >= 0 && m * m % mod == c){
        ans.push_back(m);
      }
      m += b / d * a;
    }
  }
}
void discrete_root(long long x,long long N,long long r,vector<
     long long> &ans){
  ans.clear();
  for (int i = 1; i * i <= N; ++i)
    if (N % i == 0){
      solve_sqrt(x, i, N/i, r, N, ans);
      solve_sqrt(x, N/i, i, r, N, ans);
    }
  sort(ans.begin(), ans.end());
  int sz = unique(ans.begin(),ans.end()) - ans.begin();
  ans.resize(sz);
}
```

7.5 $O(m^2\log(n))$ 求线性递推

已知 $a_0, a_1, ..., a_{m-1} a_n = c_0 * a_{n-m} + ... + c_{m-1} * a_{n-1}$ 求 $a_n = v_0 * a_0 + v_1 * a_1 + ... + v_{m-1} * a_{m-1}$

```
void linear_recurrence(long long n, int m, int a[], int c[],
     int p) {
  long long v[M] = {1 % p}, u[M << 1], msk = !!n;
  for(long long i(n); i > 1; i >>= 1) {
    msk <<= 1;
  }
  for(long long x(0); msk; msk >>= 1, x <<= 1) {
    fill_n(u, m << 1, 0);
    int b(!!(n & msk));
    x |= b;
    if(x < m) {
      u[x] = 1 % p;
    }else {
      for(int i(0); i < m; i++) {
        for(int j(0), t(i + b); j < m; j++, t++) {
          u[t] = (u[t] + v[i] * v[j]) % p;
        }
      }
      for(int i((m << 1) - 1); i >= m; i--) {
        for(int j(0), t(i - m); j < m; j++, t++) {
          u[t] = (u[t] + c[j] * u[i]) % p;
        }
      }
    }
    copy(u, u + m, v);
  }
  //a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * a[m -
       1].
  for(int i(m); i < 2 * m; i++) {
    a[i] = 0;
    for(int j(0); j < m; j++) {
      a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
    }
  }
  for(int j(0); j < m; j++) {
    b[j] = 0;
    for(int i(0); i < m; i++) {
      b[j] = (b[j] + v[i] * a[i + j]) % p;
    }
  }
  for(int j(0); j < m; j++) {
    a[j] = b[j];
  }
}
```

7.6 佩尔方程求根 $x^2 - n * y^2 = 1$

```
pair<int64, int64> solve_pell64(int64 n) {
  const static int MAXC = 111;
  int64 p[MAXC], q[MAXC], a[MAXC], g[MAXC], h[MAXC];
  p[1] = 1; p[0] = 0;
  q[1] = 0; q[0] = 1;
  a[2] = square_root(n);
  g[1] = 0; h[1] = 1;
  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 make_pair(p[i], q[i]);
  }
}
```

7.7 直线下整点个数

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

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

8 其他

8.1 某年某月某日是星期几

```
int solve(int year, int month, int day) {
    int answer;
    if (month == 1 || month == 2) {
        month += 12;
        year--;
    }
    if ((year < 1752) || (year == 1752 && month < 9) ||
        (year == 1752 && month == 9 && day < 3)) {
        answer = (day + 2 * month + 3 * (month + 1) / 5 + year
             + year / 4 + 5) % 7;
    } else {
        answer = (day + 2 * month + 3 * (month + 1) / 5 + year
             + year / 4
               - year / 100 + year / 400) % 7;
    }
    return answer;
}
```

8.2 枚举 k 子集

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

8.3 环状最长公共子串

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

```
                from[i][j] = 2;
            }
            left = max;
        }
        if (i >= n) {
            int count = 0;
            for (int x = i, y = n; y; ) {
                int t = from[x][y];
                count += t == 1;
                x += DELTA[t][0];
                y += DELTA[t][1];
            }
            ret = std::max(ret, count);
            int x = i - n + 1;
            from[x][0] = 0;
            int y = 0;
            while (y <= n && from[x][y] == 0) {
                y++;
            }
            for (; x <= i; ++x) {
                from[x][y] = 0;
                if (x == i) {
                    break;
                }
                for (; y <= n; ++y) {
                    if (from[x + 1][y] == 2) {
                        break;
                    }
                    if (y + 1 <= n && from[x + 1][y + 1] == 1)
                        {
                        y++;
                        break;
                    }
                }
            }
        }
    }
    return ret;
}
```

8.4 LL*LLmodLL

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

8.5 曼哈顿距离最小生成树

```
/*只需要考虑每个点的 pi/4*k -- pi/4*(k+1)的区间内的第一个点，这
     样只有4n条无向边。*/
const int maxn = 100000+5;
const int Inf = 1000000005;
struct TreeEdge
{
  int x,y,z;
  void make( int _x,int _y,int _z ) { x=_x; y=_y; z=_z; }
} data[maxn*4];
inline bool operator < ( const TreeEdge& x,const TreeEdge& y ){
  return x.z<y.z;
}
int x[maxn],y[maxn],px[maxn],py[maxn],id[maxn],tree[maxn],node[
     maxn],val[maxn],fa[maxn];
int n;
inline bool compare1( const int a,const int b ) { return x[a]<x
     [b]; }
inline bool compare2( const int a,const int b ) { return y[a]<y
     [b]; }
inline bool compare3( const int a,const int b ) { return (y[a]-
     x[a]<y[b]-x[b] || y[a]-x[a]==y[b]-x[b] && y[a]>y[b]); }
inline bool compare4( const int a,const int b ) { return (y[a]-
     x[a]>y[b]-x[b] || y[a]-x[a]==y[b]-x[b] && x[a]>x[b]); }
inline bool compare5( const int a,const int b ) { return (x[a]+
     y[a]>x[b]+y[b] || x[a]+y[a]==x[b]+y[b] && x[a]<x[b]); }
inline bool compare6( const int a,const int b ) { return (x[a]+
     y[a]<x[b]+y[b] || x[a]+y[a]==x[b]+y[b] && y[a]>y[b]); }
void Change_X()
{
  for(int i=0;i<n;++i) val[i]=x[i];
  for(int i=0;i<n;++i) id[i]=i;
  sort(id,id+n,compare1);
  int cntM=1, last=val[id[0]]; px[id[0]]=1;
  for(int i=1;i<n;++i)
  {
    if(val[id[i]]>last) ++cntM,last=val[id[i]];
    px[id[i]]=cntM;
  }
}
void Change_Y()
{
  for(int i=0;i<n;++i) val[i]=y[i];
  for(int i=0;i<n;++i) id[i]=i;
  sort(id,id+n,compare2);
  int cntM=1, last=val[id[0]]; py[id[0]]=1;
  for(int i=1;i<n;++i)
  {
    if(val[id[i]]>last) ++cntM,last=val[id[i]];
    py[id[i]]=cntM;
  }
}
inline int absValue( int x ) { return (x<0)?-x:x; }
inline int Cost( int a,int b ) { return absValue(x[a]-x[b])+
     absValue(y[a]-y[b]); }
int find( int x ) { return (fa[x]==x)?x:(fa[x]=find(fa[x])); }
int main()
{
//  freopen("input.txt", "r", stdin);
//  freopen("output.txt", "w", stdout);
  int test=0;
  while( scanf("%d",&n)!=EOF && n )
  {
    for(int i=0;i<n;++i) scanf("%d%d",x+i,y+i);
    Change_X();
    Change_Y();
    int cntE = 0;
    for(int i=0;i<n;++i) id[i]=i;
    sort(id,id+n,compare3);
    for(int i=1;i<=n;++i) tree[i]=Inf,node[i]=-1;
    for(int i=0;i<n;++i)
    {
      int Min=Inf, Tnode=-1;
      for(int k=py[id[i]];k<=n;k+=k&(-k)) if(tree[k]<Min) Min=
          tree[k],Tnode=node[k];
      if(Tnode>=0) data[cntE++].make(id[i],Tnode,Cost(id[i],
          Tnode));
      int tmp=x[id[i]]+y[id[i]];
      for(int k=py[id[i]];k;k-=k&(-k)) if(tmp<tree[k]) tree[k]=
          tmp,node[k]=id[i];
    }
    sort(id,id+n,compare4);
    for(int i=1;i<=n;++i) tree[i]=Inf,node[i]=-1;
    for(int i=0;i<n;++i)
    {
      int Min=Inf, Tnode=-1;
      for(int k=px[id[i]];k<=n;k+=k&(-k)) if(tree[k]<Min) Min=
          tree[k],Tnode=node[k];
      if(Tnode>=0) data[cntE++].make(id[i],Tnode,Cost(id[i],
          Tnode));
      int tmp=x[id[i]]+y[id[i]];
      for(int k=px[id[i]];k;k-=k&(-k)) if(tmp<tree[k]) tree[k]=
          tmp,node[k]=id[i];
    }
    sort(id,id+n,compare5);
    for(int i=1;i<=n;++i) tree[i]=Inf,node[i]=-1;
    for(int i=0;i<n;++i)
    {
      int Min=Inf, Tnode=-1;
      for(int k=px[id[i]];k;k-=k&(-k)) if(tree[k]<Min) Min=tree
          [k],Tnode=node[k];
      if(Tnode>=0) data[cntE++].make(id[i],Tnode,Cost(id[i],
          Tnode));
      int tmp=-x[id[i]]+y[id[i]];
      for(int k=px[id[i]];k<=n;k+=k&(-k)) if(tmp<tree[k]) tree[
          k]=tmp,node[k]=id[i];
    }
    sort(id,id+n,compare6);
    for(int i=1;i<=n;++i) tree[i]=Inf,node[i]=-1;
    for(int i=0;i<n;++i)
    {
      int Min=Inf, Tnode=-1;
      for(int k=py[id[i]];k<=n;k+=k&(-k)) if(tree[k]<Min) Min=
          tree[k],Tnode=node[k];
      if(Tnode>=0) data[cntE++].make(id[i],Tnode,Cost(id[i],
          Tnode));
      int tmp=-x[id[i]]+y[id[i]];
      for(int k=py[id[i]];k;k-=k&(-k)) if(tmp<tree[k]) tree[k]=
          tmp,node[k]=id[i];
    }
    long long Ans = 0;
    sort(data,data+cntE);
    for(int i=0;i<n;++i) fa[i]=i;
    for(int i=0;i<cntE;++i) if(find(data[i].x)!=find(data[i].y)
        )
    {
      Ans += data[i].z;
      fa[fa[data[i].x]]=fa[data[i].y];
    }
    cout<<"Case "<<++test<<": "<<"Total Weight = "<<Ans<<endl;
  }
  return 0;
}
```

8.6 极大团计数

```
void dfs(int size){
  int i, j, k, t, cnt, best = 0;
  bool bb;
  if (ne[size]==ce[size]){
    if (ce[size]==0) ++ans;
    return;
  }
  for (t=0, i=1; i<=ne[size]; ++i) {
    for (cnt=0, j=ne[size]+1; j<=ce[size]; ++j)
    if (!g[list[size][i]][list[size][j]]) ++cnt;
    if (t==0 || cnt<best) t=i, best=cnt;
  }
  if (t && best<=0) return;
  for (k=ne[size]+1; k<=ce[size]; ++k) {
    if (t>0){
      for (i=k; i<=ce[size]; ++i) if (!g[list[size][t]][list[
          size][i]]) break;
      swap(list[size][k], list[size][i]);
    }
    i=list[size][k];
    ne[size+1]=ce[size+1]=0;
    for (j=1; j<k; ++j)if (g[i][list[size][j]]) list[size+1][++
        ne[size+1]]=list[size][j];
    for (ce[size+1]=ne[size+1], j=k+1; j<=ce[size]; ++j)
    if (g[i][list[size][j]]) list[size+1][++ce[size+1]]=list[
        size][j];
    dfs(size+1);
    ++ne[size];
    --best;
    for (j=k+1, cnt=0; j<=ce[size]; ++j) if (!g[i][list[size][j
        ]]) ++cnt;
    if (t==0 || cnt<best) t=k, best=cnt;
    if (t && best<=0) break;
  }
}
void work(){
  int i;
  ne[0]=0; ce[0]=0;
  for (i=1; i<=n; ++i) list[0][++ce[0]]=i;
  ans=0;
  dfs(0);
}
```

8.7 最大团搜索

Int g[][] 为图的邻接矩阵.MC(V) 表示点集 V 的最大团. 令 Si=vi, vi+1, ..., vn, mc[i] 表示 MC(Si). 倒着算 mc[i]，那么显然 MC(V)=mc[1]. 此外有 mc[i]=mc[i+1] or mc[i]=mc[i+1]+1.

```
void init(){
  int i, j;
  for (i=1; i<=n; ++i) for (j=1; j<=n; ++j) scanf("%d", &g[i][j
      ]);
}
void dfs(int size){
  int i, j, k;
  if (len[size]==0) {
    if (size>ans) {
      ans=size; found=true;
    }
    return;
  }
  for (k=0; k<len[size] && !found; ++k) {
    if (size+len[size]-k<=ans) break;
    i=list[size][k];
    if (size+mc[i]<=ans) break;
    for (j=k+1, len[size+1]=0; j<len[size]; ++j)
    if (g[i][list[size][j]]) list[size+1][len[size+1]++]=list[
        size][j];
    dfs(size+1);
  }
}
void work(){
  int i, j;
  mc[n]=ans=1;
  for (i=n-1; i; --i) {
    found=false;
    len[1]=0;
    for (j=i+1; j<=n; ++j) if (g[i][j]) list[1][len[1]++]=j;
    dfs(1);
    mc[i]=ans;
  }
}
```

8.8 整体二分

```
void solve(int Ql, int Qr, int El, int Er) {
  if (El == Er) {
    for (int i = Ql; i <= Qr; i++)
      Q[id[i]].ans = E[El].v;
    return;
  }
  int Em = (El + Er) >> 1;
  for (int i = El; i <= Em; i++)
    modify(1, E[i].a, E[i].b, 1);
  int Qm = Ql - 1;
  for (int i = Ql; i <= Qr; i++) {
    int t = id[i];
    long long k = getcnt(1, Q[t].x, Q[t].y);
    if (k >= Q[t].k) swap(id[++Qm], id[i]);
    else Q[t].k -= k;
  }
  for (int i = El; i <= Em; i++)
    modify(1, E[i].a, E[i].b, -1);
  if (Ql <= Qm) solve(Ql, Qm, El, Em);
  if (Qm + 1 <= Qr) solve(Qm + 1, Qr, Em + 1, Er);
}
```

8.9 Dancing Links(精确覆盖及重复覆盖)

```
// HUST 1017
// 给定一个 n 行 m 列的 0/1 矩阵，选择某些行使得每一列都恰有一
    个 1
const int MAXN = 1e3 + 5;
const int MAXM = MAXN * MAXN;
const int INF = 1e9;
int ans, chosen[MAXM];
struct DancingLinks{
  int row, col, tot;
  int up[MAXM], dn[MAXM], lf[MAXM], rg[MAXM];
  int hd[MAXM], sz[MAXM];
  int posr[MAXM], posc[MAXM];
  void init(int _n, int _m){
    row = _n, col = _m;
    for(int i = 0; i <= col; ++i){
      sz[i] = 0; up[i] = dn[i] = i;
      lf[i] = i - 1; rg[i] = i + 1;
    }
    rg[col] = 0; lf[0] = col; tot = col;
    for(int i = 1; i <= row; ++i) hd[i] = -1;
  }
  void lnk(int r, int c){
    ++tot; ++sz[c];
    dn[tot] = dn[c]; up[tot] = c;
    up[dn[c]] = tot; dn[c] = tot;
    posr[tot] = r; posc[tot] = c;
    if(hd[r] < 0) hd[r] = lf[tot] = rg[tot] = tot;
    else{
      lf[tot] = hd[r]; rg[tot] = rg[hd[r]];
      lf[rg[hd[r]]] = tot; rg[hd[r]] = tot;
    }
  }
  void remove(int c){ // 删除列时删除能覆盖其的行
    rg[lf[c]] = rg[c]; lf[rg[c]] = lf[c];
    for(int i = dn[c]; i != c; i = dn[i])
      for(int j = rg[i]; j != i; j = rg[j]){
        dn[up[j]] = dn[j]; up[dn[j]] = up[j];
        --sz[posc[j]];
      }
  }
  void resume(int c){
    rg[lf[c]] = c; lf[rg[c]] = c;
    for(int i = dn[c]; i != c; i = dn[i])
      for(int j = rg[i]; j != i; j = rg[j]){
        up[dn[j]] = j; dn[up[j]] = j;
        ++sz[posc[j]];
      }
  }
  bool dance(int dpth){
    if(rg[0] == 0){
      printf("%d", dpth);
      for(int i = 0; i < dpth; ++i) printf(" %d", chosen[i]);
      puts(""); return true;
    }
    int c = rg[0];
    for(int i = rg[0]; i; i = rg[i]) if(sz[i] < sz[c]) c = i;
    remove(c); // 当前消去第c列
    for(int i = dn[c]; i != c; i = dn[i]){ // 第c列是由第i行覆
        盖的
      chosen[dpth] = posr[i];
      for(int j = rg[i]; j != i; j = rg[j]) remove(posc[j]);
      if(dance(dpth + 1)) return true;
      for(int j = lf[i]; j != i; j = lf[j]) resume(posc[j]);
    }
    resume(c);
    return false;
  }
};
DancingLinks dlx;
int n, m;
void work(){
  dlx.init(n, m);
  for(int i = 1, k, j; i <= n; ++i){
    scanf("%d", &k);
    while(k--) scanf("%d", &j), dlx.lnk(i, j);
  }
  if(!dlx.dance(0)) puts("NO");
}
// 重复覆盖
// 给定一个 n 行 m 列的 0/1 矩阵，选择某些行使得每一列至少有一
    个 1
struct DancingLinks{
  int row, col, tot;
  int up[MAXM], dn[MAXM], lf[MAXM], rg[MAXM];
  int head[MAXM], sz[MAXM];
  void init(int _n, int _m){
    row = _n, col = _m;
    for(int i = 0; i <= col; ++i){
      sz[i] = 0; up[i] = dn[i] = i;
      lf[i] = i - 1; rg[i] = i + 1;
    }
    rg[col] = 0; lf[0] = col; tot = col;
    for(int i = 1; i <= row; ++i) head[i] = -1;
  }
  void lnk(int r, int c){
    ++tot; ++sz[c];
    dn[tot] = dn[c]; up[dn[c]] = tot;
    up[tot] = c; dn[c] = tot;
    if(head[r] < 0) head[r] = lf[tot] = rg[tot] = tot;
    else{
      rg[tot] = rg[head[r]]; lf[rg[head[r]]] = tot;
      lf[tot] = head[r]; rg[head[r]] = tot;
    }
  }
  void remove(int c){ // 删除列时不删除行 因为列可被重复覆盖
    for(int i = dn[c]; i != c; i = dn[i])
      rg[lf[i]] = rg[i], lf[rg[i]] = lf[i];
  }
  void resume(int c){
    for(int i = up[c]; i != c; i = up[i])
      rg[lf[i]] = i, lf[rg[i]] = i;
  }
  void dance(int d){
    if(ans <= d) return;
    if(rg[0] == 0){ans = min(ans, d); return;}
    int c = rg[0];
    for(int i = rg[0]; i != 0; i = rg[i]) if(sz[i] < sz[c]) c =
        i;
    for(int i = dn[c]; i != c; i = dn[i]){ // 枚举c列是被哪行覆
        盖
      remove(i);
      for(int j = rg[i]; j != i; j = rg[j]) remove(j);
      dance(d + 1);
      for(int j = lf[i]; j != i; j = lf[j]) resume(j);
      resume(i);
    }
  }
};
DancingLinks dlx;
```

8.10 序列莫队

```
const int maxn = 50005;
const int maxb = 233;
int n, m, cnt[maxn], a[maxn];
long long answ[maxn], ans;
int bk, sz, bel[maxn];
int lf[maxn], rh[maxn], rnk[maxn];
bool cmp(int i, int j){
  if(bel[lf[i]] != bel[lf[j]]) return bel[lf[i]] < bel[lf[j]];
  else return bel[rh[i]] < bel[rh[j]];
}
void widden(int i){ans += cnt[a[i]]++;}
void shorten(int i){ans -= --cnt[a[i]];}
long long gcd(long long a, long long b){
  if(b == 0) return a;
  else return gcd(b, a % b);
}
int main(){
  scanf("%d%d", &n, &m);
  bk = sqrt(n); sz = n / bk;
  while(bk * sz < n) ++bk;
  for(int b = 1, i = 1; b <= bk; ++b)
    for(; i <= b * sz && i <= n; ++i) bel[i] = b;
  for(int i = 1; i <= n; ++i) scanf("%d", &a[i]);
  for(int i = 1; i <= m; ++i) scanf("%d%d", &lf[i], &rh[i]);
  for(int i = 1; i <= m; ++i) rnk[i] = i;
  sort(rnk + 1, rnk + 1 + m, cmp);
  lf[0] = rh[0] = 1; widden(1);
  for(int i = 1; i <= m; ++i){
    int k = rnk[i], kk = rnk[i-1];
    for(int j = lf[k]; j < lf[kk]; ++j) widden(j);
    for(int j = rh[k]; j > rh[kk]; --j) widden(j);
    for(int j = lf[kk]; j < lf[k]; ++j) shorten(j);
    for(int j = rh[kk]; j > rh[k]; --j) shorten(j);
    answ[k] = ans;
  }
  for(int i = 1; i <= m; ++i){
    if(answ[i] == 0){
      puts("0/1");
      continue;
    }
    int lnth = rh[i] - lf[i] + 1;
    long long t = 1LL * lnth * (lnth - 1) / 2;
    long long g = gcd(answ[i], t);
    printf("%lld/%lld\n", answ[i] / g, t / g);
  }
  return 0;
}
```

8.11 模拟退火

```
int n;
double A,B;
struct Point{
    double x,y;
    Point(){}
    Point(double x,double y):x(x),y(y){}
    void modify(){
        x = max(x,0.0);
        x = min(x,A);
        y = max(y,0.0);
        y = min(y,B);
    }
}p[1000000];
double sqr(double x){
    return x * x;
}
double Sqrt(double x){
    if(x < eps) return 0;
    return sqrt(x);
}
Point operator + (const Point &a,const Point &b){
    return Point(a.x + b.x, a.y + b.y);
}
Point operator - (const Point &a,const Point &b){
    return Point(a.x - b.x, a.y - b.y);
}
Point operator * (const Point &a,const double &k){
    return Point(a.x * k, a.y * k);
}
Point operator / (const Point &a,const double &k){
    return Point(a.x / k, a.y / k);
}
double det (const Point &a,const Point &b){
    return a.x * b.y - a.y * b.x;
}
double dist(const Point &a, const Point &b){
    return Sqrt(sqr(a.x - b.x)+sqr(a.y - b.y));
}
double work(const Point &x){
    double ans = 1e9;
    for(int i=1;i<=n;i++)
        ans = min(ans,dist(x,p[i]));
    return ans;
}
int main(){
    srand(time(NULL));
    int numcase;
    cin>>numcase;
    while (numcase--){
        scanf("%lf%lf%d",&A,&B,&n);
        for(int i=1;i<=n;i++){
            scanf("%lf%lf",&p[i].x,&p[i].y);
        }
        double total_ans = 0;
        Point total_aaa;
        for(int ii = 1;ii<=total/n;ii++){
            double ans = 0;
            Point aaa;
            Point p;
```

```
            p.x = (rand() % 10000) * A / 10000;
            p.y = (rand() % 10000) * B / 10000;
            double step = 2 * max(A,B);
            for(double T = 1e6;T > 1e-2;T = T * 0.98){
                double thi = (rand() % 10000) * pi2 / 10000;
                Point now = p + Point(cos(thi), sin(thi)) *
                    step * (rand() % 10000)/10000;
                now.modify();
                double now_ans = work(now);
                double delta = now_ans -ans;
                if(delta > 0) {
                    p = now;
                    ans = now_ans;
                    aaa = now;
                }
                else{
                    if((rand() % 10000) / 10000.0 > exp(delta /
                         T)) p = now;
                }
                step = max(step * 0.9,1e-3);
            }
            if(ans > total_ans) total_ans = ans, total_aaa =
                 aaa;
        }
        printf("The safest point is (%.1f, %.1f).\n",total_aaa.
             x,total_aaa.y);
    }
}
```

8.12 Java

```
//javac Main.java
//java Main
import java.io.*;
import java.util.*;
import java.math.*;
public class Main{
  public static BigInteger n,m;
  public static Map<BigInteger,Integer> M = new HashMap();
  public static BigInteger dfs(BigInteger x){
    if(M.get(x)!=null)return M.get(x);
    if(x.mod(BigInteger.valueOf(2))==1){
    }else{
            string p = n.toString();
    }
    M.put();
  }
    static int NNN = 1000000;
    static BigInteger N;
    static BigInteger One = new BigInteger("1");
    static BigInteger[] num_step = new BigInteger[NNN];
  public static void main(String []arg){
    Scanner cin = new Scanner(System.in);
        while(cin.hasNext())
    {
        int p = cin.nextInt();
        n = cin.nextBigInteger();
        n.multiply(m);
        M.clear();
        if(n.compareTo(BigInteger.ZERO)==0)break;
        if(n.compareTo(m)<=0){
        System.out.println(m.subtract(n));
        continue;
        }
        BigInteger[] QB = new BigInteger[5000*20];
        Integer[] QD = new Integer[5000*20];
        int head=0,tail=0;
        QB[tail]=n;
        QD[tail]=0;
        tail++;
        BigInteger ans = n.subtract(m).abs();
            if(ans.compareTo(BigInteger.valueOf(dep).add(m.
                subtract(now).abs()))>0)
                ans=BigInteger.valueOf(dep).add(m.subtract(now)
                     .abs());
            head++;
            if(now.mod(BigInteger.valueOf(2)).compareTo(
                BigInteger.ONE)!=0){
                nxt=now.divide(BigInteger.valueOf(2));
                if(M.get(nxt)==null){
                    M.put(nxt,1);
                }
            }
        System.out.println(ans);
    }
}
}
还有这样的hashset用法:
static Collection c = new HashSet();
if(c.contains(p) == false)
//读入优化
public class Main {
    BigInteger Zero = BigInteger.valueOf(0);
    BigInteger[][] a = new BigInteger[50][50];
    public void run() {
        out = new PrintWriter(System.out);
        in = new BufferedReader(new InputStreamReader(System.in
            ));
        String s;
        for ( ; ; ) {
            try {
                s = next();
                BigInteger ans = new BigInteger(s);
                ans = ans.add(Zero);
                ans = ans.subtract(Zero);
                ans = ans.multiply(ans);
                ans = ans.divide(ans);
                String t = ans.toString();
                int dig = t.length();
                if (ans.compareTo(Zero) == 1) {
                    out.println(">");
                } else if (ans.compareTo(Zero) == 0) {
                    out.println("=");
                } else if (ans.compareTo(Zero) == -1) {
                    out.println("<");
                }
            }
            catch (RuntimeException e) {break;}
        }
        out.close();
    }
    public static void main(String[] args) {new Main().run();}
    public StringTokenizer token = null;
    public BufferedReader in;
    public PrintWriter out;
    public String next() {
        while (token == null || !token.hasMoreTokens()) {
            try {token = new StringTokenizer(in.readLine());}
            catch (IOException e) {throw new RuntimeException(e
                );}
        }
        return token.nextToken();
    }
    public int nextInt() {return Integer.parseInt(next());}
    public double nextDouble() {return Double.parseDouble(next
        ());}
    public BigInteger nextBigInteger() {return new BigInteger(
        next());}
}
```

8.13 Java Rules

```
BigInteger(String val)
BigInteger(String val, int radix)
BigInteger abs()
BigInteger add(BigInteger val)
BigInteger and(BigInteger val)
BigInteger andNot(BigInteger val)
int compareTo(BigInteger val)
BigInteger divide(BigInteger val)
double doubleValue()
boolean equals(Object x)
BigInteger gcd(BigInteger val)
int hashCode()
boolean isProbablePrime(int certainty)
BigInteger  mod(BigInteger m)
BigInteger  modPow(BigInteger exponent, BigInteger m)
BigInteger  multiply(BigInteger val)
BigInteger  negate()
BigInteger  shiftLeft(int n)
BigInteger  shiftRight(int n)
String toString()
String toString(int radix)
static BigInteger valueOf(long val)
BigDecimal(BigInteger val)
BigDecimal(double / int / String val)
BigDecimal divide(BigDecimal divisor, int roundingMode)
BigDecimal divide(BigDecimal divisor, int scale, RoundingMode
     roundingMode)
```

8.14 crope

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

9 技巧

python 对拍

```
from os import system
  for i in range(1,100000):
    system("./std");
    system("./force");
    if system("diff a.out a.ans")<>0:
            break
    print i
```

关同步

```
    std::ios::sync_with_stdio(false);
```

sstream 读入

```
    char s[];
    gets(s);
    stringstream ss;
    ss << s;
    int tmp;
    while (ss >> tmp)
    // << 向ss里插入信息; >> 从ss里取出前面的信息
```

二进制文件读入 fread(地址, sizeof(数据类型), 个数, stdin) 读到文件结束!feof(stdin)

9.1 枚举子集

```
  for (int mask = (now - 1) & now; mask; mask = (mask - 1) & now)
```

9.2 真正的释放 STL 容器内存空间

```
template <typename T>
__inline void clear(T& container) {
  container.clear();  // 或者删除了一堆元素
  T(container).swap(container);
}
```

9.3 无敌的大整数相乘取模

Time complexity $O(1)$.

```
// 需要保证 x 和 y 非负
long long mult(long long x, long long y, long long MODN) {
  long long t = (x * y - (long long)((long double)x / MODN * y + 1e-3) * MODN) % MODN;
  return t < 0 ? t + MODN : t;
}
```

9.4 无敌的读入优化

```
// getchar()读入优化 << 关同步cin << 此优化
// 用isdigit()会小幅变慢
// 返回 false 表示读到文件尾
namespace Reader {
    const int L = (1 << 15) + 5;
    char buffer[L], *S, *T;
    __inline bool getchar(char &ch) {
        if (S == T) {
            T = (S = buffer) + fread(buffer, 1, L, stdin);
            if (S == T) {
        ch = EOF;
        return false;
      }
        }
    ch = *S++;
    return true;
    }
    __inline bool getint(int &x) {
    char ch; bool neg = 0;
    for (; getchar(ch) && (ch < '0' || ch > '9'); ) neg ^= ch == '-';
    if (ch == EOF) return false;
    x = ch - '0';
    for (; getchar(ch), ch >= '0' && ch <= '9'; )
      x = x * 10 + ch - '0';
    if (neg) x = -x;
    return true;
    }
}
```

9.5 梅森旋转算法

High quality pseudorandom number generator, twice as efficient as rand() with -O2. C++11 required.

```
#include <random>
int main() {
    std::mt19937 g(seed);  // std::mt19937_64
    std::cout << g() << std::endl;
}
```

10 提示

10.1 控制 cout 输出实数精度

```
std::cout << std::fixed << std::setprecision(5);
```

10.2 让 make 支持 c++11

In .bashrc or whatever:

```
export CXXFLAGS='-std=c++11 -Wall'
```

10.3 线性规划转对偶

$$\begin{array}{l}\texttt{maximize } \mathbf{c}^T\mathbf{x} \\ \texttt{subject to } \mathbf{Ax} \le \mathbf{b}, \mathbf{x} \ge 0\end{array} \Longleftrightarrow \begin{array}{l}\texttt{minimize } \mathbf{y}^T\mathbf{b} \\ \texttt{subject to } \mathbf{y}^T\mathbf{A} \ge \mathbf{c}^T, \mathbf{y} \ge 0\end{array}$$

10.4 32-bit/64-bit 随机素数

32-bit	64-bit
73550053	1249292846855685773
148898719	1701750434419805569
189560747	3605499878424114901
459874703	5648316673387803781
1202316001	6125342570814357977
1431183547	6215155308775851301
1438011109	6294606778040623451
1538762023	6347330550446020547
1557944263	7429632924303725207
1981315913	8524720079480389849

10.5 NTT 素数及其原根

Prime	Primitive root
1053818881	7
1051721729	6
1045430273	3
1012924417	5
1007681537	3

10.6 线性规划对偶

maximize $\boldsymbol{c}^T\boldsymbol{x}$, subject to $\boldsymbol{Ax} \le \boldsymbol{b}$, $\boldsymbol{x} \ge 0$.
minimize $\boldsymbol{y}^T\boldsymbol{b}$, subject to $\boldsymbol{y}^T\boldsymbol{A} \ge \boldsymbol{c}^T$, $\boldsymbol{y} \ge 0$.

10.7 博弈论相关

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

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

3. Multi-SG 游戏：可以将一堆石子分成多堆.

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

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

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

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

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

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

10.8 无向图最小生成树计数

kirchhoff 矩阵 = 度数矩阵（$i = j$, $d[i][j]$ = 度数）- 邻接矩阵（i、j 之间有边，$a[i][j] = 1$）
不同的生成树个数等于任意 n - 1 主子式行列式的绝对值

10.9 最小覆盖构造解

从 X 中所有的未盖点出发扩展匈牙利树，标记树中的所有点，则 X 中的未标记点和 Y 中的已标记点组成了所求的最小覆盖。

10.10 拉格朗日插值

$p_j(x) = \prod_{i \in I_j} \frac{x - x_i}{x_j - x_i}$ $L_n(x) = \sum_{j=1}^{n} y_i p_j(x)$

10.11 求行列式的值

行列式有很多性质，第 a 行 $*k$ 加到第 b 行上去，行列式的值不变。
三角行列式的值等于对角线元素之积。
第 a 行与第 b 行互换，行列式的值取反。
常数 * 行列式，可以把常数乘到某一行里去。
注意：全是整数并取模的话当然需要求逆元

10.12 Cayley 公式与森林计数

Cayley 公式是说，一个完全图 K_n 有 n^{n-2} 棵生成树，换句话说 n 个节点的带标号的无根树有 n^{n-2} 个。

令 $g[i]$ 表示点数为 i 的森林个数，$f[i]$ 表示点数为 i 的生成树计数（$f[i] = i^{i-2}$）那么便有

$$g[i] = \sum(g[i-j] \times cnr[i-1][j-1] \times f[j])$$

$$g[i] = \sum \frac{g[i-j] \times fac[i-1] \times f[j]}{fac[j-1] \times fac[i-j]} = fac[i-1] \times \sum(\frac{f[j]}{fac[j-1]} \times \frac{g[i-j]}{fac[i-j]})$$

10.13 常用数学公式

10.13.1 斐波那契数列

1. $fib_0=0, fib_1=1, fib_n=fib_{n-1}+fib_{n-2}$
2. $fib_{n+2}\cdot fib_n-fib_{n+1}^2=(-1)^{n+1}$
3. $fib_{-n}=(-1)^{n-1}fib_n$
4. $fib_{n+k}=fib_k\cdot fib_{n+1}+fib_{k-1}\cdot fib_n$
5. $gcd(fib_m,fib_n)=fib_{gcd(m,n)}$
6. $fib_m|fib_n^2 \Leftrightarrow nfib_n|m$

10.13.2 错排公式

1. $D_n=(n-1)(D_{n-2}-D_{n-1})=n!\cdot(1-\frac{1}{1!}+\frac{1}{2!}-\frac{1}{3!}+\ldots+\frac{(-1)^n}{n!})$

10.13.3 莫比乌斯函数

$$\mu(n)=\begin{cases}1 & 若n=1\\ (-1)^k & 若n无平方数因子，且n=p_1p_2\ldots p_k\\ 0 & 若n有大于1的平方数因数\end{cases}$$

$$\sum_{d|n}\mu(d)=\begin{cases}1 & 若n=1\\ 0 & 其他情况\end{cases}$$

$$g(n)=\sum_{d|n}f(d)\Leftrightarrow f(n)=\sum_{d|n}\mu(d)g(\frac{n}{d}), g(x)=\sum_{n=1}^{[x]}f(\frac{x}{n})\Leftrightarrow f(x)=\sum_{n=1}^{[x]}\mu(n)g(\frac{x}{n})$$

10.13.4 五边形数定理

设 $p(n)$ 是 n 的拆分数，有 $p(n)=\sum_{k\in\mathbb{Z}\setminus\{0\}}(-1)^{k-1}p\left(n-\frac{k(3k-1)}{2}\right)$

10.13.5 树的计数

1. 有根树计数：$n+1$ 个结点的有根树的个数为 $a_{n+1}=\frac{\sum_{j=1}^{n}j\cdot a_j\cdot S_{n,j}}{n}$ 其中，$S_{n,j}=\sum_{i=1}^{n/j}a_{n+1-ij}=S_{n-j,j}+a_{n+1-j}$
2. 无根树计数：当 n 为奇数时，n 个结点的无根树的个数为 $a_n-\sum_{i=1}^{n/2}a_ia_{n-i}$ 当 n 为偶数时，n 个结点的无根树的个数为 $a_n-\sum_{i=1}^{n/2}a_ia_{n-i}+\frac{1}{2}a_{\frac{n}{2}}(a_{\frac{n}{2}}+1)$
3. n 个结点的完全图的生成树个数为 n^{n-2}
4. 矩阵 - 树定理：图 G 由 n 个结点构成，设 $\boldsymbol{A}[G]$ 为图 G 的邻接矩阵、$\boldsymbol{D}[G]$ 为图 G 的度数矩阵，则图 G 的不同生成树的个数为 $\boldsymbol{C}[G]=\boldsymbol{D}[G]-\boldsymbol{A}[G]$ 的任意一个 $n-1$ 阶主子式的行列式值。

10.13.6 欧拉公式

平面图的顶点个数、边数和面的个数有如下关系：$V-E+F=C+1$ 其中，V 是顶点的数目，E 是边的数目，F 是面的数目，C 是组成图形的连通部分的数目。$V-E+F=2-2G$ 其中，G is the number of genus of surface

10.13.7 皮克定理

给定顶点坐标均是整点（或正方形格点）的简单多边形，其面积 A 和内部格点数目 i、边上格点数目 b 的关系：

$$A=i+\frac{b}{2}-1$$

10.14 平面几何公式

10.14.1 三角形和四边形的费马点

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

10.14.2 四边形

D_1, D_2 为对角线，M 对角线中点连线，A 为对角线夹角，p 为半周长

1. $a^2+b^2+c^2+d^2=D_1^2+D_2^2+4M^2$
2. $S=\frac{1}{2}D_1D_2sinA$
3. 对于圆内接四边形 $ac+bd=D_1D_2$
4. 对于圆内接四边形 $S=\sqrt{(p-a)(p-b)(p-c)(p-d)}$

10.14.3 棱台

1. 体积 $V=(A_1+A_2+\sqrt{A_1A_2})\cdot\frac{h}{3}$ A_1, A_2 为上下底面积，h 为高

10.14.4 圆台

1. 母线 $l=\sqrt{h^2+(r_1-r_2)^2}$ ，侧面积 $S=\pi(r_1+r_2)l$ ，全面积 $T=\pi r_1(l+r_1)+\pi r_2(l+r_2)$ ，体积 $V=\frac{\pi}{3}(r_1^2+r_2^2+r_1r_2)h$

10.14.5 球台

1. 侧面积 $S=2\pi rh$ ，全面积 $T=\pi(2rh+r_1^2+r_2^2)$ ，体积 $V=\frac{\pi h[3(r_1^2+r_2^2)+h^2]}{6}$

10.14.6 球扇形

1. 全面积 $T=\pi r(2h+r_0)$ h 为球冠高，r_0 为球冠底面半径，体积 $V=\frac{2}{3}\pi r^2h$

10.15 立体几何公式

10.15.1 球面三角公式

设 a,b,c 是边长，A,B,C 是所对的二面角，有余弦定理 $cosa=cosb\cdot cosc+sinb\cdot sinc\cdot cosA$ 正弦定理 $\frac{sinA}{sina}=\frac{sinB}{sinb}=\frac{sinC}{sinc}$ 三角形面积是 $A+B+C-\pi$

10.15.2 四面体体积公式

U,V,W,u,v,w 是四面体的 6 条棱，U,V,W 构成三角形，$(U,u),(V,v),(W,w)$ 互为对棱，则

$$V=\frac{\sqrt{(s-2a)(s-2b)(s-2c)(s-2d)}}{192uvw}$$

其中 $a=\sqrt{xYZ}, b=\sqrt{yZX}, c=\sqrt{zXY}, d=\sqrt{xyz}, s=a+b+c+d$

10.15.3 三次方程求根公式

对一元三次方程 $x^3+px+q=0$，令

$$A=\sqrt[3]{-\frac{q}{2}+\sqrt{(\frac{q}{2})^2+(\frac{p}{3})^3}}, B=\sqrt[3]{-\frac{q}{2}-\sqrt{(\frac{q}{2})^2+(\frac{p}{3})^3}}, \omega=\frac{(-1+\mathrm{i}\sqrt{3})}{2}$$

则 $x_j=A\omega^j+B\omega^{2j}$ (j = 0, 1, 2).

当求解 $ax^3+bx^2+cx+d=0$ 时，令 $x=y-\frac{b}{3a}$，再求解 y，即转化为 $y^3+py+q=0$ 的形式．其中，

$$p=\frac{b^2-3ac}{3a^2}, q=\frac{2b^3-9abc+27a^2d}{27a^3}$$

卡尔丹判别法：令 $\Delta=(\frac{q}{2})^2+(\frac{p}{3})^3$．当 $\Delta>0$ 时，有一个实根和一对个共轭虚根；当 $\Delta=0$ 时，有三个实根，其中两个相等；当 $\Delta<0$ 时，有三个不相等的实根．

10.15.4 椭圆

- 椭圆 $\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$，其中离心率 $e=\frac{c}{a}, c=\sqrt{a^2-b^2}$；焦点参数 $p=\frac{b^2}{a}$
- 椭圆上 (x,y) 点处的曲率半径为 $R=a^2b^2(\frac{x^2}{a^4}+\frac{y^2}{b^4})^{\frac{3}{2}}=\frac{(r_1r_2)^{\frac{3}{2}}}{ab}$，其中 r_1 和 r_2 分别为 (x,y) 与两焦点 F_1 和 F_2 的距离．
$$L_{AM}=a\int_0^{\arccos\frac{x}{a}}\sqrt{1-e^2\cos^2t}\mathrm{d}t=a\int_{\arccos\frac{x}{a}}^{\frac{\pi}{2}}\sqrt{1-e^2\sin^2t}\mathrm{d}t$$
- 椭圆的周长 $L=4a\int_0^{\frac{\pi}{2}}\sqrt{1-e^2\sin^2t}\mathrm{d}t=4aE(e,\frac{\pi}{2})$，其中
$$E(e,\frac{\pi}{2})=\frac{\pi}{2}[1-(\frac{1}{2})^2e^2-(\frac{1\times3}{2\times4})^2\frac{e^4}{3}-(\frac{1\times3\times5}{2\times4\times6})^2\frac{e^6}{5}-\cdots$$
- 设椭圆上点 $M(x,y), N(x,-y), x,y>0, A(a,0)$，原点 $O(0,0)$，扇形 OAM 的面积 $S_{OAM}=\frac{1}{2}ab\arccos\frac{x}{a}$，弓形 MAN 的面积 $S_{MAN}=ab\arccos\frac{x}{a}-xy$．
- 需要 5 个点才能确定一个圆锥曲线．
- 设 θ 为 (x,y) 点关于椭圆中心的极角，r 为 (x,y) 到椭圆中心的距离，椭圆极坐标方程：
$$x=r\cos\theta, y=r\sin\theta, r^2=\frac{b^2a^2}{b^2\cos^2\theta+a^2\sin^2\theta}$$

10.15.5 抛物线

- 标准方程 $y^2=2px$，曲率半径 $R=\dfrac{(p+2x)^{\frac{3}{2}}}{\sqrt{p}}$

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

- 弓形面积：设 M,D 是抛物线上两点，且分居一，四象限．做一条平行于 MD 且与抛物线相切的直线 L．若 M 到 L 的距离为 h．则有 $S_{MOD}=\frac{2}{3}MD\cdot h$．

10.15.6 重心

- 半径 r，圆心角为 θ 的扇形的重心与圆心的距离为 $\dfrac{4r\sin\frac{\theta}{2}}{3\theta}$

- 半径 r，圆心角为 θ 的圆弧的重心与圆心的距离为 $\dfrac{4r\sin^3\frac{\theta}{2}}{3(\theta-\sin\theta)}$

- 椭圆上半部分的重心与圆心的距离为 $\dfrac{4b}{3\pi}$

- 抛物线中弓形 MOD 的重心满足 $CQ=\frac{2}{5}PQ$，P 是直线 L 与抛物线的切点，Q 在 MD 上且 PQ 平行 x 轴，C 是重心

10.15.7 向量恒等式

- $\vec{a}\times(\vec{b}\times\vec{c})=(\vec{c}\times\vec{b})\times\vec{a}=\vec{b}(\vec{a}\cdot\vec{c})-\vec{c}(\vec{a}\cdot\vec{b})$

10.15.8 常用几何公式

- 三角形的五心
  - 重心 $\vec{G}=\frac{\vec{A}+\vec{B}+\vec{C}}{3}$，内心 $\vec{I}=\frac{a\vec{A}+b\vec{B}+c\vec{C}}{a+b+c}$，$R=\frac{2S}{a+b+c}$，外心 $x=\dfrac{\vec{A}+\vec{B}-\frac{\vec{BC}\cdot\vec{AC}}{\vec{AB}\times\vec{BC}}\vec{AB}^T}{2}$，$y=\dfrac{\vec{A}+\vec{B}+\frac{\vec{BC}\cdot\vec{AC}}{\vec{AB}\times\vec{BC}}\vec{AB}^T}{2}$，$R=\frac{abc}{4S}$，垂心 $\vec{H}=3\vec{G}-2\vec{O}$，旁心（三个）$\frac{-a\vec{A}+b\vec{B}+c\vec{C}}{-a+b+c}$

10.15.9 树的计数

- 有根数计数：令 $S_{n,j}=\sum\limits_{1\le i\le n/j}a_{n+1-ij}=S_{n-j,j}+a_{n+1-j}$

  于是，$n+1$ 个结点的有根数的总数为 $a_{n+1}=\dfrac{\sum\limits_{1\le j\le n}j\cdot a_j\cdot S_{n,j}}{n}$

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

- 无根树计数：当 n 是奇数时，则有 $a_n-\sum\limits_{1\le i\le\frac{n}{2}}a_ia_{n-i}$ 种不同的无根树

  当 n 是偶数时，则有 $a_n-\sum\limits_{1\le i\le\frac{n}{2}}a_ia_{n-i}+\frac{1}{2}a_{\frac{n}{2}}(a_{\frac{n}{2}}+1)$ 种不同的无根树

- Matrix-Tree 定理：对任意图 G，设 mat[i][i] = i 的度数，mat[i][j] = i 与 j 之间边数的相反数，则 mat[i][j] 的任意余子式的行列式就是该图的生成树个数

10.16 小知识

- lowbit 取出最低位的 1

- 勾股数：设正整数 n 的质因数分解为 $n=\prod p_i^{a_i}$，则 $x^2+y^2=n$ 有整数解的充要条件是 n 中不存在形如 $p_i\equiv 3 \pmod 4$ 且指数 a_i 为奇数的质因数 p_i．$(\frac{a-b}{2})^2+ab=(\frac{a+b}{2})^2$．

- 素勾股数：若 m 和 n 互质，而且 m 和 n 中有一个是偶数，则 $a=m^2-n^2$，$b=2mn$，$c=m^2+n^2$，则 a、b、c 是素勾股数．

- Stirling 公式：$n!\approx\sqrt{2\pi n}(\frac{n}{e})^n$

- Mersenne 素数：p 是素数且 2^p-1 的数是素数．(10000 以内的 p 有：2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217, 4253, 4423, 9689, 9941)

- 序列差分表：差分表的第 0 条对角线确定原序列．设原序列为 h_i，第 0 条对角线为 $c_0,c_1,\ldots,c_p,0,0,\ldots$．有这样两个公式：$h_n=\binom{n}{0}c_0+\binom{n}{1}c_1+\ldots+\binom{n}{p}c_p$，$\sum_{k=0}^{n}h_k=\binom{n+1}{1}c_0+\binom{n+1}{2}c_2+\ldots+\binom{n+1}{p+1}c_p$

- GCD：$\gcd(2^a-1,2^b-1)=2^{\gcd(a,b)}-1$

- Fermat 分解算法：从 $t=\sqrt{n}$ 开始，依次检查 $t^2-n,(t+1)^2-n,(t+2)^2-n,\ldots$，直到出现一个平方数 y，由于 $t^2-y^2=n$，因此分解得 $n=(t-y)(t+y)$．显然，当两个因数很接近时这个方法能很快找到结果，但如果遇到一个素数，则需要检查 $\frac{n+1}{2}-\sqrt{n}$ 个整数

- 牛顿迭代：$x_1=x_0-\frac{f(x_0)}{f'(x_0)}$

- 球与盒子的动人故事：（n 个球，m 个盒子，S 为第二类斯特林数）
  球同，盒同，无空：dp；球同，盒同，可空：dp；球同，盒不同，无空：$\binom{n-1}{m-1}$；球同，盒不同，可空：$\binom{n+m-1}{n-1}$；球不同，盒同，无空：$S(n,m)$；球不同，盒同，可空：$\sum_{k=1}^{m}S(n,k)$；球不同，盒不同，无空：$m!S(n,m)$；球不同，盒不同，可空：m^n；

- 组合数奇偶性：若 $(n\&m)=m$，则 $\binom{n}{m}$ 为奇数，否则为偶数

- 格雷码 $G(x)=x\otimes(x>>1)$

- Fibonacci 数：
  - $F_0=F_1=1$，$F_i=F_{i-1}+F_{i-2}$，$F_{-i}=(-1)^{i-1}F_i$
  - $F_i=\dfrac{1}{\sqrt{5}}((\dfrac{1+\sqrt{5}}{2})^n-(\dfrac{1-\sqrt{5}}{2})^n)$
  - $\gcd(F_n,F_m)=F_{\gcd(n,m)}$
  - $F_{i+1}F_i-F_i^2=(-1)^i$
  - $F_{n+k}=F_kF_{n+1}+F_{k-1}F_n$

- 第一类 Stirling 数：$\left[{n\atop k}\right]$ 代表第一类无符号 Stirling 数，代表将 n 阶置换群中有 k 个环的置换个数；$s(n,k)$ 代表有符号型，$s(n,k)=(-1)^{n-k}\left[{n\atop k}\right]$．
  - $(x)^{(n)}=\sum\limits_{k=0}^{n}\left[{n\atop k}\right]x^k$，$(x)_n=\sum\limits_{k=0}^{n}s(n,k)x^k$
  - $\left[{n\atop k}\right]=n\left[{n-1\atop k}\right]+\left[{n-1\atop k-1}\right]$，$\left[{0\atop 0}\right]=1$，$\left[{n\atop 0}\right]=\left[{0\atop n}\right]=0$
  - $\left[{n\atop n-2}\right]=\frac{1}{4}(3n-1)\binom{n}{3}$，$\left[{n\atop n-3}\right]=\binom{n}{2}\binom{n}{4}$
  - $\sum\limits_{k=0}^{a}\left[{n\atop k}\right]=n!-\sum\limits_{k=0}^{n}\left[{n\atop k+a+1}\right]$
  - $\sum\limits_{p=k}^{n}\left[{n\atop p}\right]\binom{p}{k}=\left[{n+1\atop k+1}\right]$

- 第二类 Stirling 数：$\left\{{n\atop k}\right\}=S(n,k)$ 代表 n 个不同的球，放到 k 个相同的盒子里，盒子非空．
  - $\left\{{n\atop k}\right\}=\frac{1}{k!}\sum\limits_{j=0}^{k}(-1)^j\binom{k}{j}(k-j)^n$
  - $\left\{{n+1\atop k}\right\}=k\left\{{n\atop k}\right\}+\left\{{n\atop k-1}\right\}$，$\left\{{0\atop 0}\right\}=1$，$\left\{{n\atop 0}\right\}=\left\{{0\atop n}\right\}=0$
  - 奇偶性：$(n-k)\&\frac{k-1}{2}==0$

- Bell 数：B_n 代表将 n 个元素划分成若干个非空集合的方案数
  - $B_0=B_1=1$，$B_n=\sum\limits_{k=0}^{n-1}\binom{n-1}{k}B_k$
  - $B_n=\sum\limits_{k=0}^{n}\left\{{n\atop k}\right\}$
  - Bell 三角形：$a_{1,1}=1$，$a_{n,1}=a_{n-1,n-1}$，$a_{n,m}=a_{n,m-1}+a_{n-1,m-1}$，$B_n=a_{n,1}$
  - 对质数 p，$B_{n+p}\equiv B_n+B_{n+1} \pmod p$

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

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

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