计算几何

1 求三点外心

```
Circle mkCir(const Point &a, const Point &b, const Point &c)
{
  double a1 = b.x - a.x, b1 = b.y - a.y, c1 = (a1 * a1 + b1 * b1) / 2;
  double a2 = c.x - a.x, b2 = c.y - a.y, c2 = (a2 * a2 + b2 * b2) / 2;
  double d = a1 * b2 - a2 * b1;
  Point tmp(a.x + (c1 * b2 - c2 * b1) / d, a.y + (a1 * c2 - a2 * c1) / d);
  return Circle(tmp, (a - tmp).len());
}
```

2 半平面交

```
struct Border {
  Point p1, p2;
  double alpha;
  void setAlpha() {
    alpha = atan2(p2.y - p1.y, p2.x - p1.x);
  }
  void read() {
    p1.read();
    p2.read();
    setAlpha();
  }
};
int n;
const int MAX_N_BORDER = 20000 + 10;
Border border[MAX_N_BORDER];
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;
}
const double LARGE = 10000;
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();
}
```

3 点到凸包的切线

```
#include<cstring>
#include<cstdio>
#include<algorithm>
using namespace std;
struct couple
{
  long long x, y;
  couple(){}
  couple(const long long  & _x, const long long &_y) {x = _x; y = _y;}
  void scan(){scanf("%lld%lld", &x, &y);}
  void print() {printf("%lld␣%lld\n", x, y);}
} q1[111111], *q, q2[111111], a[111111], x;
long long ans, ans1, s1[111111], s2[111111], *s;
int n, Q, cl1, cl2, cl, mid, lb, bs[2], frm, to;
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);}
long long operator * (const couple & a, const couple & b)
{return a.x * b.y - a.y * b.x;}
bool operator < (const couple & a, const couple & b)
{return a.x < b.x or a.x == b.x and a.y < b.y;}
typedef bool (* func) (const couple & a, const couple & b);
bool lss(const couple & a, const couple & b) {return a < b;}
bool grt(const couple & a, const couple & b) {return b < a;}
void psh(int i)
{
  while(cl > 1 and (a[i] - q[cl]) * (q[cl] - q[cl - 1]) <= 0) cl--;
  q[++cl] = a[i];
}
bool check(int mid)
{
  return (x - q[mid]) * (q[mid + 1] - x) < 0;
}
func cmp;
void calc()
{
  lb = lower_bound(q + 1, q + 1 + cl, x, cmp) - q;
  if(lb == cl + 1 or lb == 1 or (q[lb] - x) * (x - q[lb - 1]) > 0)
  {
    bs[0] = 1; bs[1] = lb - 1;
    while(bs[0] < bs[1] - 1)
    {
      mid = (bs[0] + bs[1]) / 2;
      bs[check(mid)] = mid;
    }
    frm = check(bs[0])?bs[0]:bs[1];
    bs[0] = lb - 1; bs[1] = cl - 1;
    while(bs[0] < bs[1] - 1)
    {
      mid = (bs[0] + bs[1]) / 2;
      bs[!check(mid)] = mid;
    }
    to = check(bs[1])?bs[1]:bs[0];
    if(!frm) ans1 += 0 * (x * q[1]);
    else if(to == cl) ans1 += 0 * (q[cl1] * x);
    else ans1 += q[frm] * x + x * q[to + 1] - s[to] + s[frm - 1];
  }
}
int main()
{
  scanf("%d%d", &n, &Q);
  for(int i = 1; i <= n; i++) a[i].scan();
  sort(a + 1, a + 1 + n);
  q = q1; s = s1;
  cl = 0;
  for(int i = 1; i <= n; i++)
  {
    psh(i);
  }
  s[0] = 0;
  for(int i = 1; i < cl; i++) s[i] = s[i - 1] + q[i] * q[i + 1];
  cl1 = cl;
  q = q2; s = s2;
  cl = 0;
  for(int i = n; i >= 1; i--)
  {
    psh(i);
  }
  s[0] = 0;
  for(int i = 1; i < cl; i++) s[i] = s[i - 1] + q[i] * q[i + 1];
  cl2 = cl;
  ans = s1[cl1 - 1] + s2[cl2 - 1];
  for(int i = 1; i <= Q; i++)
  {
    x.scan();
    ans1 = ans;
    cl = cl1; q = q1; s = s1; cmp = lss;
    calc();
    cl = cl2; q = q2; s = s2; cmp = grt;
    calc();
    ans1 = abs(ans1);
    printf("%lld.%c\n", ans1 / 2, ans1 % 2 == 1?'5':'0');
  }
  fclose(stdin);
  return 0;
}
```

4 圆交

```
double pi = acos(-1.0), eps = 1e-12;
double sqr(const double & x) {
  return x * x;
}
double ans[2001];
int sign(const double & x) {
  return x < -eps?-1:x > eps;
}
struct Point {
  double x, y;
  Point(){}
  Point(const double & x, const double & y) : x(x), y(y) {}
  void scan() {scanf("%lf%lf", &x, &y);}
  double sqrlen() {return sqr(x) + sqr(y);}
  double len() {return sqrt(sqrlen());}
  Point rev() {return Point(y, -x);}
  void print() {printf("%f␣%f\n", x, y);}
  Point zoom(const double & d) {double lambda = d / len(); return Point(lambda * x, lambda * y);}
} dvd, a[2001];
Point centre[2001];
double atan2(const Point & x) {
  return atan2(x.y, x.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);
}
double operator * (const Point & a, const Point & b) {
  return a.x * b.y - a.y * b.x;
}
Point operator * (const double & a, const Point & b) {
  return Point(a * b.x, a * b.y);
}
double operator % (const Point & a, const Point & b) {
  return a.x * b.x + a.y * b.y;
}
struct circle {
  double r; Point o;
  circle() {}
  void scan() {
    o.scan();
    scanf("%lf", &r);
  }
} cir[2001];
struct arc {
  double theta;
  int delta;
  Point p;
  arc() {};
  arc(const double & theta, const Point & p, int d) : theta(theta), p(p), delta(d) {}
} vec[4444];
int nV;
inline bool operator < (const arc & a, const arc & b) {
  return a.theta + eps < b.theta;
}
int cnt;
inline void psh(const double t1, const Point p1, const double t2, const Point p2) {
  if(t2 + eps < t1)
    cnt++;
  vec[nV++] = arc(t1, p1, 1);
  vec[nV++] = arc(t2, p2, -1);
}
inline double cub(const double & x) {
  return x * x * x;
}
inline void combine(int d, const double & area, const Point & o) {
  if(sign(area) == 0) return;
  centre[d] = 1 / (ans[d] + area) * (ans[d] * centre[d] + area * o);
  ans[d] += area;
}
bool equal(const double & x, const double & y) {
  return x + eps>  y and y + eps > x;
}
bool equal(const Point & a, const Point & b) {
  return equal(a.x, b.x) and equal(a.y, b.y);
}
bool equal(const circle & a, const circle & b) {
  return equal(a.o, b.o) and equal(a.r, b.r);
}
bool f[2001];
int main() {
  //freopen("hdu4895.in", "r", stdin);
  int n, m, index;
  while(EOF != scanf("%d%d%d", &m, &n, &index)) {
    index--;
    for(int i(0); i < m; i++) {
      a[i].scan();
    }
    for(int i(0); i < n; i++) {
      cir[i].scan();// n个圆
    }
    for(int i(0); i < n; i++) { // delete the same circle
      f[i] = true;
      for(int j(0); j < n; j++) if(i != j) {
        if(equal(cir[i], cir[j]) and i < j or !equal(cir[i], cir[j]) and cir[i].r < cir[j].r + eps and (
              cir[i].o - cir[j].o).sqrlen() < sqr(cir[i].r - cir[j].r) + eps) {
          f[i] = false;
          break;
        }
      }
    }
    int n1(0);
    for(int i(0); i < n; i++)
      if(f[i])
        cir[n1++] = cir[i];
    n = n1;// 去重圆结束
    fill(ans, ans + n + 1, 0);// ans[i]表示被圆覆盖至少i次的面积
    fill(centre, centre + n + 1, Point(0, 0));// centre[i]表示上面ans[i]部分的重心
    for(int i(0); i < m; i++)
      combine(0, a[i] * a[(i + 1) % m] * 0.5, 1. / 3 * (a[i] + a[(i + 1) % m]));
    for(int i(0); i < n; i++) {
      dvd = cir[i].o - Point(cir[i].r, 0);
      nV = 0;
      vec[nV++] = arc(-pi, dvd, 1);
      cnt = 0;
      for(int j(0); j < n; j++) if(j != i) {
        double d = (cir[j].o - cir[i].o).sqrlen();
        if(d < sqr(cir[j].r - cir[i].r) + eps) {
          if(cir[i].r + i * eps < cir[j].r + j * eps)
            psh(-pi, dvd, pi, dvd);
        }else if(d + eps < sqr(cir[j].r + cir[i].r)) {
          double lambda = 0.5 * (1 + (sqr(cir[i].r) - sqr(cir[j].r)) / d);
          Point cp(cir[i].o + lambda * (cir[j].o - cir[i].o));
          Point nor((cir[j].o - cir[i].o).rev().zoom(sqrt(sqr(cir[i].r) - (cp - cir[i].o).sqrlen())));
          Point frm(cp + nor);
          Point to(cp - nor);
          psh(atan2(frm - cir[i].o), frm, atan2(to - cir[i].o), to);
        }
      }
      sort(vec + 1, vec + nV);
      vec[nV++] = arc(pi, dvd, -1);
      for(int j = 0; j + 1 < nV; j++) {
        cnt += vec[j].delta;
        //if(cnt == 1) {// 如果只算ans[1]和centre[1], 可以加这个if加速.
          double theta(vec[j + 1].theta - vec[j].theta);
          double area(sqr(cir[i].r) * theta * 0.5);
          combine(cnt, area, cir[i].o + 1. / area / 3 * cub(cir[i].r) * Point(sin(vec[j + 1].theta) - sin(vec[j].theta), cos(vec[j].theta) - cos(vec[j + 1].theta)));
          combine(cnt, -sqr(cir[i].r) * sin(theta) * 0.5, 1. / 3 * (cir[i].o + vec[j].p + vec[j + 1].p));
          combine(cnt, vec[j].p * vec[j + 1].p * 0.5, 1. / 3 * (vec[j].p + vec[j + 1].p));
        //}
      }
    }// 板子部分结束 下面是题目
    combine(0, -ans[1], centre[1]);
    for(int i = 0; i < m; i++) {
      if(i != index)
        (a[index] - Point((a[i] - a[index]) * (centre[0] - a[index]), (a[i] - a[index]) % (centre[0] - a[index])).zoom((a[i] - a[index]).len())).print();
      else
        a[i].print();
    }

  }
  fclose(stdin);
  return 0;
}
```

5 判断圆存在交集

传入 n 个圆，圆心存在 cir 中，半径存在 radius 中，nlogk 判断是否存在交集

```
int n;
double sx, sy, d;
vector<Point> cir;
vector<double> radius;
int isIntersectCircleToCircle(Point c1, double r1, Point c2, double r2)
{
  double dis = c1.distTo(c2);
  return sign(dis - (r1 + r2)) <= 0;
}
void getRange(double x, Point &c, double r, double &retl, double &retr)
{
  double tmp = sqrt(max(r * r - (c.x - x) * (c.x - x), 0.0));
  retl = c.y - tmp; retr = c.y + tmp;
}
int checkInLine(double x)
{
  double minR = INF, maxL = -INF;
  double tmpl, tmpr;
  for(int i = 0; i < n; ++ i) {
    if (sign(cir[i].x + radius[i] - x) < 0 || sign(cir[i].x - radius[i] - x) > 0)
      return false;
    getRange(x, cir[i], radius[i], tmpl, tmpr);
    maxL = max(tmpl, maxL);
    minR = min(tmpr, minR);
    if (maxL > minR) return false;
  }
  return true;
}
int shouldGoLeft(double x)
{
  if (checkInLine(x)) return 2;
  int onL = 0, onR = 0;
  for(int i = 0; i < n; ++ i) {
    if (sign(cir[i].x + radius[i] - x) < 0) onL = 1;
    if (sign(cir[i].x - radius[i] - x) > 0) onR = 1;
  }
  if (onL && onR) return -1;
  if (onL) return 1;
  if (onR) return 0;

  double minR = INF, maxL = -INF, tmpl, tmpr;
  int idMinR, idMaxL;

  for(int i = 0; i < n; ++ i) {
    getRange(x, cir[i], radius[i], tmpl, tmpr);
    if (tmpr < minR) {
      minR = tmpr;
      idMinR = i;
    }
    if (tmpl > maxL) {
      maxL = tmpl;
      idMaxL = i;
    }
  }
  if (! isIntersectCircleToCircle(cir[idMinR], radius[idMinR], cir[idMaxL], radius[idMaxL]))
    return -1;
  Point p1, p2;
  intersectionCircleToCircle(cir[idMinR], radius[idMinR], cir[idMaxL], radius[idMaxL], p1, p2);
  return (p1.x < x);
}
int hasIntersectionCircles()
{
  double l = -INF, r = INF, mid;
  for(int i = 0; i < 100; ++ i) {
    mid = (l + r) * 0.5;
    int tmp = shouldGoLeft(mid);
    if (tmp < 0) return 0;
    if (tmp == 2) return 1;
    if (tmp) r = mid;
    else l = mid;
  }
  mid = (l + r) * 0.5;
  return checkInLine(mid);
}
```

6 三维点类

```
struct frac {
  long long x, y;
  frac() {};
  inline frac(const long long &_x, const long long _y) : x(_x), y(_y) {
    long long d = gcd(x, y);
    if (d < 0) d = -d;
    if (d > 1)
      x /= d, y /= d;
    if (y < 0) y = -y, x = -x;
  }
};
inline frac operator+(const frac &a, const frac &b)
{
//  long long y = a.y / gcd(a.y, b.y) * b.y;
//  return frac(y / a.y * a.x + y / b.y * b.x, y);
  return frac(a.x * b.y + a.y * b.x, a.y * b.y);
}
inline frac operator-(const frac &a, const frac &b)
{
//  long long y = a.y / gcd(a.y, b.y) * b.y;
//  return frac(y / a.y * a.x - y / b.y * b.x, y);
  return frac(a.x * b.y - a.y * b.x, a.y * b.y);
}
inline frac operator*(const frac &a, const frac &b)
{
//  long long v = gcd(a.x, b.y), w = gcd(a.y, b.x);
//  return frac((a.x / v) * (b.x / w), (a.y / w) * (b.y / v));
  return frac(a.x * b.x, a.y * b.y);
}
inline frac operator/(const frac &a, const frac &b)
{
//  long long v = gcd(a.x, b.x), w = gcd(a.y, b.y);
//  return frac((a.x / v) * (b.y / w), (a.y / w) * (b.x / v));
  return frac(a.x * b.y, a.y * b.x);
}
inline bool operator<(const frac &a, const frac &b)
{
  return a.x * b.y < a.y * b.x;
}
```

```cpp
inline bool operator==(const frac &a, const frac &b)
{
  return a.x * b.y == a.y * b.x;
}
inline bool operator<=(const frac &a, const frac &b)
{
  return a.x * b.y <= a.y * b.x;
}
inline frac sqr(const frac &a) {
  return a * a;
}
struct point {
  frac x, y, z;
  point() {};
  inline point(const frac &x, const frac &y, const frac &z) : x(x), y(y), z(z){};
  inline void scan() {
    scanf("%lld%lld%lld", &x.x, &y.x, &z.x);
    x.y = y.y = z.y = 1;
  }
  inline frac sqrlen() {
    return x * x + y * y + z * z;
  }
}A, B, C, D;
inline point operator-(const point &a, const point &b)
{
  return point(a.x - b.x, a.y - b.y, a.z - b.z);
}
inline point operator+(const point &a, const point &b)
{
  return point(a.x + b.x, a.y + b.y, a.z + b.z);
}
inline point operator*(const frac &a, const point &b)
{
  return point(a * b.x, a * b.y, a * b.z);
}
inline frac operator%(const point &a, const point &b)
{
  return a.x * b.x + a.y * b.y + a.z * b.z;
}
inline 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);
}
inline int sgn(const frac &a)
{
  return a.x < 0 ? -1 : a.x > 0;
}
inline void check(frac &ans, const point &a, const point &s, const point &t)
{
  if (sgn((a - s) % (t - s)) * sgn((a - t) % (t - s)) <= 0) //点到直线的垂足在线段上（包括端点）
    ans = min(ans, ((a - s) * (t - s)).sqrlen() / (t - s).sqrlen());//点到直线距离
}
int main()
{
  int T;
  scanf("%d", &T);
  while (T--) {
    A.scan(), B.scan();
    C.scan(), D.scan();
    frac ans = (A - C).sqrlen();
    ans = min(ans, (A - D).sqrlen());
    ans = min(ans, (B - C).sqrlen());
    ans = min(ans, (B - D).sqrlen());
    point nor = (B - A) * (D - C);
    if (!(nor.x == frac(0, 1) && nor.y == frac(0, 1) && nor.z == frac(0, 1)))//线段平行
      if (sgn((C - A) * (D - A) % nor) * sgn((C - B) * (D - B) % nor) <= 0 &&
          sgn((A - C) * (B - C) % nor) * sgn((A - D) * (B - D) % nor) <= 0)//三维跨立
        ans = min(ans, sqr(nor % (C - A)) / nor.sqrlen());
    check(ans, A, C, D);
    check(ans, B, C, D);
    check(ans, C, A, B);
    check(ans, D, A, B);
    printf("%lld␣%lld\n", ans.x, ans.y);
  }
  return 0;
}
```

7 球

计算圆心角 lat 表示纬度,$-90 \leq w \leq 90$,lng 表示经度返回两点所在大圆劣弧对应圆心角,0≤angle≤ π

```cpp
double angle (double lng1 ,double lat1 ,double lng2 ,double lat2 ) {
  double dlng = abs(lng1 - lng2) * PI / 180;
  while(dlng >= PI + PI) dlng -= PI + PI;
  if (dlng > PI) dlng = PI + PI - dlng;
  lat1 *= PI / 180 , lat2 *= PI / 180;
  return acos(cos(lat1) * cos(lat2) * cos(dlng) + sin(lat1) * sin(lat2));
}
```

计算直线距离，r 为球半径

```cpp
double line_dist(double r,double lng1,double lat1,double lng2,double lat2) {
  double dlng = abs(lng1 - lng2) * PI / 180;
  while(dlng >= PI + PI) dlng -= PI + PI;
  if (dlng > PI) dlng = PI + PI - dlng;
  lat1 *= PI / 180 , lat2 *= PI / 180;
  return r * sqrt(2 - 2 * (cos(lat1) * cos(lat2) * cos(dlng) + sin(lat1) * sin(lat2)));
}
```

计算球面距离，r 为球半径

```cpp
inline double sphere_dist(double r,double lng1,double lat1,double lng2,double lat2)
{
  return r * angle(lng1, lat1, lng2, lat2);
}
```

8 点类 + 三维凸包 N^3+ 凸包求重心

```cpp
struct triple
{
  double x, y, z;
  double sqrlen() {return x * x + y * y + z * z;}
  double len() {return sqrt(sqrlen());}
  triple(){}
  triple(double _x, double _y, double _z) : x(_x), y(_y), z(_z){}
} a[111];
char name[111][211];
bool flag, ext[111];
int l, real[111], cnt, n, f[111][111];
struct plane
{
  int a[3];
  plane(int _x, int _y, int _z)
  {
    a[0] = _x;
    a[1] = _y;
    a[2] = _z;
  }
  int & operator [] (int x)
  {
    return a[x];
  }
};
vector<plane> surf;
triple operator * (const triple & a, const triple & b)
{
  return triple(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
}
triple operator * (const double & lambda, const triple & b)
{
  return triple(lambda * b.x, lambda * b.y, lambda * b.z);
}
double operator % (const triple & a, const triple & b)
{
  return a.x * b.x + a.y * b.y + a.z * b.z;
}
triple operator - (const triple & a, const triple & b)
{
  return triple(a.x - b.x, a.y - b.y, a.z - b.z);
}
triple operator + (const triple & a, const triple & b)
{
  return triple(a.x + b.x, a.y + b.y, a.z + b.z);
}
double volume(const triple & o, int j)//volume of a tetrahedron := {a point and a triangle undersurface}
{
  return (a[surf[j][0]] - o) * (a[surf[j][1]] - o) % (a[surf[j][2]] - o);//can be negative
}
double volume(int i, int j)
```

```
{
  return volume(a[i], j);
}
double above(int i, int j) {return volume(i, j) > 0;}//point above plane
double on(int i, int j) {return volume(i, j) == 0;}//point on plane
void print(const triple & x, char ch)
{
  printf("(%lf,␣%lf,␣%lf)%c", x.x, x.y, x.z, ch);
}
double dis(const triple & o, int j)//point to plane
{
  return fabs(volume(o, j) / ((a[surf[j][1]] - a[surf[j][0]]) * (a[surf[j][2]] - a[surf[j][0]])).len());
}
int main()
{
  double ans = 0;
  for(int cv = 1; cv <= 2; cv++)
  {
    scanf("%d", &n);
    for(int i = 1; i <= n; i++)
    {
      scanf("%lf%lf%lf", &a[i].x, &a[i].y, &a[i].z);
    }
    //->degenerate checking
    flag = false;
    for(int i = 3; i <= n; i++)
    {
      if(((a[1] - a[i]) * (a[2] - a[i])).sqrlen() != 0)
      {
        swap(a[3], a[i]);
        swap(real[i], real[3]);
        for(int j = 4; j <= n; j++)
        {
          if((a[1] - a[j]) * (a[2] - a[j]) % (a[3] - a[j]) != 0)
          {
            swap(a[4], a[j]);
            swap(real[4], real[j]);
            flag = true;
            break;
          }
        }
        break;
      }
    }
    /*if(flag == false)
    {
      //degenerate!
    }else
    {*/
    //->convex polyhedra
    memset(f, 0, sizeof(f));
    surf.clear();
    surf.push_back(plane(1, 2, 3));
    surf.push_back(plane(3, 2, 1));
    for(int i = 4; i <= n; i++)
    {
      vector<plane> tmp;
      for(int j = 0; j < surf.size(); j++)
        if(above(i, j))
        {
          for(int d = 0; d < 3; d++)
          {
            f[surf[j][d]][surf[j][(d + 2) % 3]] = i;
          }
        }else
        {
          tmp.push_back(surf[j]);
        }
      surf = tmp;
      for(int j = surf.size() - 1; j >= 0; j--)
      {
        for(int d = 0; d < 3; d++)
          if(f[surf[j][d]][surf[j][(d + 1) % 3]] == i) surf.push_back(plane(surf[j][(d + 1) % 3], surf[j][d], i));
      }
    }
    //end convex polyhedra, result := surf
    //->centre of gravity
    double svol = 0;
    triple qc(0, 0, 0);
    for(int i = 0; i < surf.size(); i++)
    {
      double vol1 = volume(1, i);
      qc = qc + (vol1 / 4) * (a[1] + a[surf[i][0]] + a[surf[i][1]] + a[surf[i][2]]);
      svol += vol1;
    }
    qc = (1 / svol) * qc;
    double mn = 1e9;
    for(int i = 0; i < surf.size(); i++)
    {
      mn = min(mn, dis(qc, i));
    }
    ans += mn;
    //end centre of gravity
    //}
  }
  printf("%.5f\n", ans);
  fclose(stdin);
  return 0;
}
```

9 三维旋转

```
//sgu265
const double pi = acos(-1.0);
int n, m; char ch1; bool flag;
double a[4][4], s1, s2, x, y, z, w, b[4][4], c[4][4];
double sqr(double x)
{
  return x*x;
}
int main()
{
  scanf("%d\n", &n);
  memset(b, 0, sizeof(b));
  b[0][0] = b[1][1] = b[2][2] = b[3][3] = 1;//initial matrix
  for(int i = 1; i <= n; i++)
  {
    scanf("%c", &ch1);
    if(ch1 == 'T')
    {
      //plus each coordinate by a number (x, y, z)
      scanf("%lf␣%lf␣%lf\n", &x, &y, &z);
      memset(a, 0, sizeof(a));
      a[0][0] = 1; a[3][0] = x;
      a[1][1] = 1; a[3][1] = y;
      a[2][2] = 1; a[3][2] = z;
      a[3][3] = 1;
    }else if(ch1 == 'S')
    {
      //multiply each coordinate by a number (x, y, z)
      scanf("%lf␣%lf␣%lf\n", &x, &y, &z);
      memset(a, 0, sizeof(a));
      a[0][0] = x;
      a[1][1] = y;
      a[2][2] = z;
      a[3][3] = 1;
    }else
    {
      //rotate in a clockwise about the ray from the origin through (x, y, z);
      scanf("%lf␣%lf␣%lf␣%lf\n", &x, &y, &z, &w);
      w = w*pi/180;
      memset(a, 0, sizeof(a));
      s1 = x*x+y*y+z*z;
      a[3][3] = 1;
      a[0][0] = ((y*y+z*z)*cos(w)+x*x)/s1;
      a[0][1] = x*y*(1-cos(w))/s1+z*sin(w)/sqrt(s1);
      a[0][2] = x*z*(1-cos(w))/s1-y*sin(w)/sqrt(s1);
      a[1][0] = x*y*(1-cos(w))/s1-z*sin(w)/sqrt(s1);
      a[1][1] = ((x*x+z*z)*cos(w)+y*y)/s1;
      a[1][2] = y*z*(1-cos(w))/s1+x*sin(w)/sqrt(s1);
      a[2][0] = x*z*(1-cos(w))/s1+y*sin(w)/sqrt(s1);
      a[2][1] = y*z*(1-cos(w))/s1-x*sin(w)/sqrt(s1);
      a[2][2] = ((x*x+y*y)*cos(w)+z*z)/s1;
    }
    memset(c, 0, sizeof(c));
    for(int i = 0; i < 4; i++)
      for(int j = 0; j < 4; j++)
        for(int k = 0; k < 4; k++)
          c[i][j] += b[i][k]*a[k][j];
    memcpy(b, c, sizeof(c));
  }
  scanf("%d", &m);
  for(int i = 1; i <= m; i++)
  {
    scanf("%lf%lf%lf", &x, &y, &z);//initial vector
    printf("%lf␣%lf␣%lf\n", x*b[0][0]+y*b[1][0]+z*b[2][0]+b[3][0], x*b[0][1]+y*b[1][1]+z*b[2][1]+b[3][1], x*b[0][2]+y*b[1][2]+z*b[2][2]+b[3][2]);
  }
  return 0;
}
```

10 最小球覆盖

随机增量法，复杂度没有找到靠谱证明，暂且可以类似最小圆覆盖当线性用

```
const int nmax = 30 + 18;
const double eps = 1e-12;
struct Tpoint {
  double x, y, z;
  Tpoint() {};
  Tpoint(double _x, double _y, double _z) : x(_x), y(_y), z(_z) {};
  Tpoint operator+(const Tpoint &a) const {
    return Tpoint(x + a.x, y + a.y, z + a.z);
  }
  Tpoint operator-(const Tpoint &a) const {
    return Tpoint(x - a.x, y - a.y, z - a.z);
  }
  double operator%(const Tpoint &a) const {
    return x * a.x + y * a.y + z * a.z;
  }
  Tpoint operator/(const double &lambda) const {
    return Tpoint(x / lambda, y / lambda, z / lambda);
  }
  void scan() {
    scanf("%lf%lf%lf", &x, &y, &z);
  }
};
Tpoint operator*(const double &lambda, const Tpoint &a) {
  return Tpoint(lambda * a.x, lambda * a.y, lambda * a.z);
}
int npoint, nouter;
Tpoint pt[nmax], outer[4], res;
double radius, tmp;
double dist2(const Tpoint &p1, const Tpoint &p2)
{
  double dx = p1.x - p2.x, dy = p1.y - p2.y, dz = p1.z - p2.z;
  return dx * dx + dy * dy + dz * dz;
}
void ball()
{
  Tpoint q[3];
  double m[3][3], sol[3], L[3], det;
  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 (int i = 0; i < 2; ++i)
        q[i] = outer[i + 1] - outer[0];
      for (int i = 0; i < 2; ++i)
        for (int j = 0; j < 2; ++j)
          m[i][j] = q[i] % q[j] * 2;
      for (int i = 0; i < 2; ++i)
        sol[i] = 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] + L[0] * q[0] + L[1] * q[1];
      radius = dist2(res, outer[0]);
      break;
    case 4:
      for (int i = 0; i < 3; ++i) {
        q[i] = outer[i + 1] - outer[0];
        sol[i] = q[i] % q[i];
      }
      for (int i = 0; i < 3; ++i)
        for (int j = 0; j < 3; ++j)
          m[i][j] = 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 (int j = 0; j < 3; ++j) {
        for (int 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 (int i = 0; i < 3; ++i)
          m[i][j] = q[i] % q[j] * 2;
      }
      res = outer[0];
      for (int i = 0; i < 3; ++i)
        res = res + L[i] * q[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];
        ++nouter;
        minball(i);
        --nouter;
        if (i > 0) {
          Tpoint Tt = pt[i];
          memmove(&pt[1], &pt[0], sizeof(Tpoint) * i);
          pt[0] = Tt;
        }
      }
}
void solve()
{
  for (int i = 0; i < npoint; ++i)
    pt[i].scan();
  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) + 0.000001);
}
int main()
{
  while (1) {
    scanf("%d", &npoint);
    if (npoint == 0) break;
    solve();
  }
  return 0;
}
```

搜索

11 DLX

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#define maxn 105
#define N maxn*maxn
int a[maxn][maxn],l[N],r[N],d[N],u[N],c[N],s[maxn],head[maxn],n,m,ans;
inline int getid(int x,int y){return (x-1)*n+y;}
void remove(int x){
  l[r[x]]=l[x];r[l[x]]=r[x];
  for (int i=d[x];i!=x;i=d[i])
    for (int j=r[i];j!=i;j=r[j]){
      u[d[j]]=u[j];d[u[j]]=d[j];
      --s[c[j]];
    }
}
void resume(int x){
  for (int i=u[x];i!=x;i=u[i])
    for (int j=l[i];j!=i;j=l[j]){
      u[d[j]]=j;d[u[j]]=j;
      ++s[c[j]];
    }
  l[r[x]]=x;r[l[x]]=x;
}
void dfs(int t){
  if (t>=ans)return;
  if (!r[0]){
    if (t<ans)ans=t;
    return;
  }
  int x=0,min=1<<30;
  for (int i=r[0];i;i=r[i])
    if (s[i]<min)min=s[i],x=i;
  remove(x);
  for (int i=d[x];i!=x;i=d[i]){
    for (int j=r[i];j!=i;j=r[j])remove(c[j]);
    dfs(t+1);
    for (int j=l[i];j!=i;j=l[j])resume(c[j]);
  }
  resume(x);
}
int main()
{
  memset(a,0,sizeof(a));
  scanf("%d%d",&m,&n);
  for (int i=1;i<=n;++i){
    int x,y;scanf("%d",&x);
    for (int j=1;j<=x;++j){
      scanf("%d",&y);a[i][y]=1;
    }
  }
  for (int i=1;i<=m;++i)head[i]=n*m+i; head[0]=0;
  for (int i=1;i<=m;++i)r[head[i]]=head[i+1];
  for (int i=1;i<=m;++i)l[head[i]]=head[i-1];
  r[head[0]]=head[1];l[head[1]]=head[0];
  l[head[0]]=head[m];r[head[m]]=head[0];
  for (int i=1;i<=n;++i){
    int pre=0,first=0;
    for (int j=1;j<=m;++j)if (a[i][j]){
      if (pre)l[getid(i,j)]=getid(i,pre),r[getid(i,pre)]=getid(i,j);
      pre=j;if (!first)first=j;
    }
    if (first){
      l[getid(i,first)]=getid(i,pre);r[getid(i,pre)]=getid(i,first);
    }
  }
  for (int j=1;j<=m;++j){
    int pre=0,first=0;
    for (int i=1;i<=n;++i)if (a[i][j]){
      if (pre)u[getid(i,j)]=getid(pre,j),d[getid(pre,j)]=getid(i,j);
      pre=i;if (!first)first=i;
    }
    if (pre){
      u[getid(first,j)]=head[j];d[head[j]]=getid(first,j);
      u[head[j]]=getid(pre,j);d[getid(pre,j)]=head[j];
    }
  }
  for (int i=1;i<=n;++i)
    for (int j=1;j<=m;++j)if (a[i][j])c[getid(i,j)]=head[j];
  memset(s,0,sizeof(s));
  for (int i=1;i<=n;++i)
    for (int j=1;j<=m;++j)if (a[i][j])++s[j];
  ans=1<<30;
  dfs(0);
  if (ans==1<<30)printf("-1\n");
  else printf("%d\n",ans);
  system("pause");for (;;);
  return 0;
}
```

数学

12 多项式求根 (求导二分)

```
const double error=1e-12;
const double infi=1e+12;
double a[10],x[10];
int n;
int sign(double x) {
  return (x<-error)?(-1):(x>error);
}
double f(double a[],int n,double x) {
  double tmp=1,sum=0;
  for (int i=0;i<=n;i++) {
    sum=sum+a[i]*tmp;
    tmp=tmp*x;
  }
  return sum;
}
double binary(double l,double r,double a[],int n) {
  int sl=sign(f(a,n,l)),sr=sign(f(a,n,r));
  if (sl==0) return l;
  if (sr==0) return r;
  if (sl*sr>0) return infi;
  while (r-l>error) {
    double mid=(l+r)/2;
    int ss=sign(f(a,n,mid));
    if (ss==0) return mid;
    if (ss*sl>0) l=mid; else r=mid;
  }
  return l;
}
void solve(int n,double a[],double x[],int &nx) {
  if (n==1) {
    x[1]=-a[0]/a[1];
    nx=1;
    return;
  }
  double da[10],dx[10];
  int ndx;
  for (int i=n;i>=1;i--) da[i-1]=a[i]*i;
  solve(n-1,da,dx,ndx);
  nx=0;
  if (ndx==0) {
    double tmp=binary(-infi,infi,a,n);
    if (tmp<infi) x[++nx]=tmp;
    return;
  }
  double tmp;
  tmp=binary(-infi,dx[1],a,n);
  if (tmp<infi) x[++nx]=tmp;
  for (int i=1;i<=ndx-1;i++) {
    tmp=binary(dx[i],dx[i+1],a,n);
    if (tmp<infi) x[++nx]=tmp;
  }
  tmp=binary(dx[ndx],infi,a,n);
  if (tmp<infi) x[++nx]=tmp;
}
int main() {
  scanf("%d",&n);
  for (int i=n;i>=0;i--) scanf("%lf",&a[i]);
  int nx;
  solve(n,a,x,nx);
  for (int i=1;i<=nx;i++) printf("%0.6lf\n",x[i]);
  return 0;
}
```

13 cheat sheet

– Junru Shao ACM Honored Class, Zhiyuan College Shanghai Jiao Tong University

- $\mathrm{d}(\tan x) = \sec^2 x \mathrm{d}x$
- $\mathrm{d}(\cot x) = \csc^2 x \mathrm{d}x$
- $\mathrm{d}(\sec x) = \tan x \, \sec x \mathrm{d}x$
- $\mathrm{d}(\csc x) = -\cot x \, \csc x \mathrm{d}x$
- $d(\mathrm{arcsin} x) = \dfrac{1}{\sqrt{1-x^2}} \mathrm{d}x$
- $d(\mathrm{arccos} x) = \dfrac{-1}{\sqrt{1-x^2}} \mathrm{d}x$
- $d(\mathrm{arctan} x) = \dfrac{1}{1+x^2} \mathrm{d}x$

- $d(\text{arccot}x) = \dfrac{-1}{1+x^2}\mathrm{d}x$
- $d(\text{arcsec}x) = \dfrac{1}{x\sqrt{1-x^2}}\mathrm{d}x$
- $d(\text{arccsc}x) = \dfrac{-1}{u\sqrt{1-x^2}}\mathrm{d}x$
- $\int cu\,\mathrm{d}x = c\int u\,\mathrm{d}x$
- $\int (u+v)\,\mathrm{d}x = \int u\,\mathrm{d}x + \int v\,\mathrm{d}x$
- $\int x^n\,\mathrm{d}x = \dfrac{1}{n+1}x^{n+1}, \quad n \neq -1$
- $\int \dfrac{1}{x}\mathrm{d}x = \ln x$
- $\int e^x\,\mathrm{d}x = e^x$
- $\int \dfrac{\mathrm{d}x}{1+x^2} = \text{arctan}x$
- $\int u\dfrac{\mathrm{d}v}{\mathrm{d}x}\mathrm{d}x = uv - \int v\dfrac{\mathrm{d}u}{\mathrm{d}x}\mathrm{d}x$
- $\int \sin x\,\mathrm{d}x = -\cos x$
- $\int \cos x\,\mathrm{d}x = \sin x$
- $\int \tan x\,\mathrm{d}x = -\ln|\cos x|$
- $\int \cot x\,\mathrm{d}x = \ln|\cos x|$
- $\int \sec x\,\mathrm{d}x = \ln|\sec x + \tan x|$
- $\int \csc x\,\mathrm{d}x = \ln|\csc x + \cot x|$
- $\int \text{arcsin}\dfrac{x}{a}\mathrm{d}x = \text{arcsin}\dfrac{x}{a} + \sqrt{a^2-x^2}, \quad a > 0$
- $\int \text{arccos}\dfrac{x}{a}\mathrm{d}x = \text{arccos}\dfrac{x}{a} - \sqrt{a^2-x^2}, \quad a > 0$
- $\int \text{arctan}\dfrac{x}{a}\mathrm{d}x = x\text{arctan}\dfrac{x}{a} - \dfrac{a}{2}\ln(a^2+x^2), \quad a > 0$
- $\int \sin^2(ax)\mathrm{d}x = \dfrac{1}{2a}\big(ax - \sin(ax)\cos(ax)\big)$
- $\int \cos^2(ax)\mathrm{d}x = \dfrac{1}{2a}\big(ax + \sin(ax)\cos(ax)\big)$
- $\int \sec^2 x\,\mathrm{d}x = \tan x$
- $\int \csc^2 x\,\mathrm{d}x = -\cot x$
- $\int \sin^n x\,\mathrm{d}x = -\dfrac{\sin^{n-1}x\cos x}{n} + \dfrac{n-1}{n}\int \sin^{n-2}x\,\mathrm{d}x$
- $\int \cos^n x\,\mathrm{d}x = \dfrac{\cos^{n-1}x\sin x}{n} + \dfrac{n-1}{n}\int \cos^{n-2}x\,\mathrm{d}x$
- $\int \tan^n x\,\mathrm{d}x = \dfrac{\tan^{n-1}x}{n-1} - \int \tan^{n-2}x\,\mathrm{d}x, \quad n \neq 1$
- $\int \cot^n x\,\mathrm{d}x = -\dfrac{\cot^{n-1}x}{n-1} - \int \cot^{n-2}x\,\mathrm{d}x, \quad n \neq 1$
- $\int \sec^n x\,\mathrm{d}x = \dfrac{\tan x\sec^{n-1}x}{n-1} + \dfrac{n-2}{n-1}\int \sec^{n-2}x\,\mathrm{d}x, \quad n \neq 1$
- $\int \csc^n x\,\mathrm{d}x = -\dfrac{\cot x\csc^{n-1}x}{n-1} + \dfrac{n-2}{n-1}\int \csc^{n-2}x\,\mathrm{d}x, \quad n \neq 1$
- $\int \sinh x\,\mathrm{d}x = \cosh x$
- $\int \cosh x\,\mathrm{d}x = \sinh x$
- $\int \tanh x\,\mathrm{d}x = \ln|\cosh x|$
- $\int \coth x\,\mathrm{d}x = \ln|\sinh x|$
- $\int \text{sech}x\,\mathrm{d}x = \arctan\sinh x$
- $\int \text{csch}x\,\mathrm{d}x = \ln\left|\tanh\dfrac{x}{2}\right|$
- $\int \sinh^2 x\,\mathrm{d}x = \dfrac{1}{4}\sinh(2x) - \dfrac{1}{2}x$
- $\int \cosh^2 x\,\mathrm{d}x = \dfrac{1}{4}\sinh(2x) + \dfrac{1}{2}x$
- $\int \text{sech}^2 x\,\mathrm{d}x = \tanh x$
- $\int \text{arcsinh}\dfrac{x}{a}\mathrm{d}x = x\text{arcsinh}\dfrac{x}{a} - \sqrt{x^2+a^2}, \quad a > 0$
- $\int \text{arctanh}\dfrac{x}{a}\mathrm{d}x = x\text{arctanh}\dfrac{x}{a} + \dfrac{a}{2}\ln|a^2-x^2|$
- $\int \text{arccosh}\dfrac{x}{a}\mathrm{d}x = \begin{cases} x\text{arccosh}\dfrac{x}{a} - \sqrt{x^2+a^2}, \text{if } \text{arccosh}\dfrac{x}{a} > 0 \text{ and } a > 0 \\ x\text{arccosh}\dfrac{x}{a} + \sqrt{x^2+a^2}, \text{if } \text{arccosh}\dfrac{x}{a} < 0 \text{ and } a > 0 \end{cases}$
- $\int \dfrac{\mathrm{d}x}{\sqrt{a^2+x^2}} = \ln\left(x + \sqrt{a^2+x^2}\right), \quad a > 0$
- $\int \dfrac{\mathrm{d}x}{a^2+x^2} = \dfrac{1}{a}\text{arctan}\dfrac{x}{a}, \quad a > 0$
- $\int \sqrt{a-x^2}\,\mathrm{d}x = \dfrac{x}{2}\sqrt{a^2-x^2} + \dfrac{a^2}{2}\text{arcsin}\dfrac{x}{a}, \quad a > 0$

- $\int (a^2-x)^{3/2}\mathrm{d}x = \frac{x}{8}(5a^2-2x^2)\sqrt{a^2-x^2}+\frac{3a^4}{8}\arcsin\frac{x}{a}, \quad a>0$
- $\int \frac{\mathrm{d}x}{\sqrt{a^2-x^2}} = \arcsin\frac{x}{a}, \quad a>0$
- $\int \frac{\mathrm{d}x}{a^2-x^2} = \frac{1}{2a}\ln\left|\frac{a+x}{a-x}\right|$
- $\int \frac{\mathrm{d}x}{(a^2-x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2-x^2}}$
- $\int \sqrt{a^2\pm x^2}\,\mathrm{d}x = \frac{x}{2}\sqrt{a^2\pm x^2}\pm\frac{a^2}{2}\ln\left|x+\sqrt{a^2\pm x^2}\right|$
- $\int \frac{\mathrm{d}x}{\sqrt{x^2-a^2}} = \ln\left|x+\sqrt{x^2-a^2}\right|, \quad a>0$
- $\int \frac{\mathrm{d}x}{ax^2+bx} = \frac{1}{a}\ln\left|\frac{x}{a+bx}\right|$
- $\int x\sqrt{a+bx}\,\mathrm{d}x = \frac{2(3bx-2a)(a+bx)^{3/2}}{15b^2}$
- $\int \frac{\sqrt{a+bx}}{x}\mathrm{d}x = 2\sqrt{a+bx}+a\int\frac{1}{x\sqrt{a+bx}}\mathrm{d}x$
- $\int \frac{x}{\sqrt{a+bx}}\mathrm{d}x = \frac{1}{\sqrt{2}}\ln\left|\frac{\sqrt{a+bx}-\sqrt{a}}{\sqrt{a+bx}+\sqrt{a}}\right|, \quad a>0$
- $\int \frac{\sqrt{a^2-x^2}}{x}\mathrm{d}x = \sqrt{a^2-x^2}-a\ln\left|\frac{a+\sqrt{a^2-x^2}}{x}\right|$
- $\int x\sqrt{a-x^2}\,\mathrm{d}x = -\frac{1}{3}(a^2-x^2)^{3/2}$
- $\int x^2\sqrt{a^2-x^2}\,\mathrm{d}x = \frac{x}{8}(2x^2-a^2)\sqrt{a^2-x^2}+\frac{a^4}{8}\arcsin\frac{x}{a}, \quad a>0$
- $\int \frac{\mathrm{d}x}{\sqrt{a^2-x^2}} = -\frac{1}{a}\ln\left|\frac{a+\sqrt{a^2-x^2}}{x}\right|$
- $\int \frac{x^2\,\mathrm{d}x}{\sqrt{a^2-x^2}} = -\sqrt{a^2-x^2}$
- $\int \frac{x\,\mathrm{d}x}{\sqrt{a^2-x^2}} = -\frac{x}{2}\sqrt{a^2-x^2}+\frac{a^2}{2}\arcsin\frac{x}{a}, \quad a>0$
- $\int \frac{\sqrt{a^2+x^2}}{x}\mathrm{d}x = \sqrt{a^2+x^2}-a\ln\left|\frac{a+\sqrt{a^2+x^2}}{x}\right|$
- $\int \frac{\sqrt{x^2-a^2}}{x}\mathrm{d}x = \sqrt{x^2-a^2}-a\arccos\frac{a}{|x|}, \quad a>0$
- $\int x\sqrt{x^2\pm a^2}\,\mathrm{d}x = \frac{1}{3}(x^2\pm a^2)^{3/2}$
- $\int \frac{\mathrm{d}x}{x\sqrt{x^2+a^2}} = \frac{1}{a}\ln\left|\frac{x}{a+\sqrt{a^2+x^2}}\right|$
- $\int \frac{\mathrm{d}x}{x\sqrt{x^2-a^2}} = \frac{1}{a}\arccos\frac{a}{|x|}, \quad a>0$
- $\int \frac{\mathrm{d}x}{x^2\sqrt{x^2\pm a^2}} = \mp\frac{\sqrt{x^2\pm a^2}}{a^2x}$
- $\int \frac{x\,\mathrm{d}x}{\sqrt{x^2\pm a^2}} = \sqrt{x^2\pm a^2}$
- $\int \frac{\sqrt{x^2\pm a^2}}{x^4}\mathrm{d}x = \mp\frac{(x^2+a^2)^{3/2}}{3a^2x^3}$
- $\int \frac{\mathrm{d}x}{ax^2+bx+c} = \begin{cases} \frac{1}{\sqrt{b^2-4ac}}\ln\left|\frac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}}\right|, \text{if } b^2>4ac \\ \frac{2}{\sqrt{4ac-b^2}}\arctan\frac{2ax+b}{\sqrt{4ac-b^2}}, \text{if } b^2<4ac \end{cases}$
- $\int \frac{\mathrm{d}x}{\sqrt{ax^2+bx+c}} = \begin{cases} \frac{1}{\sqrt{a}}\ln\left|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}\right|, \text{if } a>0 \\ \frac{1}{\sqrt{-a}}\arcsin\frac{-2ax-b}{\sqrt{b^2-4ac}}, \text{if } a<0 \end{cases}$
- $\int \sqrt{ax^2+bx+c}\,\mathrm{d}x = \frac{2ax+b}{4a}\sqrt{ax^2+bx+c}+\frac{4ax-b^2}{8a}\int\frac{\mathrm{d}x}{\sqrt{ax^2+bx+c}}$
- $\int \frac{x\,\mathrm{d}x}{\sqrt{ax^2+bx+c}} = \frac{\sqrt{ax^2+bx+c}}{a}-\frac{b}{2a}\int\frac{\mathrm{d}x}{\sqrt{ax^2+bx+c}}$
- $\int \frac{\mathrm{d}x}{x\sqrt{ax^2+bx+c}} = \begin{cases} \frac{-1}{\sqrt{c}}\ln\left|\frac{2\sqrt{c}\sqrt{ax^2+bx+c}+bx+2c}{x}\right|, \text{if } c>0 \\ \frac{1}{\sqrt{-c}}\arcsin\frac{bx+2c}{|x|\sqrt{b^2-4ac}}, \text{if } c<0 \end{cases}$
- $\int x^3\sqrt{x^2+a^2}\,\mathrm{d}x = (\frac{1}{3}x^2-\frac{2}{15}a^2)(x^2+a^2)^{3/2}$
- $\int x^n\sin(ax)\,\mathrm{d}x = -\frac{1}{a}x^n\cos(ax)+\frac{n}{a}\int x^{n-1}\cos(ax)\,\mathrm{d}x$
- $\int x^n\cos(ax)\,\mathrm{d}x = \frac{1}{a}x^n\sin(ax)-\frac{n}{a}\int x^{n-1}\sin(ax)\,\mathrm{d}x$
- $\int x^ne^{ax}\,\mathrm{d}x = \frac{x^ne^{ax}}{a}-\frac{n}{a}\int x^{n-1}e^{ax}\,\mathrm{d}x$
- $\int x^n\ln(ax)\,\mathrm{d}x = x^{n+1}\left(\frac{\ln(ax)}{n+1}-\frac{1}{(n+1)^2}\right)$
- $\int x^n(\ln ax)^m\,\mathrm{d}x = \frac{x^{n+1}}{n+1}(\ln ax)^m-\frac{m}{n+1}\int x^n(\ln ax)^{m-1}\,\mathrm{d}x$

14 Polya

对于含 n 个对象的置换群 G，用 t 种颜色着色的不同方案数为：
$l = \frac{1}{|G|} \sum_{g \in G} t^{c(a_g)}$
其中 $G = a_1, a_2, ..., a_g, c(a_k)$ 为置换 a_k 的循环指标数目。

15 几何公式

三角形

1. 半周长 $P = (a + b + c)/2$
2. 面积 $S = aH_a/2 = ab\sin(C)/2 = \sqrt{P(P-a)(P-b)(P-c)}$
3. 中线 $M_a = \sqrt{2(b^2 + c^2) - a^2)}/2 = \sqrt{b^2 + c^2 + 2bc\cos(A)}/2$
4. 角平分线 $T_a = \sqrt{bc((b + c)^2 - a^2)}/(b + c) = 2bc\cos(A/2)/(b + c)$
5. 高线 $H_a = b\sin(C) = c\sin(B) = \sqrt{b^2 - ((a^2 + b^2 - c^2)/(2a))^2}$
6. 内切圆半径
$$\begin{aligned} r &= S/P = \arcsin(B/2)\sin(C/2)/\sin((B + C)/2) = 4R\sin(A/2)\sin(B/2)\sin(C/2) \\ &= \sqrt{(P-a)(P-b)(P-c)/P} = P\tan(A/2)\tan(B/2)\tan(C/2) \end{aligned}$$
7. 外接圆半径 $R = abc/(4S) = a/(2\sin(A)) = b/(2\sin(B)) = c/(2\sin(C))$

四边形

$D1, D2$ 为对角线，M 对角线中点连线，A 为对角线夹角

1. $a^2 + b^2 + c^2 + d^2 = D1^2 + D2^2 + 4M^2$
2. $S = D1D2\sin(A)/2$
3. 圆内接四边形 $ac + bd = D1D2$
4. 圆内接四边形，P 为半周长 $S = \sqrt{(P-a)(P-b)(P-c)(P-d)}$

正 n 边形

R 为外接圆半径,r 为内切圆半径

1. 中心角 $A = 2\pi/n$
2. 内角 $C = (n - 2)\pi/n$
3. 边长 $a = 2\sqrt{R^2 - r^2} = 2R\sin(A/2) = 2r\tan(A/2)$
4. 面积 $S = nar/2 = nr^2\tan(A/2) = nR^2\sin(A)/2 = na^2/(4\tan(A/2))$

圆

1. 弧长 $l = rA$
2. 弦长 $a = 2\sqrt{2hr - h^2} = 2r\sin(A/2)$
3. 弓形高 $h = r - \sqrt{r^2 - a^2/4} = r(1 - \cos(A/2)) = \arctan(A/4)/2$
4. 扇形面积 $S1 = rl/2 = r^2A/2$
5. 弓形面积 $S2 = (rl - a(r - h))/2 = r^2(A - \sin(A))/2$

棱柱

1. 体积 $V = Ah$，A 为底面积，h 为高
2. 侧面积 $S = lp$，l 为棱长，p 为直截面周长
3. 全面积 $T = S + 2A$

棱锥

1. 体积 $V = Ah$，A 为底面积，h 为高
2. 正棱锥侧面积 $S = lp$，l 为棱长，p 为直截面周长
3. 正棱锥全面积 $T = S + 2A$

棱台

1. 体积 $V = (A1 + A2 + \sqrt{A1A2})h/3$, $A1, A2$ 为上下底面积，h 为高
2. 正棱台侧面积 $S = (p1 + p2)l/2$，$p1, p2$ 为上下底面周长，l 为斜高
3. 正棱台全面积 $T = S + A1 + A2$

圆柱

1. 侧面积 $S = 2\pi rh$
2. 全面积 $T = 2\pi r(h + r)$
3. 体积 $V = \pi r^2h$

圆锥

1. 母线 $l = \sqrt{h^2 + r^2}$
2. 侧面积 $S = \pi rl$
3. 全面积 $T = \pi r(l + r)$
4. 体积 $V = \pi r^2h/3$

圆台

1. 母线 $l = \sqrt{h^2 + (r1 - r2)^2}$
2. 侧面积 $S = \pi(r1 + r2)l$
3. 全面积 $T = \pi r1(l + r1) + \pi r2(l + r2)$
4. 体积 $V = \pi(r1^2 + r2^2 + r1r2)h/3$

球

1. 全面积 $T = 4\pi r^2$
2. 体积 $V = 4\pi r^3/3$

球台

1. 侧面积 $S = 2\pi rh$
2. 全面积 $T = \pi(2rh + r1^2 + r2^2)$
3. 体积 $V = \pi h(3(r1^2 + r2^2) + h^2)/6$

球扇形

1. 全面积 $T = \pi r(2h + r0)$，h 为球冠高，$r0$ 为球冠底面半径
2. 体积 $V = 2\pi r^2 h/3$

16 长方体表面两点最短距离

```
//返回最短距离的平方
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), swap(y2, z2), swap(W, H);
    else
      swap(x1, z1), swap(x2, z2), swap(L, H);
  }
  if (z1 == H) z1 = 0, z2 = H - z2;
  r = 0x3fffffff;
  turn(0, 0, x2 - x1, y2 - y1, z2, -x1, -y1, L, W, H);
  cout << r << endl;
  return 0;
}
```

17 扩展欧几里得

```
inline void ex_gcd(int a, int b, long long &X1, long long &Y1){
     if (!b)
     {
          X1 = 1; Y1 = 0;
          return;
     }
     long long X2, Y2;
     ex_gcd(b, a % b, X2, Y2);
     X1 = Y2; Y1 = X2 - a / b * Y2;
}
```

18 中国剩余定理

包括扩展欧几里得，求逆元，和保证除数互质条件下的 CRT

```
LL x, y;
void exGcd(LL a, LL b)
{
  if (b == 0) {
    x = 1;
    y = 0;
    return;
  }
  exGcd(b, a % b);
  LL k = y;
  y = x - a / b * y;
  x = k;
}
LL inversion(LL a, LL b)
{
  exGcd(a, b);
  return (x % b + b) % b;
}
LL CRT(vector<LL> m, vector<LL> a)
{
  int N = m.size();
  LL M = 1, ret = 0;
  for(int i = 0; i < N; ++ i)
    M *= m[i];

  for(int i = 0; i < N; ++ i) {
    ret = (ret + (M / m[i]) * a[i] % M * inversion(M / m[i], m[i])) % M;
  }
  return ret;
}
```

19 Pollard_rho

```
const int tmax = 50, pmax = 7, P[pmax + 8] = {0, 2, 3, 7, 61, 24251, 17, 23};
ll n, ans[tmax + 18], k, pt, prime[tmax + 18];
int tot, p[tmax + 18], pow[tmax + 18];
ll mul(unsigned long long a, ll b, ll n)
{
  unsigned long long ans = 0;
  for (; b; b >>= 1, a = (a << 1) % n)
    if (b & 1) ans = (ans + a) % n;
  return ans;
}
ll fpm(ll a, ll b, ll n)
{
  ll ans = 1;
  for (; b; b >>= 1, a = mul(a, a, n))
    if (b & 1) ans = mul(ans, a, n);
  return ans;
}
bool witness(ll a, ll b, int c, ll n)
{
  a = fpm(a, b, n);
  for (b = a; c--; b = a) {
    a = mul(b, b, n);
    if (a == 1 && b != 1 && b != n - 1)
      return 0;
  }
  return a == 1;
}
bool Miller_Rabin(ll a)
{
  ll N = a;
  if (a == 2) return 1;
  if (a == 1 || !(a & 1)) return 0;
  int k = 0;
  for (--a; !(a & 1); ++k) a >>= 1;
  for (int i = 1; i <= pmax; ++i)
    if (N != P[i] && !witness(P[i], a, k, N))
      return 0;
  return 1;
}
ll rd()
{
  static int x = 1, y = 1;
  x += (x << 2) | 1, x &= 0x7fffffff;
  y += (y << 2) | 1, y &= 0x7fffffff;
  return (((ll)x) << 32) | y;
}
ll gcd(ll a, ll b)
{
  return b ? gcd(b, a % b) : a;
}
void Pollard_Rho(ll n)
{
  if (n == 1) return;
  if (Miller_Rabin(n)) {
    prime[++tot] = n, p[tot] = tot;
    return;
  }
  for (ll c = n - 1, x = rd() % n, y = x, k = 2, i = 1, d; 1; ++i) {
    x = (mul(x, x, n) + c) % n;
    d = gcd((y - x + n) % n, n);
    if (d != 1 && d != n) {
      Pollard_Rho(d);
      Pollard_Rho(n / d);
      return;
    }
    else if (d == n)
        y = x = rd() % n, k = 2, i = 0;
    if (i == k) k <<= 1, y = x;
  }
}
bool cmpor(int a, int b)
{
  return prime[a] < prime[b];
}
int main()
{
  while (scanf(ioll, &n), n) {
    tot = 0;
    Pollard_Rho(n);
    sort(p + 1, p + tot + 1, cmpor);
    k = prime[p[1]], pow[pt = 1] = 1, ans[1] = k;
    for (int i = 2; i <= tot; ++i)
      prime[p[i]] == k ? ++pow[pt] : (k = prime[p[i]], pow[++pt] = 1, ans[pt] = k);
    for (int i = 1; i < pt; ++i) printf(ioll "^%d␣", ans[i], pow[i]);
    printf(ioll "^%d\n", ans[pt], pow[pt]);
  }
  return 0;
}
```

20 FFT

```
const int nmax = 50000, lmax = 1 << 15, mo = 10000, wmax = 4, pe[4] = {1, 10, 100,1000};
#define Pi M_PI
struct complex
{
    double re, ur;
}A[lmax + 18], B[lmax + 18], wtmp, atmp[lmax + 18], ul[lmax + 18];
int l = 1, n;
char str[nmax + 18];
int ans[lmax + 18];
complex operator+(complex a, complex b)
{
    a.re += b.re;
    a.ur += b.ur;
    return a;
}
complex operator-(complex a, complex b)
{
    a.re -= b.re;
    a.ur -= b.ur;
    return a;
}
complex operator*(complex a, complex b)
{
    complex c;
    c.re = a.re * b.re - a.ur * b.ur;
    c.ur = a.re * b.ur + a.ur * b.re;
    return c;
}
void fft(complex *a, int n, int step)
{
    if (n == 1)
    return;
    int bs = step << 1;
    fft(a, n >> 1, bs);
    fft(a + step, n >> 1, bs);
    for (int i = 0, tmp = 0; tmp < (n >> 1); i += bs, ++tmp)
    {
    wtmp = ul[i >> 1] * a[i + step];
    atmp[tmp] = a[i] + wtmp;
    atmp[tmp + (n >> 1)] = a[i] - wtmp;
    }
    for (int i = 0, tmp = 0; tmp < n; i += step, ++tmp)
    a[i] = atmp[tmp];
}
bool get(complex *a)
{
    if (scanf("%s\n", str + 1) == EOF) return false;
    int len = strlen(str + 1), wei = 0, tmp = 0, bi = -1;
    for (int i = len; i; --i)
    {
    tmp = tmp + (str[i] - '0') * pe[wei++];
    if (wei == wmax)
        a[++bi].re = tmp, tmp = wei = 0;
    }
    if (tmp) a[++bi].re = tmp;
    if (bi > n) n = bi;
    return true;
}
int main()
{
    get(A);
    get(B);
    while (l < n) l <<= 1;
    l <<= 1;
    for (int i = 0; i < l; ++i)
    {
        ul[i].re = cos(2 * Pi * i / l);
        ul[i].ur = sin(2 * Pi * i / l);
    }
    fft(A, l, 1);
    fft(B, l, 1);
    for (int i = 0; i < l; ++i)
        A[i] = A[i] * B[i], ul[i].ur = -ul[i].ur;
    fft(A, l, 1);
    long long x = 0;
    for (int i = 0; i < l; ++i)
    {
        x += (long long) (A[i].re / l + 0.1);
        ans[i] = x % mo;
        x /= mo;
    }
    while (l && !ans[l]) --l;
    printf("%d", ans[l--]);
    for (int i = l; i >= 0; --i)
        printf("%04d", ans[i]);
    return 0;
}
```

字符串

21 最小表示

```
struct cyc_string
{
  int n, offset;
  char str[max_length];
  char & operator [] (int x)
  {return str[((offset + x) % n)];}
  cyc_string(){offset = 0;}
};
void minimum_circular_representation(cyc_string & a)
{
  int i = 0, j = 1, dlt = 0, n = a.n;
  while(i < n and j < n and dlt < n)
  {
    if(a[i + dlt] == a[j + dlt]) dlt++;
    else
    {
      if(a[i + dlt] > a[j + dlt]) i += dlt + 1; else j += dlt + 1;
      dlt = 0;
    }
  }
  a.offset = min(i, j);
}
```

22 扩展 kmp

```
while(1+j<strlen(T)&&T[0+j]==T[1+j])
  j = j + 1;
A[1]=j;
int k=1;
for(int i=2; i<strlen(T); i++)  {
  int Len = k + A[k] - 1,L = A[i-k];
  if( L < Len - i + 1 )
    A[i] = L;
  else {
    j = max(0,Len -i +1);
    while(i+j<strlen(T)&&T[i+j] == T[0+j])
      j = j + 1;
    A[i] = j,k = i;
  }
}
j = 0;
while(j<strlen(S)&&j<strlen(T)&&T[0+j]==S[0+j])
  j = j + 1;
B[0] = j,k = 0;
for(int i=1; i<strlen(S); i++) {
  int Len = k + B[k] - 1,L = A[i-k];
  if( L < Len - i + 1 )
    B[i] = L;
  else {
    j = max(0,Len -i +1);
    while(i+j<strlen(S)&&j<strlen(T)&&S[i+j] == T[0+j])
      j = j + 1;
    B[i] = j,k = i;
  }
}
```

23 manacher

```
void palindrome(char cs[], int len[], int n) {
  for (int i = 0; i < n * 2; ++i) {
    len[i] = 0;
  }
  for (int i = 0, j = 0, k; i < n * 2; i += k, j = max(j - k, 0)) {
    while (i - j >= 0 && i + j + 1 < n * 2 && cs[(i - j) / 2] == cs[(i + j + 1) / 2])
      j++;
    len[i] = j;
    for (k = 1; i - k >= 0 && j - k >= 0 && len[i - k] != j - k; k++) {
      len[i + k] = min(len[i - k], j - k);
    }
  }
}
```

24 AC 自动机

```
const int P = 1000000007;
int ans, v[201][201][501][2], n, m, request, len1, c1[201], len2, c2[201], a[201][21], cnt, len, c[201],
     f[201], type[201], fail[201], father[201];
inline void work(int xs){
    memset(v, 0, sizeof(v));
    v[0][0][0][1] = 1;
    for (int i = 1; i <= len1; i++)
    {
        if (i != 1)
        for (int l = 1; l < m; l++)
        {
            int p = 0;
            while (!a[p][l] && p) p = fail[p];
            p = a[p][l];
            if (f[p] <= request)
                v[i][p][f[p]][0]++,
                v[i][p][f[p]][0] %= P;
        }
        for (int j = 0; j <= cnt; j++)
            for (int k = 0; k <= request; k++)
            {
                if (v[i - 1][j][k][0])
                    for (int l = 0; l < m; l++)
                    {
                        int p = j;
                        while (!a[p][l] && p) p = fail[p];
                        p = a[p][l];
                        //printf("xx %d %d %d %d\n", i, p, k, f[p]);
                        if (k + f[p] <= request)
                            v[i][p][k + f[p]][0] += v[i - 1][j][k][0],
                            v[i][p][k + f[p]][0] %= P;
                    }
                if (v[i - 1][j][k][1])
                {
                    int uu = 0;
                    if (i == 1) uu = 1;
                    for (int l = uu; l <= c1[i]; l++)
                    {
                        int p = j;
                        while (!a[p][l] && p) p = fail[p];
                        p = a[p][l];
                        if (k + f[p] <= request)
                            if (l != c1[i])
                            v[i][p][k + f[p]][0] += v[i - 1][j][k][1],
                            v[i][p][k + f[p]][0] %= P;
                            else
                            v[i][p][k + f[p]][1] += v[i - 1][j][k][1],
                            v[i][p][k + f[p]][1] %= P;
                    }
                }
            }
    }
    for (int j = 0; j <= cnt; j++)
        for (int k = 0; k <= request; k++)
            if (xs == 1)
            {
                ans += v[len1][j][k][0]; ans %= P;
                ans += v[len1][j][k][1]; ans %= P;
            }
            else ans -= v[len1][j][k][0], ans += P, ans %= P;
}
inline void makefail(){
    memset(fail, 255, sizeof(fail));
    fail[0] = 0;
    int k = 0;
    for (int i = 0; i < m; ++i)
        if (a[0][i]) fail[a[0][i]] = 0, c[++k] = a[0][i];
    int M = m;
    for (int l = 1; l <= k; l++)
    {
        int m = c[l];
        if (fail[m] == -1)
        {
            int p = father[m];
            while (p && !a[fail[p]][type[m]]) p = fail[p];
            fail[m] = a[fail[p]][type[m]];
            f[m] += f[fail[m]];
        }
        for (int i = 0; i < M; ++i)
            if (a[m][i]) c[++k] = a[m][i];
    }
}
int main(){
    scanf("%d%d%d", &n, &m, &request);
    scanf("%d", &len1);
    for (int i = 1; i <= len1; i++) scanf("%d", &c1[i]);
    scanf("%d", &len2);
    for (int i = 1; i <= len2; i++) scanf("%d", &c2[i]);
    memset(a, 0, sizeof(a)); cnt = 0;
    memset(f, 0, sizeof(f));
```

```
      for (int i = 1; i <= n; i++)
      {
            scanf("%d", &len);
            for (int j = 1; j <= len; j++) scanf("%d", &c[j]);
            int now = 0;
            for (int j = 1; j <= len; j++)
            {
                  if (!a[now][c[j]]) a[now][c[j]] = ++cnt, type[cnt] = c[j], father[cnt] = now;
                  now = a[now][c[j]];
            }
            int value;
            scanf("%d", &value);
            f[now] += value;
      }
      makefail();
      ans = 0;
      work(-1);
      len1 = len2;
      for (int i = 1; i <= len1; i++) c1[i] = c2[i];
      work(1);
      printf("%d\n", ans);
}
```

25 后缀数组

25.1 shy

```
int test,n,SA[100001],c[100001],Rank[100001],tmp[100001],H[100001],f[100001];
char can[50001];

int main(){
    scanf("%d",&test);
    for (test;test;test--)
    {
             scanf("%s\n",&can);
             n=strlen(can);
             memset(f,0,sizeof(f));
             for (int i=1;i<=n;i++) f[i]=int (can[i-1]);
             memset(c,0,sizeof(c));
             for (int i=1;i<=n;i++) c[f[i]]++;
             for (int i=1;i<=1000;i++) c[i]+=c[i-1];
             for (int i=n;i;i--) SA[c[f[i]]--]=i;
             Rank[SA[1]]=1;
             for (int i=2;i<=n;i++)
                if (f[SA[i]]==f[SA[i-1]]) Rank[SA[i]]=Rank[SA[i-1]];
                else Rank[SA[i]]=Rank[SA[i-1]]+1;
             for (int L=1;L<=n;L+=L)
             {
                 if (Rank[SA[n]]==n) break;
                 memset(c,0,sizeof(c));
                 for (int i=1;i<=n;i++) c[Rank[L+i]]++;
                 for (int i=1;i<=n;i++) c[i]+=c[i-1];
                 for (int i=n;i;i--) tmp[c[Rank[L+i]]--]=i;
                 memset(c,0,sizeof(c));
                 for (int i=1;i<=n;i++) c[Rank[i]]++;
                 for (int i=1;i<=n;i++) c[i]+=c[i-1];
                 for (int i=n;i;i--) SA[c[Rank[tmp[i]]]--]=tmp[i];
                 tmp[SA[1]]=1;
                 for (int i=2;i<=n;i++)
                    if ((Rank[SA[i]]==Rank[SA[i-1]])&&(Rank[SA[i]+L]==Rank[SA[i-1]+L]))
                        tmp[SA[i]]=tmp[SA[i-1]];
                    else tmp[SA[i]]=tmp[SA[i-1]]+1;
                 for (int i=1;i<=n;i++) Rank[i]=tmp[i];
             }
    int p=0;
    for (int i=1;i<=n;i++)
    {
        int j=SA[Rank[i]-1];
        p-=1;
        if (p<0) p=0;
        while ((f[i+p]==f[j+p])) p++;
        H[i]=p;
    }
    int ans=0;
    for (int i=1;i<=n;i++)
        ans+=n-SA[i]+1-H[i];
    printf("%d\n",ans);
    }
}
```

25.2 sxy

```
int wa[N], wb[N], wv[N], ws[N], rank[N], height[N];
int cmp(int *r, int a, int b, int l){
  return r[a] == r[b] && r[a + l] == r[b + l];
}
void da(int *r, int *sa, int n, int m){
  int i, j, p, *x = wa, *y = wb, *t;
  for (i = 0; i < m; i++) ws[i] = 0;
  for (i = 0; i < n; i++) ws[x[i] = r[i]]++;
  for (i = 1; i < m; i++) ws[i] += ws[i - 1];
  for (i = n - 1; i >=0 ; i--) sa[--ws[x[i]]] = i;
  for (j = 1, p = 1;p < n; j *= 2, m = p){
    for (p = 0, i = n - j; i < n; i++) y[p++] = i;
    for (i = 0; i < n; i++) if (sa[i] >= j) y[p++] = sa[i] - j;
    for (i = 0; i < n; i++) wv[i] = x[y[i]];
    for (i = 0; i < m; i++) ws[i] = 0;
    for (i = 0; i < n; i++) ws[wv[i]]++;
    for (i = 1; i < m; i++) ws[i] += ws[i - 1];
    for (i = n - 1; i >= 0; i--) sa[--ws[wv[i]]] = y[i];
    for (t = x, x = y, y = t, p = 1, x[sa[0]] = 0,i = 1; i < n; i++) x[sa[i]] = cmp(y, sa[i - 1], sa[i],
          j) ? p - 1 : p++;
  }
}
void calheight(int *r, int *sa, int n){
  int i, j, k = 0;
  for (i = 0; i < n; i++) rank[sa[i]] = i;
  for (i = 0; i < n; h[rank[i++]] = k)
    for (k ? k-- : 0, j = sa[rank[i] - 1]; r[i + k] == r[j + k]; k++);
}
```

图论

26 桥、边双连通分量

```
const int N = 10010,M = 100000;
int n, m, x, y, ind, ind2, size, ans, tot, last[N], nt[M], pt[M], dfn[N], low[N], bh[N], p[N], nw[N], dg[
      N];
vector<int> a[N];
bool vis[N], cut[N];
void edge(int x, int y, int z){
  pt[last[x] = nt[last[x]] = ++ind] = y;
  bh[ind] = z;
}
void tarjan(int x){
  dfn[x] = low[x] = ++ind2;
  for (int i = nt[x]; pt[i]; i = nt[i])
    if (!dfn[pt[i]]){
      p[pt[i]] = bh[i];
      tarjan(pt[i]);
      low[x] = min(low[x], low[pt[i]]);
    }
    else if (bh[i] != p[x]) low[x] = min(low[x], dfn[pt[i]]);
  if (p[x] && low[x] == dfn[x]) cut[p[x]] = 1;
}
void tr(int x){
  vis[x] = 1;
  nw[x] = tot;
  for (int i = nt[x]; pt[i]; i = nt[i]) if (cut[bh[i]] == 0 && !vis[pt[i]]) tr(pt[i]);
}
int main(){
  scanf("%d%d",&n,&m);
  ind = n;
  for (int i = 1; i <= ind; ++i) last[i] = i;
  for (int i = 1; i <= m; ++i){
    scanf("%d%d", &x, &y);
    edge(x, y, i), edge(y, x, i);
  }
  for (int i = 1; i <= n; ++i) if (!dfn[i]) size = 0, tarjan(i);
  for (int i = 1; i <= n; ++i) if (!vis[i]) ++tot, tr(i);
  for (int i = 1; i <= n; ++i)
    for (int j = nt[i]; pt[j]; j = nt[j]) if (nw[i] != nw[pt[j]]) a[nw[i]].push_back(nw[pt[j]]);
  for (int i = 1; i <= tot; ++i){
    std::sort(a[i].begin(), a[i].end());
    if (a[i].size()) ++dg[i];
    for (int j = 1; j < (int)a[i].size(); ++j) if (a[i][j] != a[i][j - 1]) ++dg[i];
    if (dg[i] == 1) ++ans;
  }
  printf("%d\n", (ans + 1) / 2);
}
```

27 割点

```cpp
const int N = 10010,M = 100000;
int n, m, root, x, y, num, ind, ind2, size, ans, last[N], nt[M], pt[M], dfn[N], low[N], v[N];
void edge(int x, int y){
  pt[last[x] = nt[last[x]] = ++ind] = y;
}
void tarjan(int x){
  dfn[x] = low[x] = ++ind2;
  v[x] = 1;
  for (int i = nt[x];pt[i];i = nt[i])
    if (!dfn[pt[i]]){
      tarjan(pt[i]);
      low[x] = min(low[x], low[pt[i]]);
      if (dfn[x] <= low[pt[i]]) ++v[x];
    }
    else low[x] = min(low[x], dfn[pt[i]]);
}
int main(){
  for (; ; ){
    scanf("%d%d", &n, &m);
    if (n == 0 && m == 0) return 0;
    for (int i = 1; i <= ind; ++i) nt[i] = pt[i] = 0;
    ind = n;
    for (int i = 1; i <= ind; ++i) last[i] = i;
    for (int i = 1; i <= m; ++i){
      scanf("%d%d",&x,&y);
      ++x, ++y;
      edge(x, y), edge(y, x);
    }
    memset(dfn, 0, sizeof(dfn));
    memset(v, 0, sizeof(v));
    ans = num = ind2 = 0;
    for (int i = 1; i <= n; ++i)if (!dfn[i]){
      root = i;
      size = 0;
      ++num;
      tarjan(i);
      --v[root];
    }
    for (int i = 1; i <= n; ++i) if (v[i] + num - 1 > ans) ans = v[i] + num - 1;
    printf("%d\n", ans);
  }
}
```

28 点双连通分量

```cpp
#include <cstdio>
#include <vector>
#include <algorithm>
using namespace std;
const int N = 10010, M = 300000;
int n, m, x, y, ans1, ans2, tot1, tot2, flag, size, ind2, dfn[N], low[N], block[M], vis[N];
vector<int> a[N];
pair<int, int> stack[M];
void tarjan(int x, int p){
  dfn[x] = low[x] = ++ind2;
  for (int i = 0; i < a[x].size(); ++i) if (dfn[x] > dfn[a[x][i]] && a[x][i] != p){
    stack[++size] = make_pair(x, a[x][i]);
    if (i == a[x].size() - 1 || a[x][i] != a[x][i + 1])
    if (!dfn[a[x][i]]){
      tarjan(a[x][i], x);
      low[x] = min(low[x], low[a[x][i]]);
      if (low[a[x][i]] >= dfn[x]){
        tot1 = tot2 = 0;
        ++flag;
        for (; ; ){
          if (block[stack[size].first] != flag) ++tot1, block[stack[size].first] = flag;
          if (block[stack[size].second] != flag) ++tot1, block[stack[size].second] = flag;
          if (stack[size].first == x && stack[size].second == a[x][i]) break;
          ++tot2;
          --size;
        }
        for (; stack[size].first == x && stack[size].second == a[x][i]; --size) ++tot2;
        if (tot2 < tot1) ans1 += tot2;
        if (tot2 > tot1) ans2 += tot2;
      }
    }
    else low[x] = min(low[x], dfn[a[x][i]]);
  }
}
int main(){
  for (; ; ){
    scanf("%d%d", &n, &m);
    if (n == 0 && m == 0) return 0;
    for (int i = 1; i <= n; ++i) a[i].clear(), dfn[i] = 0;
    for (int i = 1; i <= m; ++i){
      scanf("%d%d", &x, &y);
      ++x, ++y;
      a[x].push_back(y);
      a[y].push_back(x);
    }
    for (int i = 1; i <= n; ++i) sort(a[i].begin(), a[i].end());
    ans1 = ans2 = ind2 = 0;
    for (int i = 1; i <= n; ++i) if (!dfn[i]) size = 0, tarjan(i, 0);
    printf("%d␣%d\n", ans1, ans2);
  }
}
```

29 强联通分量 + 手写栈

```cpp
int n, m, first[10001], father[10001], dfn[10001], low[10001], c[10001], pos[10001], todo[10001],
cnt, len, next[2000001], where[2000001], l, kuai, Max, color[10001], number;
bool b[10001];
int read(){
    char ch;
    for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
    int cnt = 0;
    for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
    return(cnt);
}
inline void makelist(int x, int y){
    where[++l] = y;
    next[l] = first[x];
    first[x] = l;
}
inline void tarjan(int S){
    int now = S; todo[now] = first[now];
    for (;;)
    {
        if (!now) return;
        if (first[now] == todo[now])
        {
            b[now] = true;
            dfn[now] = low[now] = ++cnt;
            c[++len] = now; pos[now] = len;
        }
        int x = todo[now];
        if (!x)
        {
            if (father[now])
                low[father[now]] = min(low[father[now]], low[now]);
            int delta = -1;
            if (father[now]) ++delta;
            for (int x = first[now]; x; x = next[x])
                if (father[where[x]] == now)
                    if (low[where[x]] >= dfn[now]) ++delta;
            Max = max(Max, delta);
            if (low[now] == dfn[now])
            {
               ++number;
               for (int i = pos[now]; i <= len; i++) color[c[i]] = number;
               len = pos[now] - 1;
            }
            now = father[now];
            continue;
        }
        todo[now] = next[todo[now]];
        if (father[now] != where[x])
            if (!b[where[x]])
            {
                father[where[x]] = now;
                now = where[x];
                todo[now] = first[now];
                continue;
            }
            else if (!color[where[x]]) low[now] = min(low[now], dfn[where[x]]);
    }
}
int main(){
    for (;;)
    {
        n = read(); m = read();
        if (!n && !m) return 0;
        memset(first, 0, sizeof(first));
        l = 0;
        for (int i = 1; i <= m; i++)
        {
            int x = read() + 1, y = read() + 1;
            makelist(x, y);
            makelist(y, x);
        }
        memset(dfn, 0, sizeof(dfn));
        memset(low, 0, sizeof(low));
        memset(color, 0, sizeof(color));
        memset(b, false, sizeof(b));
        memset(father, 0, sizeof(father));
        cnt = 0; len = 0;
        Max = - (1 << 30);
        kuai = 0; number = 0;
        for (int i = 1; i <= n; i++)
            if (!b[i]) tarjan(i), ++kuai;
        printf("%d\n", kuai + Max);
    }
}
```

30 最小树形图

```
namespace EdmondsAlgorithm { // O(ElogE + V^2) !!! 0-based !!!
  struct enode { int from, c, key, delta, dep; enode *ch[2], *next;
  } ebase[maxm], *etop, *fir[maxn], nil, *null, *inEdge[maxn], *chs[maxn];
  typedef enode *edge; typedef enode *tree;
  int n, m, setFa[maxn], deg[maxn], que[maxn];
  inline void pushDown(tree x) { if (x->delta) {
    x->ch[0]->key += x->delta; x->ch[0]->delta += x->delta;
    x->ch[1]->key += x->delta; x->ch[1]->delta += x->delta; x->delta = 0;
  }}
  tree merge(tree x, tree y) {
    if (x == null) return y; if (y == null) return x;
    if (x->key > y->key) swap(x, y); pushDown(x); x->ch[1] = merge(x->ch[1], y);
    if (x->ch[0]->dep < x->ch[1]->dep) swap(x->ch[0], x->ch[1]);
    x->dep = x->ch[1]->dep + 1; return x;
  }
  void addEdge(int u, int v, int w) {
    etop->from = u; etop->c = etop->key = w; etop->delta = etop->dep = 0;
    etop->next = fir[v]; etop->ch[0] = etop->ch[1] = null;
    fir[v] = etop; inEdge[v] = merge(inEdge[v], etop++);
  }
  void deleteMin(tree &r) { pushDown(r); r = merge(r->ch[0], r->ch[1]); }
  int findSet(int x) { return setFa[x] == x ? x : setFa[x] = findSet(setFa[x]); }
  void clear(int V, int E) {
    null = &nil; null->ch[0] = null->ch[1] = null; null->dep = -1;
    n = V; m = E; etop = ebase; Foru(i, 0, V) fir[i] = NULL; Foru(i, 0, V) inEdge[i] = null;
  }
  int solve(int root) { int res = 0, head, tail;
    for (int i = 0; i < n; ++i) setFa[i] = i;
    for ( ; ; ) { memset(deg, 0, sizeof(int) * n); chs[root] = inEdge[root];
      for (int i = 0; i < n; ++i) if (i != root && setFa[i] == i) {
        while (findSet(inEdge[i]->from) == findSet(i)) deleteMin(inEdge[i]);
        ++deg[ findSet((chs[i] = inEdge[i])->from) ];
      }
      for (int i = head = tail = 0; i < n; ++i)
        if (i != root && setFa[i] == i && deg[i] == 0) que[tail++] = i;
      while (head < tail) {
        int x = findSet(chs[que[head++]]->from);
        if (--deg[x] == 0) que[tail++] = x;
      } bool found = false;
      for (int i = 0; i < n; ++i) if (i != root && setFa[i] == i && deg[i] > 0) {
        int j = i; tree temp = null; found = true;
        do {setFa[j = findSet(chs[j]->from)] = i;
          deleteMin(inEdge[j]); res += chs[j]->key;
          inEdge[j]->key -= chs[j]->key; inEdge[j]->delta -= chs[j]->key;
          temp = merge(temp, inEdge[j]);
        } while (j != i); inEdge[i] = temp;
      } if (!found) break;
    } for (int i = 0; i < n; ++ i) if (i != root && setFa[i] == i) res += chs[i]->key;
    return res;
  }
}
namespace ChuLiu { // O(V ^ 3) !!! 1-based !!!
  int n, used[maxn], pass[maxn], eg[maxn], more, que[maxn], g[maxn][maxn];
  void combine(int id, int &sum) { int tot = 0, from, i, j, k;
    for ( ; id != 0 && !pass[id]; id = eg[id]) que[tot++] = id, pass[id] = 1;
    for (from = 0; from < tot && que[from] != id; from++);
    if (from == tot) return; more = 1;
    for (i = from; i < tot; i++) {
      sum += g[eg[que[i]]][que[i]]; if (i == from) continue;
      for (j = used[que[i]] = 1; j <= n; j++) if (!used[j])
        if (g[que[i]][j] < g[id][j]) g[id][j] = g[que[i]][j];
    }
    for (i = 1; i <= n; i++) if (!used[i] && i != id)
      for (j = from; j < tot; j++) {
        k = que[j]; if (g[i][id] > g[i][k] - g[eg[k]][k])
        g[i][id] = g[i][k] - g[eg[k]][k];
      }
  }
  void clear(int V) { n = V; Rep(i, 1, V) Rep(j, 1, V) g[i][j] = inf; }
  int solve(int root) {
    int i, j, k, sum = 0; memset(used, 0, sizeof(int) * (n + 1));
    for (more = 1; more; ) {
      more = 0; memset(eg, 0, sizeof(int) * (n + 1));
      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(int) * (n + 1));
      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;
  }
}
```

31 一般图最大匹配

```
const int nmax = 400 + 18;
int n, Next[nmax], f[nmax], mark[nmax], visited[nmax], Link[nmax], Q[nmax], head, tail;
vector<int> E[nmax];
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 solve()
{
  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;
  return ans;
}
```

32 km

```
const int oo = 1 << 30;
int ans, value[501][501], n, m, L[501], R[501], v[501];
bool bx[501], by[501];
bool find(int now){
    bx[now] = true;
    for (int i = 1; i <= m; i++)
        if (!by[i] && L[now] + R[i] == value[now][i])
        {
           by[i] = true;
           if (!v[i] || find(v[i]))
           {
              v[i] = now;
              return(true);
           }
        }
    return(false);
}
inline void km(){
    memset(L, 0, sizeof(L));
    memset(R, 0, sizeof(R));
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            L[i] = max(L[i], value[i][j]);
    ans = 0;
    memset(v, 0, sizeof(v));
    for (int i = 1; i <= min(n, m); i++)
        for (;;)
        {
            memset(bx, false, sizeof(bx));
            memset(by, false, sizeof(by));
            if (find(i)) break;
            int Min = oo;
            for (int j = 1; j <= n; j++)
                if (bx[j])
                   for (int k = 1; k <= m; k++)
                       if (!by[k])
                          Min = min(Min, L[j] + R[k] - value[j][k]);
            for (int j = 1; j <= n; j++) if (bx[j]) L[j] -= Min;
            for (int j = 1; j <= m; j++) if (by[j]) R[j] += Min;
        }
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            if (v[j] == i) ans += value[i][j];
    printf("%d\n", abs(ans));
}
int main(){
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++) scanf("%d", &value[i][j]), value[i][j] = -value[i][j];
    km();
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            value[i][j] = -value[i][j];
    km();
}
/*hint 500 * 500 1.5s
can solve problems whose n != m
must be complete graph, or should change some values of matrix to satisfy the condition*/
```

33 弦图判定

```
//弦(chord): 连接环中不相邻的两个点的边。
//弦图(chordal graph): 一个无向图称为弦图当图中任意长度大于3的环都至少有一个弦。
int n, m, first[1001], l, next[2000001], where[2000001],f[1001], a[1001], c[1001], L[1001], R[1001],
v[1001], idx[1001], pos[1001];
bool b[1001][1001];
int read(){
    char ch;
    for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
    int cnt = 0;
    for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
    return(cnt);
}
inline void makelist(int x, int y){
    where[++l] = y;
    next[l] = first[x];
    first[x] = l;
}
bool cmp(const int &x, const int &y){
    return(idx[x] < idx[y]);
}
int main(){
    for (;;)
    {
        n = read(); m = read();
        if (!n && !m) return 0;
        memset(first, 0, sizeof(first)); l = 0;
        memset(b, false, sizeof(b));
        for (int i = 1; i <= m; i++)
        {
            int x = read(), y = read();
            if (x != y && !b[x][y])
            {
               b[x][y] = true; b[y][x] = true;
               makelist(x, y); makelist(y, x);
            }
        }
        memset(f, 0, sizeof(f));
        memset(L, 0, sizeof(L));
        memset(R, 255, sizeof(R));
        L[0] = 1; R[0] = n;
        for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
        memset(idx, 0, sizeof(idx));
        memset(v, 0, sizeof(v));
        for (int i = n; i; --i)
        {
            int now = c[i];
            R[f[now]]--;
            if (R[f[now]] < L[f[now]]) R[f[now]] = -1;
            idx[now] = i; v[i] = now;
            for (int x = first[now]; x; x = next[x])
                if (!idx[where[x]])
                {
                   swap(c[pos[where[x]]], c[R[f[where[x]]]]);
                   pos[c[pos[where[x]]]] = pos[where[x]];
                   pos[where[x]] = R[f[where[x]]];
                   L[f[where[x]] + 1] = R[f[where[x]]]--;
                   if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
                   if (R[f[where[x]] + 1] == -1)
                       R[f[where[x]] + 1] = L[f[where[x]] + 1];
                   ++f[where[x]];
                }
        }
        bool ok = true;
        for (int i = 1; i <= n && ok; i++)
        {
            int cnt = 0;
            for (int x = first[v[i]]; x; x = next[x])
                if (idx[where[x]] > i) c[++cnt] = where[x];
            sort(c + 1, c + cnt + 1, cmp);
            bool can = true;
            for (int j = 2; j <= cnt; j++)
                if (!b[c[1]][c[j]])
                {
                    ok = false;
                    break;
                }
        }
        if (ok) printf("Perfect\n");
        else printf("Imperfect\n");
        printf("\n");
    }
}
```

34 弦图求团数

```
//团数：最大团大小
int n, m, first[100001], next[2000001], where[2000001], l, L[100001], R[100001], c[100001], f[100001],
pos[100001], idx[100001], v[100001], ans;
inline void makelist(int x, int y){
    where[++l] = y;
    next[l] = first[x];
    first[x] = l;
}
int read(){
    char ch;
    for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
    int cnt = 0;
    for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
    return(cnt);
}
int main(){
    memset(first, 0, sizeof(first)); l = 0;
    n = read(); m = read();
    for (int i = 1; i <= m; i++)
    {
        int x, y;
        x = read(); y = read();
        makelist(x, y); makelist(y, x);
    }
    memset(L, 0, sizeof(L));
    memset(R, 255, sizeof(R));
    memset(f, 0, sizeof(f));
    memset(idx, 0, sizeof(idx));
    for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
    L[0] = 1; R[0] = n; ans = 0;
    for (int i = n; i; --i)
    {
        int now = c[i], cnt = 1;
        idx[now] = i; v[i] = now;
        if (--R[f[now]] < L[f[now]]) R[f[now]] = -1;
        for (int x = first[now]; x; x = next[x])
            if (!idx[where[x]])
            {
                swap(c[pos[where[x]]], c[R[f[where[x]]]]);
                pos[c[pos[where[x]]]] = pos[where[x]];
                pos[where[x]] = R[f[where[x]]];
                L[f[where[x]] + 1] = R[f[where[x]]]--;
                if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
                if (R[f[where[x]] + 1] == -1) R[f[where[x]] + 1] = L[f[where[x]] + 1];
                ++f[where[x]];
            }
            else ++cnt;
        ans = max(ans, cnt);
    }
    printf("%d\n", ans);
}
```

35 最大团

```
namespace MaxClique { // 1-based
  int g[MAXN][MAXN], len[MAXN], list[MAXN][MAXN], mc[MAXN], ans, found;
  void DFS(int size) {
    if (len[size] == 0) { if (size > ans) ans = size, found = true; return; }
    for (int k = 0; k < len[size] && !found; ++k) {
      if (size + len[size] - k <= ans) break;
      int i = list[size][k]; if (size + mc[i] <= ans) break;
      for (int 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);
    }
  }
  int work(int n) {
    mc[n] = ans = 1; for (int i = n - 1; i; --i) { found = false; len[1] = 0;
      for (int j = i + 1; j <= n; ++j) if (g[i][j]) list[1][len[1]++] = j;
      DFS(1); mc[i] = ans;
    } return ans;
  }
}
```

36 最大团计数

```
namespace MaxCliqueCounting {
  int n, ans;
  int ne[MAXN], ce[MAXN];
  int g[MAXN][MAXN], list[MAXN][MAXN];
  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);
  }
}
```

37 有根树的同构

```
//http://acm.sdut.edu.cn/judgeonline/showproblem?problem_id=1861
const int mm=1051697,p=4773737;
int m,n,first[101],where[10001],next[10001],l,hash[10001],size[10001],pos[10001];
long long f[10001],rt[10001];
bool in[10001];
inline void makelist(int x,int y){
    where[++l]=y;
    next[l]=first[x];
    first[x]=l;
}
inline void hashwork(int now){
    int a[1001],v[1001],tot=0;
    size[now]=1;
    for (int x=first[now];x;x=next[x])
    {
        hashwork(where[x]);
        a[++tot]=f[where[x]];
        v[tot]=size[where[x]];
        size[now]+=size[where[x]];
    }
    a[++tot]=size[now];
    v[tot]=1;
    int len=0;
    for (int i=1;i<=tot;i++)
       for (int j=i+1;j<=tot;j++)
          if (a[j]<a[i])
          {
             int u=a[i];a[i]=a[j];a[j]=u;
             u=v[i];v[i]=v[j];v[j]=u;
          }
    f[now]=1;
    for (int i=1;i<=tot;i++)
       {
             f[now]=((f[now]*a[i])%p*rt[len])%p;
             len+=v[i];
       }
}
int main(){
    scanf("%d%d",&n,&m);
    rt[0]=1;
    for (int i=1;i<=100;i++)
        rt[i]=(rt[i-1]*mm)%p;
    for (int i=1;i<=n;i++)
    {
        memset(first,0,sizeof(first));
        memset(in,false,sizeof(in));
        l=0;
        for (int j=1;j<m;j++)
        {
            int x,y;
            scanf("%d%d",&x,&y);
            makelist(x,y);
            in[y]=true;
        }
        int root=0;
        for (int j=1;j<=m;j++)
        if (!in[j])
        {
            root=j;
            break;
        }
        memset(size,0,sizeof(size));
        memset(f,0,sizeof(f));
        hashwork(root);
        hash[i]=f[root];
    }
    for (int i=1;i<=n;i++) pos[i]=i;
    memset(in,false,sizeof(in));
    for (int i=1;i<=n;i++)
     if (!in[i])
     {
                printf("%d",i);
                for (int j=i+1;j<=n;j++)
                if (hash[j]==hash[i])
                {
                    in[j]=true;
                    printf("=%d",j);
                }
                printf("\n");
     }
}
```

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->S（下界 0，上界 oo），判断是否流量平衡
上下界最小流：不添 T->S 先流一遍，再添 T->S（下界 0，上界 oo）在残图上流一遍，答案为 S->T 的流量值
上下界最大流：添 T->S（下界 0，上界 oo）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->T 的值 + 残图中 S->T 的最大流

38 上下界最大流

```
int n, m, S, T, DS, DT, a[1001], first[1501], next[100001], where[100001], v[100001], what[100001],
l, c[1501], dist[1501], len, pre[1501], way[1501], flow[1501], out[100001], tot, cnt, ans;
inline void makelist(int x, int y, int z, int q){
    where[++l] = y;
    v[l] = z;
    what[l] = q;
    next[l] = first[x];
    first[x] = l;
}
int main(){
    for (;;)
    {
        if (scanf("%d%d", &n, &m) != 2) return 0;
        memset(first, 0, sizeof(first)); l = 1;
        memset(flow, 0, sizeof(flow));
        S = 0; T = n + m + 1; DS = T + 1; DT = DS + 1;
        for (int i = 1; i <= m; i++)
        {
            scanf("%d", &a[i]);
            flow[S] -= a[i]; flow[i] += a[i];
            makelist(S, i, 1 << 30, 0); makelist(i, S, 0, 0);
        }
        tot = 0;
        for (int i = 1; i <= n; i++)
        {
            int C, D;
            scanf("%d%d", &C, &D);
            if (D) makelist(m + i, T, D, 0), makelist(T, m + i, 0, 0);
            for (int j = 1; j <= C; j++)
            {
                int idx, x, y;
                scanf("%d%d%d", &idx, &x, &y);
                idx++;
                flow[idx] -= x; flow[i + m] += x;
                out[++tot] = x;
                if (y != x) makelist(idx, i + m, y - x, tot), makelist(i + m, idx, 0, tot);
            }
        }
        cnt = 0;
        for (int i = S; i <= T; i++)
            if (flow[i] > 0) makelist(DS, i, flow[i], 0), makelist(i, DS, 0, 0), cnt += flow[i];
            else makelist(i, DT, abs(flow[i]), 0), makelist(DT, i, 0, 0);
        makelist(T, S, 1 << 30, 0); makelist(S, T, 0, 0);
        ans = 0;
        for (; check(DS, DT);) dinic(DS, DS, DT);
        if (ans != cnt)
        {
            printf("-1\n\n");
            continue;
        }
        else
        {
            v[l] = v[l - 1] = 0;
            for (; check(S, T);) dinic(S, S, T);
            printf("%d\n", ans);
            for (int i = 3; i <= l; i += 2)
                if (what[i]) out[what[i]] += v[i];
            for (int i = 1; i <= tot; i++) printf("%d\n", out[i]);
            printf("\n");
        }
    }
}
```

39 上下界最小流

```
using namespace std;
struct {
        int x, y, down, up;
} a[10001];
int out[100001], what[100001], cnt, S, T, DS, DT, l, ans, n, m, flow[101], first[201], next[100001],
      where[100001], v[100001], dist[201], c[201], pre[201], way[201], len;
int read(){
    char ch;
    for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
    int cnt = 0;
    for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
    return(cnt);
}
inline void makelist(int x, int y, int z, int q){
    where[++l] = y;
    v[l] = z;
    what[l] = q;
    next[l] = first[x];
    first[x] = l;
}
inline void makemap(){
    memset(first, 0, sizeof(first)); l = 1;
    S = 1, T = n, DS = 0, DT = n + 1; cnt = 0;
    for (int i = 1; i <= n; i++)
        if (flow[i] > 0) makelist(DS, i, flow[i], 0), makelist(i, DS, 0, 0), cnt += flow[i];
        else makelist(i, DT, abs(flow[i]), 0), makelist(DT, i, 0, 0);
    for (int i = 1; i <= m; i++) makelist(a[i].x, a[i].y, a[i].up - a[i].down, i),
                                 makelist(a[i].y, a[i].x, 0, i);
}
inline void network(){
    for (; check();) dinic(DS);
}
int main(){
    scanf("%d%d", &n, &m);
    memset(flow, 0, sizeof(flow));
    for (int i = 1; i <= m; i++)
    {
        a[i].x = read(), a[i].y = read(), a[i].up = read();
        int status = read();
        if (status) a[i].down = a[i].up;
        else a[i].down = 0;
        flow[a[i].y] += a[i].down;
        flow[a[i].x] -= a[i].down;
    }
    makemap();
    ans = 0;
    network();
    makelist(T, S, 1 << 30, 0); makelist(S, T, 0, 0);
    network();
    if (ans != cnt)
    {
        printf("Impossible\n");
        return 0;
    }
    printf("%d\n", v[l]);
    for (int i = 3; i <= l; i += 2)
        if (what[i]) out[what[i]] = v[i];
    for (int i = 1; i <= m; i++)
    {
        printf("%d", a[i].down + out[i]);
        if (i != m) printf("␣");
        else printf("\n");
    }
}
```

40 上下界无源汇可行流

```
struct {
       int x, y, down, up, what;
} a[100001];
int S, T, DS, DT, n, m, out[100001], what[100001], first[501], pre[501], way[501], len, dist[501], c
      [501], ans, flow[201], where[100001], next[100001], v[100001], l, cnt;
inline void makelist(int x, int y, int z, int q){
    where[++l] = y;
    v[l] = z;
    what[l] = q;
    next[l] = first[x];
    first[x] = l;
}
int main(){
    scanf("%d%d", &n, &m);
    memset(flow, 0, sizeof(flow));
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d%d%d", &a[i].x, &a[i].y, &a[i].down, &a[i].up);
        flow[a[i].y] += a[i].down;
        flow[a[i].x] -= a[i].down;
    }
    cnt = 0;
    memset(first, 0, sizeof(first)); l = 1;
    S = 1; T = n; DS = 0; DT = n + 1; cnt = 0;
    for (int i = 1; i <= n; i++)
        if (flow[i] > 0) makelist(DS, i, flow[i], 0), makelist(i, DS, 0, 0), cnt += flow[i];
                    else makelist(i, DT, abs(flow[i]), 0), makelist(DT, i, 0, 0);
//   makelist(T, S, 1 << 30, 0); makelist(S, T, 0, 0);
    for (int i = 1; i <= m; i++) makelist(a[i].x, a[i].y, a[i].up - a[i].down, i),
                                 makelist(a[i].y, a[i].x, 0, i);
    ans = 0;
    for (; check();) dinic(DS);
    if (ans != cnt) printf("NO\n");
    else
    {
        printf("YES\n");
        for (int i = 3; i <= l; i += 2)
            if (what[i]) out[what[i]] = v[i];
        for (int i = 1; i <= m; i++) printf("%d\n", a[i].down + out[i]);
    }
}
```

41 上下界有源汇可行流

```
int test, n, m, Q, first[501], a1[201], a2[201], flow[501], next[100001], where[100001], v[100001], len,
l, dist[501], c[501], up[201][201], down[201][201], S, T, DS, DT, ans, out[201][201], pre[501], way[501];
inline void makelist(int x, int y, int z){
    where[++l] = y;
    v[l] = z;
    next[l] = first[x];
    first[x] = l;
}
int main(){
    scanf("%d", &test);
    for (int uu = 1; uu <= test; uu++)
    {
        scanf("%d%d", &n, &m);
        for (int i = 1; i <= n; i++) scanf("%d", &a1[i]);
        for (int i = 1; i <= m; i++) scanf("%d", &a2[i]);
        memset(up, 127, sizeof(up));
        memset(down, 0, sizeof(down));
        scanf("%d", &Q);
        for (int i = 1; i <= Q; i++)
        {
            int x, y, z;
            char str[2];
            scanf("%d%d%s%d", &x, &y, str, &z);
            int L1, L2, R1, R2;
            if (x == 0) L1 = 1, R1 = n;
            else L1 = R1 = x;
            if (y == 0) L2 = 1, R2 = m;
            else L2 = R2 = y;
            for (int j = L1; j <= R1; j++)
                for (int k = L2; k <= R2; k++)
                    if (str[0] == '>') down[j][k] = max(down[j][k], z + 1);
                    else if (str[0] == '<') up[j][k] = min(up[j][k], z - 1);
                    else down[j][k] = max(down[j][k], z), up[j][k] = min(up[j][k], z);
        }
        bool ok = true;
        for (int i = 1; i <= n && ok; i++)
            for (int j = 1; j <= m; j++)
                if (down[i][j] > up[i][j])
                {
                    ok = false;
                    break;
                }
        if (!ok)
        {
           printf("IMPOSSIBLE\n");
           if (uu != test) printf("\n");
           continue;
        }
        memset(flow, 0, sizeof(flow));
        memset(first, 0, sizeof(first)); l = 1;
        S = 0; T = n + m + 1;
        for (int i = 1; i <= n; i++) flow[S] -= a1[i], flow[i] += a1[i];
        for (int i = 1; i <= m; i++) flow[i + n] -= a2[i], flow[T] += a2[i];
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= m; j++)
            {
                flow[i] -= down[i][j]; flow[j + n] += down[i][j];
                if (down[i][j] != up[i][j]) makelist(i, j + n, up[i][j] - down[i][j]),
                                            makelist(j + n, i, 0);
            }
        DS = T + 1; DT = DS + 1;
        int cnt = 0;
        for (int i = S; i <= T; i++)
            if (flow[i] > 0) makelist(DS, i, flow[i]), makelist(i, DS, 0), cnt += flow[i];
            else if (flow[i] < 0) makelist(i, DT, abs(flow[i])), makelist(DT, i, 0);
        makelist(T, S, 1 << 30); makelist(S, T, 0);
        ans = 0;
        for (; check();) dinic(DS);
        if (ans != cnt)
        {
            printf("IMPOSSIBLE\n");
            if (uu != test) printf("\n");
            continue;
        }
        for (int i = 1; i <= n; i++)
            for (int x = first[i]; x; x = next[x])
                if (where[x] >= n + 1 && where[x] <= n + m)
                   down[i][where[x] - n] += v[x ^ 1];
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= m; j++)
            {
                printf("%d", down[i][j]);
                if (j != m) printf("␣");
                else printf("\n");
            }
         if (uu != test) printf("\n");
    }
}
```

42 zkw

```
int n, m, S, T, slk[1001], dist[1001], first[1001], l, c[1000001], next[1000001], where[1000001], ll[1000001], v[1000001];
bool b[1001];
long long ans1, ans2;
inline void makelist(int x, int y, int z, int p){
    where[++l] = y;
    ll[l] = z;
    v[l] = p;
    next[l] = first[x];
    first[x] = l;
}
inline void spfa(){
    memset(dist, 127, sizeof(dist));
    memset(b, false, sizeof(b));
    dist[T] = 0; c[1] = T;
    for (int k = 1, l = 1; l <= k; l++)
    {
        int m = c[l];
        b[m] = false;
        for (int x = first[m]; x; x = next[x])
            if (ll[x ^ 1] && dist[m] - v[x] < dist[where[x]])
            {
               dist[where[x]] = dist[m] - v[x];
               if (!b[where[x]]) b[where[x]] = true, c[++k] = where[x];
            }
    }
}
int zkw_work(int now, int cap){
    b[now] = true;
    if (now == T)
    {
        ans1 += cap;
        ans2 += (long long)cap * dist[S];
        return(cap);
    }
    int Left = cap;
    for (int x = first[now]; x; x = next[x])
        if (ll[x] && !b[where[x]])
           if (dist[now] == dist[where[x]] + v[x])
           {
               int use = zkw_work(where[x], min(Left, ll[x]));
               ll[x] -= use; ll[x ^ 1] += use;
               Left -= use;
               if (!Left) return(cap);
           }
           else slk[where[x]] = min(slk[where[x]], dist[where[x]] + v[x] - dist[now]);
    return(cap - Left);
}
bool relax(){
    int Min = 1 << 30;
    for (int i = 0; i <= T; i++)
        if (!b[i]) Min = min(Min, slk[i]);
    if (Min == 1 << 30) return(false);
    for (int i = 0; i <= T; i++)
        if (b[i]) dist[i] += Min;
    return(true);
}
inline void zkw(){
    ans1 = ans2 = 0;
    spfa();   //hint memset(dist, 0, sizeof(dist)); if all values of edges are nonnegative
    for (;;)
    {
        memset(slk, 127, sizeof(slk));
        for (;;)
        {
            memset(b, false, sizeof(b));
            if (!zkw_work(S, 1 << 30)) break;
        }
        if (!relax()) break;
    }
    printf("%I64d␣%I64d\n", ans1, ans2);
}
int main(){
    scanf("%d%d", &n, &m);
    S = 1; T = n;
    memset(first, 0, sizeof(first)); l = 1;
    for (int i = 1; i <= m; i++)
    {
        int x, y, z, q;
        scanf("%d%d%d%d", &x, &y, &z, &q);
        makelist(x, y, z, q); makelist(y, x, 0, -q);
    }
    zkw();
}
```

数据结构

43 随机可并堆

```
struct hnode
{
  hnode *l, *r;
  int k;
  hnode() {};
  hnode(int _k) : k(_k) {l = r = NULL};
};
int rd() {return x += (x << 2) | 1, x &= 0x3FFFFFF, x & 65536;}
hnode *merge(hnode *a, hnode *b)
{
    return (!a || !b) ? (a ? a : b) :
              (a->k < b->k ? merge(b, a) :
                       ((rd() ? a->l = merge(a->l, b) :
                            a->r = merge(a->r, b)), a));
}
```

44 KD-tree

```
const int MAX_N = 100000 + 10;
const int MAX_NODE = 200000 + 10;
const LL INF = 2000000000000000020LL;
int N;
struct Point
{
    int x, y, id;
};
LL dis(const Point &a, const Point &b)
{
    return 1LL * (a.x - b.x) * (a.x - b.x) + 1LL * (a.y - b.y) * (a.y - b.y);
}
struct Node
{
    Point p;
    int maxX, minX, maxY, minY;
    int l, r, d;
```

```cpp
    Node *ch[2];
};
LL ret;
LL ans[MAX_N];
Node *root;
Point p[MAX_N], queryPoint;
Node *totNode, nodePool[MAX_NODE];
int cmpx(const Point &a, const Point &b)
{
    return a.x < b.x;
}
int cmpy(const Point &a, const Point &b)
{
    return a.y < b.y;
}
Node* newNode(int l, int r, Point p, int deep)
{
    Node *t = totNode ++;
    t->l = l; t->r = r;
    t->p = p; t->d = deep;
    t->maxX = t->minX = p.x;
    t->maxY = t->minY = p.y;
    return t;
}
void updateInfo(Node *t, Node *p)
{
    t->maxX = max(t->maxX, p->maxX);
    t->maxY = max(t->maxY, p->maxY);
    t->minX = min(t->minX, p->minX);
    t->minY = min(t->minY, p->minY);
}
Node* build(int l, int r, int deep)
{
    if (l == r) return NULL;
    if (deep & 1) sort(p + l, p + r, cmpx);
    else sort(p + l, p + r, cmpy);
    int mid = (l + r) >> 1;
    Node *t = newNode(l, r, p[mid], deep & 1);
    if (l + 1 == r) return t;
    t->ch[0] = build(l, mid, deep + 1);
    t->ch[1] = build(mid + 1, r, deep + 1);
    if (t->ch[0]) updateInfo(t, t->ch[0]);
    if (t->ch[1]) updateInfo(t, t->ch[1]);
    return t;
}
void updateAns(Point p)
{
    ret = min(ret, dis(p, queryPoint));
}
LL calc(Node *t, LL d)
{
    LL tmp;
    if (d) {
        if (queryPoint.x >= t->minX && queryPoint.x <= t->maxX) tmp = 0;
        else tmp = min(abs(queryPoint.x - t->maxX), abs(queryPoint.x - t->minX));
    } else {
        if (queryPoint.y >= t->minY && queryPoint.y <= t->maxY) tmp = 0;
        else tmp = min(abs(queryPoint.y - t->maxY), abs(queryPoint.y - t->minY));
    }
    return tmp * tmp;
}
void query(Node *t)
{
    if (t == NULL) return;
    if (t->p.id != queryPoint.id) updateAns(t->p);
    if (t->l + 1 == t->r) return;
    LL dl = t->ch[0] ? calc(t->ch[0], t->d) : INF;
    LL dr = t->ch[1] ? calc(t->ch[1], t->d) : INF;
    if (dl < dr) {
        query(t->ch[0]);
        if (ret > dr) query(t->ch[1]);
    } else {
        query(t->ch[1]);
        if (ret > dl) query(t->ch[0]);
    }
}
void solve()
{
    scanf("%d", &N);
    for(int i = 0; i < N; ++ i) {
        scanf("%d%d", &p[i].x, &p[i].y);
        p[i].id = i;
    }
    totNode = nodePool;
    root = build(0, N, 1);

    for(int i = 0; i < N; ++ i) {
        queryPoint = p[i];
        ret = INF;
        query(root);
        ans[p[i].id] = ret;
    }
    for(int i = 0; i < N; ++ i)
        printf("%I64d\n", ans[i]);
}
int main()
{
    int T; scanf("%d", &T);
    for( ; T --; )
        solve();
    return 0;
}
```

45 Splay_sxy

```cpp
int n, m, op, x, l, r, root;
class Q{
public:
  int l, r, fa, sz, num, max, add, rev;
}a[50010];
void upd(int x){
  a[x].sz = a[a[x].l].sz + a[a[x].r].sz + 1;
  a[x].max = max(a[x].num, max(a[a[x].l].max + a[a[x].l].add, a[a[x].r].max + a[a[x].r].add));
}
void down(int x){
  if (a[x].add){
    a[x].max += a[x].add, a[x].num += a[x].add;
    if (a[x].l) a[a[x].l].add += a[x].add;
    if (a[x].r) a[a[x].r].add += a[x].add;
    a[x].add = 0;
  }
  if (a[x].rev){
    swap(a[x].l, a[x].r);
    if (a[x].l) a[a[x].l].rev ^= 1;
    if (a[x].r) a[a[x].r].rev ^= 1;
    a[x].rev = 0;
  }
}
void rot(int x){
  int y = a[x].fa;
  down(y), down(x);
  if (a[y].fa) if (a[a[y].fa].l == y) a[a[y].fa].l = x; else a[a[y].fa].r = x;
  a[x].fa = a[y].fa, a[y].fa = x;
  if (a[y].r == x){
    a[y].r = a[x].l;
    if (a[x].l) a[a[x].l].fa = y;
    a[x].l = y;
  }
  else{
    a[y].l = a[x].r;
    if (a[x].r) a[a[x].r].fa = y;
    a[x].r = y;
  }
  upd(y), upd(x);
}
void splay(int x, int p){
  for (;;)
    if (a[x].fa == p) return;
    else if (a[a[x].fa].fa == p){
      rot(x);
      return;
    }
    else{
      int y = a[x].fa;
      int z = a[y].fa;
      if (((a[z].l == y) ^ (a[y].l == x)) == 0) rot(y), rot(x);
      else rot(x), rot(x);
    }
}
int find(int x, int k){
  down(x);
  if (k == a[a[x].l].sz + 1) return x;
  if (k <= a[a[x].l].sz) return find(a[x].l, k);
  return find(a[x].r, k - a[a[x].l].sz - 1);
}
int main(){
  scanf("%d%d", &n, &m);
  root = 1;a[1].r = 2;a[1].sz = n + 2;
  for (int i = 2; i <= n + 1; ++i) a[i].r = i + 1, a[i].fa = i - 1, a[i].sz = n + 3 - i;
  a[n + 2].fa = n + 1, a[n + 2].sz = 1;
  for (int i = 1;i <= m;++i){
    scanf("%d%d%d", &op, &l, &r);
    splay(root = find(root, l), 0);
    splay(find(root, r + 2), root);
    if (op == 1){
      scanf("%d",&x);
      a[a[a[root].r].l].add += x;
    }
    else if (op == 2) a[a[a[root].r].l].rev ^= 1;
    else printf("%d\n", a[a[a[root].r].l].max + a[a[a[root].r].l].add);
  }
  return 0;
}
```

46 LCT

```
const int N = 300010;
int n, m, x, y, z, op, p[N], e[N][2];
class Q{
public:
  int fa, l, r, sz, x, max, add;
  bool root, rev;
}a[N];
void upd(int x){//最好下放标记
  a[x].sz = 1 + a[a[x].l].sz + a[a[x].r].sz;
  a[x].max = max(a[x].x, max(a[a[x].l].max + a[a[x].l].add, a[a[x].r].max + a[a[x].r].add));
}
void down(int x){
  if (a[x].add){
    a[x].x += a[x].add;
    a[x].max += a[x].add;
    if (a[x].l) a[a[x].l].add += a[x].add;
    if (a[x].r) a[a[x].r].add += a[x].add;
    a[x].add = 0;
  }
  if (a[x].rev){
    swap(a[x].l, a[x].r);
    if (a[x].l) a[a[x].l].rev ^= 1;
    if (a[x].r) a[a[x].r].rev ^= 1;
    a[x].rev = 0;
  }
}
void rot(int x){
  int y = a[x].fa;
  down(y), down(x);
  if (a[y].fa && !a[y].root)
    if (a[a[y].fa].l == y) a[a[y].fa].l = x; else a[a[y].fa].r = x;
  a[x].fa = a[y].fa, a[y].fa = x;
  if (a[y].r == x){
    a[y].r = a[x].l;
    if (a[x].l) a[a[x].l].fa = y;
    a[x].l = y;
  }
  else{
    a[y].l = a[x].r;
    if (a[x].r) a[a[x].r].fa = y;
    a[x].r = y;
  }
  if (a[y].root) a[x].root = 1, a[y].root = 0;
  upd(y), upd(x);
}
void splay(int x){
  p[0] = 0;
  for (int y = x; ; y = a[y].fa){
    p[++p[0]] = y;
    if (a[y].root) break;
  }
  for (int i = p[0]; i >= 1; --i) down(p[i]);
  for (; ; ){
    if (a[x].root) break;
    else if (a[a[x].fa].root){
      rot(x);
      break;
    }
    else{
      int y = a[x].fa;
      int z = a[y].fa;
      if (((a[z].l == y) ^ (a[y].l == x)) == 0) rot(y), rot(x);
      else rot(x), rot(x);
    }
  }
}
void access(int x){
  splay(x);
  if (a[x].r) a[a[x].r].root = 1;
  a[x].r = 0;
  upd(x);
  for (; a[x].fa; ){
    splay(a[x].fa);
    int z = a[x].fa;
    if (a[z].r) a[a[z].r].root = 1;
    a[x].root = 0;
    a[z].r = x;
    upd(z);
    splay(x);
  }
}
void setroot(int x){
  access(x);
  a[x].rev ^= 1;
}
int findroot(int x){
  access(x);
  for (; a[x].l; x = a[x].l);
  splay(x);
  return x;
}
bool join(int x, int y){
  if (findroot(x) == findroot(y)) return false;
  setroot(x);
  access(x);
  a[x].fa = y;
  access(x);
  return true;
}
void cut(int x, int y){//y == 0 cut x和x的父亲,y != 0 cut x和y(有直接连边)
  if (y) setroot(y);
  access(x);
  a[a[x].l].root = 1;
  a[a[x].l].fa = a[x].fa;
  a[x].l = 0;
  upd(x);
}
int main(){
  for (; ; ){
    scanf("%d", &n);
    if (feof(stdin)) return 0;
    for (int i = 1; i <= n; ++i) a[i].x = a[i].max = a[i].fa = a[i].l = a[i].r = a[i].add = 0, a[i].sz = 1, a[i].root = 1, a[i].rev = 0;
    for (int i = 1; i < n; ++i) scanf("%d%d", &e[i][0], &e[i][1]);
    for (int i = 1; i <= n; ++i){
      scanf("%d", &x);
      a[i].x = a[i].max = x;
    }
    for (int i = 1; i < n; ++i) join(e[i][0], e[i][1]);
    scanf("%d", &m);
    for (int i = 1; i <= m; ++i){
      scanf("%d", &op);
      if (op == 1){
        scanf("%d%d", &x, &y);
        if (!join(x, y)) printf("-1\n");
      }
      else if (op == 2){
        scanf("%d%d", &x, &y);
        if (x == y || findroot(x) != findroot(y)){
          printf("-1\n");
          continue;
        }
        setroot(x);
        cut(y, 0);
      }
      else if (op == 3){
        scanf("%d%d%d", &z, &x, &y);
        if (findroot(x) != findroot(y)){
          printf("-1\n");
          continue;
        }
        setroot(x);
        access(y);
        a[y].add += z;
      }
      else{
        scanf("%d%d", &x, &y);
        if (findroot(x) != findroot(y)){
          printf("-1\n");
          continue;
        }
        setroot(x);
        access(y);
        printf("%d\n", a[y].max + a[y].add);
      }
    }
    printf("\n");
  }
  return 0;
}
```

47 树链剖分

```
const int N = 10000,M = 30000;
int T, n, x, y, ind, totw, ans, nt[M], pt[M], last[N], len[M], fa[N], dep[N], siz[N], son[N], w[N], wf[N
      ], top[N], f[N];
char s[10];
bool vis[N];
class edge{
public:
  int x, y, z, w;
};
edge e[N];
int a[N * 10];
void edge(int x, int y, int z){
  last[x] = nt[last[x]] = ++ind;
  pt[ind] = y,len[ind] = z;
}
void mkt(int ind, int l, int r){
  if (l == r){
    a[ind] = f[wf[l]];
    return;
  }
  int mid = (l + r) / 2;
  mkt(ind * 2, l, mid);
  mkt(ind * 2 + 1, mid + 1, r);
  a[ind] = max(a[ind * 2], a[ind * 2 + 1]);
}
void change(int ind, int l, int r, int x, int y){
  if (l == r && l == x){
    a[ind] = y;
    return;
  }
  int mid = (l + r) / 2;
  if (x <= mid) change(ind * 2, l, mid, x, y);
  else change(ind * 2 + 1, mid + 1, r, x, y);
  a[ind] = max(a[ind * 2], a[ind * 2 + 1]);
}
int query(int ind, int l, int r, int x, int y){
  if (l == x && r == y) return a[ind];
  int mid = (l + r) / 2;
  if (y <= mid) return query(ind * 2, l, mid, x, y);
  else if (x > mid) return query(ind * 2 + 1, mid + 1, r, x, y);
  else return max(query(ind * 2, l, mid, x, mid), query(ind * 2 + 1, mid + 1, r, mid + 1, y));
}
void dfs1(int x, int y){
  vis[x] = 1;
  dep[x] = y;
  siz[x] = 1;
  int max = 0,maxi = 0;
  for (int i = nt[x]; pt[i]; i = nt[i]) if (!vis[pt[i]]){
    fa[pt[i]] = x;
    dfs1(pt[i],y + 1);
    siz[x] += siz[pt[i]];
    if (siz[pt[i]] > max) max = siz[pt[i]], maxi = pt[i];
  }
  son[x] = maxi;
}
void dfs2(int x){
  w[x] = ++totw;
  if (son[x]){
    top[son[x]] = top[x];
    dfs2(son[x]);
  }
  for (int i = nt[x]; pt[i]; i = nt[i]) if (pt[i] != fa[x] && pt[i] != son[x]){
    top[pt[i]] = pt[i];
    dfs2(pt[i]);
  }
}
//处理点权、提取路径子程序，“totw = 0;mkt(1, 1, n);”
void go(int x, int y){
  int f1, f2;
  Q ans;
  bool first = true;
  t1 = t2 = 0;
  for (; ; ){
    f1 = top[x], f2 = top[y];
    if (f1 == f2){
      if (dep[x] < dep[y]) g2[++t2][0] = w[x], g2[t2][1] = w[y];
      else g1[++t1][0] = w[y], g1[t1][1] = w[x];
      break;
    }
    else if (dep[f1] > dep[f2]){
      g1[++t1][0] = w[f1], g1[t1][1] = w[x];
      x = fa[f1];
    }
    else{
      g2[++t2][0] = w[f2], g2[t2][1] = w[y];
      y = fa[f2];
    }
  }
  for (int i = 1; i <= t1; ++i){
    if (first) ans = rev(query(1, 1, n, g1[i][0], g1[i][1]));
    else ans = merge(ans, rev(query(1, 1, n, g1[i][0], g1[i][1])));
    first = false;
  }
  for (int i = t2; i >= 1; --i){
    if (first) ans = query(1, 1, n, g2[i][0], g2[i][1]);
    else ans = merge(ans, query(1, 1, n, g2[i][0], g2[i][1]));
    first = false;
  }
  printf("%d\n", max(max(ans.L[0], ans.L[1]), 0));
}
int main(){
  for (scanf("%d",&T); T--; ){
    scanf("%d", &n);
    for (int i = 1; i <= ind; ++i) nt[i] = pt[i] = len[i] = 0;
    ind = n;
    for (int i = 1; i <= n; ++i) last[i] = i;
    for (int i = 1; i < n; ++i){
      scanf("%d%d%d", &e[i].x, &e[i].y, &e[i].z);
      edge(e[i].x, e[i].y, e[i].z);
      edge(e[i].y, e[i].x, e[i].z);
    }
    memset(vis, 0, sizeof(vis));
    dfs1(1, 1);
    for (int i = 1; i < n; ++i)
      if (dep[e[i].x] > dep[e[i].y]) f[e[i].x] = e[i].z,e[i].w = e[i].x;
      else f[e[i].y] = e[i].z,e[i].w = e[i].y;
    totw = -1;//处理点权的时候是0
    top[1] = 1;
    dfs2(1);
    for (int i = 1; i <= n; ++i) wf[w[i]] = i;
    mkt(1, 1, n - 1);//处理点权的时候是n，下同
    for (; ; ){
      scanf("%s", s + 1);
      if (s[1] == 'D') break;
      scanf("%d%d", &x, &y);
      if (s[1] == 'C') change(1, 1, n - 1, w[e[x].w], y);
      else{
        ans = 0;
        int f1, f2;
        for (; ; ){
          if (x == y) break;//处理点权时去掉
          f1 = top[x], f2 = top[y];
          if (f1 == f2){
            if (dep[x] < dep[y]) ans = max(ans, query(1, 1, n - 1, w[x] + 1, w[y]));
            else ans = max(ans, query(1, 1, n - 1, w[y] + 1, w[x]));
            break;
          }
          else if (dep[f1] > dep[f2]){
            ans = max(ans, query(1, 1, n - 1, w[f1], w[x]));
            x = fa[f1];
          }
          else{
            ans = max(ans,query(1, 1, n - 1, w[f2], w[y]));
            y = fa[f2];
          }
        }
        printf("%d\n", ans);
      }
    }
  }
  return 0;
}
```

杂

48 求某年某月某日星期几

```
int whatday(int d, int m, int y)
{
  int ans;
  if (m == 1 || m == 2) {
    m += 12; y --;
  }
  if ((y < 1752) || (y == 1752 && m < 9) || (y == 1752 && m == 9 && d < 3))
    ans = (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 + 5) % 7;
  else ans = (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 - y / 100 + y / 400) % 7;
  return ans;
}
int main()
{
  cout << whatday(30, 10, 2013) << endl;
}
```

49 Java Reference

```
import java.io.*;
import java.math.*;
import java.util.*;
public class Main {
  public void run() {
//    BigInteger f[] = new BigInteger[101];
    int TT = 0, cases = 0;
    try {
      TT = reader.nextInt();
    } catch (IOException ex) {
    }
//    while (reader.hasNext()) {
    while ((TT--) > 0) {
//    while (true) {
    try {
    } catch (IOException ex) {
    }
    }
    writer.close();
  }
  InputReader reader;
  PrintWriter writer;
  Main() {
    reader = new InputReader();
    writer = new PrintWriter(System.out);
  }
  public static void main(String [] args) {
    new Main().run();
  }
  void debug(Object...os) {
    System.err.println(Arrays.deepToString(os));
  }
}
class InputReader {
  BufferedReader reader;
  StringTokenizer tokenizer;
  InputReader() {
    reader = new BufferedReader(new InputStreamReader(System.in));
    tokenizer = new StringTokenizer("");
  }
  boolean hasNext() {
    while (tokenizer == null || !tokenizer.hasMoreTokens())
      try {
        tokenizer = new StringTokenizer(reader.readLine());
      }
      catch (Exception e) {
        return false;
      }
    return true;
  }
  String next() throws IOException {
    while (!tokenizer.hasMoreTokens()) {
      tokenizer = new StringTokenizer(reader.readLine());
    }
    return tokenizer.nextToken();
  }
  Integer nextInt() throws IOException {
    return Integer.parseInt(next());
  }
}
//________________________________________----
Scanner cin;
void solve() {
  cin = new Scanner(new BufferedInputStream(System.in));
  cin.hasNextBigInteger();
  cin.nextBigInteger();
  System.out.println();
}
//Arrays
int a[];
.fill(a, 0); | .sort(a)
//String
String s;
.charAt(int i); | compareTo(String) | compareToIgnoreCase() | contains(String)
length() | substring(int l, int len)
//BigInteger
.abs() | .add() | bitLength() | subtract() | divide() | remainder() | divideAndRemainder() | modPow(b, c)
       | pow(int) | multiply() | compareTo() |
gcd() | intValue() | longValue() | isProbablePrime(int c) (1 - 1/2^c) |
nextProbablePrime() | shiftLeft(int) | valueOf()
//BigDecimal
.ROUND_CEILING | ROUND_DOWN_FLOOR | ROUND_HALF_DOWN | ROUND_HALF_EVEN | ROUND_HALF_UP | ROUND_UP
.divide(BigDecimal b, int scale, int round_mode) | doubleValue() | movePointLeft(int) | pow(int) |
      setScale(int scale, int round_mode) | stripTrailingZeros()
//StringBuilder
StringBuilder sb = new StringBuilder();
sb.append(elem) | out.println(sb)
```