Dracarys

Team Referrence Library

May 10, 2016

Contents

多边形与圆面积交

```cpp
point ORI;
double r;
int n;
point info[maxn];
// 用有向面积，划分成一个三角形和圆的面积的交
double area2(point pa, point pb) {
  if (pa.len() < pb.len()) swap(pa, pb);
  if (pb.len() < eps) return 0;
  double a, b, c, B, C, sinB, cosB, sinC, cosC, S, h, theta;
  a = pb.len();
  b = pa.len();
  c = (pb - pa).len();
  cosB = dot(pb, pb - pa) / a / c;
  B = acos(cosB);
  cosC = dot(pa, pb) / a / b;
  C = acos(cosC);
  if (a > r) {
    S = (C/2)*r*r;
    h = a*b*sin(C)/c;
    if (h < r && B < PI/2) S -= (acos(h/r)*r*r - h*sqrt(r*r-h*h));
  } else if (b > r) {
    theta = PI - B - asin(sin(B)/r*a);
    S = .5*a*r*sin(theta) + (C-theta)/2*r*r;
  } else {
    S = .5*sin(C)*a*b;
  }
  //printf("res = %.4f\n", S);
  return S;
}
double area() {
  double S = 0;
  for (int i = 0; i < n; ++i) {
    S += area2(info[i], info[i + 1]) * Sign(cross(info[i], info[i + 1]));
  }
  return fabs(S);
}
```

二维几何

```cpp
#include <iostream>
#include <cmath>
#include <vector>

using namespace std;

const double PI = acos(-1.0);
const double EPS = 1e-8;

int sign(double x)
{
  return x < -EPS ? -1 : x > EPS;
}

double newSqrt(double x)
{
  return x < 0 ? 0 : sqrt(x);
}

struct Point {
  double x, y;
  Point(double x = 0, double y = 0) : x(x), y(y) {}
  Point operator + (const Point &that) const {
    return Point(x + that.x, y + that.y);
  }
  Point operator - (const Point &that) const {
    return Point(x - that.x, y - that.y);
  }
  Point operator * (const double &that) const {
    return Point(x * that, y * that);
  }
  Point operator / (const double &that) const {
    return Point(x / that, y / that);
  }
  Point rotate(const double ang) { // 逆时针旋转 ang 弧度
    return Point(cos(ang) * x - sin(ang) * y, cos(ang) * y + sin(ang) * x);
  }
  Point turn90() { // 逆时针旋转 90 度
    return Point(-y, x);
  }
  double len2() const {
    return x * x + y * y;
  }
  double len() const {
    return sqrt(x * x + y * y);
  }
  Point unit() const {
    return *this / len();
  }
};
double det(Point a, Point b)
```

```cpp
{
  return a.x * b.y - b.x * a.y;
}
double dot(Point a, Point b)
{
  return a.x * b.x + a.y * b.y;
}

struct Line {
  Point a, b;
  Line(Point a, Point b) : a(a), b(b) {}
};

Point isLL(const Line &l0, const Line &l1) {
  double s0 = det(l1.b - l1.a, l0.a - l1.a),
       s1 = -det(l1.b - l1.a, l0.b - l1.a);
  return (l0.a * s1 + l0.b * s0) / (s0 + s1);
}
bool onSeg(const Line &l, const Point &p) { // 点在线段上
  return sign(det(p - l.a, l.b - l.a)) == 0 && sign(dot(p - l.a, p - l.b)) <= 0;
}
Point projection(const Line &l, const Point &p) { // 点到直线投影
  return l.a + (l.b - l.a) * (dot(p - l.a, l.b - l.a) / (l.b - l.a).len2());
}
double disToLine(const Line &l, const Point &p) {
  return abs(det(p - l.a, l.b - l.a) / (l.b - l.a).len());
}
double disToSeg(const Line &l, const Point &p) { // 点到线段距离
  return sign(dot(p - l.a, l.b - l.a)) * sign(dot(p - l.b, l.a - l.b)) != 1 ?
    disToLine(l, p) : min((p - l.a).len(), (p - l.b).len());
}
Point symmetryPoint(const Point a, const Point b) { // 点 b 关于点 a 的中心对称点
  return a + a - b;
}
Point reflection(const Line &l, const Point &p) { // 点关于直线的对称点
  return symmetryPoint(projection(l, p), p);
}
struct Circle {
  Point o;
  double r;
  Circle (Point o = Point(0, 0), double r = 0) : o(o), r(r) {}
};
// 求圆与直线的交点
bool isCL(Circle a, Line l, Point &p1, Point &p2) {
  if (sign(det(l.a - a.o, l.b - a.o) / (l.a - l.b).len()) > 0) return false;
  Point o = isLL(Line(a.o, a.o + (l.b - l.a).turn90()), l);
  Point delta = (l.b - l.a).unit() * newSqrt(a.r * a.r - (o - a.o).len2());
  p1 = o + delta;
  p2 = o - delta;
  return true;
}

// 求圆与圆的交面积
double areaCC(const Circle &c1, const Circle &c2) {
  double d = (c1.o - c2.o).len();
  if (sign(d - (c1.r + c2.r)) > 0) {
    return 0;
  }
  if (sign(d - abs(c1.r - c2.r)) < 0) {
    double r = min(c1.r, c2.r);
    return r * r * PI;
  }
  double x = (d * d + c1.r * c1.r - c2.r * c2.r) / (2 * d),
       t1 = acos(x / c1.r), t2 = acos((d - x) / c2.r);
  return c1.r * c1.r * t1 + c2.r * c2.r * t2 - d * c1.r * sin(t1);
}

// 求圆与圆的交点，注意调用前要先判定重圆
bool isCC(Circle a, Circle b, Point &p1, Point &p2) {
  double s1 = (a.o - b.o).len();
  if (sign(s1 - a.r - b.r) > 0 || sign(s1 - abs(a.r - b.r)) < 0) return false;
  double s2 = (a.r * a.r - b.r * b.r) / s1;
  double aa = (s1 + s2) * 0.5, bb = (s1 - s2) * 0.5;
  Point o = (b.o - a.o) * (aa / (aa + bb)) + a.o;
  Point delta = (b.o - a.o).unit().turn90() * newSqrt(a.r * a.r - aa * aa);
  p1 = o + delta, p2 = o - delta;
  return true;
}

// 求点到圆的切点，按关于点的左手方向返回两个点
bool tanCP(const Circle &c, const Point &p0, Point &p1, Point &p2)
{
  double x = (p0 - c.o).len2(), d = x - c.r * c.r;
  if (d < EPS) return false;
  Point p = (p0 - c.o) * (c.r * c.r / x);
  Point delta = ((p0 - c.o) * (-c.r * sqrt(d) / x)).turn90();
  p1 = c.o + p + delta;
  p2 = c.o + p - delta;
  return true;
}
```

```cpp

// 求圆到圆的外共切线，按关于 c1.o 的左手方向返回两条线
vector<Line> extanCC(const Circle &c1, const Circle &c2)
{
  vector<Line> ret;
  if (sign(c1.r - c2.r) == 0) {
    Point dir = c2.o - c1.o;
    dir = (dir * (c1.r / dir.len())).turn90();
    ret.push_back(Line(c1.o + dir, c2.o + dir));
    ret.push_back(Line(c1.o - dir, c2.o - dir));
  } else {
    Point p = (c1.o * -c2.r + c2.o * c1.r) / (c1.r - c2.r);
    Point p1, p2, q1, q2;
    if (tanCP(c1, p, p1, p2) && tanCP(c2, p, q1, q2)) {
      if (c1.r < c2.r) swap(p1, p2), swap(q1, q2);
      ret.push_back(Line(p1, q1));
      ret.push_back(Line(p2, q2));
    }
  }
  return ret;
}

// 求圆到圆的内共切线，按关于 c1.o 的左手方向返回两条线
vector<Line> intanCC(const Circle &c1, const Circle &c2)
{
  vector<Line> ret;
  Point p = (c1.o * c2.r + c2.o * c1.r) / (c1.r + c2.r);
  Point p1, p2, q1, q2;
  if (tanCP(c1, p, p1, p2) && tanCP(c2, p, q1, q2)) {
    ret.push_back(Line(p1, q1));
    ret.push_back(Line(p2, q2));
  }
  return ret;
}


int main()
{
  return 0;
}
```

$n \log n$ 半平面交

```cpp
#define cross(p1,p2,p3)((p2.x-p1.x)*(p3.y-p1.y)-(p3.x-p1.x)*(p2.y-p1.y))
#define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))
Point isSS(Point p1,Point p2,Point q1,Point q2){
  double a1=cross(q1,q2,p1),a2=-cross(q1,q2,p2);
  return(p1*a2+p2*a1)/(a1+a2);
}
struct Border{
  void setAlpha(){ alpha=atan2(p2.y-p1.y,p2.x-p1.x);}
}border[MAX_N_BORDER];
int n;
bool operator<(const Border&a,const Border&b){
  int c=sign(a.alpha-b.alpha);
  if(c!=0) return c==1;
  return crossOp(b.p1,b.p2,a.p1)>=0;
}
bool operator==(const Border&a,const Border&b){ return sign(a.alpha-b.alpha)==0;}
void add(double x,double y,double nx,double ny){
  border[n].p1=Point(x,y);border[n].p2=Point(nx,ny);
  border[n].setAlpha();n++;
}
Point isBorder(const Border&a,const Border&b){ return isSS(a.p1,a.p2,b.p1,b.p2);}
Border que[MAX_N_BORDER]; int qh,qt;
bool check(const Border&a,const Border&b,const Border&me){
  Point is=isBorder(a,b); return crossOp(me.p1,me.p2,is)>0;
}
void convexIntersection(){
  qh=qt=0; sort(border,border+n); n=unique(border,border+n)-border;
  for(int i=0;i<n;++i){
    Border cur=border[i];
    while(qh+1<qt&&!check(que[qt-2],que[qt-1],cur)) --qt;
    while(qh+1<qt&&!check(que[qh],que[qh+1],cur)) ++qh;
    que[qt++]=cur;
  }
  while(qh+1<qt&&!check(que[qt-2],que[qt-1],que[qh])) --qt;
  while(qh+1<qt&&!check(que[qh],que[qh+1],que[qt-1])) ++qh;
}
void calcArea(){
  static Point ps[MAX_N_BORDER]; int cnt=0;
  if(qt-qh<=2){ puts("0.0"); return; }
  for(int i=qh;i<qt;++i){
    int next=i+1==qt?qh:i+1; ps[cnt++]=isBorder(que[i],que[next]);
  }
  double area=0;
  for(int i=0;i<cnt;++i) area += ps[i].det(ps[(i+1) % cnt]);
  area/=2; area=fabsl(area);
  cout.setf(ios::fixed); cout.precision(1); cout<<area<<endl;
}
void halfPlaneIntersection(){
  cin>>n; for(int i=0;i<n;++i) border[i].read();
  add(0,0,LARGE,0); add(LARGE,0,LARGE,LARGE);
  add(LARGE,LARGE,0,LARGE); add(0,LARGE,0,0);
  convexIntersection(); calcArea();
}
```

Delaunay 三角剖分

```cpp
/*
Delaunay Triangulation 随机增量算法 :
节点数至少为点数的 6 倍, 空间消耗较大注意计算内存使用
建图的过程在 build 中, 注意初始化内存池和初始三角形的坐标范围 (Triangulation::LOTS)
Triangulation::find 返回包含某点的三角形
Triangulation::add_point 将某点加入三角剖分
某个 Triangle 在三角剖分中当且仅当它的 has_children 为 0
如果要找到三角形 u 的邻域, 则枚举它的所有 u.edge[i].tri, 该条边的两个点为 u.p[(i+1)%3],
  ↪ u.p[(i+2)%3]
*/
const int N = 100000 + 5;
const int MAX_TRIS = N * 6;
const double EPSILON = 1e-6;
const double PI = acos(-1.0);
using namespace std;

struct Point {
  double x,y;
  Point() : x(0), y(0) {}
  Point(double x, double y) : x(x), y(y) {}
  inline bool operator == (Point const& that) const {
    return x == that.x && y == that.y;
  }
};

inline double sqr(double x) { return x*x; }
double dist_sqr(Point const& a, Point const& b) {
  return sqr(a.x-b.x) + sqr(a.y-b.y);
}
bool in_circumcircle(Point const& p1, Point const& p2, Point const& p3, Point const&
  ↪ p4) {
  double u11 = p1.x - p4.x;
  double u21 = p2.x - p4.x;
  double u31 = p3.x - p4.x;
  double u12 = p1.y - p4.y;
  double u22 = p2.y - p4.y;
  double u32 = p3.y - p4.y;
  double u13 = sqr(p1.x) - sqr(p4.x) + sqr(p1.y) - sqr(p4.y);
  double u23 = sqr(p2.x) - sqr(p4.x) + sqr(p2.y) - sqr(p4.y);
  double u33 = sqr(p3.x) - sqr(p4.x) + sqr(p3.y) - sqr(p4.y);
  double det = -u13*u22*u31 + u12*u23*u31 + u13*u21*u32 - u11*u23*u32 - u12*u21*u33
    ↪ + u11*u22*u33;
  return det > EPSILON;
}
double side(Point const& a, Point const& b, Point const& p) {
  return (b.x-a.x)*(p.y-a.y) - (b.y-a.y)*(p.x-a.x);
}

typedef int SideRef;
struct Triangle;
typedef Triangle* TriangleRef;
struct Edge {
  TriangleRef tri;
  SideRef     side;
  Edge() : tri(0), side(0) {}
  Edge(TriangleRef tri, SideRef side) : tri(tri), side(side) {}
};
struct Triangle {
  Point p[3];
  Edge  edge[3];
  TriangleRef children[3];
  Triangle() {}
  Triangle(Point const& p0, Point const& p1, Point const& p2) {
    p[0] = p0; p[1] = p1; p[2] = p2;
    children[0] = children[1] = children[2] = 0;
  }
  bool has_children() const {
    return children[0] != 0;
  }
  int num_children() const {
    return children[0] == 0 ? 0
      : children[1] == 0 ? 1
      : children[2] == 0 ? 2 : 3;
  }
  bool contains(Point const& q) const {
    double a = side(p[0],p[1],q);
    double b = side(p[1],p[2],q);
    double c = side(p[2],p[0],q);
    return a >= -EPSILON && b >= -EPSILON && c >= -EPSILON;
  }
} triange_pool[MAX_TRIS], *tot_triangles;
void set_edge(Edge a, Edge b) {
  if (a.tri) a.tri->edge[a.side] = b;
  if (b.tri) b.tri->edge[b.side] = a;
  if (a.tri && b.tri) {
    assert(a.tri->p[(a.side+1)%3] == b.tri->p[(b.side+2)%3]);
    assert(a.tri->p[(a.side+2)%3] == b.tri->p[(b.side+1)%3]);
  }
}
class Triangulation {
  public:
    Triangulation() {
      const double LOTS = 1e6;
      the_root = new(tot_triangles++)
        ↪ Triangle(Point(-LOTS,-LOTS),Point(+LOTS,-LOTS),Point(0,+LOTS));
    }
    ~Triangulation() {}
    TriangleRef find(Point p) const {
      return find(the_root,p);
    }
    void add_point(Point const& p) {
      add_point(find(the_root,p),p);
    }
  private:
    TriangleRef the_root;
    static TriangleRef find(TriangleRef root, Point const& p) {
      for( ; ; ) {
        assert(root->contains(p));
        if (!root->has_children()) {
          return root;
        } else {
          int flag = true;
          for (int i = 0; i < 3 && root->children[i] ; ++i) {
            if (root->children[i]->contains(p)) {
              root = root->children[i];
              break;
            }
          }
          assert(flag&&"point not found");
        }
      }
    }
    void add_point(TriangleRef root, Point const& p) {
      TriangleRef tab,tbc,tca;
      /* split it into three triangles */
      tab = new(tot_triangles++) Triangle(root->p[0], root->p[1], p);
      tbc = new(tot_triangles++) Triangle(root->p[1], root->p[2], p);
      tca = new(tot_triangles++) Triangle(root->p[2], root->p[0], p);
      set_edge(Edge(tab,0), Edge(tbc,1));
      set_edge(Edge(tbc,0), Edge(tca,1));
      set_edge(Edge(tca,0), Edge(tab,1));
      set_edge(Edge(tab,2), root->edge[2]);
      set_edge(Edge(tbc,2), root->edge[0]);
      set_edge(Edge(tca,2), root->edge[1]);
      root->children[0] = tab;
      root->children[1] = tbc;
      root->children[2] = tca;
      flip(tab,2);
      flip(tbc,2);
      flip(tca,2);
    }
    void flip(TriangleRef tri, SideRef pi) {
      TriangleRef trj = tri->edge[pi].tri;
      int pj = tri->edge[pi].side;
      if (!trj) return;
      if (!in_circumcircle(tri->p[0],tri->p[1],tri->p[2],trj->p[pj])) return;
      assert(tri->p[(pi+2)%3] == trj->p[(pj+1)%3]);
      assert(tri->p[(pi+1)%3] == trj->p[(pj+2)%3]);
      /* flip edge between tri,trj */
      TriangleRef trk = new(tot_triangles++) Triangle(tri->p[(pi+1)%3], trj->p[pj],
        ↪ tri->p[pi]);
      TriangleRef trl = new(tot_triangles++) Triangle(trj->p[(pj+1)%3], tri->p[pi],
        ↪ trj->p[pj]);
      set_edge(Edge(trk,0), Edge(trl,0));
      set_edge(Edge(trk,1), tri->edge[(pi+2)%3]);
      set_edge(Edge(trk,2), trj->edge[(pj+1)%3]);
      set_edge(Edge(trl,1), trj->edge[(pj+2)%3]);
      set_edge(Edge(trl,2), tri->edge[(pi+1)%3]);
      tri->children[0] = trk; tri->children[1] = trl; tri->children[2] = 0;
      trj->children[0] = trk; trj->children[1] = trl; trj->children[2] = 0;
      flip(trk,1);
      flip(trk,2);
      flip(trl,1);
      flip(trl,2);
    }
};

int n;
Point ps[N];

void build()
{
  tot_triangles = triange_pool;
  cin >> n;
  for(int i = 0; i < n; ++ i) {
    int x, y;
    scanf("%d%d", &x, &y);
    ps[i].x = x; ps[i].y = y;
  }
  random_shuffle(ps, ps + n);
  Triangulation tri;
  for(int i = 0; i < n; ++ i) {
    tri.add_point(ps[i]);
  }
}

int main()
{
  build();
  return 0;
}
```

三维几何操作合并

```
struct Point3D {
  double x, y, z;
};

Point3D det(const Point3D &a, const Point3D &b) {
  return Point3D(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
}
// 平面法向量 : 平面上两个向量叉积
// 点共平面 : 平面上一点与之的向量点积法向量为 0
// 点在线段 ( 直线 ) 上 : 共线且两边点积非正
// 点在三角形内 ( 不包含边界，需再判断是与某条边共线 )
bool pointInTri(const Point3D &a, const Point3D &b, const Point3D &c, const Point3D &p) {
  return sign(det(a - b, a - c).len() - det(p - a, p - b).len() - det(p - b, p - c).len() - det(p - c, p - a).len()) == 0;
}
// 共平面的两点是否在这平面上一条直线的同侧
bool sameSide(const Point3D &a, const Point3D &b, const Point3D &p0, const Point3D &p1) {
  return sign(dot(det(a - b, p0 - b), det(a - b, p1 - b))) > 0;
}
// 两点在平面同侧 : 点积法向量符号相同
// 两直线平行 / 垂直 : 同二维
// 平面平行 / 垂直 : 判断法向量
// 线面垂直 : 法向量和直线平行
// 判断空间线段是否相交 : 四点共面两线段不平行相互在异侧
// 线段和三角形是否相交 : 线段在三角形平面不同侧 三角形任意两点在线段和第三点组成的平面的不同侧
// 求空间直线交点
Point3D intersection(const Point3D &a0, const Point3D &b0, const Point3D &a1, const Point3D &b1) {
  double t = ((a0.x - a1.x) * (a1.y - b1.y) - (a0.y - a1.y) * (a1.x - b1.x)) / ((a0.x - b0.x) * (a1.y - b1.y) - (a0.y - b0.y) * (a1.x - b1.x));
  return a0 + (b0 - a0) * t;
}
// 求平面和直线的交点
Point3D intersection(const Point3D &a, const Point3D &b, const Point3D &c, const Point3D &l0, const Point3D &l1) {
  Point3D p = pVec(a, b, c); // 平面法向量
  double t = (p.x * (a.x - l0.x) + p.y * (a.y - l0.y) + p.z * (a.z - l0.z)) / (p.x * (l1.x - l0.x) + p.y * (l1.y - l0.y) + p.z * (l1.z - l0.z));
  return l0 + (l1 - l0) * t;
}
// 求平面交线 : 取不平行的一条直线的一个交点，以及法向量叉积得到直线方向
// 点到直线距离 : 叉积得到三角形的面积除以底边
// 点到平面距离 : 点积法向量
// 直线间距离 : 平行时随便取一点求距离，否则叉积方向向量得到方向点积计算长度
// 直线夹角 : 点积 平面夹角 : 法向量点积
// 三维向量旋转操作(绕向量 s 旋转 ang 角度)，对于右手系 s 指向观察者时逆时针
// 矩阵版
void rotate(const Point3D &s, double ang) {
  double l = s.len(), x = s.x / l, y = s.y / l, z = s.z / l, sinA = sin(ang), cosA = cos(ang);
  double p[4][4]= {CosA + (1 - CosA) * x * x, (1 - CosA) * x * y - SinA * z, (1 - CosA) * x * z + SinA * y, 0,
    (1 - CosA) * y * x + SinA * z, CosA + (1 - CosA) * y * y, (1 - CosA) * y * z - SinA * x, 0,
    (1 - CosA) * z * x - SinA * y, (1 - CosA) * z * y + SinA * x, CosA + (1 - CosA) * z * z, 0,
    0, 0, 0, 1 };
}
// 计算版 : 把需要旋转的向量按照 s 分解，做二维旋转，再回到三维
```

三维凸包

```
#define SIZE(X) (int(X.size()))
#define PI 3.14159265358979323846264338327950288
struct Point {
  Point cross(const Point &p) const
  { return Point(y * p.z - z * p.y, z * p.x - x * p.z, x * p.y - y * p.x); }
```

```cpp
} info[1005];
int mark[1005][1005],n, cnt;;
double mix(const Point &a, const Point &b, const Point &c)
{ return a.dot(b.cross(c)); }
double area(int a, int b, int c)
{ return ((info[b] - info[a]).cross(info[c] - info[a])).length(); }
double volume(int a, int b, int c, int d)
{ return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]); }
struct Face {
  int a, b, c; Face() {}
  Face(int a, int b, int c): a(a), b(b), c(c) {}
  int &operator [](int k)
  { if (k == 0) return a; if (k == 1) return b; return c; }
};
vector <Face> face;
inline void insert(int a, int b, int c) { face.push_back(Face(a, b, c)); }
void add(int v) {
  vector <Face> tmp; int a, b, c; cnt++;
  for (int i = 0; i < SIZE(face); i++) {
    a = face[i][0]; b = face[i][1]; c = face[i][2];
    if (Sign(volume(v, a, b, c)) < 0)
    mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] = mark[a][c] =
      ↪ cnt;
    else tmp.push_back(face[i]);
  } face = tmp;
  for (int i = 0; i < SIZE(tmp); i++) {
    a = face[i][0]; b = face[i][1]; c = face[i][2];
    if (mark[a][b] == cnt) insert(b, a, v);
    if (mark[b][c] == cnt) insert(c, b, v);
    if (mark[c][a] == cnt) insert(a, c, v);
}}
int Find() {
  for (int i = 2; i < n; i++) {
    Point ndir = (info[0] - info[i]).cross(info[1] - info[i]);
    if (ndir == Point()) continue; swap(info[i], info[2]);
    for (int j = i + 1; j < n; j++) if (Sign(volume(0, 1, 2, j)) != 0) {
      swap(info[j], info[3]); insert(0, 1, 2); insert(0, 2, 1); return 1;
} } return 0; }
int main() {
  for (; scanf("%d", &n) == 1; ) {
    for (int i = 0; i < n; i++) info[i].Input();
    sort(info, info + n); n = unique(info, info + n) - info;
    face.clear(); random_shuffle(info, info + n);
    if (Find()) { memset(mark, 0, sizeof(mark)); cnt = 0;
      for (int i = 3; i < n; i++) add(i); vector<Point> Ndir;
      for (int i = 0; i < SIZE(face); ++i) {
        Point p = (info[face[i][0]] - info[face[i][1]]).cross(
            info[face[i][2]] - info[face[i][1]]);
        p = p / p.length(); Ndir.push_back(p);
      } sort(Ndir.begin(), Ndir.end());
      int ans = unique(Ndir.begin(), Ndir.end()) - Ndir.begin();
      printf("%d\n", ans);
    } else printf("1\n");
} }
// 求重心
double calcDist(const Point &p, int a, int b, int c)
{ return fabs(mix(info[a] - p, info[b] - p, info[c] - p) / area(a, b, c)); }
//compute the minimal distance of center of any faces
double findDist() { //compute center of mass
  double totalWeight = 0; Point center(.0, .0, .0);
  Point first = info[face[0][0]];
  for (int i = 0; i < SIZE(face); ++i) {
    Point p = (info[face[i][0]]+info[face[i][1]]+info[face[i][2]]+first)*.25;
    double weight = mix(info[face[i][0]] - first, info[face[i][1]]
        - first, info[face[i][2]] - first);
    totalWeight += weight; center = center + p * weight;
  } center = center / totalWeight;
  double res = 1e100; //compute distance
  for (int i = 0; i < SIZE(face); ++i)
    res = min(res, calcDist(center, face[i][0], face[i][1], face[i][2]));
    return res; }
```

凸包上快速询问

```cpp
/*
   给定凸包, log n 内完成各种询问, 具体操作有 :
   1. 判定一个点是否在凸包内
   2. 询问凸包外的点到凸包的两个切点
   3. 询问一个向量关于凸包的切点
   4. 询问一条直线和凸包的交点
   INF 为坐标范围, 需要定义点类大于号
   改成实数只需修改 sign 函数，以及把 long long 改为 double 即可
   构造函数时传入凸包要求无重点, 面积非空, 以及 pair(x,y) 的最小点放在第一个
*/
#include <vector>
#include <functional>
using namespace std;

const int INF = 1000000000;

struct Convex
```

```cpp
{
  int n;
  vector<Point> a;
  vector<Point> upper, lower;
  Convex(vector<Point> _a) : a(_a) {
    n = a.size();
    int ptr = 0;
    for(int i = 1; i < n; ++ i) if (a[ptr] < a[i]) ptr = i;
    for(int i = 0; i <= ptr; ++ i) lower.push_back(a[i]);
    for(int i = ptr; i < n; ++ i) upper.push_back(a[i]);
    upper.push_back(a[0]);
  }
  int sign(long long x) {
    return x < 0 ? -1 : x > 0;
  }
  pair<long long, int> get_tangent(vector<Point> &convex, Point vec) {
    int l = 0, r = (int)convex.size() - 2;
    for( ; l + 1 < r; ) {
      int mid = (l + r) / 2;
      if (sign((convex[mid + 1] - convex[mid]).det(vec)) > 0) r = mid;
      else l = mid;
    }
    return max(make_pair(vec.det(convex[r]), r), make_pair(vec.det(convex[0]), 0));
  }
  void update_tangent(const Point &p, int id, int &i0, int &i1) {
    if ((a[i0] - p).det(a[id] - p) > 0) i0 = id;
    if ((a[i1] - p).det(a[id] - p) < 0) i1 = id;
  }
  void binary_search(int l, int r, Point p, int &i0, int &i1) {
    if (l == r) return;
    update_tangent(p, l % n, i0, i1);
    int sl = sign((a[l % n] - p).det(a[(l + 1) % n] - p));
    for( ; l + 1 < r; ) {
      int mid = (l + r) / 2;
      int smid = sign((a[mid % n] - p).det(a[(mid + 1) % n] - p));
      if (smid == sl) l = mid;
      else r = mid;
    }
    update_tangent(p, r % n, i0, i1);
  }
  int binary_search(Point u, Point v, int l, int r) {
    int sl = sign((v - u).det(a[l % n] - u));
    for( ; l + 1 < r; ) {
      int mid = (l + r) / 2;
      int smid = sign((v - u).det(a[mid % n] - u));
      if (smid == sl) l = mid;
      else r = mid;
    }
    return l % n;
  }
  // 判定点是否在凸包内，在边界返回 true
  bool contain(Point p) {
    if (p.x < lower[0].x || p.x > lower.back().x) return false;
    int id = lower_bound(lower.begin(), lower.end(), Point(p.x, -INF)) - lower.begin();
    if (lower[id].x == p.x) {
      if (lower[id].y > p.y) return false;
    } else if ((lower[id - 1] - p).det(lower[id] - p) < 0) return false;
    id = lower_bound(upper.begin(), upper.end(), Point(p.x, INF), greater<Point>()) - upper.begin();
    if (upper[id].x == p.x) {
      if (upper[id].y < p.y) return false;
    } else if ((upper[id - 1] - p).det(upper[id] - p) < 0) return false;
    return true;
  }
  // 求点 p 关于凸包的两个切点，如果在凸包外则有序返回编号，多解返回任意一个�否则返回 false
  bool get_tangent(Point p, int &i0, int &i1) {
    if (contain(p)) return false;
    i0 = i1 = 0;
    int id = lower_bound(lower.begin(), lower.end(), p) - lower.begin();
    binary_search(0, id, p, i0, i1);
    binary_search(id, (int)lower.size(), p, i0, i1);
    id = lower_bound(upper.begin(), upper.end(), p, greater<Point>()) - upper.begin();
    binary_search((int)lower.size() - 1, (int)lower.size() - 1 + id, p, i0, i1);
    binary_search((int)lower.size() - 1 + id, (int)lower.size() - 1 + (int)upper.size(), p, i0, i1);
    return true;
  }
  // 求凸包上和向量 vec 叉积最大的点，返回编号，有多个返回任意一个
  int get_tangent(Point vec) {
    pair<long long, int> ret = get_tangent(upper, vec);
    ret.second = (ret.second + (int)lower.size() - 1) % n;
    ret = max(ret, get_tangent(lower, vec));
    return ret.second;
  }
  // 求凸包和直线 u,v 的交点，如果无严格相交返回 false 。如果有则是和 (i,next(i)) 的交点，两个点无序，交在点上不确定返回两条线段之一。
  bool get_intersection(Point u, Point v, int &i0, int &i1) {
    int p0 = get_tangent(u - v), p1 = get_tangent(v - u);
    if (sign((v - u).det(a[p0] - u)) * sign((v - u).det(a[p1] - u)) < 0) {
      if (p0 > p1) swap(p0, p1);
      i0 = binary_search(u, v, p0, p1);
      i1 = binary_search(u, v, p1, p0 + n);
      return true;
    } else {
      return false;
    }
  }
};
```

圆的面积模板 $(n^2 \log n)$

```cpp
// Area[i] 表示覆盖次数大于等于 i 的面积
struct Tevent {
  Point p; double ang; int add;
  Tevent() {}
  Tevent(const Point &_p, double _ang, int _add): p(_p), ang(_ang), add(_add) {}
  bool operator <(const Tevent &a) const {
    return ang < a.ang;
  }
} eve[N * 2];
int E, cnt, C;
Circle c[N];
bool g[N][N], overlap[N][N];
double Area[N];
int cX[N], cY[N], cR[N];
bool contain(int i, int j) {
  return (sign(c[i].r - c[j].r) > 0|| sign(c[i].r - c[j].r) == 0 && i < j) &&
    ↪ c[i].contain(c[j], -1);
}
int main() {
  scanf("%d", &C);
  for (int i = 0; i < C; ++i) {
    scanf("%d%d%d", cX+i, cY+i, cR+i);
    c[i].o = Point(cX[i], cY[i]);
    c[i].r = cR[i];
  }
  for (int i = 0; i <= C; ++i) Area[i] = 0;
  for (int i = 0; i < C; ++i) for (int j = 0; j < C; ++j)
      overlap[i][j] = contain(i, j);
  for (int i = 0; i < C; ++i) for (int j = 0; j < C; ++j)
    g[i][j] = !(overlap[i][j] || overlap[j][i] || c[i].disjuct(c[j], -1));
  for (int i = 0; i < C; ++i) {
    E = 0; cnt = 1;
    for (int j = 0; j < C; ++j) if (j != i && overlap[j][i]) cnt++;
    for (int j = 0; j < C; ++j) {
      if (i != j && g[i][j]) {
        Point aa, bb;
        isCC(c[i], c[j], aa, bb);
        double A = atan2(aa.y - c[i].o.y, aa.x - c[i].o.x);
        double B = atan2(bb.y - c[i].o.y, bb.x - c[i].o.x);
        eve[E++] = Tevent(bb, B, 1);
        eve[E++] = Tevent(aa, A, -1);
        if (B > A) cnt++;
      }
    }
    if (E == 0) { //cnt 表示覆盖次数超过 cnt
      Area[cnt] += PI * c[i].r * c[i].r;
    } else {
      sort(eve, eve + E);
      eve[E] = eve[0];
      for (int j = 0; j < E; ++j) {
        cnt += eve[j].add;
        Area[cnt] += eve[j].p.det(eve[j + 1].p) * .5;
        double theta = eve[j + 1].ang - eve[j].ang;
        if (theta < 0) theta += PI * 2.;
        Area[cnt] += theta * c[i].r * c[i].r * .5 - sin(theta) * c[i].r * c[i].r *
          ↪ .5;
      }
    }
  }
  for(int i = 1; i <= C; ++ i) printf("[%d] = %.3f\n", i, Area[i] - Area[i + 1]);
}
```

三角形的心

```cpp
Point inCenter(const Point &A, const Point &B, const Point &C) { // 内心
  double a = (B - C).len(), b = (C - A).len(), c = (A - B).len(),
    p = (a + b + c) / 2,
    s = sqrt(p * (p - a) * (p - b) * (p - c)),
    r = s / p;
  return (A * a + B * b + C * c) / (a + b + c);
}

Point circumCenter(const Point &a, const Point &b, const Point &c) { // 外心
  Point bb = b - a, cc = c - a;
  double db = bb.len2(), dc = cc.len2(), d = 2 * det(bb, cc);
  return a + Point(bb.y * dc - cc.y * db, cc.x * db - bb.x * dc) / d;
}

```

```cpp
Point othroCenter(const Point &a, const Point &b, const Point &c) { // 垂心
  Point ba = b - a, ca = c - a, bc = b - c;
  double Y = ba.y * ca.y * bc.y,
       A = ca.x * ba.y - ba.x * ca.y,
       x0 = (Y + ca.x * ba.y * b.x - ba.x * ca.y * c.x) / A,
       y0 = -ba.x * (x0 - c.x) / ba.y + ca.y;
  return Point(x0, y0);
}
```

最小覆盖球

```cpp
int npoint, nouter; Tpoint pt[200000], outer[4],res; double radius,tmp;
void ball() {
  Tpoint q[3]; double m[3][3], sol[3], L[3], det;
  int i,j; res.x = res.y = res.z = radius = 0;
  switch ( nouter ) {
  case 1: res=outer[0]; break;
  case 2: res=(outer[0]+outer[1])/2; radius=dist2(res, outer[0]); break;
  case 3:
    for (i=0; i<2; ++i) q[i]=outer[i+1]-outer[0];
    for (i=0; i<2; ++i) for(j=0; j<2; ++j) m[i][j]=dot(q[i], q[j])*2;
    for (i=0; i<2; ++i) sol[i]=dot(q[i], q[i]);
    if (fabs(det=m[0][0]*m[1][1]-m[0][1]*m[1][0])<eps) return;
    L[0]=(sol[0]*m[1][1]-sol[1]*m[0][1])/det;
    L[1]=(sol[1]*m[0][0]-sol[0]*m[1][0])/det;
    res=outer[0]+q[0]*L[0]+q[1]*L[1];
    radius=dist2(res, outer[0]);
    break;
  case 4:
    for (i=0; i<3; ++i) q[i]=outer[i+1]-outer[0], sol[i]=dot(q[i], q[i]);
    for (i=0;i<3;++i) for(j=0;j<3;++j) m[i][j]=dot(q[i],q[j])*2;
    det= m[0][0]*m[1][1]*m[2][2]
    + m[0][1]*m[1][2]*m[2][0]
    + m[0][2]*m[2][1]*m[1][0]
    - m[0][2]*m[1][1]*m[2][0]
    - m[0][1]*m[1][0]*m[2][2]
    - m[0][0]*m[1][2]*m[2][1];
    if ( fabs(det)<eps ) return;
    for (j=0; j<3; ++j) {
      for (i=0; i<3; ++i) m[i][j]=sol[i];
      L[j]=( m[0][0]*m[1][1]*m[2][2]
      + m[0][1]*m[1][2]*m[2][0]
      + m[0][2]*m[2][1]*m[1][0]
      - m[0][2]*m[1][1]*m[2][0]
      - m[0][1]*m[1][0]*m[2][2]
      - m[0][0]*m[1][2]*m[2][1]
      ) / det;
      for (i=0; i<3; ++i) m[i][j]=dot(q[i], q[j])*2;
    } res=outer[0];
    for (i=0; i<3; ++i ) res = res + q[i] * L[i];
    radius=dist2(res, outer[0]);
}}
void minball(int n) { ball();
  if ( nouter<4 ) for (int i=0; i<n; ++i)
  if (dist2(res, pt[i])-radius>eps) {
    outer[nouter++]=pt[i]; minball(i); --nouter;
    if (i>0) { Tpoint Tt = pt[i];
      memmove(&pt[1], &pt[0], sizeof(Tpoint)*i); pt[0]=Tt;
}}}
int main0(){
  scanf("%d", &npoint);
  for (int i=0;i<npoint;i++) scanf("%lf%lf%lf",&pt[i].x,&pt[i].y,&pt[i].z);
  random_shuffle(pt,pt+npoint); radius=-1;
  for (int i=0;i<npoint;i++) if (dist2(res,pt[i])-radius>eps)
    nouter=1, outer[0]=pt[i], minball(i);
  printf("%.5f\n",sqrt(radius));
}
```

经纬度求球面最短距离

```cpp
//lati 为纬度 longi 为经度 R 为半径
double Dist(double lati1,double longi1,double lati2,double longi2,double R) {
  double pi=acos(-1.0); lati1*=pi/180,longi1*=pi/180,lati2*=pi/180,longi2*=pi/180;
  double x1=cos(lati1)*sin(longi1),y1=cos(lati1)*cos(longi1),z1=sin(lati1);
  double x2=cos(lati2)*sin(longi2),y2=cos(lati2)*cos(longi2),z2=sin(lati2);
  double theta=acos(x1*x2+y1*y2+z1*z2); return(R*theta);
}
```

长方体表面两点最短距离

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

最大团

```
// Super Fast Maximum Clique
// To Build Graph: Maxclique(Edges, Number of Nodes)
// To Get Answer: mcqdyn(AnswerNodes Index Array, AnswserLength)
typedef bool BB[N];
struct Maxclique {
  const BB* e; int pk, level; const float Tlimit;
  struct Vertex{ int i, d; Vertex(int i):i(i),d(0){} };
  typedef vector<Vertex> Vertices; typedef vector<int> ColorClass;
  Vertices V; vector<ColorClass> C; ColorClass QMAX, Q;
  static bool desc_degree(const Vertex &vi, const Vertex &vj){
    return vi.d > vj.d;
  }
  void init_colors(Vertices &v){
    const int max_degree = v[0].d;
    for(int i = 0; i < (int)v.size(); i++) v[i].d = min(i, max_degree) + 1;
  }
  void set_degrees(Vertices &v){
    for(int i = 0, j; i < (int)v.size(); i++)
      for(v[i].d = j = 0; j < int(v.size()); j++)
        v[i].d += e[v[i].i][v[j].i];
  }
  struct StepCount{ int i1, i2; StepCount():i1(0),i2(0){} };
  vector<StepCount> S;
  bool cut1(const int pi, const ColorClass &A){
    for(int i = 0; i < (int)A.size(); i++) if (e[pi][A[i]]) return true;
    return false;
  }
  void cut2(const Vertices &A, Vertices &B){
    for(int i = 0; i < (int)A.size() - 1; i++)
      if(e[A.back().i][A[i].i])
        B.push_back(A[i].i);
  }
  void color_sort(Vertices &R){
    int j = 0, maxno = 1, min_k = max((int)QMAX.size() - (int)Q.size() + 1, 1);
    C[1].clear(), C[2].clear();
    for(int i = 0; i < (int)R.size(); i++) {
      int pi = R[i].i, k = 1;
      while(cut1(pi, C[k])) k++;
      if(k > maxno) maxno = k, C[maxno + 1].clear();
      C[k].push_back(pi);
      if(k < min_k) R[j++].i = pi;
    }
    if(j > 0) R[j - 1].d = 0;
    for(int k = min_k; k <= maxno; k++)
      for(int i = 0; i < (int)C[k].size(); i++)
        R[j].i = C[k][i], R[j++].d = k;
  }
  void expand_dyn(Vertices &R){// diff -> diff with no dyn
    S[level].i1 = S[level].i1 + S[level - 1].i1 - S[level].i2;//diff
    S[level].i2 = S[level - 1].i1;//diff
    while((int)R.size()) {
      if((int)Q.size() + R.back().d > (int)QMAX.size()){
        Q.push_back(R.back().i); Vertices Rp; cut2(R, Rp);
        if((int)Rp.size()){
          if((float)S[level].i1 / ++pk < Tlimit) degree_sort(Rp);//diff
          color_sort(Rp);
          S[level].i1++, level++;//diff
          expand_dyn(Rp);
          level--;//diff
        }
        else if((int)Q.size() > (int)QMAX.size()) QMAX = Q;
        Q.pop_back();
      }
      else return;
      R.pop_back();
    }
  }
  void mcqdyn(int* maxclique, int &sz){
    set_degrees(V); sort(V.begin(),V.end(), desc_degree); init_colors(V);
    for(int i = 0; i < (int)V.size() + 1; i++) S[i].i1 = S[i].i2 = 0;
    expand_dyn(V); sz = (int)QMAX.size();
    for(int i = 0; i < (int)QMAX.size(); i++) maxclique[i] = QMAX[i];
  }
  void degree_sort(Vertices &R){
    set_degrees(R); sort(R.begin(), R.end(), desc_degree);
  }
  Maxclique(const BB* conn, const int sz, const float tt = 0.025) \
   : pk(0), level(1), Tlimit(tt){
    for(int i = 0; i < sz; i++) V.push_back(Vertex(i));
    e = conn, C.resize(sz + 1), S.resize(sz + 1);
  }
};
```

极大团计数

```
//Bool g[][] 为图的邻接矩阵，图点的标号由 1 至 n
void dfs(int size){
  int i, j, k, t, cnt, best = 0;
  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(){
  ne[0]=0; ce[0]=0;
  for (int i=1; i<=n; ++i) list[0][++ce[0]]=i;
  ans=0; dfs(0);
}
```

KM

```
//Bool g[][] 为图的邻接矩阵，图点的标号由 1 至 n
void dfs(int size){
  int i, j, k, t, cnt, best = 0;
  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(){
  ne[0]=0; ce[0]=0;
  for (int i=1; i<=n; ++i) list[0][++ce[0]]=i;
  ans=0; dfs(0);
}
```

最小树形图

```
//Bool g[][] 为图的邻接矩阵，图点的标号由 1 至 n
void dfs(int size){
  int i, j, k, t, cnt, best = 0;
  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(){
  ne[0]=0; ce[0]=0;
  for (int i=1; i<=n; ++i) list[0][++ce[0]]=i;
  ans=0; dfs(0);
}
```

无向图最小割

```
int cost[maxn][maxn],seq[maxn],len[maxn],n,m,pop,ans;
bool used[maxn];
void Init(){
  int i,j,a,b,c;
  for(i=0;i<n;i++) for(j=0;j<n;j++) cost[i][j]=0;
  for(i=0;i<m;i++){
    scanf("%d %d %d",&a,&b,&c); cost[a][b]+=c; cost[b][a]+=c;
  }
  pop=n; for(i=0;i<n;i++) seq[i]=i;
}
void Work(){
  ans=inf; int i,j,k,l,mm,sum,pk;
  while(pop > 1){
    for(i=1;i<pop;i++) used[seq[i]]=0; used[seq[0]]=1;
    for(i=1;i<pop;i++) len[seq[i]]=cost[seq[0]][seq[i]];
    pk=0; mm=-inf; k=-1;
    for(i=1;i<pop;i++) if(len[seq[i]] > mm){ mm=len[seq[i]]; k=i; }
    for(i=1;i<pop;i++){
      used[seq[l=k]]=1;
      if(i==pop-2) pk=k;
      if(i==pop-1) break;
      mm=-inf;
      for(j=1;j<pop;j++) if(!used[seq[j]])
        if((len[seq[j]]+=cost[seq[l]][seq[j]]) > mm)
          mm=len[seq[j]], k=j;
    }
    sum=0;
    for(i=0;i<pop;i++) if(i != k) sum+=cost[seq[k]][seq[i]];
    ans=min(ans,sum);
    for(i=0;i<pop;i++)
      cost[seq[k]][seq[i]]=cost[seq[i]][seq[k]]+=cost[seq[pk]][seq[i]];
    seq[pk]=seq[--pop];
  }
  printf("%d\n",ans);
}
```

带花树

```
vector<int> link[maxn];
int n,match[maxn],Queue[maxn],head,tail;
int pred[maxn],base[maxn],start,finish,newbase;
bool InQueue[maxn],InBlossom[maxn];
void push(int u){ Queue[tail++]=u;InQueue[u]=true; }
int pop(){ return Queue[head++]; }
int FindCommonAncestor(int u,int v){
  bool InPath[maxn];
  for(int i=0;i<n;i++) InPath[i]=0;
  while(true){ u=base[u];InPath[u]=true;if(u==start) break;u=pred[match[u]]; }
  while(true){ v=base[v];if(InPath[v]) break;v=pred[match[v]]; }
  return v;
}
void ResetTrace(int u){
  int v;
  while(base[u]!=newbase){
    v=match[u];
    InBlossom[base[u]]=InBlossom[base[v]]=true;
    u=pred[v];
    if(base[u]!=newbase) pred[u]=v;
  }
}
void BlossomContract(int u,int v){
  newbase=FindCommonAncestor(u,v);
  for (int i=0;i<n;i++)
  InBlossom[i]=0;
  ResetTrace(u);ResetTrace(v);
  if(base[u]!=newbase) pred[u]=v;
  if(base[v]!=newbase) pred[v]=u;
  for(int i=0;i<n;++i)
  if(InBlossom[base[i]]){
    base[i]=newbase;
    if(!InQueue[i]) push(i);
  }
}
bool FindAugmentingPath(int u){
  bool found=false;
  for(int i=0;i<n;++i) pred[i]=-1,base[i]=i;
  for (int i=0;i<n;i++) InQueue[i]=0;
  start=u;finish=-1; head=tail=0; push(start);
  while(head<tail){
    int u=pop();
    for(int i=link[u].size()-1;i>=0;i--){
      int v=link[u][i];
      if(base[u]!=base[v]&&match[u]!=v)
        if(v==start||(match[v]>=0&&pred[match[v]]>=0))
          BlossomContract(u,v);
        else if(pred[v]==-1){
          pred[v]=u;
          if(match[v]>=0) push(match[v]);
          else{ finish=v; return true; }
        }
    }
  }
  return found;
}
void AugmentPath(){
  int u=finish,v,w;
  while(u>=0){ v=pred[u];w=match[v];match[v]=u;match[u]=v;u=w; }
}
void FindMaxMatching(){
  for(int i=0;i<n;++i) match[i]=-1;
  for(int i=0;i<n;++i) if(match[i]==-1) if(FindAugmentingPath(i)) AugmentPath();
}
```

动态最小生成树

```
/* 动态最小生成树 Q(logQ)^2
  (qx[i],qy[i]) 表示将编号为 qx[i] 的边的权值改为 qy[i]
  删除一条边相当于将其权值改为 ∞
  加入一条边相当于将其权值从 ∞ 变成某个值 */
const int qsize=maxm+3*maxq;
int x[qsize],y[qsize],z[qsize], qx[maxq],qy[maxq],n,m,Q;
void init(){
  scanf("%d%d",&n,&m);
  for(int i=0;i<m;i++) scanf("%d%d%d",x+i,y+i,z+i);
  scanf("%d",&Q);
  for(int i=0;i<Q;i++){ scanf("%d%d",qx+i,qy+i); qx[i]--; }
}
int a[maxn],*tz;
int find(int x){
  int root=x; while(a[root]) root=a[root];
  int next; while(next=a[x]){ a[x]=root; x=next; }
  return root;
}
inline bool cmp(const int &a,const int &b){ return tz[a]<tz[b]; }
int kx[maxn],ky[maxn],kt, vd[maxn],id[maxm], app[maxm];
bool extra[maxm];
void solve(int *qx,int *qy,int Q,int n,int *x,int *y,int *z,int m,long long ans){
  if(Q==1){
    for(int i=1;i<=n;i++) a[i]=0;
    z[ qx[0] ]=qy[0];
    for(int i=0;i<m;i++) id[i]=i;tz=z;
    sort(id,id+m,cmp); int ri,rj;
    for(int i=0;i<m;i++){
    ri=find(x[id[i]]); rj=find(y[id[i]]);
    if(ri!=rj){ ans+=z[id[i]]; a[ri]=rj; }
  }
  printf("%I64d\n",ans);
  return;
}
int ri,rj;
//contract
kt=0;
for(int i=1;i<=n;i++) a[i]=0;
for(int i=0;i<Q;i++){
  ri=find(x[qx[i]]); rj=find(y[qx[i]]); if(ri!=rj) a[ri]=rj;
}
int tm=0;
for(int i=0;i<m;i++) extra[i]=true;
for(int i=0;i<Q;i++) extra[ qx[i] ]=false;
for(int i=0;i<m;i++) if(extra[i]) id[tm++]=i;
tz=z; sort(id,id+tm,cmp);
for(int i=0;i<tm;i++){
  ri=find(x[id[i]]); rj=find(y[id[i]]);
  if(ri!=rj){
    a[ri]=rj; ans += z[id[i]];
    kx[kt]=x[id[i]]; ky[kt]=y[id[i]]; kt++;
  }
}
for(int i=1;i<=n;i++) a[i]=0;
for(int i=0;i<kt;i++) a[ find(kx[i]) ]=find(ky[i]);
int n2=0;
for(int i=1;i<=n;i++) if(a[i]==0)
vd[i]=++n2;
for(int i=1;i<=n;i++) if(a[i])
vd[i]=vd[find(i)];
int m2=0, *Nx=x+m, *Ny=y+m, *Nz=z+m;
for(int i=0;i<m;i++) app[i]=-1;
for(int i=0;i<Q;i++) if(app[qx[i]]==-1){
  Nx[m2]=vd[ x[ qx[i] ] ]; Ny[m2]=vd[ y[ qx[i] ] ]; Nz[m2]=z[ qx[i] ];
  app[qx[i]]=m2; m2++;
}
for(int i=0;i<Q;i++){ z[ qx[i] ]=qy[i]; qx[i]=app[qx[i]]; }
for(int i=1;i<=n2;i++) a[i]=0;
for(int i=0;i<tm;i++){
  ri=find(vd[ x[id[i]] ]);  rj=find(vd[ y[id[i]] ]);
  if(ri!=rj){
    a[ri]=rj; Nx[m2]=vd[ x[id[i]] ];
    Ny[m2]=vd[ y[id[i]] ]; Nz[m2]=z[id[i]]; m2++;
    }
  }
  int mid=Q/2;
  solve(qx,qy,mid,n2,Nx,Ny,Nz,m2,ans);
  solve(qx+mid,qy+mid,Q-mid,n2,Nx,Ny,Nz,m2,ans);
}
void work(){ if(Q) solve(qx,qy,Q,n,x,y,z,m,0); }
int main(){init(); work(); return 0; }
```

Hopcroft

```
int from[1010],wh[1010],g[1010];
int num[100010],nxt[100010],tot;
int n,m,ans,h,t,q[1010],dx[1010],dy[1010];
bool bfs(){
  bool ret=false;
  h=0;t=0;
  for(int i=0;i<n;i++) if(wh[i]==-1) t++, q[t]=i;
  memset(dx,0,sizeof(dx)), memset(dy,0,sizeof(dy));
  while(h++<t){
    for(int i=g[q[h]];i!=0;i=nxt[i])
      if(dy[num[i]]==0){
        dy[num[i]]=dx[q[h]]+1;
        if(from[num[i]]==-1) ret=true;
        else{
          dx[from[num[i]]]=dx[q[h]]+2;
          q[++t]=from[num[i]];
        }
      }
  }
  return ret;
}
bool dfs(int x){
  for(int i=g[x];i!=0;i=nxt[i]){
    if(dy[num[i]]==dx[x]+1){
      dy[num[i]]=0;
      if(from[num[i]]==-1||dfs(from[num[i]])){
        wh[x]=num[i];from[num[i]]=x;return true;
      }
    }
  }
  return false;
}
void hopcroft(){
  memset(from,-1,sizeof(from)), memset(wh,-1,sizeof(wh));
  while(bfs())
    for(int i=0;i<n;i++)
      if(wh[i]==-1&&dfs(i)) ans++;
}
void insert(int x,int y){ tot++;num[tot]=y;nxt[tot]=g[x];g[x]=tot; }
int main(){
  while(scanf("%d %d",&n,&m)==2){
    tot=0; memset(g,0,sizeof(g));
    for(int i=0;i<n;i++){
      int x; scanf("%d",&x);
      for(int j=0;j<x;j++){
        int y; scanf("%d",&y);
        y--; insert(i,y);
      }
    }
    ans=0; hopcroft(); printf("%d\n",ans);
  }
}
```

素数判定

```
int strong_pseudo_primetest(long long n,int base) {
    long long n2=n-1,res;
    int s=0;
    while(n2%2==0) n2>>=1,s++;
    res=powmod(base,n2,n);
    if((res==1)||(res==n-1)) return 1;
    s--;
    while(s>=0) {
        res=mulmod(res,res,n);
        if(res==n-1) return 1;
        s--;
    }
    return 0; // n is not a strong pseudo prime
}
int isprime(long long n) {
  static LL testNum[]={2,3,5,7,11,13,17,19,23,29,31,37};
  static LL lim[]={4,0,1373653LL,25326001LL,25000000000LL,2152302898747LL, \
  3474749660383LL,341550071728321LL,0,0,0,0};
  if(n<2||n==3215031751LL) return 0;
  for(int i=0;i<12;++i){
    if(n<lim[i]) return 1;
    if(strong_pseudo_primetest(n,testNum[i])==0) return 0;
  }
  return 1;
}
```

启发式分解

```
int ansn; LL ans[1000];
LL func(LL x,LL n){ return(mod_mul(x,x,n)+1)%n; }
LL Pollard(LL n){
  LL i,x,y,p;
  if(Rabin_Miller(n)) return n;
  if(!(n&1)) return 2;
  for(i=1;i<20;i++){
    x=i; y=func(x,n); p=gcd(y-x,n);
    while(p==1) {x=func(x,n); y=func(func(y,n),n); p=gcd((y-x+n)%n,n)%n;}
    if(p==0||p==n) continue;
    return p;
  }
}
void factor(LL n){
  LL x;
  x=Pollard(n);
  if(x==n){ ans[ansn++]=x; return; }
  factor(x), factor(n/x);
}
```

二次剩余

```
void calcH(int &t, int &h, const int p) {
  int tmp = p - 1; for (t = 0; (tmp & 1) == 0; tmp /= 2) t++; h = tmp;
}
// solve equation x^2 mod p = a
bool solve(int a, int p, int &x, int &y) {
  srand(19920225);
  if (p == 2) { x = y = 1; return true; }
  int p2 = p / 2, tmp = power(a, p2, p);
  if (tmp == p - 1) return false;
  if ((p + 1) % 4 == 0) {
    x = power(a, (p + 1) / 4, p); y = p - x; return true;
  } else {
    int t, h, b, pb; calcH(t, h, p);
    if (t >= 2) {
      do {b = rand() % (p - 2) + 2;
      } while (power(b, p / 2, p) != p - 1);
      pb = power(b, h, p);
    } int s = power(a, h / 2, p);
    for (int step = 2; step <= t; step++) {
      int ss = (((long long)(s * s) % p) * a) % p;
      for (int i = 0; i < t - step; i++) ss = ((long long)ss * ss) % p;
      if (ss + 1 == p) s = (s * pb) % p; pb = ((long long)pb * pb) % p;
    } x = ((long long)s * a) % p; y = p - x;
  } return true;
}
```

Pell 方程

```
ULL A,B,p[maxn],q[maxn],a[maxn],g[maxn],h[maxn];
int main() {
  for (int test=1, n;scanf("%d",&n) && n;++test) {
    printf("Case %d: ",test);
    if (fabs(sqrt(n)-floor(sqrt(n)+1e-7))<=1e-7)   {
      int a=(int)(floor(sqrt(n)+1e-7)); printf("%d %d\n",a,1);
    } else {
      // 求 x^2 - ny^2 = 1 的最小正整数根, n 不是完全平方数
      p[1]=q[0]=h[1]=1;p[0]=q[1]=g[1]=0;
      a[2]=(int)(floor(sqrt(n)+1e-7));
      for (int i=2;i;++i) {
        g[i]=-g[i-1]+a[i]*h[i-1]; h[i]=(n-sqr(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 (sqr((ULL)(p[i]))-n*sqr((ULL)(q[i]))==1){
          A=p[i];B=q[i];break; }
      } cout << A << ' ' << B <<endl;
    }}}
```

蔡勒公式

```
int zeller(int y,int m,int d) {
  if (m<=2) y--,m+=12; int c=y/100; y%=100;
  int w=((c>>2)-(c<<1)+y+(y>>2)+(13*(m+1)/5)+d-1)%7;
  if (w<0) w+=7; return(w);
}
```

Schreier-Sims

```
namespace Schreier_Sims_Algorithm{
  struct Permutation{
    vector<int> P;
    Permutation(){}
    Permutation(int n){
      P.resize(n);
    }
    Permutation inv()const{
      Permutation ret(P.size());
      for(int i = 0; i < int(P.size()); ++i) ret.P[P[i]] = i;
      return ret;
    }
  int &operator [](const int &dn){
    return P[dn];
  }
  void resize(const size_t &sz){
    P.resize(sz);
  }
  size_t size()const{
    return P.size();
  }
  const int &operator [](const int &dn)const{
    return P[dn];
  }
};
Permutation operator *(const Permutation &a, const Permutation &b){
  Permutation ret(a.size());
  for(int i = 0; i < (int)a.size(); ++i){
    ret[i] = b[a[i]];
  }
  return ret;
}

typedef vector<Permutation> Bucket;
typedef vector<int> Table;
typedef pair<int,int> pii;
int n, m;
vector<Bucket> buckets, bucketsInv;
vector<Table> lookupTable;

int fastFilter(const Permutation &g, bool addToGroup = true){
  int n = buckets.size();
  Permutation p;
  for(int i = 0; i < n; ++i){
    int res = lookupTable[i][p[i]];
    if(res == -1){
      if(addToGroup){
        buckets[i].push_back(p);
        bucketsInv[i].push_back(p.inv());
        lookupTable[i][p[i]] = (int)buckets[i].size() - 1;
      }
      return i;
    }
    p = p * bucketsInv[i][res];
    swap(i1,i2);
  }
  return -1;
}

long long calcTotalSize(){
  long long ret = 1;
  for(int i = 0; i < n; ++i){
    ret *= buckets[i].size();
  }
  return ret;
}

bool inGroup(const Permutation &g){
  return fastFilter(g, false) == -1;
}

void solve(const Bucket &gen,int _n){// m perm[0..n - 1]s
  n = _n, m = gen.size();
  {//clear all
    vector<Bucket> _buckets(n);
    swap(buckets, _buckets);
    vector<Bucket> _bucketsInv(n);
    swap(bucketsInv, _bucketsInv);
    vector<Table> _lookupTable(n);
    swap(lookupTable, _lookupTable);
  }
  for(int i = 0; i < n; ++i){
    lookupTable[i].resize(n);
    fill(lookupTable[i].begin(), lookupTable[i].end(), -1);
  }
  Permutation id(n);
  for(int i = 0; i < n; ++i){
    id[i] = i;
  }
  for(int i = 0; i < n; ++i){
    buckets[i].push_back(id);
    bucketsInv[i].push_back(id);
    lookupTable[i][i] = 0;
  }
  for(int i = 0; i < m; ++i){
    fastFilter(gen[i]);
  }
  queue<pair<point,point> > toUpdate;
  for(int i = 0; i < n; ++i){
    for(int j = i; j < n; ++j){
      for(int k = 0; k < (int)buckets[i].size(); ++k){
        for(int l = 0; l < (int)buckets[j].size(); ++l){
            toUpdate.push(make_pair(pii(i,k), pii(j,l)));
          }
        }
      }
    }
    while(!toUpdate.empty()){
      pii a = toUpdate.front().first;
      pii b = toUpdate.front().second;
      toUpdate.pop();
      int res = fastFilter(buckets[a.first][a.second] * buckets[b.first][b.second]);
      if(res==-1) continue;
      pii newPair(res, (int)buckets[res].size() - 1);
      for(int i = 0; i < n; ++i){
        for(int j = 0; j < (int)buckets[i].size(); ++j){
          if(i <= res){
            toUpdate.push(make_pair(pii(i , j), newPair));
          }
          if(res<=i){
            toUpdate.push(make_pair(newPair, pii(i, j)));
          }
        }
      }
    }
  }
}
```

Romberg

```
template<class T>
double romberg(const T&f,double a,double b,double eps=1e-8){
  std::vector<double>t; double h=b-a,last,curr; int k=1,i=1;
  t.push_back(h*(f(a)+f(b))/2); // 梯形
  do{ last=t.back(); curr=0; double x=a+h/2;
    for(int j=0;j<k;++j) curr+=f(x),x+=h;
    curr=(t[0]+h*curr)/2; double k1=4.0/3.0,k2=1.0/3.0;
    for(int j=0;j<i;j++){ double temp=k1*curr-k2*t[j];
      t[j]=curr; curr=temp; k2/=4*k1-k2; k1=k2+1; // 防止溢出
    } t.push_back(curr); k*=2; h/=2; i++;
  } while(std::fabs(last-curr)>eps);
  return t.back();
}
```

线性规划

```
// 求max{cx | Ax ≤ b, x ≥ 0}的解
typedef vector<double> VD;
VD simplex(vector<VD> A, VD b, VD c) {
  int n = A.size(), m = A[0].size() + 1, r = n, s = m - 1;
  vector<VD> D(n + 2, VD(m + 1, 0)); vector<int> ix(n + m);
  for (int i = 0; i < n + m; ++ i) ix[i] = i;
  for (int i = 0; i < n; ++ i) {
    for (int j = 0; j < m - 1; ++ j) D[i][j] = -A[i][j];
    D[i][m - 1] = 1; D[i][m] = b[i];
    if (D[r][m] > D[i][m]) r = i;
  }
  for (int j = 0; j < m - 1; ++ j) D[n][j] = c[j];
  D[n + 1][m - 1] = -1;
  for (double d; ; ) {
    if (r < n) {
      int t = ix[s]; ix[s] = ix[r + m]; ix[r + m] = t;
      D[r][s] = 1.0 / D[r][s]; vector<int> speedUp;
      for (int j = 0; j <= m; ++ j) if (j != s) {
        D[r][j] *= -D[r][s];
        if(D[r][j]) speedUp.push_back(j);
      }
      for (int i = 0; i <= n + 1; ++ i) if (i != r) {
        for(int j = 0; j < speedUp.size(); ++ j)
        D[i][speedUp[j]] += D[r][speedUp[j]] * D[i][s];
        D[i][s] *= D[r][s];
    }} r = -1; s = -1;
    for (int j = 0; j < m; ++ j) if (s < 0 || ix[s] > ix[j])
      if (D[n + 1][j] > EPS || (D[n + 1][j] > -EPS && D[n][j] > EPS)) s = j;
    if (s < 0) break;
    for (int i = 0; i < n; ++ i) if (D[i][s] < -EPS)
      if (r < 0 || (d = D[r][m] / D[r][s] - D[i][m] / D[i][s]) < -EPS
          || (d < EPS && ix[r + m] > ix[i + m])) r = i;
    if (r < 0) return VD(); // 无边界
  }
  if (D[n + 1][m] < -EPS) return VD(); // 无解
  VD x(m - 1);
  for (int i = m; i < n + m; ++ i) if (ix[i] < m - 1) x[ix[i]] = D[i - m][m];
  return x; // 最优值在 D[n][m]
}
```

FFT

```
/*
  double□□□10^9+7□□□□□□□□2^20
*/

const int MOD = 1000003;

```

```cpp
const double PI = acos(-1);

typedef complex<double> Complex;

const int N = 65536, L = 15, MASK = (1 << L) - 1;

Complex w[N];

void FFTInit() {
  for (int i = 0; i < N; ++i) {
    w[i] = Complex(cos(2 * i * PI / N), sin(2 * i * PI / N));
  }
}

void FFT(Complex p[], int n) {
  for (int i = 1, j = 0; i < n - 1; ++i) {
    for (int s = n; j ^= s >>= 1, ~j & s;);
    if (i < j) {
      swap(p[i], p[j]);
    }
  }
  for (int d = 0; (1 << d) < n; ++d) {
    int m = 1 << d, m2 = m * 2, rm = n >> (d + 1);
    for (int i = 0; i < n; i += m2) {
      for (int j = 0; j < m; ++j) {
        Complex &p1 = p[i + j + m], &p2 = p[i + j];
        Complex t = w[rm * j] * p1;
        p1 = p2 - t;
        p2 = p2 + t;
      }
    }
  }
}

Complex A[N], B[N], C[N], D[N];

void mul(int a[N], int b[N]) {
  for (int i = 0; i < N; ++i) {
    A[i] = Complex(a[i] >> L, a[i] & MASK);
    B[i] = Complex(b[i] >> L, b[i] & MASK);
  }
  FFT(A, N);
  FFT(B, N);
  for (int i = 0; i < N; ++i) {
    int j = (N - i) % N;
    Complex da = (A[i] - conj(A[j])) * Complex(0, -0.5),
        db = (A[i] + conj(A[j])) * Complex(0.5, 0),
        dc = (B[i] - conj(B[j])) * Complex(0, -0.5),
        dd = (B[i] + conj(B[j])) * Complex(0.5, 0);
    C[j] = da * dd + da * dc * Complex(0, 1);
    D[j] = db * dd + db * dc * Complex(0, 1);
  }
  FFT(C, N);
  FFT(D, N);
  for (int i = 0; i < N; ++i) {
    long long da = (long long)(C[i].imag() / N + 0.5) % MOD,
          db = (long long)(C[i].real() / N + 0.5) % MOD,
          dc = (long long)(D[i].imag() / N + 0.5) % MOD,
          dd = (long long)(D[i].real() / N + 0.5) % MOD;
    a[i] = ((dd << (L * 2)) + ((db + dc) << L) + da) % MOD;
  }
}
```

NTT

```cpp
//R 是 2^n*q+1 形质数 p 的原根
void NFT(int P[], int n, int oper) {
  for (int i = 1, j = 0; i < n - 1; ++i) {
    for (int s = n; j ^= s >>= 1, ~j & s;);
    if (i < j) {
      swap(P[i], P[j]);
    }
  }
  for (int d = 0; (1 << d) < n; ++d) {
    int m = 1 << d, m2 = m * 2;
    int unit_p0 = powmod(R, (MOD - 1) / m2);
    if (oper < 0) {
      unit_p0 = inverse(unit_p0);
    }
    for (int i = 0; i < n; i += m2) {
      int unit = 1;
      for (int j = 0; j < m; ++j) {
        int &P1 = P[i + j + m],
          &P2 = P[i + j];
        int t = (long long)unit * P1 % MOD;
        P1 = (P2 - t + MOD) % MOD;
        P2 = (P2 + t) % MOD;
        unit = (long long)unit * unit_p0 % MOD;
      }
    }
  }
}
```

FWT

```
void FWT(int a[N], int N) {
  for (int d = 1; d < N; d <<= 1) {
    int d2 = d << 1;
    for (int i = 0; i < N; i += d2) {
      int *x = a + i, *y = a + i + d;
      for (int j = 0; j < d; ++j, ++x, ++y) {
        if ((*x += *y) >= MOD) {
          *x -= MOD;
        }
        if ((*y = *x - (*y << 1)) < 0) {
          if ((*y += MOD) < 0) {
            *y += MOD;
          }
        }
      }
    }
  }
}

void xorPow(int a[N], int n, int b[N]) {
  memset(b, 0, sizeof(int) * N);
  b[0] = 1;
  FWT(a, N);
  FWT(b, N);
  while(n) {
    if (n & 1) {
      dot(b, a, N);
    }
    dot(a, a, N);
    n >>= 1;
  }
  FWT(b, N);
  norm(b, N);
}
```

回文串 manacher

```
for(int i=1,j=0;i!=(n<<1)-1;++i){
  int p=i>>1,q=i-p,r=((j+1)>>1)+l[j]-1;
  l[i]=r<q?0:min(r-q+1,l[(j<<1)-i]);
  while(p-l[i]!=-1&&q+l[i]!=n&&s[p-l[i]]==s[q+l[i]]) l[i]++;
  if(q+l[i]-1>r) j=i;
```

```
  a+=l[i];
}
```

后缀数组（倍增）

```
int rank[MAX_N],height[MAX_N];
int cmp(int *x,int a,int b,int d){
  return x[a]==x[b]&&x[a+d]==x[b+d];
}
void doubling(int *a,int N,int M){
  static int sRank[MAX_N],tmpA[MAX_N],tmpB[MAX_N];
  int *x=tmpA,*y=tmpB;
  for(int i=0;i<M;++i) sRank[i]=0;
  for(int i=0;i<N;++i) ++sRank[x[i]=a[i]];
  for(int i=1;i<M;++i) sRank[i]+=sRank[i-1];
  for(int i=N-1;i>=0;--i) sa[--sRank[x[i]]]=i;
  for(int d=1,p=0;p<N;M=p,d<<=1){
    p=0; for(int i=N-d;i<N;++i) y[p++]=i;
    for(int i=0;i<N;++i) if(sa[i]>=d) y[p++]=sa[i]-d;
    for(int i=0;i<M;++i) sRank[i]=0;
    for(int i=0;i<N;++i) ++sRank[x[i]];
    for(int i=1;i<M;++i) sRank[i]+=sRank[i-1];
    for(int i=N-1;i>=0;--i) sa[--sRank[x[y[i]]]]=y[i];
    swap(x,y); x[sa[0]]=0; p=1;
    for(int i=1;i<N;++i) x[sa[i]]=cmp(y,sa[i],sa[i-1],d)?p-1:p++;
  }
}
void calcHeight(){
  for(int i=0;i<N;++i) rank[sa[i]]=i;
  int cur=0; for(int i=0;i<N;++i)
  if(rank[i]){
    if(cur) cur--;
    for(;a[i+cur]==a[sa[rank[i]-1]+cur];++cur);
    height[rank[i]]=cur;
  }
}
```

后缀数组 (DC3)

```
// 待排序的字符串放在 r 数组中, 从 r[0] 到 r[n-1], 长度为 n, 且最大值小于 m
// 约定除 r[n-1] 外所有的 r[i] 都大于 0, r[n-1]=0
// 函数结束后, 结果放在 sa 数组中, 从 sa[0] 到 sa[n-1]
#define maxn 10000
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int wa[maxn],wb[maxn],wv[maxn],wss[maxn]; // 必须这么大
int s[maxn*3],sa[maxn*3];
int c0(int *r,int a,int b){return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];}
int c12(int k,int *r,int a,int b){
  if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
  else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
}
void sort(int *r,int *a,int *b,int n,int m){
  int i; for(i=0;i<n;i++) wv[i]=r[a[i]];
  for(i=0;i<m;i++) wss[i]=0; for(i=0;i<n;i++) wss[wv[i]]++;
  for(i=1;i<m;i++) wss[i]+=wss[i-1];
  for(i=n-1;i>=0;i--) b[--wss[wv[i]]]=a[i];
}
void dc3(int *r,int *sa,int n,int m){
  int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
  r[n]=r[n+1]=0;
  for(i=0;i<n;i++) if(i%3!=0) wa[tbc++]=i;
  sort(r+2,wa,wb,tbc,m); sort(r+1,wb,wa,tbc,m); sort(r,wa,wb,tbc,m);
  for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
    rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
  if(p<tbc) dc3(rn,san,tbc,p);
  else for(i=0;i<tbc;i++) san[rn[i]]=i;
  for (i=0;i<tbc;i++) if(san[i]<tb) wb[ta++]=san[i]*3;
  if(n%3==1) wb[ta++]=n-1;
  sort(r,wb,wa,ta,m); for(i=0;i<tbc;i++) wv[wb[i]=G(san[i])]=i;
  for(i=0,j=0,p=0;i<ta && j<tbc;p++)
    sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
  for(;i<ta;p++) sa[p]=wa[i++]; for(;j<tbc;p++) sa[p]=wb[j++];}
int main(){
  int n,m=0;  scanf("%d",&n);
  for (int i=0;i<n;i++) scanf("%d",&s[i]),s[i]++,m=max(s[i]+1,m);
  printf("%d\n",m); s[n++]=0; dc3(s,sa,n,m);
  for (int i=0;i<n;i++) printf("%d ",sa[i]);printf("\n");
}
```

后缀自动机

```
struct State {
  int length;
  State *parent,*go[C];
  State(int length):length(length),parent(NULL){
    memset(go,0,sizeof(go));
  }
  State* extend(State *start,int token){
    State *p=this;
    State *np=new State(this->length+1);
    while(p!=NULL&&p->go[token]==NULL)
      p->go[token]=np, p=p->parent;
    if(p==NULL) np->parent=start;
    else{
      State *q=p->go[token];
      if(p->length+1==q->length) np->parent=q;
      else{
        State *nq=new State(p->length+1);
        memcpy(nq->go,q->go,sizeof(q->go));
        nq->parent=q->parent;
        np->parent=q->parent=nq;
        while(p!=NULL&&p->go[token]==q)
          p->go[token]=nq, p=p->parent;
      }
    }
    return np;
  }
};
```

后缀树 (With Walk)

```
struct State {
        int length;
        State *parent,*go[C];
        State(int length):length(length),parent(NULL){
                memset(go,0,sizeof(go));
        }
        State* extend(State *start,int token){
                State *p=this;
                State *np=new State(this->length+1);
                while(p!=NULL&&p->go[token]==NULL)
                        p->go[token]=np, p=p->parent;
                if(p==NULL) np->parent=start;
                else{
                        State *q=p->go[token];
                        if(p->length+1==q->length) np->parent=q;
                        else{
                                State *nq=new State(p->length+1);
                                memcpy(nq->go,q->go,sizeof(q->go));
                                nq->parent=q->parent;
                                np->parent=q->parent=nq;
                                while(p!=NULL&&p->go[token]==q)
                                        p->go[token]=nq, p=p->parent;
                        }
                }
                return np;
        }
};
```

后缀树 (With Pop Front)

```
struct State {
        int length;
        State *parent,*go[C];
        State(int length):length(length),parent(NULL){
                memset(go,0,sizeof(go));
        }
        State* extend(State *start,int token){
                State *p=this;
                State *np=new State(this->length+1);
                while(p!=NULL&&p->go[token]==NULL)
                        p->go[token]=np, p=p->parent;
                if(p==NULL) np->parent=start;
                else{
                        State *q=p->go[token];
                        if(p->length+1==q->length) np->parent=q;
                        else{
                                State *nq=new State(p->length+1);
                                memcpy(nq->go,q->go,sizeof(q->go));
                                nq->parent=q->parent;
                                np->parent=q->parent=nq;
                                while(p!=NULL&&p->go[token]==q)
                                        p->go[token]=nq, p=p->parent;
                        }
                }
                return np;
        }
};
```

字符串最小表示

```
std::string find(std::string s) {
  int i,j,k,l,N=s.length(); s+=s;
  for(i=0,j=1;j<N;){
    for(k=0;k<N&&s[i+k]==s[j+k];k++);
    if(k>=N) break;
    if(s[i+k]<s[j+k]) j+=k+1;
    else l=i+k,i=j,j=max(l,j)+1;
  }
  return s.substr(i,N);
}
```

轻重链剖分

```
struct Tree(){}*root[N];
int father[N],size[N],depth[N];
int bfsOrd[N],pathId[N],ordInPath[N],sqn[N];
void doBfs(int s){
  int qh=0,qt=0,*que=bfsOrd; father[s]=-1; depth[s]=0;
  for(que[qt++]=s;qh<qt;){
    int u=que[qh++];
    foreach(iter,adj[u]){
      int v=*iter; if(v==father[u]) continue;
      father[v]=u; depth[v]=depth[u]+1; que[qt++]=v;
    }
  }
}
void doSplit(){
  for(int i=N-1;i>=0;--i){
    int u=bfsOrd[i]; size[u]=1;
    foreach(iter,adj[u]){
      int v=*iter; if(v==father[u]) continue; size[u]+=size[v];
    }
  }
  memset(pathId,-1,sizeof pathId);
  for(int i=0;i<N;++i){
    int top=bfsOrd[i],cnt=0;
    if(pathId[top]!=-1) continue;
    for(int next,u=top;u!=-1;u=next){
      sqn[cnt]=val[u]; ordInPath[u]=cnt; pathId[u]=top; ++cnt;
      next=-1;
      foreach(iter,adj[u]){
        int v=*iter; if(v==father[u]) continue;
        if(next<0||size[next]<size[v]) next=v;
      }
    }
    root[top]=new Tree(0,cnt,sqn);
  }
}
void prepare(){ doBfs(0); doSplit(); }
```

KD Tree

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

using namespace std;
// 带插入版本 ， 没有写内存回收 ， 空间复杂度 n log n ， 如果不需要插入可以大大简化
// N 为最大点数, D 为每个点的最大维度, d 为实际维度
// 以查找最近点为例 ret 为当前最近点的距离的平方 ， 用来剪枝 ， 查询 k 近或 k 远的方法类似
// 使用时注意先 initNull
const long long INF = (int)1e9 + 10;
const int N = 2000000 + 10;
const int D = 5;
const double SCALE = 0.75;
struct Point { int x[D]; } buf[N];
int d;
struct Node {
  int depth, size;
  Node *ch[2], *p;
  Point val, maxv, minv;
  void set(Node *t, int d) { ch[d] = t; t->p = this; }
  bool dir() { return this == p->ch[1]; }
  bool balanced() {
    return (double)max(ch[0]->size, ch[1]->size) <= (double)size * SCALE;
  }
  void update() {
    size = ch[0]->size + ch[1]->size + 1;
    for(int i = 0; i < d; ++ i) {
      maxv.x[i] = max(val.x[i], max(ch[0]->maxv.x[i], ch[1]->maxv.x[i]));
      minv.x[i] = min(val.x[i], min(ch[0]->minv.x[i], ch[1]->minv.x[i]));
    }
  }
} nodePool[N], *totNode, *null;
Node* newNode(Point p, int depth) {
  Node *t = totNode ++;
  t->ch[0] = t->ch[1] = t->p = null;
  t->depth = depth;
  t->val = t->maxv = t->minv = p;
  t->size = 1;
  return t;
}
long long ret;
int ctr;
int cmp(const Point &a, const Point &b) { return a.x[ctr] < b.x[ctr]; }
struct KDTree {
  Node *root;
  KDTree() { root = null; }
  KDTree(Point *a, int n) {
    root = build(a, 0, n - 1, 0);
  }
  Node *build(Point *a, int l, int r, int depth) {
    if (l > r) return null;
    ctr = depth;
    sort(a + l, a + r + 1, cmp);
    int mid = (l + r) >> 1;
    Node *t = newNode(a[mid], depth);
    t->set(build(a, l, mid - 1, (depth + 1) % d), 0);
    t->set(build(a, mid + 1, r, (depth + 1) % d), 1);
    t->update();
    return t;
  }
  void tranverse(Node *t, Point *vec, int &tot) {
    if (t == null) return;
    vec[tot ++] = t->val;
    tranverse(t->ch[0], vec, tot);
    tranverse(t->ch[1], vec, tot);
  }
  void rebuild(Node *t) {
    Node *p = t->p;
    int tot = 0;
    tranverse(t, buf, tot);
    Node *u = build(buf, 0, tot - 1, t->depth);
    p->set(u, t->dir());
    for( ; p != null; p = p->p) p->update();
    if (t == root) root = u;
  }
  void insert(Point p) {
    if (root == null) { root = newNode(p, 0); return; }
    Node *cur = root, *last = null;
    int dir = 0;
    for( ; cur != null; ) {
      last = cur;
      dir = (p.x[cur->depth] > cur->val.x[cur->depth]);
      cur = cur->ch[dir];
    }
    Node *t = newNode(p, (last->depth + 1) % d), *bad = null;
    last->set(t, dir);
    for( ; t != null; t = t->p) {
      t->update();
      if (!t->balanced()) bad = t;
    }
    if (bad != null) rebuild(bad);
  }
  long long calcEval(Point u, Node *t, int d) {
    long long l = t->minv.x[d], r = t->maxv.x[d], x = u.x[d];
    if (x >= l && x <= r) return 0LL;
    long long ret = min(abs(x - l), abs(x - r));
    return ret * ret;
  }
  void updateAns(Point u, Point p) {
    // 在这里更新答案
  }
  void query(Node *t, Point p) {
    if (t == null) return;
    updateAns(t->val, p);
    long long evalLeft = calcEval(p, t->ch[0], t->depth);
    long long evalRight = calcEval(p, t->ch[1], t->depth);
    if (evalLeft <= evalRight) {
      query(t->ch[0], p);
      if (ret > evalRight) query(t->ch[1], p);
    } else {
      query(t->ch[1], p);
      if (ret > evalLeft) query(t->ch[0], p);
    }
  }
  void query(Point p) {
    query(root, p);
  }
};
void initNull() {
  totNode = nodePool;
  null = totNode ++;
  null->size = 0;
  for(int i = 0; i < d; ++ i) {
    null->maxv.x[i] = -INF;
    null->minv.x[i] = INF;
  }
}
```

Splay Tree

```
// 注意初始化内存池和 null 节点
struct Node{
  int rev,size; Node *ch[2],*p;
  void set(Node*,int); int dir(); void update(); void relax(); void appRev();
} nodePool[MAX_NODE],*curNode,*null;
Node *newNode(){
  Node *t=curNode++; t->rev=0, t->size=1;
  t->ch[0]=t->ch[1]=t->p=null; return t;
}
struct Splay{
  Node *root;
  Splay(){ root=newNode(); root->set(newNode(),0); root->update(); }
  void rot(Node *t){
    Node *p=t->p; int d=t->dir();
    p->relax(); t->relax();
    if(p==root) root=t;
    p->set(t->ch[!d],d); p->p->set(t,p->dir()); t->set(p,!d);
    p->update();
  }
  void splay(Node *t,Node *f=null){
    for(t->relax();t->p!=f;)
      if(t->p->p==f) rot(t);
      else t->dir()==t->p->dir()?(rot(t->p),rot(t)):(rot(t),rot(t));
    t->update();
  }
};
void initNull(){ curNode=nodePool;null=curNode++;null->size=0; }
void Node::set(Node *t,int _d){ ch[_d]=t; t->p=this; }
int Node::dir(){ return this==p->ch[1]; }
void Node::update(){ size=ch[0]->size+ch[1]->size+1;}
void Node::relax(){ if(rev) ch[0]->appRev(), ch[1]->appRev(), rev=false; }
void Node::appRev(){ if(this==null) return; rev^=true; swap(ch[0],ch[1]); }
```

Link Cut Tree

```
// 注意初始化 null 节点，单点的 is_root 初始为 true
struct Node{
  Node *ch[2], *p;
  int is_root, rev;
  bool dir();
  void set(Node*, bool);
  void update();
  void relax();
  void app_rev();
} *null;
void rot(Node *t){
  Node *p=t->p; bool d=t->dir();
  p->relax(); t->relax(); p->set(t->ch[!d],d);
  if(p->is_root) t->p=p->p,swap(p->is_root,t->is_root);
  else p->p->set(t,p->dir());
  t->set(p,!d); p->update();
}
void splay(Node *t){
  for(t->relax();!t->is_root;)
    if(t->p->is_root) rot(t);
    else t->dir()==t->p->dir() ?(rot(t->p),rot(t)) :(rot(t),rot(t));
  t->update();
}
void access(Node *t){
  for(Node *s=null; t!=null; s=t,t=t->p){
    splay(t);
    if (t->p == null) { /*TODO*/ }
    t->ch[1]->is_root=true; s->is_root=false;
    t->ch[1]=s; t->update();
  }
}
bool Node::dir(){ return this==p->ch[1]; }
void Node::set(Node *t,bool _d){ ch[_d]=t; t->p=this; }
void Node::update(){ }
void Node::app_rev(){ if (this == null) return; rev ^= true; swap(ch[0], ch[1]); }
void Node::relax() { if(this==null) return; if (rev) { ch[0]->app_rev();
  ↪ ch[1]->app_rev(); rev = false; } }
```

```
void make_root(Node *u) { access(u); splay(u); u->app_rev(); }
```

Dominator Tree

```
vector<int> prec[N], succ[N];

vector<int> ord;

int stamp, vis[N];

int num[N];

int fa[N];

void dfs(int u) {
  vis[u] = stamp;
  num[u] = ord.size();
  ord.push_back(u);
  for (int i = 0; i < (int)succ[u].size(); ++i) {
    int v = succ[u][i];
    if (vis[v] != stamp) {
      fa[v] = u;
      dfs(v);
    }
  }
}

int fs[N], mins[N];

int dom[N], sem[N];

int find(int u) {
  if (u != fs[u]) {
    int v = fs[u];
    fs[u] = find(fs[u]);
    if (mins[v] != -1 && num[sem[mins[v]]] < num[sem[mins[u]]]) {
      mins[u] = mins[v];
    }
  }
  return fs[u];
}

void merge(int u, int v) {
  fs[u] = v;
}

vector<int> buf[N];

int buf2[N];

void mark(int source) {
  ord.clear();
  ++stamp;
  dfs(source);
  for (int i = 0; i < (int)ord.size(); ++i) {
    int u = ord[i];
    fs[u] = u;
    mins[u] = -1;
    buf2[u] = -1;
  }
  for (int i = (int)ord.size() - 1; i > 0; --i) {
    int u = ord[i], p = fa[u];
    sem[u] = p;
    for (int j = 0; j < (int)prec[u].size(); ++j) {
      int v = prec[u][j];
      if (use[v] != stamp) {
        continue;
      }
      if (num[v] > num[u]) {
        find(v);
        v = sem[mins[v]];
      }
      if (num[v] < num[sem[u]]) {
        sem[u] = v;
      }
    }
    buf[sem[u]].push_back(u);
    mins[u] = u;
    merge(u, p);
    while (buf[p].size()) {
      int v = buf[p].back();
      buf[p].pop_back();
      find(v);
      if (sem[v] == sem[mins[v]]) {
        dom[v] = sem[v];
      } else {
        buf2[v] = mins[v];
      }
    }
  }
  dom[ord[0]] = ord[0];
  for (int i = 0; i < (int)ord.size(); ++i) {
    int u = ord[i];
    if (~buf2[u]) {
      dom[u] = dom[buf2[u]];
    }
  }
}
```

DancingLinks

```
struct node{
  node *left,*right,*up,*down,*col; int row,cnt;
}*head,*col[MAXC],Node[MAXNODE],*ans[MAXNODE];
int totNode;
void insert(const std::vector<int> &V,int rownum){
  std::vector<node*> N;
  for(int i=0;i<int(V.size());++i){
    node* now=Node+(totNode++); now->row=rownum;
    now->col=now->up=col[V[i]], now->down=col[V[i]]->down;
    now->up->down=now, now->down->up=now;
    now->col->cnt++; N.push_back(now);
  }
  for(int i=0;i<int(V.size());++i)
    N[i]->right=N[(i+1)%V.size()], N[i]->left=N[(i-1+V.size())%V.size()];
}
void Remove(node *x){
  x->left->right=x->right, x->right->left=x->left;
  for(node *i=x->down;i!=x;i=i->down)
    for(node *j=i->right;j!=i;j=j->right)
      j->up->down=j->down, j->down->up=j->up, --(j->col->cnt);
}
void Resume(node *x){
  for(node *i=x->up;i!=x;i=i->up)
    for(node *j=i->left;j!=i;j=j->left)
      j->up->down=j->down->up=j, ++(j->col->cnt);
  x->left->right=x, x->right->left=x;
}
bool search(int tot){
  if(head->right==head) return true;
  node *choose=NULL;
  for(node *i=head->right;i!=head;i=i->right){
    if(choose==NULL||choose->cnt>i->cnt) choose=i;
    if(choose->cnt<2) break;
  }
  Remove(choose);
  for(node *i=choose->down;i!=choose;i=i->down){
    for(node *j=i->right;j!=i;j=j->right) Remove(j->col);
    ans[tot]=i;
    if(search(tot+1)) return true;
    ans[tot]=NULL;
    for(node *j=i->left;j!=i;j=j->left) Resume(j->col);
  }
  Resume(choose);
  return false;
}
void prepare(int totC){
  head=Node+totC;
  for(int i=0;i<totC;++i) col[i]=Node+i;
  totNode=totC+1;
  for(int i=0;i<=totC;++i){
    (Node+i)->right=Node+(i+1)%(totC+1);
    (Node+i)->left=Node+(i+totC)%(totC+1);
    (Node+i)->up=(Node+i)->down=Node+i;
  }
}
```

弦图相关

1. 团数 $\le$ 色数，弦图团数 $=$ 色数
2. 设 $next(v)$ 表示 $N(v)$ 中最前的点．令 w* 表示所有满足 $A \in B$ 的 w 中最后的一个点，判断 $v \cup N(v)$ 是否为极大团，只需判断是否存在一个 w，满足 $Next(w) = v$ 且 $|N(v)| + 1 \le |N(w)|$ 即可．
3. 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
4. 最大独立集：完美消除序列从前往后能选就选
5. 弦图最大独立集数 $=$ 最小团覆盖数，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

图同构 Hash

$$F_t(i) = (F_{t-1}(i) \times A + \sum_{i \to j} F_{t-1}(j) \times B + \sum_{j \to i} F_{t-1}(j) \times C + D \times (i = a)) \ mod \ P$$

枚举点 a 迭代 K 次后求得的就是 a 点所对应的 hash 值
其中 K，A，B，C，D，P 为 hash 参数，可自选

直线下有多少个格点

```
LL solve(LL n,LL a,LL b,LL m){
  // 计算 for (int i=0;i<n;++i) s+=floor((a+b*i)/m)
  //n,m,a,b>0
  if(b==0) return n*(a/m);
  if(a>=m) return n*(a/m)+solve(n,a%m,b,m);
  if(b>=m) return (n-1)*n/2*(b/m)+solve(n,a,b%m,m);
  return solve((a+b*n)/m,(a+b*n)%m,m,b);
}
```

费用流

```
// Q is a priority_queue<PII, vector<PII>, greater<PII> >
// for an edge(s, t): u is the capacity, v is the cost, nxt is the next edge,
// op is the opposite edge
// this code can not deal with negative cycles
typedef pair<int,int> PII;
struct edge{ int t,u,v; edge *nxt,*op; }E[MAXE],*V[MAXV];
int D[MAXN], dist[MAXN], maxflow, mincost; bool in[MAXN];
bool modlabel(){
  while(!Q.empty()) Q.pop();
  for(int i=S;i<=T;++i){
    if(in[i]) D[i]=0,Q.push(PII(0,i));
    else D[i]=inf;
  }
  while(!Q.empty()){
    int x=Q.top().first,y=Q.top().second;
    Q.pop();
    if(y==T) break;
    if(D[y]<x) continue;
    for(edge *ii=V[y];ii;ii=ii->nxt) if(ii->u){
      if(x+(ii->v+dist[ii->t]-dist[y])<D[ii->t]){
        D[ii->t]=x+(ii->v+dist[ii->t]-dist[y]);
        Q.push(PII(D[ii->t],ii->t));
      }
    }
  }
  if(D[T]==inf) return false;
  for(int i=S;i<=T;++i) if(D[i]>D[T]) dist[i]+=D[T]-D[i];
  return true;
}
int aug(int p,int limit){
  if(p==T) return maxflow+=limit,mincost+=limit*dist[S],limit;
  in[p]=1; int kk,ll=limit;
  for(edge *ii=V[p];ii;ii=ii->nxt) if(ii->u){
    if(!in[ii->t]&&dist[ii->t]+ii->v==dist[p]){
      kk=aug(ii->t,min(ii->u,ll));
      ll-=kk,ii->u-=kk,ii->op->u+=kk;
      if(!ll) return in[p]=0,limit;
    }
  }
  return limit-ll;
}
PII mincostFlow(){
  for(int i=S;i<=T;++i) dist[i]=i==T?inf:0;
  while(!Q.empty()) Q.pop();
  Q.push(PII(0,T));
  while(!Q.empty()){
    int x=Q.top().first,y=Q.top().second;
    Q.pop();
    if(dist[y]<x) continue;
    for(edge *ii=V[y];ii;ii=ii->nxt) if(ii->op->u&&ii->v+x<dist[ii->t]){
      dist[ii->t]=ii->v+x;
      Q.push(PII(dist[ii->t],ii->t));
    }
  }
  maxflow=mincost=0;
  do{
    do{
      memset(in,0,sizeof(in));
    }while(aug(S,maxflow));
  }while(modlabel());
  return PII(maxflow,mincost);
}
```

综合

定理 1: 最小覆盖数 = 最大匹配数

定理 2: 最大独立集 S 与 最小覆盖集 T 互补

算法:

1. 做最大匹配 ，没有匹配的空闲点 $\in S$
2. 如果 $u \in S$ 那么 u 的临点必然属于 T
3. 如果一对匹配的点中有一个属于 T 那么另外一个属于 S
4. 还不能确定的 ，把左子图的放入 S，右子图放入 T

算法结束

上下界无源汇可行流 ：不用添 $T->S$，判断是否流量平衡

上下界有源汇可行流 ：添 $T \to S$(下界 0，上界 ∞)， 判断是否流量平衡

上下界最小流 ：不添 $T \to S$ 先流一遍 ，再添 $T \to S$(下界 0，上界 ∞) 在残图上流一遍 ，答案为 $S \to T$ 的流量值

上下界最大流 ：添 $T \to S$（下界 0，上界 ∞）流一遍，再在残图上流一遍S到T的最大流，答案为前者的 $S \to T$ 的值 + 残图中 $S \to T$ 的最大流

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

Stirling 数

第一类 :n 个元素的项目分作 k 个环排列的方法数目

$s(n,k) = (-1)^{n+k}|s(n,k)|$

$|s(n,0)| = 0, |s(1,1)| = 1,$

$|s(n,k)| = |s(n-1,k-1)| + (n-1) * |s(n-1,k)|$

第二类 :n 个元素的集定义 k 个等价类的方法数

$S(n,1) = S(n,n) = 1, S(n,k) = S(n-1,k-1) + k * S(n-1,k)$

积分表

Integrals of Rational Functions

$\int \frac{1}{1+x^2}dx = \tan^{-1} x$ $\int \frac{1}{a^2+x^2}dx = \frac{1}{a}\tan^{-1}\frac{x}{a}$ $\int \frac{x}{a^2+x^2}dx = \frac{1}{2}\ln|a^2+x^2|$ $\int \frac{x^2}{a^2+x^2}dx = x - a\tan^{-1}\frac{x}{a}$ $\int \frac{x^3}{a^2+x^2}dx = \frac{1}{2}x^2 - \frac{1}{2}a^2\ln|a^2+x^2|$

$\int \frac{1}{ax^2+bx+c}dx = \frac{2}{\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}}$ $\int \frac{1}{(x+a)(x+b)}dx = \frac{1}{b-a}\ln\frac{a+x}{b+x},\ a \neq b$ $\int \frac{x}{(x+a)^2}dx = \frac{a}{a+x} + \ln|a+x|$ $\int \frac{x}{ax^2+bx+c}dx = \frac{1}{2a}\ln|ax^2+bx+c| - \frac{b}{a\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}}$

Integrals with Roots

$\int \frac{x}{\sqrt{x\pm a}}dx = \frac{2}{3}(x \mp 2a)\sqrt{x \pm a}$ $\int \sqrt{\frac{x}{a-x}}dx = -\sqrt{x(a-x)} - a\tan^{-1}\frac{\sqrt{x(a-x)}}{x-a}$ $\int \sqrt{\frac{x}{a+x}}dx = \sqrt{x(a+x)} - a\ln\left[\sqrt{x}+\sqrt{x+a}\right]$ $\int x\sqrt{x^2 \pm a^2}dx = \frac{1}{3}\left(x^2 \pm a^2\right)^{3/2}$

$\int x\sqrt{ax+b}dx = \frac{2}{15a^2}(-2b^2+abx+3a^2x^2)\sqrt{ax+b}$ $\int \sqrt{x(ax+b)}dx = \frac{1}{4a^{3/2}}\left[(2ax+b)\sqrt{ax(ax+b)}\ \ -b^2\ln\left|a\sqrt{x}+\sqrt{a(ax+b)}\right|\right]$ $\int \sqrt{x^2 \pm a^2}dx = \frac{1}{2}x\sqrt{x^2 \pm a^2} \pm \frac{1}{2}a^2\ln\left|x+\sqrt{x^2 \pm a^2}\right|$

$\int \sqrt{x^3(ax+b)}dx = \left[\frac{b}{12a} - \frac{b^2}{8a^2x} + \frac{x}{3}\right]\sqrt{x^3(ax+b)} + \frac{b^3}{8a^{5/2}}\ln\left|a\sqrt{x}+\sqrt{a(ax+b)}\right|$ $\int \sqrt{a^2-x^2}dx = \frac{1}{2}x\sqrt{a^2-x^2} + \frac{1}{2}a^2\tan^{-1}\frac{x}{\sqrt{a^2-x^2}}$ $\int \frac{x^2}{\sqrt{x^2 \pm a^2}}dx = \frac{1}{2}x\sqrt{x^2 \pm a^2} \mp \frac{1}{2}a^2\ln\left|x+\sqrt{x^2 \pm a^2}\right|$

$\int \frac{1}{\sqrt{x^2 \pm a^2}}dx = \ln\left|x+\sqrt{x^2 \pm a^2}\right|$ $\int \frac{1}{\sqrt{a^2-x^2}}dx = \sin^{-1}\frac{x}{a}$ $\int \frac{x}{\sqrt{x^2 \pm a^2}}dx = \sqrt{x^2 \pm a^2}$ $\int \frac{x}{\sqrt{a^2-x^2}}dx = -\sqrt{a^2-x^2}$ $\int \sqrt{ax^2+bx+c}dx = \frac{b+2ax}{4a}\sqrt{ax^2+bx+c} + \frac{4ac-b^2}{8a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx^+c)}\right|$

$\int x\sqrt{ax^2+bx+c} = \frac{1}{48a^{5/2}}\left(2\sqrt{a}\sqrt{ax^2+bx+c}\ \times\left(-3b^2+2abx+8a(c+ax^2)\right)\ +3(b^3-4abc)\ln\left|b+2ax+2\sqrt{a}\sqrt{ax^2+bx+c}\right|\right)$ $\int \frac{1}{\sqrt{ax^2+bx+c}}dx = \frac{1}{\sqrt{a}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right|$

$\int \frac{x}{\sqrt{ax^2+bx+c}}dx = \frac{1}{a}\sqrt{ax^2+bx+c} - \frac{b}{2a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right|$ $\int \frac{dx}{(a^2+x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2+x^2}}$

Integrals with Logarithms

$\int \ln(ax+b)dx = \left(x+\frac{b}{a}\right)\ln(ax+b) - x, a \neq 0$

$\int \frac{\ln ax}{x}dx = \frac{1}{2}(\ln ax)^2$ $\int \ln(x^2+a^2)\,\mathrm{dx} = x\ln(x^2+a^2) + 2a\tan^{-1}\frac{x}{a} - 2x$ $\int \ln(x^2-a^2)\,\mathrm{dx} = x\ln(x^2-a^2) + a\ln\frac{x+a}{x-a} - 2x$ $\int x\ln(ax+b)dx = \frac{bx}{2a} - \frac{1}{4}x^2 + \frac{1}{2}\left(x^2 - \frac{b^2}{a^2}\right)\ln(ax+b)$

$\int \ln\left(ax^2+bx+c\right)dx = \frac{1}{a}\sqrt{4ac-b^2}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}} - 2x + \left(\frac{b}{2a}+x\right)\ln\left(ax^2+bx+c\right)$ $\int x\ln\left(a^2-b^2x^2\right)dx = -\frac{1}{2}x^2 + \frac{1}{2}\left(x^2-\frac{a^2}{b^2}\right)\ln\left(a^2-b^2x^2\right)$

Integrals with Exponentials

$\int x^ne^{ax}\,\mathrm{d}x = \dfrac{x^ne^{ax}}{a} - \dfrac{n}{a}\int x^{n-1}e^{ax}\,\mathrm{d}x$ $\int xe^{-ax^2}\ \mathrm{dx} = -\dfrac{1}{2a}e^{-ax^2}$

Integrals with Trigonometric Functions

$\int \sin^3 axdx = -\frac{3\cos ax}{4a} + \frac{\cos 3ax}{12a}$ $\int \cos^2 axdx = \frac{x}{2} + \frac{\sin 2ax}{4a}$

$\int \cos^3 axdx = \frac{3\sin ax}{4a} + \frac{\sin 3ax}{12a}$ $\int \cos ax\sin bxdx = \frac{\cos[(a-b)x]}{2(a-b)} - \frac{\cos[(a+b)x]}{2(a+b)}, a \neq b$ $\int \sin^2 ax\cos bxdx = -\frac{\sin[(2a-b)x]}{4(2a-b)} + \frac{\sin bx}{2b} - \frac{\sin[(2a+b)x]}{4(2a+b)}$ $\int \sin^2 x\cos xdx = \frac{1}{3}\sin^3 x$

$\int \cos^2 ax\sin bxdx = \frac{\cos[(2a-b)x]}{4(2a-b)} - \frac{\cos bx}{2b} - \frac{\cos[(2a+b)x]}{4(2a+b)}$ $\int \cos^2 ax\sin axdx = -\frac{1}{3a}\cos^3 ax$ $\int \sin^2 ax\cos^2 bxdx = \frac{x}{4} - \frac{\sin 2ax}{8a} - \frac{\sin[2(a-b)x]}{16(a-b)} + \frac{\sin 2bx}{8b} - \frac{\sin[2(a+b)x]}{16(a+b)}$ $\int \sin^2 ax\cos^2 axdx = \frac{x}{8} - \frac{\sin 4ax}{32a}$

$\int \tan axdx = -\frac{1}{a}\ln\cos ax$ $\int \tan^2 axdx = -x + \frac{1}{a}\tan ax$ $\int \tan^3 axdx = \frac{1}{a}\ln\cos ax + \frac{1}{2a}\sec^2 ax$ $\int \sec xdx = \ln|\sec x + \tan x| = 2\tanh^{-1}\left(\tan\frac{x}{2}\right)$ $\int \sec^2 axdx = \frac{1}{a}\tan ax$

$\int \sec^3 x\,\mathrm{dx} = \frac{1}{2}\sec x\tan x + \frac{1}{2}\ln|\sec x + \tan x|$ $\int \sec x\tan xdx = \sec x$ $\int \sec^2 x\tan xdx = \frac{1}{2}\sec^2 x$ $\int \sec^n x\tan xdx = \frac{1}{n}\sec^n x, n \neq 0$ $\int \csc xdx = \ln\left|\tan\frac{x}{2}\right| = \ln|\csc x - \cot x| + C$

$\int \csc^2 axdx = -\frac{1}{a}\cot ax$ $\int \csc^3 xdx = -\frac{1}{2}\cot x\csc x + \frac{1}{2}\ln|\csc x - \cot x|$ $\int \csc^n x\cot xdx = -\frac{1}{n}\csc^n x, n \neq 0$ $\int \sec x\csc xdx = \ln|\tan x|$

Products of Trigonometric Functions and Monomials

$\int x\cos xdx = \cos x + x\sin x$ $\int x\cos axdx = \frac{1}{a^2}\cos ax + \frac{x}{a}\sin ax$ $\int x^2\cos xdx = 2x\cos x + \left(x^2-2\right)\sin x$ $\int x^2\cos axdx = \frac{2x\cos ax}{a^2} + \frac{a^2x^2-2}{a^3}\sin ax$ $\int x\sin xdx = -x\cos x + \sin x$

$\int x\sin axdx = -\frac{x\cos ax}{a} + \frac{\sin ax}{a^2}$ $\int x^2\sin xdx = \left(2-x^2\right)\cos x + 2x\sin x$ $\int x^2\sin axdx = \frac{2-a^2x^2}{a^3}\cos ax + \frac{2x\sin ax}{a^2}$

Products of Trigonometric Functions and Exponentials

$\int e^x\sin xdx = \frac{1}{2}e^x(\sin x - \cos x)$ $\int e^{bx}\sin axdx = \frac{1}{a^2+b^2}e^{bx}(b\sin ax - a\cos ax)$ $\int e^{bx}\cos axdx = \frac{1}{a^2+b^2}e^{bx}(a\sin ax + b\cos ax)$ $\int xe^x\sin xdx = \frac{1}{2}e^x(\cos x - x\cos x + x\sin x)$

$\int xe^x\cos xdx = \frac{1}{2}e^x(x\cos x - \sin x + x\sin x)$ $\int e^x\cos xdx = \frac{1}{2}e^x(\sin x + \cos x)$

Java

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

public class Main{
  BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
  PrintWriter writer = new PrintWriter(System.out);
  StringTokenizer tokenizer = null;

  void solve() throws Exception {
  }
  void run()throws Exception{
    try{
      while (true) {
        solve();
      }
    }
    catch(Exception e){
    }
    finally{
      reader.close();
      writer.close();
    }
  }
  String next()throws Exception{
    for(;tokenizer == null || !tokenizer.hasMoreTokens();){
      tokenizer = new StringTokenizer(reader.readLine());
    }
    return tokenizer.nextToken();
  }
  int nextInt()throws Exception{
    return Integer.parseInt(next());
  }
  double nextDouble()throws Exception{
    return Double.parseDouble(next());
  }
  BigInteger nextBigInteger()throws Exception{
    return new BigInteger(next());
  }
  public static void main(String args[])throws Exception{
    (new Main()).run();
  }
}
```

Vimrc

```
\begin{lstlisting}
set nu ai ci si mouse=a ts=4 sts=4 sw=4

nmap<C-A> ggVG
vmap<C-C> "+y

nmap<F3> : vs %<.in <CR>
nmap<F5> : !./%< <CR>
nmap<F8> : !./%< < %<.in <CR>
nmap<F9> : !g++ % -o %< -Wall <CR>

"nmap<F4> : !gedit % <CR>
"autocmd BufNewFile *.cpp 0r ~/temp.cpp
"set hlsearch incseach

"syntax on
"filetype plugin indent on
\end{lstlisting}
```