SuperNova

Standard Code Library

version 3.141

October, 2013

Contents

1 二维几何 **7**
- 1.1 Naive Tips . . . 7
- 1.2 几何公式 . . . 7
- 1.3 点类 . . . 10
- 1.4 基本操作 . . . 12
- 1.5 球面 . . . 15
- 1.6 半平面交 . . . 16
- 1.7 最小圆覆盖 . . . 18
- 1.8 求直线与凸包的交点 . . . 20
  - 1.8.1 tEJtM . . . 20
  - 1.8.2 Seraphim . . . 22
- 1.9 点到凸包的切线 . . . 24
- 1.10 判断圆存在交集 $NlogK$ . . . 26
- 1.11 圆与三角形的交面积 . . . 27
- 1.12 圆与圆的交并面积 . . . 30
- 1.13 Farmland . . . 32
  - 1.13.1 Logic_IU . . . 32
  - 1.13.2 tEJtM . . . 35

2 三维几何 **39**
- 2.1 三维几何 Seraphim Version . . . 39
  - 2.1.1 基本操作 . . . 39
  - 2.1.2 三维旋转操作 . . . 42
  - 2.1.3 三维凸包随机增量 . . . 42
- 2.2 基本操作 tEJtM Version . . . 45
- 2.3 点类 + 三维凸包 N^3+ 凸包求重心 . . . 48
- 2.4 三维旋转 . . . 52
- 2.5 最小球覆盖 . . . 53

3 图论 **57**
- 3.1 Dijkstra . . . 57
- 3.2 最大流 . . . 57
- 3.3 上下界流 . . . 59
  - 3.3.1 上下界无源汇可行流 . . . 59
  - 3.3.2 上下界最大流 . . . 61
  - 3.3.3 上下界最小流 . . . 64
  - 3.3.4 上下界有源汇可行流 . . . 66

3.4 费用流 . . . 70
3.4.1 Logic_IU+ 负费用路 . . . 70
3.4.2 shytangyuan+ZKW . . . 71
3.5 强联通分量 . . . 73
3.5.1 Logic_IU . . . 73
3.5.2 shytangyuan+ 手写栈 . . . 74
3.6 KM . . . 76
3.6.1 tEJtM . . . 76
3.6.2 Logic_IU . . . 78
3.6.3 shytangyuan+ 邻接阵 . . . 79
3.6.4 shytangyuan+ 链表 . . . 81
3.7 Hopcroft . . . 83
3.8 一般图最大匹配 . . . 84
3.9 无向图最小割 . . . 87
3.10 最小树形图 $(ElogE + V^2)$ . . . 88
3.11 最小树形图 (V^3) . . . 90

4 数据结构 **93**
4.1 KD 树 . . . 93
4.1.1 tEJtM+ 高维 . . . 93
4.1.2 Logic_IU . . . 95
4.2 后缀自动机 . . . 98
4.2.1 tEJtM+LCA 非递归 Tarjan . . . 98
4.2.2 Logic_IU . . . 100
4.3 Splay 树 . . . 101
4.3.1 Logic_IU . . . 101
4.3.2 shytangyuan . . . 104
4.4 动态树 . . . 107
4.5 二叉堆 . . . 108
4.6 左偏树 . . . 109
4.7 Treap . . . 110
4.8 线段树 . . . 111
4.9 轻重链剖分 . . . 112
4.10 KMP . . . 114
4.11 扩展 KMP . . . 114
4.12 Manacher . . . 115
4.13 AC 自动机 . . . 115
4.13.1 Logic_IU . . . 115
4.13.2 shytangyuan . . . 116
4.14 后缀数组 . . . 117
4.14.1 Logic_IU . . . 117
4.14.2 shytangyuan . . . 118
4.14.3 DC3 . . . 119

5 杂 **123**
5.1 m^2logn 求线性递推第 n 项 . . . 123
5.2 FFT . . . 123
5.3 中国剩余定理 . . . 129
5.4 Pollard's Rho+Miller-Rabbin . . . 129

5.5 素数判定 (long long 内确定性算法) . . . 131
5.6 求前 P 个数的逆元 . . . 132
5.7 广义离散对数 (不需要互质) . . . 132
5.8 n 次剩余 . . . 133
5.9 二次剩余 . . . 137
5.10 长方体表面两点最短距离 . . . 139
5.11 字符串的最小表示 . . . 140
5.11.1 Logic_IU . . . 140
5.11.2 tEJtM . . . 140
5.12 牛顿迭代开根号 . . . 141
5.13 求某年某月某日星期几 . . . 141
5.14 A* . . . 141
5.15 Dancing Links . . . 144
5.16 弦图判定 . . . 146
5.17 弦图求团数 . . . 148
5.18 有根树的同构 . . . 150
5.19 极大团搜索算法 . . . 152
5.20 极大团的计数 . . . 153
5.21 多项式求根 (求导二分) . . . 154
5.22 有多少个点在多边形内 . . . 155
5.23 斜线下格点统计 . . . 155
5.24 杂知识 . . . 156
5.25 Language Reference . . . 157
5.25.1 C++ Tips . . . 157
5.25.2 Java Reference . . . 157
5.26 vimrc . . . 161

Chapter 1

二维几何

1.1 Naive Tips

1. 注意舍入方式 (0.5 的舍入方向)，防止输出 -0
2. 几何题注意多测试不对称数据
3. 整数几何注意避免出界
4. 符点几何注意 EPS 的使用
5. 公式化简后再代入
6. atan2(0,0)=0, atan2 的值域为 [-π, π]
7. 使用 acos, asin, sqrt 等函数时，注意定义域

1.2 几何公式

高维球

1. 体积 $V_0 = 1$, $V_{n+1} = S_n/(n+1)$
2. 表面积 $S_0 = 2$, $S_{n+1} = 2\pi V_n$

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

$$r = S/P = \arcsin(B/2)\sin(C/2)/\sin((B+C)/2) = 4R\sin(A/2)\sin(B/2)\sin(C/2)$$
$$= \sqrt{(P-a)(P-b)(P-c)/P} = P\tan(A/2)\tan(B/2)\tan(C/2)$$

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 $V = \pi r^2 h$

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 $V = \pi r^2 h/3$

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 $V = 2\pi r^2 h/3$

1.3 点类

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <cstring>
#include <iostream>
#include <algorithm>
#define foreach(e,x) for(__typeof(x.begin()) e=x.begin();e!=x.end();++e)
using namespace std;

const double PI = acos(-1.);
const double EPS = 1e-8;
inline int sign(double a) {
    return a < -EPS ? -1 : a > EPS;
}

struct Point {
    double x, y;
    Point() {
    }
    Point(double _x, double _y) :
            x(_x), y(_y) {
    }
    Point operator+(const Point&p) const {
        return Point(x + p.x, y + p.y);
    }
    Point operator-(const Point&p) const {
        return Point(x - p.x, y - p.y);
    }
    Point operator*(double d) const {
        return Point(x * d, y * d);
    }
    Point operator/(double d) const {
        return Point(x / d, y / d);
    }
    bool operator<(const Point&p) const {
        int c = sign(x - p.x);
        if (c)
            return c == -1;
        return sign(y - p.y) == -1;
    }
    double dot(const Point&p) const {
```

```
42          return x * p.x + y * p.y;
43      }
44      double det(const Point&p) const {
45          return x * p.y - y * p.x;
46      }
47      double alpha() const {
48          return atan2(y, x);
49      }
50      double distTo(const Point&p) const {
51          double dx = x - p.x, dy = y - p.y;
52          return hypot(dx, dy);
53      }
54      double alphaTo(const Point&p) const {
55          double dx = x - p.x, dy = y - p.y;
56          return atan2(dy, dx);
57      }
58      //clockwise
59      Point rotAlpha(const double &alpha, const Point &o = Point(0, 0)) const {
60          double nx = cos(alpha) * (x - o.x) + sin(alpha) * (y - o.y);
61          double ny = -sin(alpha) * (x - o.x) + cos(alpha) * (y - o.y);
62          return Point(nx, ny) + o;
63      }
64      Point rot90() const {
65          return Point(-y, x);
66      }
67      Point unit() {
68          return *this / abs();
69      }
70      void read() {
71          scanf("%lf%lf", &x, &y);
72      }
73      double abs() {
74          return hypot(x, y);
75      }
76      double abs2() {
77          return x * x + y * y;
78      }
79      void write() {
80          cout << "(" << x << "," << y << ")" << endl;
81      }
82  };
83
84  #define cross(p1,p2,p3) ((p2.x-p1.x)*(p3.y-p1.y)-(p3.x-p1.x)*(p2.y-p1.y))
85  #define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))
86
87  Point isSS(Point p1, Point p2, Point q1, Point q2) {
88      double a1 = cross(q1,q2,p1), a2 = -cross(q1,q2,p2);
89      return (p1 * a2 + p2 * a1) / (a1 + a2);
90  }
```

```
91
92 double minDiff(double a, double b) // a, b in[0, 2 * PI)
93 {
94     return min(abs(a - b), 2 * PI - abs(a - b));
95 }
```

1.4 基本操作

顺时针或逆时针传入一个凸多边形，返回被半平面 $\overrightarrow{q1q2}$ 逆时针方向切割掉之后的凸多边形

```
1  vector<Point> convexCut(const vector<Point>&ps, Point q1, Point q2) {
2      vector<Point> qs;
3      int n = ps.size();
4      for (int i = 0; i < n; ++i) {
5          Point p1 = ps[i], p2 = ps[(i + 1) % n];
6          int d1 = crossOp(q1,q2,p1), d2 = crossOp(q1,q2,p2);
7          if (d1 >= 0)
8              qs.push_back(p1);
9          if (d1 * d2 < 0)
10             qs.push_back(isSS(p1, p2, q1, q2));
11     }
12     return qs;
13 }
```

返回 ps 的有向面积

```
1  double calcArea(const vector<Point>&ps) {
2      int n = ps.size();
3      double ret = 0;
4      for (int i = 0; i < n; ++i) {
5          ret += ps[i].det(ps[(i + 1) % n]);
6      }
7      return ret / 2;
8  }
```

返回点集 ps 组成的凸包

```
1  vector<Point> convexHull(vector<Point> ps) {
2      int n = ps.size();
3      if (n <= 1)
4          return ps;
5      sort(ps.begin(), ps.end());
6      vector<Point> qs;
7      for (int i = 0; i < n; qs.push_back(ps[i++])) {
8          while (qs.size() > 1 && crossOp(qs[qs.size()-2],qs.back(),ps[i]) <= 0)
9              qs.pop_back();
10     }
11     for (int i = n - 2, t = qs.size(); i >= 0; qs.push_back(ps[i--])) {
12         while ((int)qs.size() > t && crossOp(qs[(int)qs.size()-2],qs.back(),ps[i]) <=
               0)
13             qs.pop_back();
14     }
```

```
15      qs.pop_back();
16      return qs;
17  }
```

返回凸包 ps 的直径

```
1   double convexDiameter(const vector<Point>&ps) {
2       int n = ps.size();
3       int is = 0, js = 0;
4       for (int i = 1; i < n; ++i) {
5           if (ps[i].x > ps[is].x)
6               is = i;
7           if (ps[i].x < ps[js].x)
8               js = i;
9       }
10      double maxd = ps[is].distTo(ps[js]);
11      int i = is, j = js;
12      do {
13          if ((ps[(i + 1) % n] - ps[i]).det(ps[(j + 1) % n] - ps[j]) >= 0)
14              (++j) %= n;
15          else
16              (++i) %= n;
17          maxd = max(maxd, ps[i].distTo(ps[j]));
18      } while (i != is || j != js);
19      return maxd;
20  }
```

判断点 p 在线段 q1q2 上，端点重合返回 true

```
1   int onSegment(Point p, Point q1, Point q2)
2   {
3       return crossOp(q1, q2, p) == 0 && sign((p - q1).dot(p - q2)) <= 0;
4   }
```

判断线段 p1p2 和 q1q2 是否严格相交，重合或端点相交返回 false

```
1   int isIntersect(Point p1, Point p2, Point q1, Point q2)
2   {
3       return crossOp(p1, p2, q1) * crossOp(p1, p2, q2) < 0 && crossOp(q1, q2, p1) *
            cross(q1, q2, p2) < 0;
4   }
```

判断直线 p1p2 和 q1q2 是否平行

```
1   int isParallel(Point p1, Point p2, Point q1, Point q2)
2   {
3       return sign((p2 - p1).det(q2 - q1)) == 0;
4   }
```

返回点 p 到直线 uv 的距离

```
1   double distPointToLine(Point p, Point u, Point v)
2   {
3       return abs((u - p).det(v - p)) / u.distTo(v);
4   }
```

判断点 q 是否在简单多边形 p 内部，边界返回 false

```
int insidePolygon(Point q, vector<Point> &p)
{
    int n = p.size();
    for(int i = 0; i < n; ++ i) {
        if (onSegment(q, p[i], p[(i + 1) % n])) return false;
    }
    Point q2;
    double offsite = LIM;
    for( ; ; ) {
        int flag = true;
        int rnd = rand() % 10000;
        q2.x = cos(rnd) * offsite;
        q2.y = sin(rnd) * offsite;
        for(int i = 0; i < n; ++ i) {
            if (onSegment(p[i], q, q2)) {
                flag = false;
                break;
            }
        }
        if (flag) break;
    }
    int cnt = 0;
    for(int i = 0; i < n; ++ i) {
        cnt += isIntersect(p[i], p[(i + 1) % n], q, q2);
    }
    return cnt & 1;
}
```

判断直线 l1l2 是否与圆相交，相切返回 true

```
int isIntersectLineToCircle(Point c, double r, Point l1, Point l2)
{
    return (distPointToLine(c, l1, l2) - r) <= 0;
}
```

判断圆与线段是否有公共点，线段在圆内部返回 true

```
int isIntersectSegmentToCircle(Point c, double r, Point p1, Point p2)
{
    if ((distPointToLine(c, p1, p2) - r) > 0) return false;
    if (sign(c.distTo(p1) - r) <= 0 || sign(c.distTo(p2) - r) <= 0) return true;
    Point c2 = (p2 - p1).rot90() + c;
    return crossOp(c, c2, p1) * crossOp(c, c2, p2) <= 0;
}
```

判断圆与圆是否相交，外切或内切返回 true

```
int isIntersectCircleToCircle(Point c1, double r1, Point c2, double r2)
{
    double dis = c1.distTo(c2);
    return sign(dis - abs(r1 - r2)) >= 0 && sign(dis - (r1 + r2)) <= 0;
```

```
5 }
```

求直线与圆的两个交点

```
1 void intersectionLineToCircle(Point c, double r, Point l1, Point l2, Point& p1, Point
    & p2) {
2     Point c2 = c + (l2 - l1).rot90();
3     c2 = isSS(c, c2, l1, l2);
4     double t = sqrt(r * r - (c2 - c).abs2());
5     p1 = c2 + (l2 - l1).unit() * t;
6     p2 = c2 - (l2 - l1).unit() * t;
7 }
```

求圆与圆的两个交点

```
1 void intersectionCircleToCircle(Point c1, double r1, Point c2, double r2, Point &p1,
    Point &p2) {
2     double t = (1 + (r1 * r1 - r2 * r2) / (c1 - c2).abs2()) / 2;
3     Point u = c1 + (c2 - c1) * t;
4     Point v = u + (c2 - c1).rot90();
5     intersectionLineToCircle(c1, r1, u, v, p1, p2);
6 }
```

1.5 球面

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

```
1 double angle(double lng1,double lat1,double lng2,double lat2) {
2     double dlng = abs(lng1 - lng2) * PI / 180;
3     while(dlng >= PI + PI) dlng -= PI + PI;
4     if (dlng > PI) dlng = PI + PI - dlng;
5     lat1 *= PI / 180, lat2 *= PI / 180;
6     return acos(cos(lat1) * cos(lat2) * cos(dlng) + sin(lat1) * sin(lat2));
7 }
```

计算直线距离,r 为球半径

```
1 double line_dist(double r,double lng1,double lat1,double lng2,double lat2) {
2     double dlng = abs(lng1 - lng2) * PI / 180;
3     while(dlng >= PI + PI) dlng -= PI + PI;
4     if (dlng > PI) dlng = PI + PI - dlng;
5     lat1 *= PI / 180, lat2 *= PI / 180;
6     return r * sqrt(2 - 2 * (cos(lat1) * cos(lat2) * cos(dlng) + sin(lat1) * sin(lat2
          )));
7 }
```

计算球面距离,r 为球半径

```
1 inline double sphere_dist(double r,double lng1,double lat1,double lng2,double lat2){
2     return r * angle(lng1, lat1, lng2, lat2);
3 }
```

1.6 半平面交

```
1  struct Border {
2      Point p1, p2;
3      double alpha;
4      void setAlpha() {
5          alpha = atan2(p2.y - p1.y, p2.x - p1.x);
6      }
7      void read() {
8          p1.read();
9          p2.read();
10         setAlpha();
11     }
12 };
13
14 int n;
15 const int MAX_N_BORDER = 20000 + 10;
16 Border border[MAX_N_BORDER];
17
18 bool operator<(const Border&a, const Border&b) {
19     int c = sign(a.alpha - b.alpha);
20     if (c != 0)
21         return c == 1;
22     return crossOp(b.p1,b.p2,a.p1) >= 0;
23 }
24
25 bool operator==(const Border&a, const Border&b) {
26     return sign(a.alpha - b.alpha) == 0;
27 }
28
29 const double LARGE = 10000;
30
31 void add(double x, double y, double nx, double ny) {
32     border[n].p1 = Point(x, y);
33     border[n].p2 = Point(nx, ny);
34     border[n].setAlpha();
35     n++;
36 }
37
38 Point isBorder(const Border&a, const Border&b) {
39     return isSS(a.p1, a.p2, b.p1, b.p2);
40 }
41
42 Border que[MAX_N_BORDER];
43 int qh, qt;
44
45 bool check(const Border&a, const Border&b, const Border&me) {
46     Point is = isBorder(a, b);
47     return crossOp(me.p1,me.p2,is) > 0;
```

```
48 }
49
50 void convexIntersection() {
51     qh = qt = 0;
52     sort(border, border + n);
53     n = unique(border, border + n) - border;
54     for (int i = 0; i < n; ++i) {
55         Border cur = border[i];
56         while (qh + 1 < qt && !check(que[qt - 2], que[qt - 1], cur))
57             --qt;
58         while (qh + 1 < qt && !check(que[qh], que[qh + 1], cur))
59             ++qh;
60         que[qt++] = cur;
61     }
62     while (qh + 1 < qt && !check(que[qt - 2], que[qt - 1], que[qh]))
63         --qt;
64     while (qh + 1 < qt && !check(que[qh], que[qh + 1], que[qt - 1]))
65         ++qh;
66 }
67
68 void calcArea() {
69     static Point ps[MAX_N_BORDER];
70     int cnt = 0;
71
72     if (qt - qh <= 2) {
73         puts("0.0");
74         return;
75     }
76
77     for (int i = qh; i < qt; ++i) {
78         int next = i + 1 == qt ? qh : i + 1;
79         ps[cnt++] = isBorder(que[i], que[next]);
80     }
81
82     double area = 0;
83     for (int i = 0; i < cnt; ++i) {
84         area += ps[i].det(ps[(i + 1) % cnt]);
85     }
86     area /= 2;
87     area = fabsl(area);
88     cout.setf(ios::fixed);
89     cout.precision(1);
90     cout << area << endl;
91 }
92
93 void halfPlaneIntersection()
94 {
95     cin >> n;
96     for (int i = 0; i < n; ++i) {
```

```
97          border[i].read();
98      }
99      add(0, 0, LARGE, 0);
100     add(LARGE, 0, LARGE, LARGE);
101     add(LARGE, LARGE, 0, LARGE);
102     add(0, LARGE, 0, 0);
103
104     convexIntersection();
105     calcArea();
106 }
```

1.7 最小圆覆盖

```
1  #include<cmath>
2  #include<cstdio>
3  #include<algorithm>
4  using namespace std;
5  const double eps=1e-6;
6  struct couple
7  {
8      double x, y;
9      couple(){}
10     couple(const double &xx, const double &yy)
11     {
12         x = xx; y = yy;
13     }
14 } a[100001];
15 int n;
16 bool operator < (const couple & a, const couple & b)
17 {
18     return a.x < b.x - eps or (abs(a.x - b.x) < eps and a.y < b.y - eps);
19 }
20 bool operator == (const couple & a, const couple & b)
21 {
22     return !(a < b) and !(b < a);
23 }
24 inline couple operator - (const couple &a, const couple &b)
25 {
26     return couple(a.x-b.x, a.y-b.y);
27 }
28 inline couple operator + (const couple &a, const couple &b)
29 {
30     return couple(a.x+b.x, a.y+b.y);
31 }
32 inline couple operator * (const couple &a, const double &b)
33 {
34     return couple(a.x*b, a.y*b);
35 }
36 inline couple operator / (const couple &a, const double &b)
```

```
37 {
38     return a*(1/b);
39 }
40 inline double operator * (const couple &a, const couple &b)
41 {
42     return a.x*b.y-a.y*b.x;
43 }
44 inline double len(const couple &a)
45 {
46     return a.x*a.x+a.y*a.y;
47 }
48 inline double di2(const couple &a, const couple &b)
49 {
50     return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
51 }
52 inline double dis(const couple &a, const couple &b)
53 {
54     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
55 }
56 struct circle
57 {
58     double r; couple c;
59 } cir;
60 inline bool inside(const couple & x)
61 {
62     return di2(x, cir.c) < cir.r*cir.r+eps;
63 }
64 inline void p2c(int x, int y)
65 {
66     cir.c.x = (a[x].x+a[y].x)/2;
67     cir.c.y = (a[x].y+a[y].y)/2;
68     cir.r = dis(cir.c, a[x]);
69 }
70 inline void p3c(int i, int j, int k)
71 {
72     couple x = a[i], y = a[j], z = a[k];
73     cir.r = sqrt(di2(x,y)*di2(y,z)*di2(z,x))/fabs(x*y+y*z+z*x)/2;
74     couple t1((x-y).x, (y-z).x), t2((x-y).y, (y-z).y), t3((len(x)-len(y))/2, (len(y)-
        len(z))/2);
75     cir.c = couple(t3*t2, t1*t3)/(t1*t2);
76 }
77 inline circle mi()
78 {
79     sort(a + 1, a + 1 + n);
80     n = unique(a + 1, a + 1 + n) - a - 1;
81     if(n == 1)
82     {
83         cir.c = a[1];
84         cir.r = 0;
```

```
85          return cir;
86      }
87      random_shuffle(a + 1, a + 1 + n);
88      p2c(1, 2);
89      for(int i = 3; i <= n; i++)
90          if(!inside(a[i]))
91          {
92              p2c(1, i);
93              for(int j = 2; j < i; j++)
94                  if(!inside(a[j]))
95                  {
96                      p2c(i, j);
97                      for(int k = 1; k < j; k++)
98                          if(!inside(a[k]))
99                              p3c(i,j, k);
100                 }
101         }
102     return cir;
103 }
```

1.8 求直线与凸包的交点

1.8.1 tEJtM

```
1  #include<cstring>
2  #include<cstdio>
3  #include<cmath>
4  #include<algorithm>
5  using namespace std;
6  struct couple
7  {
8      double x, y;
9      couple(){}
10     couple(const double & _x, const double & _y) : x(_x), y(_y) {}
11     void scan() {scanf("%lf%lf", &x, &y); }
12 } cp1, cp2, x, y;
13 double operator * (const couple & a, const couple & b) {return a.x * b.y - a.y * b.x
       ;}
14 couple operator - (const couple & a, const couple & b) {return couple(a.x - b.x, a.y
       - b.y);}
15 couple operator + (const couple & a, const couple & b) {return couple(a.x + b.x, a.y
       + b.y);}
16 couple operator * (const double & a, const couple & b) {return couple(a * b.x, a * b.
       y);}
17 bool les(const couple & a, const couple & b) {return a.x < b.x or a.x == b.x and a.y
       < b.y;}
18 bool mor(const couple & a, const couple & b) {return a.x > b.x or a.x == b.x and a.y
       > b.y;}
19 int n, m, mxi, mni, t1, t2, c1, c2, mi;
```

```
20 double eps = 1e-12;
21 int sign(const double & x) {return x > eps?1:x < -eps?-1:0;}
22 couple cross(const couple &a, const couple &b, const couple &c, const couple &d)
23 {
24     if(sign((b - a) * (d - c)) == 0) return a;
25     double lambda = (c - a) * (d - c) / ((b - a) * (d - c));
26     return a + lambda * (b - a);
27 }
28 double s[50001];
29 struct convex_polygon
30 {
31     couple a[50000];
32     couple & operator [] (int x) {return a[(x % n + n) % n];}
33     int get_max(bool (*cmp)(const couple & a, const couple & b)) {int rtn = 0; for(
           int i = 1; i < n; i++) if(cmp(a[i], a[rtn])) rtn = i; return rtn;}
34 } a;
35 int check(int id)
36 {
37     return (sign((y - x) * (a[id - 1] - a[id]))) * sign((y - x) * (a[id + 1] - a[id])
           ) == 0?-sign((y - x) * (a[id + 1] - a[id])):sign((sign((y - x) * (a[id - 1] -
           a[id]))) - sign((y - x) * (a[id + 1] - a[id])));
38 }
39 int check1(int id)
40 {
41     return sign((y - x) * (a[id] - x)) * sign((y - x) * (a[id + 1] - x)) <= 0?0:sign
           ((y - x) * (a[id] - x));
42 }
43 int di(int (*check)(int), int le, int ri)
44 {
45     int nor = check(le), mid;
46     if(le > ri) ri += n;
47     while(le != ri)
48     {
49         mid = (le + ri) / 2;
50         if(0 == check(mid)) return mid;
51         else if(nor == check(mid)) le = mid + 1;
52         else ri = mid - 1;
53     }
54     return le;
55 }
56 double area(int le, int ri)
57 {
58     le %= n; ri %= n;
59     return (le <= ri)?(s[ri] - s[le]):(s[n] - s[le] + s[ri]);
60 }
61 int main()
62 {
63     freopen("sgu345.in", "r", stdin);
64     scanf("%d", &n);
```

```
65      for(int i = 1; i <= n; i++)
66      {
67          a[i].scan();
68          if(i >= 3 and fabs((a[i] - a[i - 1]) * (a[i - 1] - a[i - 2])) < eps)
69          {
70              a[i - 1] = a[i];
71              i--; n--;
72          }
73      }
74      s[0] = 0;
75      for(int i = 1; i <= n; i++)
76      {
77          s[i] = s[i - 1] + a[i - 1] * a[i];
78      }
79      mni = a.get_max(les);
80      mxi = a.get_max(mor);
81      scanf("%d", &m);
82      for(int i = 1; i <= m; i++)
83      {
84          x.scan(); y.scan();
85          if(check(mni)== 0 or check(mxi)== 0)
86          {
87              if(check(mxi) == 0) mi = mxi; else mi = mni;
88              t1 = mi;
89              t2 = di(check, mi + 1, mi - 1);
90          }else
91          {
92              t1 = di(check, mni, mxi);
93              t2 = di(check, mxi, mni);
94          }
95          c1 = di(check1, t1, t2);
96          c2 = di(check1, t2, t1);
97          if(check1(c1) and check1(c2)) {printf("0\n"); continue;}
98          cp1 = cross(a[c1], a[c1 + 1], x, y);
99          cp2 = cross(a[c2], a[c2 + 1], x, y);
100         printf("%.10f\n", min(fabs(area(c1 + 1, c2) + cp1 * a[c1 + 1] + a[c2] * cp2 +
                cp2 * cp1) / 2, fabs(area(c2 + 1, c1) + a[c1] * cp1 + cp1 * cp2 + cp2 * a
                [c2 + 1]) / 2));
101     }
102     fclose(stdin);
103     return 0;
104 }
```

1.8.2 Seraphim

```
1  double calc(point a, point b){
2      double k=atan2(b.y-a.y , b.x-a.x); if (k<0) k+=2*pi;return k;
3  }
4  //= the convex must compare y, then £¬xa[0] is the lower-right point
5  //======= three is no 3 points in line. a[] is convex 0~n-1
```

```
6  void prepare(point a[] ,double w[],int &n) {
7      int i; rep(i,n) a[i+n]=a[i];
8      a[2*n]=a[0];
9      rep(i,n) { w[i]=calc(a[i],a[i+1]);w[i+n]=w[i];}
10 }
11 int find(double k,int n , double w[]){
12     if (k<=w[0] || k>w[n-1]) return 0; int l,r,mid; l=0; r=n-1;
13     while (l<=r) { mid=(l+r)/2;if (w[mid]>=k) r=mid-1; else l=mid+1;
14     }return r+1;
15 }
16 int dic(const point &a, const point &b , int l ,int r , point c[]) {
17     int s; if (area(a,b,c[l])<0) s=-1; else s=1; int mid;
18     while (l<=r) {
19         mid=(l+r)/2; if (area(a,b,c[mid])*s <= 0) r=mid-1;
20         else l=mid+1;
21     }return r+1;
22 }
23 point get(const point &a, const point &b, point s1, point s2) {
24     double k1,k2; point tmp; k1=area(a,b,s1); k2=area(a,b,s2);
25     if (cmp(k1)==0) return s1; if (cmp(k2)==0) return s2;
26     tmp=(s1*k2 - s2*k1) / (k2-k1);
27     return tmp;
28 }
29 bool line_cross_convex(point a, point b ,point c[] , int n, point &cp1, point &cp2 ,
       double w[]) {
30     int i,j;
31     i=find(calc(a,b),n,w);
32     j=find(calc(b,a),n,w);
33     double k1,k2;
34     k1=area(a,b,c[i]); k2=area(a,b,c[j]);
35     if (cmp(k1)*cmp(k2)>0) return false; //no cross
36     if (cmp(k1)==0 || cmp(k2)==0) {
37         //cross a point or a line in the convex
38         if (cmp(k1)==0) {
39             if (cmp(area(a,b,c[i+1]))==0) {cp1=c[i]; cp2=c[i+1];}
40             else cp1=cp2=c[i];
41             return true;
42         }
43         if (cmp(k2)==0) {
44             if (cmp(area(a,b,c[j+1]))==0) {cp1=c[j];cp2=c[j+1];
45             }else cp1=cp2=c[j];
46         }return true;
47     }
48     if (i>j) swap(i,j); int x,y;
49     x=dic(a,b,i,j,c); y=dic(a,b,j,i+n,c);
50     cp1=get(a,b,c[x-1],c[x]); cp2=get(a,b,c[y-1],c[y]);
51     return true;
52 }
```

1.9 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];
```

```
48          bs[0] = lb - 1; bs[1] = cl - 1;
49          while(bs[0] < bs[1] - 1)
50          {
51              mid = (bs[0] + bs[1]) / 2;
52              bs[!check(mid)] = mid;
53          }
54          to = check(bs[1])?bs[1]:bs[0];
55          if(!frm) ans1 += 0 * (x * q[1]);
56          else if(to == cl) ans1 += 0 * (q[cl1] * x);
57          else ans1 += q[frm] * x + x * q[to + 1] - s[to] + s[frm - 1];
58      }
59  }
60  int main()
61  {
62      scanf("%d%d", &n, &Q);
63      for(int i = 1; i <= n; i++) a[i].scan();
64      sort(a + 1, a + 1 + n);
65      q = q1; s = s1;
66      cl = 0;
67      for(int i = 1; i <= n; i++)
68      {
69          psh(i);
70      }
71      s[0] = 0;
72      for(int i = 1; i < cl; i++) s[i] = s[i - 1] + q[i] * q[i + 1];
73      cl1 = cl;
74      q = q2; s = s2;
75      cl = 0;
76      for(int i = n; i >= 1; i--)
77      {
78          psh(i);
79      }
80      s[0] = 0;
81      for(int i = 1; i < cl; i++) s[i] = s[i - 1] + q[i] * q[i + 1];
82      cl2 = cl;
83      ans = s1[cl1 - 1] + s2[cl2 - 1];
84      for(int i = 1; i <= Q; i++)
85      {
86          x.scan();
87          ans1 = ans;
88          cl = cl1; q = q1; s = s1; cmp = lss;
89          calc();
90          cl = cl2; q = q2; s = s2; cmp = grt;
91          calc();
92          ans1 = abs(ans1);
93          printf("%lld.%c\n", ans1 / 2, ans1 % 2 == 1?'5':'0');
94      }
95      fclose(stdin);
96      return 0;
```

```
97 }
```

1.10 判断圆存在交集 $NlogK$

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

```
1  int n;
2  double sx, sy, d;
3  vector<Point> cir;
4  vector<double> radius;
5
6  int isIntersectCircleToCircle(Point c1, double r1, Point c2, double r2)
7  {
8      double dis = c1.distTo(c2);
9      return sign(dis - (r1 + r2)) <= 0;
10 }
11
12 void getRange(double x, Point &c, double r, double &retl, double &retr)
13 {
14     double tmp = sqrt(max(r * r - (c.x - x) * (c.x - x), 0.0));
15     retl = c.y - tmp; retr = c.y + tmp;
16 }
17
18 int checkInLine(double x)
19 {
20     double minR = INF, maxL = -INF;
21     double tmpl, tmpr;
22     for(int i = 0; i < n; ++ i) {
23         if (sign(cir[i].x + radius[i] - x) < 0 || sign(cir[i].x - radius[i] - x) > 0)
24             return false;
25         getRange(x, cir[i], radius[i], tmpl, tmpr);
26         maxL = max(tmpl, maxL);
27         minR = min(tmpr, minR);
28         if (maxL > minR) return false;
29     }
30     return true;
31 }
32
33 int shouldGoLeft(double x)
34 {
35     if (checkInLine(x)) return 2;
36     int onL = 0, onR = 0;
37     for(int i = 0; i < n; ++ i) {
38         if (sign(cir[i].x + radius[i] - x) < 0) onL = 1;
39         if (sign(cir[i].x - radius[i] - x) > 0) onR = 1;
40     }
41     if (onL && onR) return -1;
42     if (onL) return 1;
43     if (onR) return 0;
44
```

```
45      double minR = INF, maxL = -INF, tmpl, tmpr;
46      int idMinR, idMaxL;
47
48      for(int i = 0; i < n; ++ i) {
49          getRange(x, cir[i], radius[i], tmpl, tmpr);
50          if (tmpr < minR) {
51              minR = tmpr;
52              idMinR = i;
53          }
54          if (tmpl > maxL) {
55              maxL = tmpl;
56              idMaxL = i;
57          }
58      }
59      if (! isIntersectCircleToCircle(cir[idMinR], radius[idMinR], cir[idMaxL], radius[
            idMaxL]))
60          return -1;
61      Point p1, p2;
62      intersectionCircleToCircle(cir[idMinR], radius[idMinR], cir[idMaxL], radius[
            idMaxL], p1, p2);
63      return (p1.x < x);
64  }
65
66  int hasIntersectionCircles()
67  {
68      double l = -INF, r = INF, mid;
69      for(int i = 0; i < 100; ++ i) {
70          mid = (l + r) * 0.5;
71          int tmp = shouldGoLeft(mid);
72          if (tmp < 0) return 0;
73          if (tmp == 2) return 1;
74          if (tmp) r = mid;
75          else l = mid;
76      }
77      mid = (l + r) * 0.5;
78      return checkInLine(mid);
79  }
```

1.11 圆与三角形的交面积

```
1  #include<cstring>
2  #include<cstdio>
3  #include<algorithm>
4  #include<cmath>
5  using namespace std;
6  const double eps = 1e-12, PI = acos(-1.);
7  int sign(double x)
8  {
```

```
9       return x < -eps?-1:(x > eps?1:0);
10  }
11  struct triple
12  {
13      double x, y, z;
14      triple(){}
15      triple(const double & _x, const double & _y, const double & _z) : x(_x), y(_y), z
            (_z){}
16      void scan() {scanf("%lf%lf%lf", &x, &y, &z);}
17      void print(char ch) {printf("%lf %lf %lf%c", x, y, z, ch);}
18      double sqrlen() const {return x * x + y * y + z * z;}
19      double len() const {return sqrt(sqrlen());}
20  } p1, p2, p3, dir, co;
21  double sign(triple x)
22  {
23      return sign(x.x) == 0?(sign(x.y) == 0?sign(x.z):sign(x.y)):sign(x.x);
24  }
25  triple operator + (const triple & a, const triple & b)
26  {
27      return triple(a.x + b.x, a.y + b.y, a.z + b.z);
28  }
29  triple operator - (const triple & a, const triple & b)
30  {
31      return triple(a.x - b.x, a.y - b.y, a.z - b.z);
32  }
33  triple operator * (const double & a, const triple & b)
34  {
35      return triple(a * b.x, a * b.y, a * b.z);
36  }
37  triple operator * (const triple & a, const triple & b)
38  {
39      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
            );
40  }
41  double operator % (const triple & a, const triple & b)
42  {
43      return a.x * b.x + a.y * b.y + a.z * b.z;
44  }
45
46  double t, ans, r;
47  double fix(double x)
48  {
49      if(x > 1) return 1;
50      else if(x < -1) return -1;
51      else return x;
52  }
53  double calc(triple pa, triple pb)
54  {
55          if(pa.len() < pb.len()) swap(pa, pb);
```

```
56          if(pb.len() < eps) return 0;
57          double a, b, c, B, C, sinB, cosB, sinC, cosC, S, h, theta;
58          a = pb.len();
59          b = pa.len();
60          c = (pb - pa).len();
61          cosB = fix(pb % (pb - pa) / a / c);
62          sinB = fix((pb * (pb - pa)).len() / a / c);
63          B = acos(cosB);
64          cosC =  fix(pa % pb / a / b);
65          sinC = fix((pa * pb).len() / a / b);
66          C = acos(cosC);
67          if(a > r)
68          {
69              S = C / 2 * r * r;
70              h = a * b * sinC / c;
71              if(h < r and B < PI / 2) S -= (acos(h / r) * r * r - h * sqrt(r * r - h *
                    h));
72          }else if(b > r)
73          {
74              theta = PI - B - asin(fix(sinB / r * a));
75              S = .5 * a * r * sin(theta) + (C - theta) / 2 * r * r;
76          }else
77              S = .5 * sinC * a * b;
78          //printf("%lf\n", S);
79          return S;
80  }
81  int main()
82  {
83      p1.scan();
84      p2.scan();
85      p3.scan();
86      dir = (p2 - p1) * (p3 - p1);
87      double t = dir % p1 / (dir % dir);
88      co = t * dir;
89      //co.print('\n');
90      if(co.sqrlen() > 10000)
91      {
92          printf("0\n");
93          return 0;
94      }
95      r = sqrt(10000 - co.sqrlen());
96      p1 = p1 - co;
97      p2 = p2 - co;
98      p3 = p3 - co;
99      double ans = 0;
100     ans += calc(p1, p2) * sign(p1 * p2);
101     ans += calc(p2, p3) * sign(p2 * p3);
102     ans += calc(p3, p1) * sign(p3 * p1);
103     printf("%.10f\n", fabs(ans));
```

```
104     fclose(stdin);
105     return 0;
106 }
```

1.12 圆与圆的交并面积

```
1  #include<cstring>
2  #include<cstdio>
3  #include<algorithm>
4  #include<cmath>
5  #include<vector>
6  using namespace std;
7  double pi = acos(-1.0), eps = 1e-12;
8  double sqr(const double & a) {return a * a;}
9  double ans[1111];
10 int sign(const double & a) {return a > eps?1:a < -eps?-1:0;}
11 int n, cnt;
12 struct couple
13 {
14     double x, y;
15     couple(){}
16     couple(const double & _x, const double & _y) : x(_x), y(_y){}
17     void scan() {scanf("%lf%lf", &x, &y);}
18     double sqrlen() {return sqr(x) + sqr(y);}
19     double len() {return sqrt(sqrlen());}
20     couple rev() {return couple(y, -x);}
21     couple zoom(const double & d) {double lambda = d / len(); return couple(lambda *
           x, lambda * y);}
22 } dvd;
23 double atan2(const couple & x) {return atan2(x.y, x.x);}
24 couple operator - (const couple & a, const couple & b)
25 {
26     return couple(a.x - b.x, a.y - b.y);
27 }
28 couple operator + (const couple & a, const couple & b)
29 {
30     return couple(a.x + b.x, a.y + b.y);
31 }
32 double operator * (const couple & a, const couple & b)
33 {
34     return a.x * b.y - a.y * b.x;
35 }
36 couple operator * (const double & a, const couple & b)
37 {
38     return couple(a * b.x, a * b.y);
39 }
40 struct circle
41 {
42     double r; couple o;
```

```
43      circle(){}
44      void scan() {o.scan(); scanf("%lf", &r);}
45  } cir[1111];
46  struct arc
47  {
48      double theta;
49      int delta;
50      couple p;
51      arc(const double & _theta, const couple & _p, int _d) :theta(_theta), p(_p),
            delta(_d){}
52  };
53  bool operator < (const arc & a, const arc & b) {return a.theta < b.theta;}
54  vector<arc> vec;
55  void psh(const double t1, const couple p1, const double t2, const couple p2)
56  {
57      if(t1 < t2)
58      {
59          vec.push_back(arc(t1, p1, 1));
60          vec.push_back(arc(t2, p2, -1));
61      }else
62      {
63          vec.push_back(arc(t1, p1, 1));
64          vec.push_back(arc(pi, dvd, -1));
65          vec.push_back(arc(-pi, dvd, 1));
66          vec.push_back(arc(t2, p2, -1));
67      }
68  }
69  int main()
70  {
71      freopen("cirut.in", "r", stdin);
72
73      scanf("%d", &n);
74      for(int i = 1; i <= n; i++) cir[i].scan(), ans[i] = 0;
75      for(int i = 1; i <= n; i++)
76      {
77          vec.clear();
78          dvd = cir[i].o - couple(cir[i].r, 0);
79          vec.push_back(arc(-pi, dvd, 1));
80          for(int j = 1; j <= n; j++) if(j != i)
81          {
82              double d = (cir[j].o - cir[i].o).sqrlen();
83              if(d <= sqr(cir[j].r - cir[i].r))
84              {
85                  if(cir[i].r < cir[j].r)
86                      psh(-pi, dvd, pi, dvd);
87              }else if(d < sqr(cir[j].r + cir[i].r))
88              {
89                  double lambda = 0.5 * (1 + (sqr(cir[i].r) - sqr(cir[j].r))/d);
90                  couple cp = cir[i].o + lambda * (cir[j].o - cir[i].o);
```

```
91                  couple frm = cp + (cir[j].o - cir[i].o).rev().zoom(sqrt(sqr(cir[i].r)
                        - (cp - cir[i].o).sqrlen()));
92                  couple to = cp - (cir[j].o - cir[i].o).rev().zoom(sqrt(sqr(cir[i].r)
                       - (cp - cir[i].o).sqrlen()));
93                  psh(atan2(frm - cir[i].o), frm, atan2(to - cir[i].o), to);
94              }
95          }
96          sort(vec.begin() + 1, vec.end());
97          vec.push_back(arc(pi, dvd, -1));
98          cnt = 0;
99          for(int j = 0; j + 1 < vec.size(); j++)
100         {
101             cnt += vec[j].delta;
102             double theta = vec[j + 1].theta - vec[j].theta;
103             ans[cnt] += sqr(cir[i].r) * (theta - sin(theta)) * 0.5;
104             ans[cnt] += vec[j].p * vec[j + 1].p * 0.5;
105         }
106     }
107     ans[n + 1] = 0;
108     for(int i = 1; i <= n; i++) printf("[%d] = %.3lf\n", i, ans[i] - ans[i + 1]);
109     fclose(stdin);
110     return 0;
111 }
```

1.13 Farmland

1.13.1 Logic_IU

```
1  #include<cstdio>
2  #include<cstring>
3  #include<vector>
4  #include<cmath>
5  #include<iostream>
6  #include<algorithm>
7
8  using namespace std;
9
10 #define foreach(e, x) for(__typeof(x.begin()) e = x.begin(); e != x.end(); ++ e)
11
12 typedef long long LL;
13 typedef unsigned long long ULL;
14 typedef pair<int, int> PII;
15
16 const int N = 200 + 10;
17
18 struct Point
19 {
20     double x, y;
21     Point() {}
```

```
22      Point(double _x, double _y) {
23          x = _x; y = _y;
24      }
25      Point operator - (const Point &that) const {
26          return Point(x - that.x, y - that.y);
27      }
28      double det(const Point &that) const {
29          return x * that.y - y * that.x;
30      }
31      double alpha() {
32          return atan2(y, x);
33      }
34      void read() {
35          scanf("%lf%lf", &x, &y);
36      }
37  };
38
39  int n, m;
40  int vis[N][N];
41  int prev[N][N];
42  int in[N];
43  Point point[N], o;
44  vector<int> adj[N];
45  vector<int> sqn;
46
47  int cmp(const int &u, const int &v)
48  {
49      Point pu = point[u] - o, pv = point[v] - o;
50      return pu.alpha() < pv.alpha();
51  }
52
53  double calcArea(vector<int> &ps)
54  {
55      double area = 0;
56      for(int i = 0; i < (int)ps.size(); ++ i) {
57          int j = i == (int)ps.size() - 1 ? 0 : i + 1;
58          area += point[ps[i]].det(point[ps[j]]);
59      }
60      return area;
61  }
62
63  void dfs(int u, int v)
64  {
65      if (vis[u][v]) return;
66      vis[u][v] = true;
67      sqn.push_back(u);
68      int w = prev[u][v];
69      dfs(v, w);
70  }
```

```
71
72  void solve()
73  {
74      cin >> n;
75      for(int i = 0; i < n; ++ i) {
76          adj[i].clear();
77      }
78      int u, v;
79      for(int i = 0; i < n; ++ i) {
80          cin >> u; -- u;
81          point[u].read();
82          int tot; cin >> tot;
83          for(int j = 0; j < tot; ++ j) {
84              scanf("%d", &v); -- v;
85              adj[u].push_back(v);
86          }
87      }
88      cin >> m;
89
90      for(int i = 0; i < n; ++ i) {
91          o = point[i];
92          sort(adj[i].begin(), adj[i].end(), cmp);
93      }
94
95      for(int i = 0; i < n; ++ i) {
96          for(int j = 0; j < (int)adj[i].size(); ++ j) {
97              vis[i][adj[i][j]] = false;
98              int tmp = j == 0 ? adj[i].size() - 1 : j - 1;
99              prev[adj[i][j]][i] = adj[i][tmp];
100         }
101     }
102
103     int ret = 0;
104     for(int i = 0; i < n; ++ i) {
105         for(int j = 0; j < (int)adj[i].size(); ++ j) {
106             int v = adj[i][j];
107             if (vis[i][v]) continue;
108             sqn.clear();
109             dfs(i, v);
110             int flag = true;
111             if (calcArea(sqn) > 0) {
112                 memset(in, 0, sizeof in);
113                 for(int k = 0; k < (int)sqn.size(); ++ k) {
114                     v = sqn[k];
115                     if (in[v]) {
116                         flag = false;
117                         break;
118                     }
119                     in[v] = true;
```

```
120                 }
121                 if ((int)sqn.size() == m)
122                     ret += flag;
123             }
124         }
125     }
126     cout << ret << endl;
127 }
128
129 int main()
130 {
131     int T; cin >> T;
132     for(int i = 1; i <= T; ++ i) {
133         solve();
134     }
135     return 0;
136 }
```

1.13.2 tEJtM

```
1  #include<cstring>
2  #include<cstdio>
3  #include<cmath>
4  #include<vector>
5  #include<algorithm>
6  #include<map>
7  using namespace std;
8  double eps = 1e-12;
9  struct couple
10 {
11     int x, y;
12     couple(){}
13     couple(int _x, int _y) : x(_x), y(_y) {}
14     void scan() {scanf("%d%d", &x, &y);}
15 } a[222], c;
16 int operator * (const couple & x, const couple & y) {return x.x * y.y - x.y * y.x;}
17 int operator % (const couple & x, const couple & y) {return x.x * y.x + x.y * y.y;}
18 couple operator - (const couple & x, const couple & y) {return couple(x.x - y.x, x.y
       - y.y);}
19 int k, n, x, y, ans, x1, Q, id[222];
20 bool flag, flag1;
21 int area;
22 vector<int> gen;
23 vector<int> vec[222];
24 vector<bool> f[222];
25 map<int, int> mp;
26 struct Polar
27 {
28     couple o;
29     Polar(const couple & _o) : o(_o){}
```

```
30      bool operator () (const couple & a, const couple & b)
31      {
32          return atan2((a - o).y, (a - o).x) < atan2((b - o).y, (b - o).x);
33      }
34      bool operator () (int x, int y)
35      {
36          return this->operator () (a[x], a[y]);
37      }
38  };
39  int main()
40  {
41      freopen("01.in", "r", stdin);
42      scanf("%d", &Q);
43      for(int qq = 1; qq <= Q; qq++)
44      {
45          scanf("%d", &n);
46          mp.clear();
47          for(int i = 1; i <= n; i++)
48          {
49              scanf("%d", &id[i]);
50              a[i].scan();
51              vec[i].clear();
52              f[i].clear();
53              scanf("%d", &x);
54              for(int j = 0; j < x; j++) {scanf("%d", &y); vec[i].push_back(y); f[i].
                    push_back(true);}
55          }
56          scanf("%d", &k);
57          for(int i = 1; i <= n; i++)
58              sort(vec[i].begin(), vec[i].end(), Polar(a[i]));
59          ans = 0;
60          for(int i = 1; i <= n; i++) for(int j = 0; j < vec[i].size(); j++) if(f[i][j
                ])
61          {
62              x = i;
63              y = j;
64              gen.clear();
65              for(;;){
66                  f[x][y] = false;
67                  gen.push_back(x);
68                  x1 = x;
69                  x = vec[x][y];
70                  y = lower_bound(vec[x].begin(), vec[x].end(), x1, Polar(a[x])) - vec[
                        x].begin();
71                  y = (y + 1) % vec[x].size();
72                  if(x == i and y == j) break;//break when x == i and y == j, not only
                        x == i!
73              }
74              if(gen.size() != k) continue;
```

```
75              gen.push_back(gen.front());
76              area = 0;
77              for(int i = 0; i + 1 < gen.size(); i++) area += a[gen[i]] * a[gen[i +
                    1]];//printf("%d %d %lf\n", gen[i], gen[i + 1], a[gen[i]] * a[gen[i +
                    1]]);}
78              if(area >= 0) continue;
79              flag = true;
80              for(int i = 0; i + 2 < gen.size(); i++) {for(int j = i + 1; j + 1 < gen.
                    size(); j++) if(gen[i] == gen[j]) {flag = false; break;} if(flag ==
                    false) break;}
81              ans += flag;
82          }
83          printf("%d\n", ans);
84      }
85      fclose(stdin);
86      return 0;
87  }
```

Chapter 2

三维几何

2.1 三维几何 Seraphim Version

2.1.1 基本操作

```
1  //vlen(point3 P):length of vector; zero(double x):if fabs(x)<eps) return true;
2  double vlen(point3 p);
3  //平面法向量
4  point3 pvec(point3 s1,point3 s2,point3 s3){return det((s1-s2),(s2-s3));}
5  //共线check
6  int dots_inline(point3 p1,point3 p2,point3 p3){
7      return vlen(det(p1-p2,p2-p3))<eps;}
8  //共平面check
9  int dots_onplane(point3 a,point3 b,point3 c,point3 d){
10     return zero(dot(pvec(a,b,c),d-a));}
11 //在线段上check(end point inclusive)
12 int dot_online_in(point3 p,line3 l)
13 int dot_online_in(point3 p,point3 l1,point3 l2){return zero(vlen(det(p-l1,p-l2)))&&(
       l1.x-p.x)*(l2.x-p.x)<eps&&(l1.y-p.y)*(l2.y-p.y)<eps&&(l1.z-p.z)*(l2.z-p.z)<eps;
           }
14 //在线段上check(end point exclusive)
15 int dot_online_ex(point3 p,line3 l)
16 int dot_online_ex(point3 p,point3 l1,point3 l2){ return dot_online_in(p,l1,l2)&&(!
       zero(p.x-l1.x)||!zero(p.y-l1.y)||!zero(p.z-l1.z))&&(!zero(p.x-l2.x)||!zero(p.y-l2.
       y)||!zero(p.z-l2.z));
17 }
18 //一个点是否在三角形里check(inclusive)
19 int dot_inplane_in(point3 p,plane3 s)
20 int dot_inplane_in(point3 p,point3 s1,point3 s2,point3 s3){
21     return zero(vlen(det(s1-s2,s1-s3))-vlen(det(p-s1,p-s2))-
22         vlen(det(p-s2,p-s3))-vlen(det(p-s3,p-s1)));
23 }
24 //一个点是否在三角形里check(exclusive)
25 int dot_inplane_ex(point3 p,plane3 s)
26 int dot_inplane_ex(point3 p,point3 s1,point3 s2,point3 s3){
```

```
27      return dot_inplane_in(p,s1,s2,s3)&&vlen(det(p-s1,p-s2))>eps&&
28          vlen(det(p-s2,p-s3))>eps&&vlen(det(p-s3,p-s1))>eps;
29  }
30  //check if two point and a segment in one plane have the same side
31  int same_side(point3 p1,point3 p2,point3 l1,point3 l2)
32  int same_side(point3 p1,point3 p2,line3 l){
33      return dot(det(l.a-l.b,p1-l.b),det(l.a-l.b,p2-l.b))>eps;
34  }
35  //check if two point and a segment in one plane have the opposite side
36  int opposite_side(point3 p1,point3 p2,point3 l1,point3 l2)
37  int opposite_side(point3 p1,point3 p2,line3 l){
38      return dot(det(l.a-l.b,p1-l.b), det(l.a-l.b,p2-l.b))<-eps;
39  }
40  //check if two point is on the same side of a plane
41  int same_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3)
42  int same_side(point3 p1,point3 p2,plane3 s){
43      return dot(pvec(s),p1-s.a)*dot(pvec(s),p2-s.a)>eps;
44  }
45  //check if two point is on the opposite side of a plane
46  int opposite_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3)
47  int opposite_side(point3 p1,point3 p2,plane3 s){
48      return dot(pvec(s),p1-s.a)*dot(pvec(s),p2-s.a)<-eps;
49  }
50  //check if two straight line is parallel
51  int parallel(point3 u1,point3 u2,point3 v1,point3 v2)
52  int parallel(line3 u,line3 v){    return vlen(det(u.a-u.b,v.a-v.b))<eps; }
53  //check if two plane is parallel
54  int parallel(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)
55  int parallel(plane3 u,plane3 v){return vlen(det(pvec(u),pvec(v)))<eps;}
56  //check if a plane and a line is parallel
57  int parallel(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
58  int parallel(line3 l,plane3 s){ return zero(dot(l.a-l.b,pvec(s))); }
59  //check if two line is perpendicular
60  int perpendicular(point3 u1,point3 u2,point3 v1,point3 v2)
61  int perpendicular(line3 u,line3 v){return zero(dot(u.a-u.b,v.a-v.b)); }
62  //check if two plane is perpendicular
63  int perpendicular(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)
64  int perpendicular(plane3 u,plane3 v){    return zero(dot(pvec(u),pvec(v))); }
65  //check if plane and line is perpendicular
66  int perpendicular(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
67  int perpendicular(line3 l,plane3 s){return vlen(det(l.a-l.b,pvec(s)))<eps;}
68  //check 两条线段是否有交点(end point inclusive)
69  int intersect_in(point3 u1,point3 u2,point3 v1,point3 v2)
70  int intersect_in(line3 u,line3 v){
71      if (!dots_onplane(u.a,u.b,v.a,v.b)) return 0;
72      if (!dots_inline(u.a,u.b,v.a)||!dots_inline(u.a,u.b,v.b))
73          return !same_side(u.a,u.b,v)&&!same_side(v.a,v.b,u);
74      return dot_online_in(u.a,v)||dot_online_in(u.b,v)||
75  dot_online_in(v.a,u)||dot_online_in(v.b,u);
```

```
76 }
77 //check 两条线段是否有交点(end point exclusive)
78 int intersect_ex(point3 u1,point3 u2,point3 v1,point3 v2)
79 int intersect_ex(line3 u,line3 v){
80     return dots_onplane(u.a,u.b,v.a,v.b)&&opposite_side(u.a,u.b,v)&&
81 opposite_side(v.a,v.b,u);
82 }
83 //线段和三角形是否有交点check(end point and border inclusive)
84 int intersect_in(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
85 int intersect_in(line3 l,plane3 s){
86     return !same_side(l.a,l.b,s)&&!same_side(s.a,s.b,l.a,l.b,s.c)&&
87         !same_side(s.b,s.c,l.a,l.b,s.a)&&!same_side(s.c,s.a,l.a,l.b,s.b);
88 }
89 //线段和三角形是否有交点check(end point and border exclusive)
90 int intersect_ex(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
91 int intersect_ex(line3 l,plane3 s){
92     return opposite_side(l.a,l.b,s)&&opposite_side(s.a,s.b,l.a,l.b,s.c)&&
        opposite_side(s.b,s.c,l.a,l.b,s.a)&&opposite_side(s.c,s.a,l.a,l.b,s.b);}
93 //calculate the intersection of two line
94 //Must you should ensure they are co-plane and not parallel
95 point3 intersection(point3 u1,point3 u2,point3 v1,point3 v2)
96 point3 intersection(line3 u,line3 v){
97     point3 ret=u.a;
98     double t=((u.a.x-v.a.x)*(v.a.y-v.b.y)-(u.a.y-v.a.y)*(v.a.x-v.b.x))
99             /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
100 ret+=(u.b-u.a)*t;     return ret;
101 }
102 //calculate the intersection of plane and line
103 point3 intersection(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
104 point3 intersection(line3 l,plane3 s){
105     point3 ret=pvec(s);
106 double t=(ret.x*(s.a.x-l.a.x)+ret.y*(s.a.y-l.a.y)+ret.z*(s.a.z-l.a.z))/
107         (ret.x*(l.b.x-l.a.x)+ret.y*(l.b.y-l.a.y)+ret.z*(l.b.z-l.a.z));
108     ret=l.a + (l.b-l.a)*t;      return ret;
109 }
110 //calculate the intersection of two plane
111 bool intersection(plane3 pl1 , plane3 pl2 , line3 &li) {
112     if (parallel(pl1,pl2)) return false;
113     li.a=parallel(pl2.a,pl2.b, pl1) ? intersection(pl2.b,pl2.c, pl1.a,pl1.b,pl1.c) :
        intersection(pl2.a,pl2.b, pl1.a,pl1.b,pl1.c);
114     point3 fa; fa=det(pvec(pl1),pvec(pl2));
115     li.b=li.a+fa;
116     return true;
117 }
118 //distance from point to line
119 double ptoline(point3 p,point3 l1,point3 l2)
120 double ptoline(point3 p,line3 l){
121     return vlen(det(p-l.a,l.b-l.a))/distance(l.a,l.b);}
122 //distance from point to plane
```

```
123 double ptoplane(point3 p,plane3 s){
124     return fabs(dot(pvec(s),p-s.a))/vlen(pvec(s));}
125 double ptoplane(point3 p,point3 s1,point3 s2,point3 s3)
126 //distance between two line 当      u,平行时有问题v
127 double linetoline(line3 u,line3 v){
128     point3 n=det(u.a-u.b,v.a-v.b); return fabs(dot(u.a-v.a,n))/vlen(n);
129 }
130 double linetoline(point3 u1,point3 u2,point3 v1,point3 v2)
131 //cosine value of the angle formed by two lines
132 double angle_cos(line3 u,line3 v){
133     return dot(u.a-u.b,v.a-v.b)/vlen(u.a-u.b)/vlen(v.a-v.b);
134 }
135 double angle_cos(point3 u1,point3 u2,point3 v1,point3 v2)
136 //cosine value of the angle formed by two planes
137 double angle_cos(plane3 u,plane3 v){
138     return dot(pvec(u),pvec(v))/vlen(pvec(u))/vlen(pvec(v));}
139 double angle_cos(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3)
140 //cosine value of the angle formed by plane and line
141 double angle_sin(line3 l,plane3 s){
142     return dot(l.a-l.b,pvec(s))/vlen(l.a-l.b)/vlen(pvec(s));}
143 double angle_sin(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
```

2.1.2 三维旋转操作

a 点绕 Ob 向量，逆时针旋转弧度 angle,sin(angle),cos(angle) 先求出来，减少精度问题。

```
1  point e1,e2,e3;
2  point Rotate( point a, point b, double angle ){
3      b.std();//单位化，注意不能为b(0,0,0)
4      e3=b; double lens=a*e3;//dot(a,e3)
5      e1=a -  e3*lens; if (e1.len()>(1e-8)) e1.std(); else return a;
6      e2=e1/e3; //det(e1,e3)
7      double x1,y1,x,y;
8      y1=a*e1; x1=a*e2;
9      x=x1*cos(angle) - y1*sin(angle);
10     y=x1*sin(angle) + y1*cos(angle);
11     return e3*lens + e1*y + e2*x;
12 }
```

2.1.3 三维凸包随机增量

```
1 #include<iostream>
2 #include<cstring>
3 #include<algorithm>
4 #include<vector>
5 #include<cmath>
6 #include<cstdio>
7 using namespace std;
8 #define SIZE(X) (int(X.size()))
9 #define Eps 1E-8
```

```
10 #define PI 3.14159265358979323846264338327950288
11 inline int Sign(double x) {
12     return x < -Eps ? -1 : (x > Eps ? 1 : 0);
13 }
14 inline double Sqrt(double x) {
15     return x < 0 ? 0 : sqrt(x);
16 }
17 struct Point {
18     double x, y, z;
19     Point() {
20         x = y = z = 0;
21     }
22     Point(double x, double y, double z): x(x), y(y), z(z) {}
23     bool operator <(const Point &p) const {
24         return x < p.x || x == p.x && y < p.y || x == p.x && y == p.y && z < p.z;
25     }
26     bool operator ==(const Point &p) const {
27         return Sign(x - p.x) == 0 && Sign(y - p.y) == 0 && Sign(z - p.z) == 0;
28     }
29     Point operator +(const Point &p) const {
30         return Point(x + p.x, y + p.y, z + p.z);
31     }
32     Point operator -(const Point &p) const {
33         return Point(x - p.x, y - p.y, z - p.z);
34     }
35     Point operator *(const double k) const {
36         return Point(x * k, y * k, z * k);
37     }
38     Point operator /(const double k) const {
39         return Point(x / k, y / k, z / k);
40     }
41     Point cross(const Point &p) const {
42         return Point(y * p.z - z * p.y, z * p.x - x * p.z, x * p.y - y * p.x);
43     }
44     double dot(const Point &p) const {
45         return x * p.x + y * p.y + z * p.z;
46     }
47     double norm() {
48         return dot(*this);
49     }
50     double length() {
51         return Sqrt(norm());
52     }
53     void read() {
54         scanf("%lf%lf%lf", &x, &y, &z);
55     }
56     void write() {
57         printf("(%.10f, %.10f, %.10f)\n", x, y, z);
58     }
```

```
59  };
60  int mark[1005][1005];
61  Point info[1005];
62  int n, cnt;
63  double mix(const Point &a, const Point &b, const Point &c) {
64      return a.dot(b.cross(c));
65  }
66  double area(int a, int b, int c) {
67      return ((info[b] - info[a]).cross(info[c] - info[a])).length();
68  }
69  double volume(int a, int b, int c, int d) {
70      return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]);
71  }
72  struct Face {
73      int a, b, c;
74      Face() {}
75      Face(int a, int b, int c): a(a), b(b), c(c) {}
76      int &operator [](int k) {
77          if (k == 0) return a;
78          if (k == 1) return b;
79          return c;
80      }
81  };
82  vector <Face> face;
83  inline void insert(int a, int b, int c) {
84      face.push_back(Face(a, b, c));
85  }
86  void add(int v) {
87      vector <Face> tmp;
88      int a, b, c;
89      cnt ++;
90      for (int i = 0; i < SIZE(face); i ++) {
91          a = face[i][0];
92          b = face[i][1];
93          c = face[i][2];
94          if (Sign(volume(v, a, b, c)) < 0)
95              mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] =
96                  mark[a][c] = cnt;
97          else
98              tmp.push_back(face[i]);
99      }
100     face = tmp;
101     for (int i = 0; i < SIZE(tmp); i ++) {
102         a = face[i][0];
103         b = face[i][1];
104         c = face[i][2];
105         if (mark[a][b] == cnt) insert(b, a, v);
106         if (mark[b][c] == cnt) insert(c, b, v);
107         if (mark[c][a] == cnt) insert(a, c, v);
```

```
108     }
109 }
110 int Find() {
111     for (int i = 2; i < n; i ++) {
112         Point ndir = (info[0] - info[i]).cross(info[1] - info[i]);
113         if (ndir == Point())
114             continue;
115         swap(info[i], info[2]);
116         for (int j = i + 1; j < n; j ++)
117             if (Sign(volume(0, 1, 2, j)) != 0) {
118                 swap(info[j], info[3]);
119                 insert(0, 1, 2);
120                 insert(0, 2, 1);
121                 return 1;
122             }
123     }
124     return 0;
125 }
126 int main() {
127     double ans, ret;
128     int Case;
129     for (scanf("%d", &Case); Case; Case --) {
130         scanf("%d", &n);
131         for (int i = 0; i < n; i ++)
132             info[i].read();
133         sort(info, info + n);
134         n = unique(info, info + n) - info;
135         face.clear();
136         random_shuffle(info, info + n);
137         ans = ret = 0;
138         if (Find()) {
139             memset(mark, 0, sizeof(mark));
140             cnt = 0;
141             for (int i = 3; i < n; i ++) add(i);
142             int first = face[0][0];
143             for (int i = 0; i < SIZE(face); i ++) {
144                 ret += area(face[i][0], face[i][1], face[i][2]);
145                 ans += fabs(volume(first, face[i][0], face[i][1], face[i][2]));
146             }
147             ans /= 6;
148             ret /= 2;
149         }
150         printf("%.3f␣%.3f\n", ret, ans);
151     }
152     return 0;
153 }
```

2.2 基本操作 tEJtM Version

```
1  struct triple
2  {
3      double x, y, z;
4      triple(){}
5      triple(const double & _x, const doule & _y, const double & _z) : x(_x), y(_y), z(
           _z);
6      double len2() const {return x * x + y * y + z * z;}
7      double len() const {return sqrt(len2());}
8      void scan() {scanf("%lf%lf%lf", &x, &y, &z);}
9  };
10 triple operator + (const triple & a, const triple & b)
11 {return triple(a.x + b.x, a.y + b.y, a.z + b.z);}
12 triple operator - (const triple & a, const triple & b)
13 {return triple(a.x - b.x, a.y - b.y, a.z - b.z);}
14 triple operator * (const double & a, const triple & b)
15 {return triple(a * b.x, a * b.y, a * b.z);}
16 triple operator * (const triple & a, const triple & b)
17 {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);}
18 double operator % (const triple & a, const triple & b)
19 {return a.x * b.x + a.y * b.y + a.z * b.z;}
20 struct line
21 {
22     triple s, d;
23     line(){}
24     line(const triple & _s, const triple & _d) : s(_s), d(_d){}
25     triple operator () const {return d;}
26 };
27 struct plane
28 {
29     triple a, n;
30     plane()
31     plane(const triple & _a, const triple & _n) :a(_a), n(_n){}
32     plane(const triple & _a, const triple & _b, const triple & _c)
33     {a = _a; n = (b - a) * (b - c);}//valid for non-colinear _a, _b, _c.
34 };
35 const double eps = 1e-12;
36 int sign(const double & x)
37 {return x < -eps?-1:x > eps?1:0;}
38 //判断 相交平行垂直
39
40 bool parallel(const triple & a, const triple & b)//向量平行
41 {
42     return sign((a * b).len2()) == 0;
43 }
44 bool orthogonal(const triple & a, const triple & b)//向量垂直
45 {
46     return sign(a % b) == 0;
47 }
48 bool dis(const triple & a, const line & b)//点到直线距离 无正负
```

```
49 {
50     return ((a - b.s) * b.d).len() / b.d.len();
51 }//做三角形求高法
52 bool perpendicular(const triple & a, const triple & b)//点到直线垂线
53 {
54     return line(a, b.d * (b.s - a) * b.d);
55 }//若是点在直线上则求出来方向为的直线0
56 bool on(const triple & a, const line & b)//点在直线上
57 {
58     return parallel(a - b.s, b.d);
59 }
60 bool parallel(const line & a, const line & b)//直线平行
61 {
62     return parallel(a.d, b.d);
63 }//重合也算平行
64 bool orthogonal(const line & a, const line & b)//直线垂直
65 {
66     return orthogonal(a.d, b.d);
67 }
68 double dis(const line & a, const line & b)//直线间最近距离 无正负
69 {
70     triple nor = a.d * b.d;
71     if(sign(nor.len2()) == 0) return dis(b.s, a);//平行直线距离
72     else return nor % (a.s - b.s) / nor.len();//不平行直线最近距离
73 }
74 bool intersect(const line & a, const line & b)//直线相交距离为:0
75 {
76     return sign(dis(a, b)) == 0;
77 }
78
79 triple intersection(const line & a, const line & b)//直线交点 假设不平行且交点存在
80 {
81     double lambda = (b.s - a.s) * b.d % (a.d * b.d) / (a.d * b.d).len2();
82     return a.s + lambda * a.d;
83 }//若是不平行交点也不存在则求出来的是条直线最近距离两点中第一条直线上的那个点2
84
85 bool onRay(const triple & a, const line & b) //点在射线上 含
86 {
87     return on(a, b) and sign((a - b.s) % (b.d)) >= 0;
88 }
89 bool onSeg(const triple & a, const triple & b, const triple & c)//点在线段上 含
90 {
91     return onRay(a, line(b, c - b)) and onRay(a, line(c, b - c));
92 }//有了上面个函数就可以处理射线2 线段和直线相互之间存在交点的问题了
93
94 //下面是有关平面的算法
95 int dis(const triple & a, const plane & b)//点到平面距离 有正负
96 {
97     return (a - b.a) % b.n;
```

```
98  }
99  int above(const triple & a, const plane & b)//above_1 on_0 under_-1
100 {
101     return sign(dis(a, b));//和法向同向的算是平面的上面
102 }
103 bool parallel(const line & a, const plane & b)//直线和平面平行 重合也算平行
104 {
105     return orthogonal(a.d, b.n);
106 }
107
108 bool parallel(const plane & a, const plane & b)//平面和平面平行
109 {
110     return parallel(a.n, b.n);
111 }
112 bool intersect(const line & a, const plane & b)//直线和平面相交?
113 {
114     return !parallel(a, b) or sign(a.s * b.n - b.s * b.n) == 0; //不平行或在平面内部
115 }
116 triple intersection(const line & a, const plane & b)//直线和平面的交点
117 {
118     double lambda = b.n % (b.a - a.s) / (a.d % b.n);
119     return a.s + lambda * a.d;
120 }
121 line intersection(const plane & a, const plane & b)//平面和平面的交线
122 {
123     return line(intersection(line(a.a, a.n * b.n * a.n)), a.n * b.n);
124 }
```

2.3 点类 + 三维凸包 N^3+ 凸包求重心

```
1  #include<cstring>
2  #include<cstdio>
3  #include<vector>
4  #include<algorithm>
5  #include<set>
6  #include<string>
7  #include<cmath>
8  using namespace std;
9  multiset<string> st;
10 struct triple
11 {
12     double x, y, z;
13     double sqrlen() {return x * x + y * y + z * z;}
14     double len() {return sqrt(sqrlen());}
15     triple(){}
16     triple(double _x, double _y, double _z) : x(_x), y(_y), z(_z){}
17 } a[111];
18 char name[111][211];
19 bool flag, ext[111];
```

```
20 int l, real[111], cnt, n, f[111][111];
21 struct plane
22 {
23     int a[3];
24     plane(int _x, int _y, int _z)
25     {
26         a[0] = _x;
27         a[1] = _y;
28         a[2] = _z;
29     }
30     int & operator [] (int x)
31     {
32         return a[x];
33     }
34 };
35 vector<plane> surf;
36 triple operator * (const triple & a, const triple & b)
37 {
38     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
       );
39 }
40 triple operator * (const double & lambda, const triple & b)
41 {
42     return triple(lambda * b.x, lambda * b.y, lambda * b.z);
43 }
44 double operator % (const triple & a, const triple & b)
45 {
46     return a.x * b.x + a.y * b.y + a.z * b.z;
47 }
48 triple operator - (const triple & a, const triple & b)
49 {
50     return triple(a.x - b.x, a.y - b.y, a.z - b.z);
51 }
52 triple operator + (const triple & a, const triple & b)
53 {
54     return triple(a.x + b.x, a.y + b.y, a.z + b.z);
55 }
56 double volume(const triple & o, int j)//volume of a tetrahedron := {a point and a
    triangle undersurface}
57 {
58     return (a[surf[j][0]] - o) * (a[surf[j][1]] - o) % (a[surf[j][2]] - o);//can be
       negative
59 }
60 double volume(int i, int j)
61 {
62     return volume(a[i], j);
63 }
64 double above(int i, int j) {return volume(i, j) > 0;}//point above plane
65 double on(int i, int j) {return volume(i, j) == 0;}//point on plane
```

```
66  void print(const triple & x, char ch)
67  {
68      printf("(%lf, %lf, %lf)%c", x.x, x.y, x.z, ch);
69  }
70  double dis(const triple & o, int j)//point to plane
71  {
72      return fabs(volume(o, j) / ((a[surf[j][1]] - a[surf[j][0]]) * (a[surf[j][2]] - a[
            surf[j][0]])).len());
73  }
74  int main()
75  {
76      double ans = 0;
77      for(int cv = 1; cv <= 2; cv++)
78      {
79          scanf("%d", &n);
80          for(int i = 1; i <= n; i++)
81          {
82              scanf("%lf%lf%lf", &a[i].x, &a[i].y, &a[i].z);
83          }
84          //->degenerate checking
85          flag = false;
86          for(int i = 3; i <= n; i++)
87          {
88              if(((a[1] - a[i]) * (a[2] - a[i])).sqrlen() != 0)
89              {
90                  swap(a[3], a[i]);
91                  swap(real[i], real[3]);
92                  for(int j = 4; j <= n; j++)
93                  {
94                      if((a[1] - a[j]) * (a[2] - a[j]) % (a[3] - a[j]) != 0)
95                      {
96                          swap(a[4], a[j]);
97                          swap(real[4], real[j]);
98                          flag = true;
99                          break;
100                     }
101                 }
102                 break;
103             }
104         }
105         /*if(flag == false)
106         {
107             //degenerate!
108         }else
109         {*/
110         //->convex polyhedra
111         memset(f, 0, sizeof(f));
112         surf.clear();
113         surf.push_back(plane(1, 2, 3));
```

```
114             surf.push_back(plane(3, 2, 1));
115             for(int i = 4; i <= n; i++)
116             {
117                 vector<plane> tmp;
118                 for(int j = 0; j < surf.size(); j++)
119                     if(above(i, j))
120                     {
121                         for(int d = 0; d < 3; d++)
122                         {
123                             f[surf[j][d]][surf[j][(d + 2) % 3]] = i;
124                         }
125                     }else
126                     {
127                         tmp.push_back(surf[j]);
128                     }
129                 surf = tmp;
130                 for(int j = surf.size() - 1; j >= 0; j--)
131                 {
132                     for(int d = 0; d < 3; d++)
133                         if(f[surf[j][d]][surf[j][(d + 1) % 3]] == i) surf.push_back(plane
                            (surf[j][(d + 1) % 3], surf[j][d], i));
134                 }
135             }
136             //end convex polyhedra, result := surf
137             //->centre of gravity
138             double svol = 0;
139             triple qc(0, 0, 0);
140             for(int i = 0; i < surf.size(); i++)
141             {
142                 double vol1 = volume(1, i);
143                 qc = qc + (vol1 / 4) * (a[1] + a[surf[i][0]] + a[surf[i][1]] + a[surf[i
                        ][2]]);
144                 svol += vol1;
145             }
146             qc = (1 / svol) * qc;
147             double mn = 1e9;
148             for(int i = 0; i < surf.size(); i++)
149             {
150                 mn = min(mn, dis(qc, i));
151             }
152             ans += mn;
153             //end centre of gravity
154             //}
155         }
156         printf("%.5f\n", ans);
157         fclose(stdin);
158         return 0;
159     }
```

2.4 三维旋转

```
1  //sgu265
2  #include<cstring>
3  #include<cstdio>
4  #include<cmath>
5  #include<algorithm>
6  using namespace std;
7  const double pi = acos(-1.0);
8  int n, m; char ch1; bool flag;
9  double a[4][4], s1, s2, x, y, z, w, b[4][4], c[4][4];
10 double sqr(double x)
11 {
12     return x*x;
13 }
14 int main()
15 {
16     scanf("%d\n", &n);
17     memset(b, 0, sizeof(b));
18     b[0][0] = b[1][1] = b[2][2] = b[3][3] = 1;//initial matrix
19     for(int i = 1; i <= n; i++)
20     {
21         scanf("%c", &ch1);
22         if(ch1 == 'T')
23         {
24             //plus each coordinate by a number (x, y, z)
25             scanf("%lf %lf %lf\n", &x, &y, &z);
26             memset(a, 0, sizeof(a));
27             a[0][0] = 1; a[3][0] = x;
28             a[1][1] = 1; a[3][1] = y;
29             a[2][2] = 1; a[3][2] = z;
30             a[3][3] = 1;
31         }else if(ch1 == 'S')
32         {
33             //multiply each coordinate by a number (x, y, z)
34             scanf("%lf %lf %lf\n", &x, &y, &z);
35             memset(a, 0, sizeof(a));
36             a[0][0] = x;
37             a[1][1] = y;
38             a[2][2] = z;
39             a[3][3] = 1;
40         }else
41         {
42             //rotate in a clockwise about the ray from the origin through (x, y, z);
43             scanf("%lf %lf %lf %lf\n", &x, &y, &z, &w);
44             w = w*pi/180;
45             memset(a, 0, sizeof(a));
46             s1 = x*x+y*y+z*z;
47             a[3][3] = 1;
```

```
48                  a[0][0] = ((y*y+z*z)*cos(w)+x*x)/s1;
49                  a[0][1] = x*y*(1-cos(w))/s1+z*sin(w)/sqrt(s1);
50                  a[0][2] = x*z*(1-cos(w))/s1-y*sin(w)/sqrt(s1);
51                  a[1][0] = x*y*(1-cos(w))/s1-z*sin(w)/sqrt(s1);
52                  a[1][1] = ((x*x+z*z)*cos(w)+y*y)/s1;
53                  a[1][2] = y*z*(1-cos(w))/s1+x*sin(w)/sqrt(s1);
54                  a[2][0] = x*z*(1-cos(w))/s1+y*sin(w)/sqrt(s1);
55                  a[2][1] = y*z*(1-cos(w))/s1-x*sin(w)/sqrt(s1);
56                  a[2][2] = ((x*x+y*y)*cos(w)+z*z)/s1;
57              }
58              memset(c, 0, sizeof(c));
59              for(int i = 0; i < 4; i++)
60                  for(int j = 0; j < 4; j++)
61                      for(int k = 0; k < 4; k++)
62                          c[i][j] += b[i][k]*a[k][j];
63              memcpy(b, c, sizeof(c));
64          }
65          scanf("%d", &m);
66          for(int i = 1; i <= m; i++)
67          {
68              scanf("%lf%lf%lf", &x, &y, &z);//initial vector
69              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]);
70          }
71          return 0;
72      }
```

2.5 最小球覆盖

```
1   #include<iostream>
2   #include<cstring>
3   #include<algorithm>
4   #include<cstdio>
5   #include<cmath>
6
7   using namespace std;
8
9   const int eps = 1e-8;
10
11  struct Tpoint
12  {
13      double x, y, z;
14  };
15
16  int npoint, nouter;
17
18  Tpoint pt[200000], outer[4],res;
19  double radius,tmp;
20  inline double dist(Tpoint p1, Tpoint p2) {
```

```
21      double dx=p1.x-p2.x, dy=p1.y-p2.y, dz=p1.z-p2.z;
22      return ( dx*dx + dy*dy + dz*dz );
23  }
24  inline double dot(Tpoint p1, Tpoint p2) {
25      return p1.x*p2.x + p1.y*p2.y + p1.z*p2.z;
26  }
27  void ball() {
28      Tpoint q[3]; double m[3][3], sol[3], L[3], det;
29      int i,j;
30      res.x = res.y = res.z = radius = 0;
31      switch ( nouter ) {
32          case 1: res=outer[0]; break;
33          case 2:
34                  res.x=(outer[0].x+outer[1].x)/2;
35                  res.y=(outer[0].y+outer[1].y)/2;
36                  res.z=(outer[0].z+outer[1].z)/2;
37                  radius=dist(res, outer[0]);
38                  break;
39          case 3:
40                  for (i=0; i<2; ++i ) {
41                      q[i].x=outer[i+1].x-outer[0].x;
42                      q[i].y=outer[i+1].y-outer[0].y;
43                      q[i].z=outer[i+1].z-outer[0].z;
44                  }
45                  for (i=0; i<2; ++i) for(j=0; j<2; ++j)
46                      m[i][j]=dot(q[i], q[j])*2;
47                  for (i=0; i<2; ++i ) sol[i]=dot(q[i], q[i]);
48                  if (fabs(det=m[0][0]*m[1][1]-m[0][1]*m[1][0])<eps)
49                      return;
50                  L[0]=(sol[0]*m[1][1]-sol[1]*m[0][1])/det;
51                  L[1]=(sol[1]*m[0][0]-sol[0]*m[1][0])/det;
52                  res.x=outer[0].x+q[0].x*L[0]+q[1].x*L[1];
53                  res.y=outer[0].y+q[0].y*L[0]+q[1].y*L[1];
54                  res.z=outer[0].z+q[0].z*L[0]+q[1].z*L[1];
55                  radius=dist(res, outer[0]);
56                  break;
57          case 4:
58                  for (i=0; i<3; ++i) {
59                      q[i].x=outer[i+1].x-outer[0].x;
60                      q[i].y=outer[i+1].y-outer[0].y;
61                      q[i].z=outer[i+1].z-outer[0].z;
62                      sol[i]=dot(q[i], q[i]);
63                  }
64                  for (i=0;i<3;++i)
65                      for(j=0;j<3;++j) m[i][j]=dot(q[i],q[j])*2;
66                  det= m[0][0]*m[1][1]*m[2][2]
67                      + m[0][1]*m[1][2]*m[2][0]
68                      + m[0][2]*m[2][1]*m[1][0]
69                      - m[0][2]*m[1][1]*m[2][0]
```

```
70                      - m[0][1]*m[1][0]*m[2][2]
71                      - m[0][0]*m[1][2]*m[2][1];
72                  if ( fabs(det)<eps ) return;
73                  for (j=0; j<3; ++j) {
74                      for (i=0; i<3; ++i) m[i][j]=sol[i];
75                      L[j]=( m[0][0]*m[1][1]*m[2][2]
76                              + m[0][1]*m[1][2]*m[2][0]
77                              + m[0][2]*m[2][1]*m[1][0]
78                              - m[0][2]*m[1][1]*m[2][0]
79                              - m[0][1]*m[1][0]*m[2][2]
80                              - m[0][0]*m[1][2]*m[2][1]
81                           ) / det;
82                      for (i=0; i<3; ++i)
83                          m[i][j]=dot(q[i], q[j])*2;
84                  }
85                  res=outer[0];
86                  for (i=0; i<3; ++i ) {
87                      res.x += q[i].x * L[i];
88                      res.y += q[i].y * L[i];
89                      res.z += q[i].z * L[i];
90                  }
91                  radius=dist(res, outer[0]);
92      }
93  }
94  void minball(int n) {
95      ball();
96      //printf("(%.3lf,%.3lf,%.3lf) %.3lf\n", res.x,res.y,res.z,radius);
97      if ( nouter<4 )
98          for (int i=0; i<n; ++i)
99              if (dist(res, pt[i])-radius>eps) {
100                 outer[nouter]=pt[i];
101                 ++nouter;
102                 minball(i);
103                 --nouter;
104                 if (i>0) {
105                     Tpoint Tt = pt[i];
106                     memmove(&pt[1], &pt[0], sizeof(Tpoint)*i);
107                     pt[0]=Tt;
108                 }
109             }
110 }
111 void solve()
112 {
113     for (int i=0;i<npoint;i++) scanf("%lf%lf%lf",&pt[i].x,&pt[i].y,&pt[i].z);
114     random_shuffle(pt, pt + npoint);
115     radius=-1;
116     for (int i=0;i<npoint;i++){
117         if (dist(res,pt[i])-radius>eps){
118             nouter=1;
```

```
119                 outer[0]=pt[i];
120                 minball(i);
121             }
122         }
123         printf("%.5f\n",sqrt(radius));
124     }
125     int main(){
126         for( ; cin >> npoint && npoint; )
127             solve();
128         return 0;
129     }
```

Chapter 3

图论

3.1 Dijkstra

求 s 到其他点的最短路

```
1  int used[MAX_N], dis[MAX_N];
2  void dijstra(int s) {
3      fill(dis, dis + N, INF); dis[s] = 0;
4      priority_queue<pair<int, int> > que;
5      que.push(make_pair(-dis[s], s));
6      while (!que.empty()) {
7          int u = que.top().second; que.pop();
8          if (used[u]) continue;
9          used[u] = true;
10         foreach(e, E[u])
11             if (dis[u] + e->w < dis[e->t]) {
12                 dis[e->t] = dis[u] + e->w;
13                 que.push(make_pair(-dis[e->t], e->t));
14             }
15     }
16 }
```

3.2 最大流

iSAP 算法求 S 到 T 的最大流，点数为 cntN，边表存储在 *E[] 中

```
1  struct Edge
2  {
3      int t, c;
4      Edge *n, *r;
5  } *E[MAX_V], edges[MAX_M], *totEdge;
6
7  Edge* makeEdge(int s, int t, int c)
8  {
9      Edge *e = totEdge ++;
```

```
10      e->t = t; e->c = c; e->n = E[s];
11      return E[s] = e;
12  }
13
14  void addEdge(int s, int t, int c)
15  {
16      Edge *p = makeEdge(s, t, c), *q = makeEdge(t, s, 0);
17      p->r = q; q->r = p;
18  }
19
20  int maxflow()
21  {
22      static int  cnt     [MAX_V];
23      static int  h       [MAX_V];
24      static int  que     [MAX_V];
25      static int  aug     [MAX_V];
26      static Edge *cur    [MAX_V];
27      static Edge *prev   [MAX_V];
28      fill(h, h + cntN, cntN);
29      memset(cnt, 0, sizeof cnt);
30      int qt = 0, qh = 0; h[T] = 0;
31      for(que[qt ++] = T; qh < qt; ) {
32          int u = que[qh ++];
33          ++ cnt[h[u]];
34          for(Edge *e = E[u]; e; e = e->n)
35              if (e->r->c && h[e->t] == cntN) {
36                  h[e->t] = h[u] + 1;
37                  que[qt ++] = e->t;
38              }
39      }
40      memcpy(cur, E, sizeof E);
41      aug[S] = INF; Edge *e;
42      int flow = 0, u = S;
43      while (h[S] < cntN) {
44          for(e = cur[u]; e; e = e->n)
45              if (e->c && h[e->t] + 1 == h[u])
46                  break;
47          if (e) {
48              int v = e->t;
49              cur[u] = prev[v] = e;
50              aug[v] = min(aug[u], e->c);
51              if ((u = v) == T) {
52                  int by = aug[T];
53                  while (u != S) {
54                      Edge *p = prev[u];
55                      p->c -= by;
56                      p->r->c += by;
57                      u = p->r->t;
58                  }
```

```
59                  flow += by;
60              }
61          } else {
62              if (!-- cnt[h[u]]) return flow;
63              h[u] = cntN;
64              for(e = E[u]; e; e = e->n)
65                  if (e->c && h[u] > h[e->t] + 1)
66                      h[u] = h[e->t] + 1, cur[u] = e;
67              ++ cnt[h[u]];
68              if (u != S) u = prev[u]->r->t;
69          }
70      }
71      return flow;
72  }
```

3.3 上下界流

上下界无源汇可行流: 不用添 T—>S，判断是否流量平衡
上下界无源汇可行流: 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

3.3.1 上下界无源汇可行流

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <cstring>
4  #include <ctime>
5  #include <cmath>
6  #include <iostream>
7  #include <algorithm>
8
9  using namespace std;
10
11 struct {
12         int x, y, down, up, what;
13 } a[100001];
14
15 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;
16
17 inline void makelist(int x, int y, int z, int q){
18     where[++l] = y;
19     v[l] = z;
20     what[l] = q;
```

```
21      next[l] = first[x];
22      first[x] = l;
23  }
24
25  bool check(){
26      memset(dist, 0, sizeof(dist));
27      dist[DS] = 1; c[1] = DS;
28      for (int k = 1, l = 1; l <= k; l++)
29      {
30          int m = c[l];
31          if (m == DT)
32          {
33              len = dist[m] + 1;
34              return(true);
35          }
36          for (int x = first[m]; x; x = next[x])
37              if (v[x] && !dist[where[x]]) dist[where[x]] = dist[m] + 1, c[++k] = where
                    [x];
38      }
39      return(false);
40  }
41
42  inline void dinic(int now){
43      if (now == DT)
44      {
45          int Minflow = 1 << 30;
46          for (; now != DS; now = pre[now]) Minflow = min(Minflow, v[way[now]]);
47          ans += Minflow;
48          for (now = DT; now != DS; now = pre[now])
49          {
50              v[way[now]] -= Minflow;
51              v[way[now] ^ 1] += Minflow;
52              if (!v[way[now]]) len = dist[now];
53          }
54          return;
55      }
56      for (int x = first[now]; x; x = next[x])
57          if (v[x] && dist[now] + 1 == dist[where[x]])
58          {
59              pre[where[x]] = now;
60              way[where[x]] = x;
61              dinic(where[x]);
62              if (dist[now] >= len) return;
63              len = dist[DT] + 1;
64          }
65      dist[now] = -1;
66  }
67
68  int main(){
```

```
69    //  freopen("194.in", "r", stdin);
70    //  freopen("194.out", "w", stdout);
71      scanf("%d%d", &n, &m);
72      memset(flow, 0, sizeof(flow));
73      for (int i = 1; i <= m; i++)
74      {
75          scanf("%d%d%d%d", &a[i].x, &a[i].y, &a[i].down, &a[i].up);
76          flow[a[i].y] += a[i].down;
77          flow[a[i].x] -= a[i].down;
78      }
79      cnt = 0;
80      memset(first, 0, sizeof(first)); l = 1;
81      S = 1; T = n; DS = 0; DT = n + 1; cnt = 0;
82      for (int i = 1; i <= n; i++)
83          if (flow[i] > 0) makelist(DS, i, flow[i], 0), makelist(i, DS, 0, 0), cnt +=
                flow[i];
84                      else makelist(i, DT, abs(flow[i]), 0), makelist(DT, i, 0, 0);
85   //   makelist(T, S, 1 << 30, 0); makelist(S, T, 0, 0);
86      for (int i = 1; i <= m; i++) makelist(a[i].x, a[i].y, a[i].up - a[i].down, i),
87                                   makelist(a[i].y, a[i].x, 0, i);
88      ans = 0;
89      for (; check();) dinic(DS);
90      if (ans != cnt) printf("NO\n");
91      else
92      {
93          printf("YES\n");
94          for (int i = 3; i <= l; i += 2)
95              if (what[i]) out[what[i]] = v[i];
96          for (int i = 1; i <= m; i++) printf("%d\n", a[i].down + out[i]);
97      }
98  }
```

3.3.2 上下界最大流

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <cstring>
4  #include <ctime>
5  #include <cmath>
6  #include <iostream>
7  #include <algorithm>
8
9  using namespace std;
10
11 int n, m, S, T, DS, DT, a[1001], first[1501], next[100001], where[100001], v[100001],
       what[100001],
12 l, c[1501], dist[1501], len, pre[1501], way[1501], flow[1501], out[100001], tot, cnt,
       ans;
13
14 inline void makelist(int x, int y, int z, int q){
```

```
15      where[++l] = y;
16      v[l] = z;
17      what[l] = q;
18      next[l] = first[x];
19      first[x] = l;
20  }
21
22  bool check(int S, int T){
23      memset(dist, 0, sizeof(dist));
24      c[1] = S; dist[S] = 1;
25      for (int k = 1, l = 1; l <= k; l++)
26      {
27          int m = c[l];
28          if (m == T)
29          {
30              len = dist[m] + 1;
31              return(true);
32          }
33          for (int x = first[m]; x; x = next[x])
34              if (v[x] && !dist[where[x]])
35              {
36                  dist[where[x]] = dist[m] + 1;
37                  c[++k] = where[x];
38              }
39      }
40      return(false);
41  }
42
43  inline void dinic(int now, int S, int T){
44      if (now == T)
45      {
46          int Minflow = 1 << 30;
47          for (; now != S; now = pre[now]) Minflow = min(Minflow, v[way[now]]);
48          ans += Minflow;
49          for (now = T; now != S; now = pre[now])
50          {
51              v[way[now]] -= Minflow;
52              v[way[now] ^ 1] += Minflow;
53              if (!v[way[now]]) len = dist[now];
54          }
55          return;
56      }
57      for (int x = first[now]; x; x = next[x])
58          if (v[x] && dist[where[x]] == dist[now] + 1)
59          {
60              pre[where[x]] = now;
61              way[where[x]] = x;
62              dinic(where[x], S, T);
63              if (dist[now] >= len) return;
```

```
64              len = dist[T] + 1;
65          }
66      dist[now] = -1;
67  }
68
69  int main(){
70  //      freopen("3229.in", "r", stdin);
71  //      freopen("3229.out", "w", stdout);
72      for (;;)
73      {
74          if (scanf("%d%d", &n, &m) != 2) return 0;
75          memset(first, 0, sizeof(first)); l = 1;
76          memset(flow, 0, sizeof(flow));
77          S = 0; T = n + m + 1; DS = T + 1; DT = DS + 1;
78          for (int i = 1; i <= m; i++)
79          {
80              scanf("%d", &a[i]);
81              flow[S] -= a[i]; flow[i] += a[i];
82              makelist(S, i, 1 << 30, 0); makelist(i, S, 0, 0);
83          }
84          tot = 0;
85          for (int i = 1; i <= n; i++)
86          {
87              int C, D;
88              scanf("%d%d", &C, &D);
89              if (D) makelist(m + i, T, D, 0), makelist(T, m + i, 0, 0);
90              for (int j = 1; j <= C; j++)
91              {
92                  int idx, x, y;
93                  scanf("%d%d%d", &idx, &x, &y);
94                  idx++;
95                  flow[idx] -= x; flow[i + m] += x;
96                  out[++tot] = x;
97                  if (y != x) makelist(idx, i + m, y - x, tot), makelist(i + m, idx, 0,
                        tot);
98              }
99          }
100         cnt = 0;
101         for (int i = S; i <= T; i++)
102             if (flow[i] > 0) makelist(DS, i, flow[i], 0), makelist(i, DS, 0, 0), cnt
                    += flow[i];
103             else makelist(i, DT, abs(flow[i]), 0), makelist(DT, i, 0, 0);
104         makelist(T, S, 1 << 30, 0); makelist(S, T, 0, 0);
105         ans = 0;
106         for (; check(DS, DT);) dinic(DS, DS, DT);
107         if (ans != cnt)
108         {
109             printf("-1\n\n");
110             continue;
```

```
111         }
112         else
113         {
114             v[l] = v[l - 1] = 0;
115             for (; check(S, T);) dinic(S, S, T);
116             printf("%d\n", ans);
117             for (int i = 3; i <= l; i += 2)
118                 if (what[i]) out[what[i]] += v[i];
119             for (int i = 1; i <= tot; i++) printf("%d\n", out[i]);
120             printf("\n");
121         }
122     }
123 }
```

3.3.3 上下界最小流

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <cstring>
4  #include <ctime>
5  #include <cmath>
6  #include <iostream>
7  #include <algorithm>
8
9  using namespace std;
10
11 struct {
12         int x, y, down, up;
13 } a[10001];
14 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;
15
16 int read(){
17     char ch;
18     for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
19     int cnt = 0;
20     for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
21     return(cnt);
22 }
23
24 inline void makelist(int x, int y, int z, int q){
25     where[++l] = y;
26     v[l] = z;
27     what[l] = q;
28     next[l] = first[x];
29     first[x] = l;
30 }
31
32 inline void makemap(){
```

```
33      memset(first, 0, sizeof(first)); l = 1;
34      S = 1, T = n, DS = 0, DT = n + 1; cnt = 0;
35      for (int i = 1; i <= n; i++)
36          if (flow[i] > 0) makelist(DS, i, flow[i], 0), makelist(i, DS, 0, 0), cnt +=
                flow[i];
37          else makelist(i, DT, abs(flow[i]), 0), makelist(DT, i, 0, 0);
38      for (int i = 1; i <= m; i++) makelist(a[i].x, a[i].y, a[i].up - a[i].down, i),
39                                   makelist(a[i].y, a[i].x, 0, i);
40  }
41
42  bool check(){
43      memset(dist, 0, sizeof(dist));
44      dist[DS] = 1; c[1] = DS;
45      for (int k = 1, l = 1; l <= k; l++)
46      {
47          int m = c[l];
48          if (m == DT)
49          {
50              len = dist[m] + 1;
51              return(true);
52          }
53          for (int x = first[m]; x; x = next[x])
54              if (v[x] && !dist[where[x]]) dist[where[x]] = dist[m] + 1, c[++k] = where
                    [x];
55      }
56      return(false);
57  }
58
59  inline void dinic(int now){
60      if (now == DT)
61      {
62          int Minflow = 1 << 30;
63          for (; now != DS; now = pre[now]) Minflow = min(Minflow, v[way[now]]);
64          ans += Minflow;
65          for (now = DT; now != DS; now = pre[now])
66          {
67              v[way[now]] -= Minflow;
68              v[way[now] ^ 1] += Minflow;
69              if (!v[way[now]]) len = dist[now];
70          }
71          return;
72      }
73      for (int x = first[now]; x; x = next[x])
74          if (dist[where[x]] == dist[now] + 1 && v[x])
75          {
76              pre[where[x]] = now;
77              way[where[x]] = x;
78              dinic(where[x]);
79              if (dist[now] >= len) return;
```

```
80              len = dist[DT] + 1;
81          }
82      dist[now] = -1;
83  }
84
85  inline void network(){
86      for (; check();) dinic(DS);
87  }
88
89  int main(){
90     // freopen("176.in", "r", stdin);
91     // freopen("176.out", "w", stdout);
92      scanf("%d%d", &n, &m);
93      memset(flow, 0, sizeof(flow));
94      for (int i = 1; i <= m; i++)
95      {
96          a[i].x = read(), a[i].y = read(), a[i].up = read();
97          int status = read();
98          if (status) a[i].down = a[i].up;
99          else a[i].down = 0;
100         flow[a[i].y] += a[i].down;
101         flow[a[i].x] -= a[i].down;
102     }
103     makemap();
104     ans = 0;
105     network();
106     makelist(T, S, 1 << 30, 0); makelist(S, T, 0, 0);
107     network();
108     if (ans != cnt)
109     {
110         printf("Impossible\n");
111         return 0;
112     }
113     printf("%d\n", v[l]);
114     for (int i = 3; i <= l; i += 2)
115         if (what[i]) out[what[i]] = v[i];
116     for (int i = 1; i <= m; i++)
117     {
118         printf("%d", a[i].down + out[i]);
119         if (i != m) printf("␣");
120         else printf("\n");
121     }
122 }
```

3.3.4 上下界有源汇可行流

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <cstring>
4  #include <ctime>
```

```
5  #include <cmath>
6  #include <iostream>
7  #include <algorithm>
8
9  using namespace std;
10
11 int test, n, m, Q, first[501], a1[201], a2[201], flow[501], next[100001], where
       [100001], v[100001], len,
12 l, dist[501], c[501], up[201][201], down[201][201], S, T, DS, DT, ans, out[201][201],
       pre[501], way[501];
13
14 inline void makelist(int x, int y, int z){
15     where[++l] = y;
16     v[l] = z;
17     next[l] = first[x];
18     first[x] = l;
19 }
20
21 bool check(){
22     memset(dist, 0, sizeof(dist));
23     dist[DS] = 1; c[1] = DS;
24     for (int k = 1, l = 1; l <= k; l++)
25     {
26         int m = c[l];
27         if (m == DT)
28         {
29             len = dist[m] + 1;
30             return(true);
31         }
32         for (int x = first[m]; x; x = next[x])
33             if (v[x] && !dist[where[x]])
34             {
35                 dist[where[x]] = dist[m] + 1;
36                 c[++k] = where[x];
37             }
38     }
39     return(false);
40 }
41
42 inline void dinic(int now){
43     if (now == DT)
44     {
45         int Minflow = 1 << 30;
46         for (; now != DS; now = pre[now]) Minflow = min(Minflow, v[way[now]]);
47         ans += Minflow;
48         for (now = DT; now != DS; now = pre[now])
49         {
50             v[way[now]] -= Minflow;
51             v[way[now] ^ 1] += Minflow;
```

```
52              if (!v[way[now]]) len = dist[now];
53          }
54          return;
55      }
56      for (int x = first[now]; x; x = next[x])
57          if (v[x] && dist[now] + 1 == dist[where[x]])
58          {
59              pre[where[x]] = now;
60              way[where[x]] = x;
61              dinic(where[x]);
62              if (dist[now] >= len) return;
63              len = dist[DT] + 1;
64          }
65      dist[now] = -1;
66  }
67
68  int main(){
69    //  freopen("2396.in", "r", stdin);
70    //  freopen("2396.out", "w", stdout);
71      scanf("%d", &test);
72      for (int uu = 1; uu <= test; uu++)
73      {
74          scanf("%d%d", &n, &m);
75          for (int i = 1; i <= n; i++) scanf("%d", &a1[i]);
76          for (int i = 1; i <= m; i++) scanf("%d", &a2[i]);
77          memset(up, 127, sizeof(up));
78          memset(down, 0, sizeof(down));
79          scanf("%d", &Q);
80          for (int i = 1; i <= Q; i++)
81          {
82              int x, y, z;
83              char str[2];
84              scanf("%d%d%s%d", &x, &y, str, &z);
85              int L1, L2, R1, R2;
86              if (x == 0) L1 = 1, R1 = n;
87              else L1 = R1 = x;
88              if (y == 0) L2 = 1, R2 = m;
89              else L2 = R2 = y;
90              for (int j = L1; j <= R1; j++)
91                  for (int k = L2; k <= R2; k++)
92                      if (str[0] == '>') down[j][k] = max(down[j][k], z + 1);
93                      else if (str[0] == '<') up[j][k] = min(up[j][k], z - 1);
94                      else down[j][k] = max(down[j][k], z), up[j][k] = min(up[j][k], z)
                            ;
95          }
96          bool ok = true;
97          for (int i = 1; i <= n && ok; i++)
98              for (int j = 1; j <= m; j++)
99                  if (down[i][j] > up[i][j])
```

```
100                 {
101                     ok = false;
102                     break;
103                 }
104         if (!ok)
105         {
106             printf("IMPOSSIBLE\n");
107             if (uu != test) printf("\n");
108             continue;
109         }
110         memset(flow, 0, sizeof(flow));
111         memset(first, 0, sizeof(first)); l = 1;
112         S = 0; T = n + m + 1;
113         for (int i = 1; i <= n; i++) flow[S] -= a1[i], flow[i] += a1[i];
114         for (int i = 1; i <= m; i++) flow[i + n] -= a2[i], flow[T] += a2[i];
115         for (int i = 1; i <= n; i++)
116             for (int j = 1; j <= m; j++)
117             {
118                 flow[i] -= down[i][j]; flow[j + n] += down[i][j];
119                 if (down[i][j] != up[i][j]) makelist(i, j + n, up[i][j] - down[i][j])
                        ,
120                                             makelist(j + n, i, 0);
121             }
122         DS = T + 1; DT = DS + 1;
123         int cnt = 0;
124         for (int i = S; i <= T; i++)
125             if (flow[i] > 0) makelist(DS, i, flow[i]), makelist(i, DS, 0), cnt +=
                    flow[i];
126             else if (flow[i] < 0) makelist(i, DT, abs(flow[i])), makelist(DT, i, 0);
127         makelist(T, S, 1 << 30); makelist(S, T, 0);
128         ans = 0;
129         for (; check();) dinic(DS);
130         if (ans != cnt)
131         {
132             printf("IMPOSSIBLE\n");
133             if (uu != test) printf("\n");
134             continue;
135         }
136         for (int i = 1; i <= n; i++)
137             for (int x = first[i]; x; x = next[x])
138                 if (where[x] >= n + 1 && where[x] <= n + m)
139                     down[i][where[x] - n] += v[x ^ 1];
140         for (int i = 1; i <= n; i++)
141             for (int j = 1; j <= m; j++)
142             {
143                 printf("%d", down[i][j]);
144                 if (j != m) printf("␣");
145                 else printf("\n");
146             }
```

```
147             if (uu != test) printf("\n");
148         }
149     }
```

3.4 费用流

3.4.1 Logic_IU+ 负费用路

注意图的初始化，费用和流的类型依题目而定

```
1   int flow, cost;
2
3   struct Edge
4   {
5       int t, c, w;
6       Edge *n, *r;
7   } *totEdge, edges[MAX_M], *E[MAX_V];
8
9   Edge* makeEdge(int s, int t, int c, int w)
10  {
11      Edge *e = totEdge ++;
12      e->t = t; e->c = c; e->w = w; e->n = E[s];
13      return E[s] = e;
14  }
15
16  void addEdge(int s, int t, int c, int w)
17  {
18      Edge *st = makeEdge(s, t, c, w), *ts = makeEdge(t, s, 0, -w);
19      st->r = ts; ts->r = st;
20  }
21
22  int SPFA()
23  {
24      static int que[MAX_V];
25      static int aug[MAX_V];
26      static int in[MAX_V];
27      static int dist[MAX_V];
28      static Edge *prev[MAX_V];
29      int qh = 0, qt = 0;
30
31      int u, v;
32      fill(dist, dist + cntN, INF); dist[S] = 0;
33      fill(in, in + cntN, 0); in[S] = true;
34      que[qt ++] = S; aug[S] = INF;
35      for( ; qh != qt; ) {
36          u = que[qh]; qh = (qh + 1) % MAX_N;
37          for(Edge *e = E[u]; e; e = e->n) {
38              if (! e->c) continue;
39              v = e->t;
40              if (dist[v] > dist[u] + e->w) {
```

```
41                      dist[v] = dist[u] + e->w;
42                      aug[v] = min(aug[u], e->c);
43                      prev[v] = e;
44                      if (! in[v]) {
45                          in[v] = true;
46                          if (qh != qt && dist[v] <= dist[que[qh]]) {
47                              qh = (qh - 1 + MAX_N) % MAX_N;
48                              que[qh] = v;
49                          } else {
50                              que[qt] = v;
51                              qt = (qt + 1) % MAX_N;
52                          }
53                      }
54                  }
55              }
56              in[u] = false;
57          }
58
59          if (dist[T] == INF) return false;
60          cost += dist[T] * aug[T];
61          flow += aug[T];
62          for(u = T; u != S; ) {
63              prev[u]->c -= aug[T];
64              prev[u]->r->c += aug[T];
65              u = prev[u]->r->t;
66          }
67          return true;
68      }
69
70      int minCostFlow()
71      {
72          flow = cost = 0;
73          while(SPFA());
74          return cost;
75      }
```

3.4.2 shytangyuan+ZKW

```
1   #include <cstdio>
2   #include <cstdlib>
3   #include <cstring>
4   #include <ctime>
5   #include <cmath>
6   #include <iostream>
7   #include <algorithm>
8
9   using namespace std;
10
```

```
11 int n, m, S, T, slk[1001], dist[1001], first[1001], l, c[1000001], next[1000001],
       where[1000001], ll[1000001], v[1000001];
12 bool b[1001];
13 long long ans1, ans2;
14
15 inline void makelist(int x, int y, int z, int p){
16     where[++l] = y;
17     ll[l] = z;
18     v[l] = p;
19     next[l] = first[x];
20     first[x] = l;
21 }
22
23 inline void spfa(){
24     memset(dist, 127, sizeof(dist));
25     memset(b, false, sizeof(b));
26     dist[T] = 0; c[1] = T;
27     for (int k = 1, l = 1; l <= k; l++)
28     {
29         int m = c[l];
30         b[m] = false;
31         for (int x = first[m]; x; x = next[x])
32             if (ll[x ^ 1] && dist[m] - v[x] < dist[where[x]])
33             {
34                 dist[where[x]] = dist[m] - v[x];
35                 if (!b[where[x]]) b[where[x]] = true, c[++k] = where[x];
36             }
37     }
38 }
39
40 int zkw_work(int now, int cap){
41     b[now] = true;
42     if (now == T)
43     {
44         ans1 += cap;
45         ans2 += (long long)cap * dist[S];
46         return(cap);
47     }
48     int Left = cap;
49     for (int x = first[now]; x; x = next[x])
50         if (ll[x] && !b[where[x]])
51             if (dist[now] == dist[where[x]] + v[x])
52             {
53                 int use = zkw_work(where[x], min(Left, ll[x]));
54                 ll[x] -= use; ll[x ^ 1] += use;
55                 Left -= use;
56                 if (!Left) return(cap);
57             }
```

```
58              else slk[where[x]] = min(slk[where[x]], dist[where[x]] + v[x] - dist[now])
                    ;
59      return(cap - Left);
60  }
61
62  bool relax(){
63      int Min = 1 << 30;
64      for (int i = 0; i <= T; i++)
65          if (!b[i]) Min = min(Min, slk[i]);
66      if (Min == 1 << 30) return(false);
67      for (int i = 0; i <= T; i++)
68          if (b[i]) dist[i] += Min;
69      return(true);
70  }
71
72  inline void zkw(){
73      ans1 = ans2 = 0;
74      spfa();    //hint memset(dist, 0, sizeof(dist)); if all values of edges are
            nonnegative
75      for (;;)
76      {
77          memset(slk, 127, sizeof(slk));
78          for (;;)
79          {
80              memset(b, false, sizeof(b));
81              if (!zkw_work(S, 1 << 30)) break;
82          }
83          if (!relax()) break;
84      }
85      printf("%I64d %I64d\n", ans1, ans2);
86  }
87
88  int main(){
89      scanf("%d%d", &n, &m);
90      S = 1; T = n;
91      memset(first, 0, sizeof(first)); l = 1;
92      for (int i = 1; i <= m; i++)
93      {
94          int x, y, z, q;
95          scanf("%d%d%d%d", &x, &y, &z, &q);
96          makelist(x, y, z, q); makelist(y, x, 0, -q);
97      }
98      zkw();
99  }
```

3.5 强联通分量

3.5.1 Logic_IU

N 个点的图求 SCC，totID 为时间标记，top 为栈顶，totCol 为强联通分量个数，注意初始化

```
int totID, totCol;
int col[MAX_N], low[MAX_N], dfn[MAX_N];
int top, stack[MAX_N], instack[MAX_N];

int tarjan(int u)
{
    low[u] = dfn[u] = ++ totID;
    instack[u] = true; stack[++ top] = u;

    int v;
    foreach(it, adj[u]) {
        v = it->first;
        if (dfn[v] == -1)
            low[u] = min(low[u], tarjan(v));
        else if (instack[v])
            low[u] = min(low[u], dfn[v]);
    }

    if (low[u] == dfn[u]) {
        do {
            v = stack[top --];
            instack[v] = false;
            col[v] = totCol;
        } while(v != u);
        ++ totCol;
    }
    return low[u];
}

void solve()
{
    totID = totCol = top = 0;
    fill(dfn, dfn + N, 0);
    for(int i = 0; i < N; ++ i)
        if (! dfn[i])
            tarjan(i);
}
```

3.5.2 shytangyuan+ 手写栈

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <cmath>
#include <iostream>
#include <algorithm>

```

```
 9  using namespace std;
10
11  int n, m, first[10001], father[10001], dfn[10001], low[10001], c[10001], pos[10001],
        todo[10001],
12  cnt, len, next[2000001], where[2000001], l, kuai, Max, color[10001], number;
13  bool b[10001];
14
15  int read(){
16      char ch;
17      for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
18      int cnt = 0;
19      for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
20      return(cnt);
21  }
22
23  inline void makelist(int x, int y){
24      where[++l] = y;
25      next[l] = first[x];
26      first[x] = l;
27  }
28
29  inline void tarjan(int S){
30      int now = S; todo[now] = first[now];
31      for (;;)
32      {
33          if (!now) return;
34          if (first[now] == todo[now])
35          {
36              b[now] = true;
37              dfn[now] = low[now] = ++cnt;
38              c[++len] = now; pos[now] = len;
39          }
40          int x = todo[now];
41          if (!x)
42          {
43              if (father[now])
44                  low[father[now]] = min(low[father[now]], low[now]);
45              int delta = -1;
46              if (father[now]) ++delta;
47              for (int x = first[now]; x; x = next[x])
48                  if (father[where[x]] == now)
49                      if (low[where[x]] >= dfn[now]) ++delta;
50              Max = max(Max, delta);
51              if (low[now] == dfn[now])
52              {
53                 ++number;
54                  for (int i = pos[now]; i <= len; i++) color[c[i]] = number;
55                  len = pos[now] - 1;
56              }
```

```
57              now = father[now];
58              continue;
59          }
60          todo[now] = next[todo[now]];
61          if (father[now] != where[x])
62              if (!b[where[x]])
63              {
64                  father[where[x]] = now;
65                  now = where[x];
66                  todo[now] = first[now];
67                  continue;
68              }
69              else if (!color[where[x]]) low[now] = min(low[now], dfn[where[x]]);
70      }
71  }
72
73  int main(){
74      // freopen("2117.in", "r", stdin);
75      // freopen("2117.out", "w", stdout);
76      for (;;)
77      {
78          n = read(); m = read();
79          if (!n && !m) return 0;
80          memset(first, 0, sizeof(first));
81          l = 0;
82          for (int i = 1; i <= m; i++)
83          {
84              int x = read() + 1, y = read() + 1;
85              makelist(x, y);
86              makelist(y, x);
87          }
88          memset(dfn, 0, sizeof(dfn));
89          memset(low, 0, sizeof(low));
90          memset(color, 0, sizeof(color));
91          memset(b, false, sizeof(b));
92          memset(father, 0, sizeof(father));
93          cnt = 0; len = 0;
94          Max = -(1 << 30);
95          kuai = 0; number = 0;
96          for (int i = 1; i <= n; i++)
97              if (!b[i]) tarjan(i), ++kuai;
98          printf("%d\n", kuai + Max);
99      }
100 }
```

3.6 KM

3.6.1 tEJtM

```
1  //sgu206
2  #include<cstring>
3  #include<cstdio>
4  #include<algorithm>
5  using namespace std;
6  struct recmap
7  {
8      int y, c;
9      recmap *next;
10 } *p, ma1[1000], *id1[1001];
11 int n, m, l1, l2, c[1001], i, x, y, z, g[401][401], gn, ll[1001], lr[1001], rm[1001],
       exd, t;
12 bool f[1001], vl[1001];
13 void build(int x, int y)
14 {
15     ma1[++l1].y = y;
16     ma1[l1].next = id1[x];
17     id1[x] = &ma1[l1];
18 }
19 bool dfs(int v)
20 {
21     if(v == y) return true;
22     f[v] = false;
23     for(recmap *p=id1[v];p;p=p->next) if(f[p->y] and dfs(p->y))
24     {
25         g[(p-ma1)>>1][i-n+1] = c[(p-ma1)>>1]-z;
26         return true;
27     }
28     return false;
29 }
30 int hgry(int v)
31 {
32
33     vl[v] = true;
34     for(int u = 1; u <= gn; u++)
35         if(f[u])
36         {
37             if((t=ll[v]+lr[u] - g[v][u])==0)
38             {
39                 f[u] = false;
40                 if(rm[u] == 0 or hgry(rm[u])) return rm[u] = v;
41             }else exd = min(exd, t);
42         }
43     return 0;
44 }
45 void KM()
46 {
47     memset(ll, 0x7f, sizeof(ll));
48     memset(lr, 0, sizeof(lr));
```

```
49      memset(rm, 0, sizeof(rm));
50      for(int i = 1; i < n; i++)
51      {
52          for(;;)
53          {
54              memset(vl, false, sizeof(vl));
55              memset(f, true, sizeof(f));
56              exd = 0x7fffffff;
57              if(hgry(i)) break;
58              for(int i = 1; i < n; i++) if(vl[i]) ll[i] -= exd;
59              for(int i = n; i <= m; i++) if(!f[i-n+1]) lr[i-n+1] += exd;
60          }
61      }
62  }
63  int main()
64  {
65      scanf("%d%d", &n, &m);
66      memset(id1, 0, sizeof(id1));
67      l1 = 1;
68      for(i = 1; i < n; i++)
69      {
70          scanf("%d%d%d", &x, &y, &c[i]);
71          build(x, y);
72          build(y, x);
73      }
74      memset(g, 0, sizeof(g));
75      for(i = n; i <= m; i++)
76      {
77          scanf("%d%d%d", &x, &y, &z);
78          c[i] = z;
79          memset(f, true, sizeof(f));
80          dfs(x);
81      }
82      gn = max(n-1, m-n+1);
83      KM();
84      for(int i = 1; i < n; i++) printf("%d\n", c[i]-ll[i]);
85      for(int i = n; i <= m; i++) printf("%d\n", c[i] + lr[i-n+1]);
86      return 0;
87  }
```

3.6.2 Logic_IU

求完备匹配的最大权匹配，建好的完全图用 w[][] 存储，点数为 N

```
1  const int MAX_N = 200 + 10;
2  const int INF = 1000000000;
3
4  int N, flag;
5  int w[MAX_N][MAX_N];
6  int fx[MAX_N], fy[MAX_N], lx[MAX_N], ly[MAX_N], slk[MAX_N], mat[MAX_N];
7
```

```
8  int DFS(int x) {
9      fx[x] = flag;
10     for(int i = 1; i <= N; ++ i)
11         if (lx[x] + ly[i] != w[x][i])
12             slk[i] = min(slk[i], lx[x] + ly[i] - w[x][i]);
13         else if (fy[i] != flag) {
14             fy[i] = flag;
15             if (mat[i] < 0 || DFS(mat[i])) {
16                 mat[i] = x;
17                 return true;
18             }
19         }
20     return false;
21 }
22
23 int KM() {
24     for(int i = 1; i <= N; ++ i) {
25         mat[i] = -1;
26         fx[i] = 0; fy[i] = 0;
27         ly[i] = 0; lx[i] = -INF;
28         for(int j = 1; j <= N; ++ j)
29             lx[i] = max(lx[i], w[i][j]);
30     }
31     for(int now = 1; now <= N; ++ now) {
32         ++ flag; for(int i = 1; i <= N; ++ i) slk[i] = INF;
33         while (! DFS(now)) {
34             int p(INF); for(int i = 1; i <= N; ++ i)
35                 if (fy[i] != flag) p = min(p, slk[i]);
36             for(int i = 1; i <= N; ++ i) {
37                 if (fx[i] == flag) lx[i] -= p;
38                 if (fy[i] == flag) ly[i] += p;
39                 slk[i] = INF;
40             }
41             ++ flag;
42         }
43     }
44     long long ans = 0;
45     for(int i = 1; i <= N; ++ i) ans += lx[i], ans += ly[i];
46     return ans;
47 }
```

3.6.3 shytangyuan+ 邻接阵

```
1 #include <cstdio>
2 #include <cstdlib>
3 #include <cstring>
4 #include <ctime>
5 #include <cmath>
```

```
6  #include <iostream>
7  #include <algorithm>
8
9  using namespace std;
10
11 const int oo = 1 << 30;
12 int ans, value[501][501], n, m, L[501], R[501], v[501];
13 bool bx[501], by[501];
14
15
16 bool find(int now){
17     bx[now] = true;
18     for (int i = 1; i <= m; i++)
19         if (!by[i] && L[now] + R[i] == value[now][i])
20         {
21             by[i] = true;
22             if (!v[i] || find(v[i]))
23             {
24                 v[i] = now;
25                 return(true);
26             }
27         }
28     return(false);
29 }
30
31 inline void km(){
32     memset(L, 0, sizeof(L));
33     memset(R, 0, sizeof(R));
34     for (int i = 1; i <= n; i++)
35         for (int j = 1; j <= m; j++)
36             L[i] = max(L[i], value[i][j]);
37     ans = 0;
38     memset(v, 0, sizeof(v));
39     for (int i = 1; i <= min(n, m); i++)
40         for (;;)
41         {
42             memset(bx, false, sizeof(bx));
43             memset(by, false, sizeof(by));
44             if (find(i)) break;
45             int Min = oo;
46             for (int j = 1; j <= n; j++)
47                 if (bx[j])
48                     for (int k = 1; k <= m; k++)
49                         if (!by[k])
50                             Min = min(Min, L[j] + R[k] - value[j][k]);
51             for (int j = 1; j <= n; j++) if (bx[j]) L[j] -= Min;
52             for (int j = 1; j <= m; j++) if (by[j]) R[j] += Min;
53         }
54     for (int i = 1; i <= n; i++)
```

```
55          for (int j = 1; j <= m; j++)
56              if (v[j] == i) ans += value[i][j];
57      printf("%d\n", abs(ans));
58  }
59
60  int main(){
61      scanf("%d%d", &n, &m);
62      for (int i = 1; i <= n; i++)
63          for (int j = 1; j <= m; j++) scanf("%d", &value[i][j]), value[i][j] = -value[
                i][j];
64      km();
65      for (int i = 1; i <= n; i++)
66          for (int j = 1; j <= m; j++)
67              value[i][j] = -value[i][j];
68      km();
69  }
70
71  /*hint 500 * 500 1.5s
72  can solve problems whose n != m
73  must be complete graph, or should change some values of matrix to satisfy the
        condition*/
```

3.6.4 shytangyuan+ 链表

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <cstring>
4  #include <ctime>
5  #include <cmath>
6  #include <iostream>
7  #include <algorithm>
8
9  using namespace std;
10
11 const int oo = 1 << 30;
12 int ans, first[501], l, where[250001], next[250001], value[250001], n, m, L[501], R
       [501], v[501];
13 bool bx[501], by[501];
14
15 inline void makelist(int x, int y, int z){
16     where[++l] = y;
17     value[l] = z;
18     next[l] = first[x];
19     first[x] = l;
20 }
21
22 bool find(int now){
23     bx[now] = true;
24     for (int x = first[now]; x; x = next[x])
25         if (!by[where[x]] && L[now] + R[where[x]] == value[x])
```

```
26          {
27              by[where[x]] = true;
28              if (!v[where[x]] || find(v[where[x]]))
29              {
30                  v[where[x]] = now;
31                  return(true);
32              }
33          }
34      return(false);
35  }
36
37  inline void km(){
38      memset(L, 0, sizeof(L));
39      memset(R, 0, sizeof(R));
40      for (int i = 1; i <= n; i++)
41          for (int x = first[i]; x; x = next[x])
42              L[i] = max(L[i], value[x]);
43      ans = 0;
44      memset(v, 0, sizeof(v));
45      for (int i = 1; i <= min(n, m); i++)
46          for (;;)
47          {
48              memset(bx, false, sizeof(bx));
49              memset(by, false, sizeof(by));
50              if (find(i)) break;
51              int Min = oo;
52              for (int j = 1; j <= n; j++)
53                  if (bx[j])
54                      for (int x = first[j]; x; x = next[x])
55                          if (!by[where[x]])
56                              Min = min(Min, L[j] + R[where[x]] - value[x]);
57              for (int j = 1; j <= n; j++) if (bx[j]) L[j] -= Min;
58              for (int j = 1; j <= m; j++) if (by[j]) R[j] += Min;
59          }
60      for (int i = 1; i <= n; i++)
61          for (int x = first[i]; x; x = next[x])
62              if (v[where[x]] == i) ans += value[x];
63      printf("%d\n", abs(ans));
64  }
65
66  int main(){
67      scanf("%d%d", &n, &m);
68      memset(first, 0, sizeof(first)); l = 0;
69      for (int i = 1; i <= n; i++)
70          for (int j = 1; j <= m; j++)
71          {
72              int x;
73              scanf("%d", &x);
74              makelist(i, j, -x);
```

```
75          }
76      km();
77      for (int i = 1; i <= l; i++) value[i] = -value[i];
78      km();
79  }
80
81  //hint 500 * 500 2.2s
82  //can solve problems whose n != m
```

3.7 Hopcroft

```
1  #include <cstdio>
2  #include <cstring>
3  #define maxn 50005
4  #define maxm 150005
5  int cx[maxn],cy[maxn],mk[maxn],q[maxn],src[maxn],pre[maxn];
6  int head[maxn],vtx[maxm],next[maxm],tot,n,m;
7  inline void Add(int a,int b)
8  {
9      vtx[tot]=b;
10     next[tot]=head[a];
11     head[a]=tot++;
12 }
13 inline int Maxmatch()
14 {
15     memset(mk,-1,sizeof(mk));
16     memset(cx,-1,sizeof(cx));
17     memset(cy,-1,sizeof(cy));
18     for (int p=1,fl=1,h,tail;fl;++p)
19     {
20         fl=0;
21         h=tail=0;
22         for (int i=0;i<n;++i)
23             if (cx[i]==-1)
24                 q[++tail]=i,pre[i]=-1,src[i]=i;
25         for (h=1;h<=tail;++h)
26         {
27             int u=q[h];
28             if (cx[src[u]]!=-1) continue;
29             for (int pp=head[u],v=vtx[pp];pp;pp=next[pp],v=vtx[pp])
30                 if (mk[v]!=p)
31                 {
32                     mk[v]=p;
33                     q[++tail]=cy[v];
34                     if (cy[v]>=0)
35                     {
36                         pre[cy[v]]=u;
37                         src[cy[v]]=src[u];
38                         continue;
```

```
39                          }
40                          int d,e,t;
41                          for
42                              (--tail,fl=1,d=u,e=v;d!=-1;t=cx[d],cx[d]=e,cy[e]=d,e=t,d=pre[
                                    d]);
43                          break;
44                      }
45              }
46          }
47          int res=0;
48          for (int i=0;i<n;++i)
49              res+=(cx[i]!=-1);
50          return res;
51      }
52      int main()
53      {
54          freopen("4206.in","r",stdin);
55          freopen("4206.out","w",stdout);
56          int P;
57          scanf("%d%d%d",&n,&m,&P);
58          tot=2;
59          for (int i=0;i<P;++i)
60          {
61              int a,b;
62              scanf("%d%d",&a,&b);
63              --a;--b;
64              Add(a,b);
65          }
66          printf("%d\n",Maxmatch());
67          return 0;
68      }
```

3.8 一般图最大匹配

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <cstring>
4  #include <iostream>
5  #include <algorithm>
6  using namespace std;
7  const int N=250;
8  int n;//点数(1->n)
9  int head;
10 int tail;
11 int Start;
12 int Finish;
13 int match[N];//表示哪个点匹配了哪个点
14 int Father[N];//增广路的Father
15 int Base[N];//该点属于哪朵花
```

```
16 int Q[N];
17 bool mark[N];
18 bool mat[N][N]; //邻接矩阵
19 bool InBlossom[N];
20 bool in_Queue[N];
21
22 void BlossomContract(int x,int y){
23     memset(mark,0,sizeof(mark));
24     memset(InBlossom,0,sizeof(InBlossom));
25 #define pre Father[match[i]]
26     int lca,i;
27     for (i=x;i;i=pre) {i=Base[i]; mark[i]=true; }
28     for (i=y;i;i=pre) {i=Base[i]; if (mark[i]) {lca=i; break;} }//寻找lca一定要注
           意,i=Base[i]
29     for (i=x;Base[i]!=lca;i=pre){
30         if (Base[pre]!=lca) Father[pre]=match[i];//对于树中的父边是匹配边的点，向后
               跳BFSFather
31         InBlossom[Base[i]]=true;
32         InBlossom[Base[match[i]]]=true;
33     }
34     for (i=y;Base[i]!=lca;i=pre){
35         if (Base[pre]!=lca) Father[pre]=match[i];//同理
36         InBlossom[Base[i]]=true;
37         InBlossom[Base[match[i]]]=true;
38     }
39 #undef pre
40     if (Base[x]!=lca) Father[x]=y;//注意不能从这个奇环的关键点跳回来lca
41     if (Base[y]!=lca) Father[y]=x;
42     for (i=1;i<=n;i++)
43         if (InBlossom[Base[i]]){
44             Base[i]=lca;
45             if (!in_Queue[i]){
46                 Q[++tail]=i;
47                 in_Queue[i]=true;//要注意如果本来连向树中父结点的边是非匹配边的点，可能是没有入
                       队的BFS
48             }
49         }
50 }
51
52 void Change(){
53     int x,y,z;
54     z=Finish;
55     while (z){
56         y=Father[z];
57         x=match[y];
58         match[y]=z;
59         match[z]=y;
60         z=x;
61     }
```

```
62 }
63
64 void FindAugmentPath(){
65     memset(Father,0,sizeof(Father));
66     memset(in_Queue,0,sizeof(in_Queue));
67     for (int i=1;i<=n;i++) Base[i]=i;
68     head=0; tail=1;
69     Q[1]=Start;
70     in_Queue[Start]=1;
71     while (head!=tail){
72         int x=Q[++head];
73         for (int y=1;y<=n;y++)
74             if (mat[x][y] && Base[x]!=Base[y] && match[x]!=y) {//无意义的边
75                 if ( Start==y || (match[y] && Father[match[y]]))//精髓地用表示该点是
                        否Father
76                     BlossomContract(x,y);
77                 else if (!Father[y]){
78                     Father[y]=x;
79                     if (match[y]){
80                         Q[++tail]=match[y];
81                         in_Queue[match[y]]=true;
82                     }
83                     else{
84                         Finish=y;
85                         Change();
86                         return;
87                     }
88                 }
89             }
90     }
91 }
92
93 void Edmonds(){
94     memset(match,0,sizeof(match));
95     for (Start=1;Start<=n;Start++)
96         if (match[Start]==0)
97             FindAugmentPath();
98 }
99
100 void output(){
101     memset(mark,0,sizeof(mark));
102     int cnt=0;//一般图最大匹配 最大点数
103     for (int i=1;i<=n;i++)
104         if (match[i]) cnt++;
105     printf("%d\n",cnt);
106     for (int i=1;i<=n;i++)
107         if (!mark[i] && match[i]){
108             mark[i]=true;//和imatch[i匹配]
109             mark[match[i]]=true;
```

```
110             printf("%d %d\n",i,match[i]);
111         }
112 }
113
114 int main(){
115     int x,y;
116     scanf("%d",&n);
117     memset(mat,0,sizeof(mat));
118     while (scanf("%d%d",&x,&y)!=EOF)
119         mat[x][y]=mat[y][x]=1;
120     Edmonds();
121     output();
122     return 0;
123 }
```

3.9 无向图最小割

```
1  const int V = 100;
2  #define typec int
3  const typec inf = 0x3f3f3f; // max of res
4  const typec maxw = 1000; // maximum edge weight
5  typec g[V][V], w[V]; //g[i][j] = g[j][i]
6  int a[V], v[V], na[V];
7  typec mincut(int n) {
8      int i, j, pv, zj;
9      typec best = maxw * n * n;
10     for (i = 0; i < n; i++) v[i] = i; // vertex: 0 ~ n-1
11     while (n > 1) {
12         for (a[v[0]] = 1, i = 1; i < n; i++) {
13             a[v[i]] = 0; na[i - 1] = i;
14             w[i] = g[v[0]][v[i]];
15         }
16         for (pv = v[0], i = 1; i < n; i++) {
17             for (zj = -1, j = 1; j < n; j++)
18                 if (!a[v[j]] && (zj < 0 || w[j] > w[zj]))
19                     zj = j;
20             a[v[zj]] = 1;
21             if (i == n - 1) {
22                 if (best > w[zj]) best = w[zj];
23                 for (i = 0; i < n; i++)
24                     g[v[i]][pv] = g[pv][v[i]] +=
25                         g[v[zj]][v[i]];
26                 v[zj] = v[--n];
27                 break;
28             }
29             pv = v[zj];
30             for (j = 1; j < n; j++)
31                 if(!a[v[j]])
32                     w[j] += g[v[zj]][v[j]];
```

```
33          }
34      }
35      return best;
36  }
37
38  int main()
39  {
40      return 0;
41  }
```

3.10 最小树形图 ($ElogE + V^2$)

```
1   const int N = 1111;
2   const int M = 1111111;
3   int n, m, a, b, c, x[N], y[N], z[N],
4       edgeCnt, firstEdge[N], from[M], length[M], nextEdge[M],
5       inEdge[N], key[M], delta[M], depth[M], child[M][2],
6       parent[N], choosen[N], degree[N], queue[N];
7   void pass (int x) {
8       if (delta[x] != 0) {
9           key[child[x][0]] += delta[x];
10          delta[child[x][0]] += delta[x];
11          key[child[x][1]] += delta[x];
12          delta[child[x][1]] += delta[x];
13          delta[x] = 0;
14      }
15  }
16  int merge (int x, int y) {
17      if (x == 0 or y == 0) {
18          return x ^ y;
19      }
20      if (key[x] > key[y]) {
21          swap(x, y);
22      }
23      pass(x);
24      child[x][1] = merge(child[x][1], y);
25      if (depth[child[x][0]] < depth[child[x][1]]) {
26          swap(child[x][0], child[x][1]);
27      }
28      depth[x] = depth[child[x][1]] + 1;
29      return x;
30  }
31  void addEdge (int u, int v, int w) {
32      from[++ edgeCnt] = u;
33      length[edgeCnt] = w;
34      nextEdge[edgeCnt] = firstEdge[v];
35      firstEdge[v] = edgeCnt;
36      key[edgeCnt] = w;
37      delta[edgeCnt] = 0;
```

```
38      depth[edgeCnt] = 0;
39      child[edgeCnt][0] = child[edgeCnt][1] = 0;
40      inEdge[v] = merge(inEdge[v], edgeCnt);
41  }
42  void deleteMin (int &r) {
43      pass(r);
44      r = merge(child[r][0], child[r][1]);
45  }
46  int findRoot (int u) {
47      if (parent[u] != u) {
48          parent[u] = findRoot(parent[u]);
49      }
50      return parent[u];
51  }
52  void clear () {
53      edgeCnt = 0;
54      depth[0] = -1;
55      memset(inEdge, 0, sizeof(inEdge));
56      memset(firstEdge, 0, sizeof(firstEdge));
57  }
58  int solve (int root) {
59      int result = 0;
60      for (int i = 0; i < n; ++ i) {
61          parent[i] = i;
62      }
63      while (true) {
64          memset(degree, 0, sizeof(degree));
65          for (int i = 0; i < n; ++ i) {
66              if (i == root or parent[i] != i) {
67                  continue;
68              }
69              while (findRoot(from[inEdge[i]]) == findRoot(i)) {
70                  deleteMin(inEdge[i]);
71              }
72              choosen[i] = inEdge[i];
73              degree[findRoot(from[choosen[i]])] += 1;
74          }
75          int head = 0, tail = 0;
76          for (int i = 0; i < n; ++ i) {
77              if (i != root and parent[i] == i and degree[i] == 0) {
78                  queue[tail ++] = i;
79              }
80          }
81          while (head < tail) {
82              if (-- degree[findRoot(from[choosen[queue[head]]])] == 0) {
83                  queue[tail ++] = findRoot(from[choosen[queue[head]]]);
84              }
85              head += 1;
86          }
```

```
87          bool found = false;
88          for (int i = 0; i < n; ++ i) {
89              if (i != root and parent[i] == i and degree[i] > 0) {
90                  found = true;
91                  int j = i, temp = 0;
92                  do{
93                      j = findRoot(from[choosen[j]]);
94                      parent[j] = i;
95                      deleteMin(inEdge[j]);
96                      result += key[choosen[j]];
97                      key[inEdge[j]] -= key[choosen[j]];
98                      delta[inEdge[j]] -= key[choosen[j]];
99                      temp = merge(temp, inEdge[j]);
100                 } while (j != i);
101                 inEdge[i] = temp;
102             }
103         }
104         if (not found) {
105             break;
106         }
107     }
108     for (int i = 0; i < n; ++ i) {
109         if (i != root and parent[i] == i) {
110             result += key[choosen[i]];
111         }
112     }
113     return result;
114 }
```

3.11 最小树形图 (V^3)

```
1  const int maxn=1100;
2  int n,m , g[maxn][maxn] , used[maxn] , pass[maxn] , eg[maxn] , more , queue[maxn];
3  void combine (int id , int &sum ) {
4      int tot = 0 , from , i , j , k ;
5      for ( ; id!=0 && !pass[ id ] ; id=eg[id] ) {
6          queue[tot++]=id ; pass[id]=1;
7      }
8      for ( from=0; from<tot && queue[from]!=id ; from++);
9      if
10         ( from==tot ) return ;
11     more = 1 ;
12     for ( i=from ; i<tot ; i++) {
13         sum+=g[eg[queue[i]]][queue[i]] ;
14         if ( i!=from ) {
15             used[queue[i]]=1;
16             for ( j = 1 ; j <= n ; j++) if ( !used[j] )
17                 if ( g[queue[i]][j]<g[id][j] ) g[id][j]=g[queue[i]][j] ;
18         }
```

```
19      }
20      for ( i=1; i<=n ; i++) if ( !used[i] && i!=id ) {
21          for ( j=from ; j<tot ; j++){
22              k=queue[j];
23              if ( g[i][id]>g[i][k]-g[eg[k]][k] ) g[i][id]=g[i][k]-g[eg[k]][k];
24          }
25      }
26  }
27  int mdst( int root ) { // return the total length of MDST
28      int i , j , k , sum = 0 ;
29      memset ( used , 0 , sizeof ( used ) ) ;
30      for ( more =1; more ; ) {
31          more = 0 ;
32          memset (eg,0,sizeof(eg)) ;
33          for ( i=1 ; i <= n ; i ++) if ( !used[i] && i!=root ) {
34              for ( j=1 , k=0 ; j <= n ; j ++) if ( !used[j] && i!=j )
35                  if ( k==0 || g[j][i] < g[k][i] ) k=j ;
36              eg[i] = k ;
37          }
38          memset(pass,0,sizeof(pass));
39          for ( i=1; i<=n ; i++) if ( !used[i] && !pass[i] && i!= root ) combine
40              ( i , sum ) ;
41      }
42      for ( i =1; i<=n ; i ++) if ( !used[i] && i!= root ) sum+=g[eg[i]][i];
43      return sum ;
44  }
45  int main(){
46      freopen("input.txt","r",stdin);
47      freopen("output.txt","w",stdout);
48      int i,j,k,test,cases;
49      cases=0;
50      scanf("%d",&test);
51      while (test){
52          test--;
53          //if (n==0) break;
54          scanf("%d%d",&n,&m);
55          //
56          memset(g,60,sizeof(g));
57          foru(i,1,n)
58              foru(j,1,n) g[i][j]=1000001;
59          foru(i,1,m) {
60              scanf("%d%d",&j,&k);
61              j++;k++;
62              scanf("%d",&g[j][k]);
63          }
64          cases++;
65          printf("Case␣#%d:␣",cases);
66          k=mdst(1);
67          if (k>1000000) printf("Possums!\n");
```

```
68            //===no
69            else printf("%d\n",k);
70        }
71        return 0;
72    }
```

Chapter 4

数据结构

4.1 KD 树

4.1.1 tEJtM+ 高维

```
1  #include<cstring>
2  #include<cstdio>
3  #include<algorithm>
4  #include<cmath>
5  using namespace std;
6  int nkd = 0, n, d, m;
7  template<typename T> struct vector
8  {
9      T a[2];
10     T & operator [] (int x) {return a[x];}
11     const T & operator [] (int x) const {return a[x];}
12 };
13 struct kd
14 {
15    kd * s[2];
16    int i;
17    vector<int> x;
18    vector<int> find();
19 } kdpool[222222], *root;
20 vector<int> u, v, vec[111111];
21 long long dis(const vector<int> & a, const vector<int> & b)
22 {
23    long long rtn = 0;
24    for(int i = 0; i < d; i++) rtn += (long long)(a[i] - b[i]) * (a[i] - b[i]);
25    return rtn;
26 }
27 void bize(int le, int ri, int index, int i)
28 {
29     if(ri <= le) return;
30    if(ri == le + 1)
```

```
31      {
32          if(vec[le][i] > vec[ri][i]) swap(vec[le], vec[ri]);
33          return;
34      }
35      int l = le, r = ri, x = vec[le + rand() % (ri - le + 1)][i];
36      for(;;)
37      {
38          while(vec[l][i] < x) l++;
39          while(vec[r][i] > x) r--;
40          if(l < r)
41          {
42              swap(vec[l], vec[r]);
43              l++; r--;
44          }
45          if(l >= r) break;
46      }
47      int posi = le;
48      while(posi <= ri and vec[posi][i] <= x) posi++;
49      if(index <= posi - 1) bize(le, posi - 1, index, i);
50      if(index >= posi) bize(posi, ri, index, i);
51  }
52  kd * build(int le, int ri, int i)
53  {
54      if(le > ri) return 0;
55      kd * p = &kdpool[++nkd];
56      bize(le, ri, (le + ri) / 2, i);
57      p->x = vec[(le + ri) / 2];
58      p->i = i;
59      if(le != ri)
60      {
61          p->s[0] = build(le, (le + ri) / 2 - 1, (i + 1) % d);
62          p->s[1] = build((le + ri) / 2 + 1, ri, (i + 1) % d);
63      }else p->s[0] = p->s[1] = 0;
64      return p;
65  }
66  vector<int> kd::find()
67  {
68      vector<int> rtn(x), tmp;
69      double l; int go = v[i] > x[i];
70      if(s[go])
71      {
72          tmp = s[go]->find();
73          if(dis(tmp, v) < dis(rtn, v)) rtn = tmp;
74      }
75      l = sqrt(dis(rtn, v));
76      if(v[i] - l < x[i] and x[i] < v[i] + l and s[go ^ 1])
77      {
78          tmp = s[go ^ 1]->find();
79          if(dis(tmp, v) < dis(rtn, v)) rtn = tmp;
```

```
80      }
81      return rtn;
82  }
83  int main()
84  {
85      scanf("%d%d", &n, &d);
86      for(int i = 1; i <= n; i++)
87      {
88          for(int j = 0; j < d; j++)
89              scanf("%d", &vec[i][j]);
90      }
91      root = build(1, n, 0);
92      scanf("%d", &m);
93      for(int i = 1; i <= m; i++)
94      {
95          for(int j = 0; j < d; j++)
96              scanf("%d", &v[j]);
97          u = root->find();
98          for(int j = 0; j < d; j++)
99          {
100             printf("%d%c", u[j], j == d - 1?'\n':' ');
101         }
102     }
103     return 0;
104 }
```

4.1.2 Logic_IU

读入 N 个点，输出距离每个点的最近点。

```
1   const int MAX_N = 100000 + 10;
2   const int MAX_NODE = 200000 + 10;
3   const LL INF = 200000000000000020LL;
4
5   int N;
6
7   struct Point
8   {
9       int x, y, id;
10  };
11
12  LL dis(const Point &a, const Point &b)
13  {
14      return 1LL * (a.x - b.x) * (a.x - b.x) + 1LL * (a.y - b.y) * (a.y - b.y);
15  }
16
17  struct Node
18  {
19      Point p;
20      int maxX, minX, maxY, minY;
21      int l, r, d;
```

```
22      Node *ch[2];
23  };
24
25  LL ret;
26  LL ans[MAX_N];
27  Node *root;
28  Point p[MAX_N], queryPoint;
29  Node *totNode, nodePool[MAX_NODE];
30
31  int cmpx(const Point &a, const Point &b)
32  {
33      return a.x < b.x;
34  }
35  int cmpy(const Point &a, const Point &b)
36  {
37      return a.y < b.y;
38  }
39
40  Node* newNode(int l, int r, Point p, int deep)
41  {
42      Node *t = totNode ++;
43      t->l = l; t->r = r;
44      t->p = p; t->d = deep;
45      t->maxX = t->minX = p.x;
46      t->maxY = t->minY = p.y;
47      return t;
48  }
49
50  void updateInfo(Node *t, Node *p)
51  {
52      t->maxX = max(t->maxX, p->maxX);
53      t->maxY = max(t->maxY, p->maxY);
54      t->minX = min(t->minX, p->minX);
55      t->minY = min(t->minY, p->minY);
56  }
57
58  Node* build(int l, int r, int deep)
59  {
60      if (l == r) return NULL;
61      if (deep & 1) sort(p + l, p + r, cmpx);
62      else sort(p + l, p + r, cmpy);
63      int mid = (l + r) >> 1;
64      Node *t = newNode(l, r, p[mid], deep & 1);
65      if (l + 1 == r) return t;
66      t->ch[0] = build(l, mid, deep + 1);
67      t->ch[1] = build(mid + 1, r, deep + 1);
68      if (t->ch[0]) updateInfo(t, t->ch[0]);
69      if (t->ch[1]) updateInfo(t, t->ch[1]);
70      return t;
```

```
71  }
72
73  void updateAns(Point p)
74  {
75      ret = min(ret, dis(p, queryPoint));
76  }
77
78  LL calc(Node *t, LL d)
79  {
80      LL tmp;
81      if (d) {
82          if (queryPoint.x >= t->minX && queryPoint.x <= t->maxX) tmp = 0;
83          else tmp = min(abs(queryPoint.x - t->maxX), abs(queryPoint.x - t->minX));
84      } else {
85          if (queryPoint.y >= t->minY && queryPoint.y <= t->maxY) tmp = 0;
86          else tmp = min(abs(queryPoint.y - t->maxY), abs(queryPoint.y - t->minY));
87      }
88      return tmp * tmp;
89  }
90
91  void query(Node *t)
92  {
93      if (t == NULL) return;
94      if (t->p.id != queryPoint.id) updateAns(t->p);
95      if (t->l + 1 == t->r) return;
96      LL dl = t->ch[0] ? calc(t->ch[0], t->d) : INF;
97      LL dr = t->ch[1] ? calc(t->ch[1], t->d) : INF;
98      if (dl < dr) {
99          query(t->ch[0]);
100         if (ret > dr) query(t->ch[1]);
101     } else {
102         query(t->ch[1]);
103         if (ret > dl) query(t->ch[0]);
104     }
105 }
106
107 void solve()
108 {
109     scanf("%d", &N);
110     for(int i = 0; i < N; ++ i) {
111         scanf("%d%d", &p[i].x, &p[i].y);
112         p[i].id = i;
113     }
114     totNode = nodePool;
115     root = build(0, N, 1);
116
117     for(int i = 0; i < N; ++ i) {
118         queryPoint = p[i];
119         ret = INF;
```

```
120         query(root);
121         ans[p[i].id] = ret;
122     }
123     for(int i = 0; i < N; ++ i)
124         printf("%I64d\n", ans[i]);
125 }
126
127 int main()
128 {
129     int T; scanf("%d", &T);
130     for( ; T --; )
131         solve();
132     return 0;
133 }
```

4.2 后缀自动机

4.2.1 tEJtM+LCA 非递归 Tarjan

```
1  #include<cstdio>
2  #include<cstring>
3  #include<ctime>
4  int np=0, rela[2000022], n, nod[2000011], l, cl, v, p, idx[2000022],iddx[2000022], ll
       , Q, siz[2000022];
5  int ans[1000011], x, y;
6  struct recq
7  {
8      int v, p;
9  } q[2000022];
10 bool f[2000022];
11 char c;
12 struct recmap
13 {
14     int y, next, i;
15 } map[2000022], map1[2000011];
16 void build(int x, int y)
17 {
18     map[++l].y = y;
19     map[l].next = idx[x];
20     idx[x] = l;
21 }
22 void build(int x, int y, int z)
23 {
24     map1[++ll].y = y;
25     map1[ll].i = z;
26     map1[ll].next = iddx[x];
27     iddx[x] = ll;
28 }
```

```
29 int getr(int x)
30 {
31     int p = x, p1, p2;
32     while(rela[p]!=p) p = rela[p];
33     p1 = p; p = x;
34     while(rela[p]!=p) {p2 = rela[p]; rela[p] = p1; p = p2;}
35     return p1;
36 }
37 struct recsam
38 {
39     int l, v;
40     recsam * go[26], *p;
41 } *leaf, *root, polsam[2000022];
42 recsam * newrecsam(int _l)
43 {
44     recsam * p = &polsam[++np];
45     p->l = _l; p->v = np; p->p=0;
46     memset(p->go, 0, sizeof(p->go));
47     return p;
48 }
49 recsam * newrecsam(recsam & x)
50 {
51     recsam * p = &polsam[++np];
52     *p=x;
53     p->v = np;
54     return p;
55 }
56 int main()
57 {
58     scanf("%d\n", &n);
59     root = newrecsam(0); leaf = root;
60     memset(siz, 0, sizeof(siz));
61     for(int i = 1; i <= n; i++)
62     {
63         scanf("%c", &c); c -= 'a';
64         recsam * np = newrecsam(i);
65         nod[i] = np->v; siz[np->v] = 1;
66         recsam * p = leaf;
67         for(;p and p->go[c] == 0;p=p->p)p->go[c] = np;
68         if(!p) np->p = root;
69         else
70         {
71             recsam * q;
72             if((q=p->go[c])->l == p->l+1) np->p = q;
73             else
74             {
75                 recsam * nq = newrecsam(*q);
76                 nq->l = p->l+1;
77                 nq->p = q->p;
```

```
78                  q->p = np->p = nq;
79                  for(;p and p->go[c] == q; p=p->p) p->go[c] = nq;
80              }
81          }
82          leaf = np;
83      }
84      l = 0; memset(idx, 0, sizeof(idx));
85      for(int i = 1; i <= np; i++) if(polsam[i].p)build(polsam[i].p->v, polsam[i].v);
86      scanf("%d", &Q);
87      ll = 0; memset(iddx, 0, sizeof(iddx));
88      for(int i = 1; i <= Q; i++)
89      {
90          scanf("%d%d", &x, &y);
91          build(nod[x], nod[y], i);
92          build(nod[y], nod[x], i);
93      }
94      q[cl=1].v = 1;
95      q[cl=1].p = idx[1];
96      memset(f, false, sizeof(f));
97      for(int i = 1; i <= np; i++) rela[i] = i;
98      while(cl)
99      {
100         v = q[cl].v;
101         q[cl].p = map[p=q[cl].p].next;
102         if(p)
103         {
104             q[++cl].v = map[p].y;
105             q[cl].p = idx[q[cl].v];
106         }else
107         {
108             f[v] = true;
109             for(int p = iddx[v]; p; p = map1[p].next)
110                 if(f[map1[p].y] == true) ans[map1[p].i] = getr(map1[p].y);
111             siz[q[cl-1].v] += siz[v];
112             rela[getr(v)] = getr(q[cl-1].v);
113             cl--;
114         }
115     }
116     for(int i = 1; i <= Q; i++) printf("%d\n", ans[i]!=1?siz[ans[i]]:0);
117     return 0;
118 }
```

4.2.2 Logic_IU

```
1 struct State
2 {
3     int val;
4     State *suf, *go[26];
5 } *root, *last;
6
```

```
7  State statePool[MAX_N], *curState;
8
9  void extend(int w)
10 {
11     State *p = last, *np = curState ++;
12     np->val = p->val + 1;
13     for( ; p && ! p->go[w]; p = p->suf)
14         p->go[w] = np;
15     if (! p)
16         np->suf = root;
17     else {
18         State *q = p->go[w];
19         if (q->val == p->val + 1)
20             np->suf = q;
21         else {
22             State *nq = curState ++;
23             nq->val = p->val + 1;
24             memcpy(nq->go, q->go, sizeof q->go);
25             nq->suf = q->suf;
26             q->suf = np->suf = nq;
27             for( ; p && p->go[w] == q; p = p->suf)
28                 p->go[w] = nq;
29         }
30     }
31     last = np;
32 }
```

4.3 Splay 树

4.3.1 Logic_IU

注意初始化内存池和 null 节点，以及根据需要修改 update 和 relax，区间必须是 1-based

```
1  const int MAX_NODE = 50000 + 10;
2  const int INF = 2000000000;
3
4  struct Node *null;
5
6  struct Node
7  {
8      int rev, add;
9      int val, maxv, size;
10     Node *ch[2], *p;
11
12     void set(Node *t, int _d) {
13         ch[_d] = t;
14         t->p = this;
15     }
16     int dir() {
17         return this == p->ch[1];
```

```
18      }
19      void update() {
20          maxv = max(max(ch[0]->maxv, ch[1]->maxv), val);
21          size = ch[0]->size + ch[1]->size + 1;
22      }
23      void relax() {
24          if (add) {
25              ch[0]->appAdd(add);
26              ch[1]->appAdd(add);
27              add = 0;
28          }
29          if (rev) {
30              ch[0]->appRev();
31              ch[1]->appRev();
32              rev = false;
33          }
34      }
35      void appAdd(int x) {
36          if (this == null) return;
37          add += x;
38          val += x;
39          maxv += x;
40      }
41      void appRev() {
42          if (this == null) return;
43          rev ^= true;
44          swap(ch[0], ch[1]);
45      }
46  };
47
48  Node nodePool[MAX_NODE], *curNode;
49
50  Node *newNode(int val = 0)
51  {
52      Node *t = curNode ++;
53      t->maxv = t->val = val;
54      t->rev = t->add = 0;
55      t->size = 1;
56      t->ch[0] = t->ch[1] = t->p = null;
57      return t;
58  }
59
60  struct Splay
61  {
62      Node *root;
63
64      Splay() {
65          root = newNode();
66          root->set(newNode(), 0);
```

```
67          root->update();
68      }
69
70      Splay(int *a, int N) { //sequence is 1-based
71          root = build(a, 0, N + 1);
72      }
73
74      Node* build(int *a, int l, int r) {
75          if (l > r) return null;
76          int mid = l + r >> 1;
77          Node *t = newNode(a[mid]);
78          t->set(build(a, l, mid - 1), 0);
79          t->set(build(a, mid + 1, r), 1);
80          t->update();
81          return t;
82      }
83
84      void rot(Node *t)
85      {
86          Node *p = t->p; int d = t->dir();
87          p->relax(); t->relax();
88          if (p == root) root = t;
89          p->set(t->ch[! d], d);
90          p->p->set(t, p->dir());
91          t->set(p, ! d);
92          p->update();
93      }
94
95      void splay(Node *t, Node *f = null)
96      {
97          for(t->relax(); t->p != f; ) {
98              if (t->p->p == f) rot(t);
99              else t->dir() == t->p->dir() ? (rot(t->p), rot(t)) : (rot(t), rot(t));
100         }
101         t->update();
102     }
103
104     Node* getKth(int k) {
105         Node *t = root;
106         int tmp;
107         for( ; ; ) {
108             t->relax();
109             tmp = t->ch[0]->size + 1;
110             if (tmp == k) return t;
111             if (tmp < k) {
112                 k -= tmp;
113                 t = t->ch[1];
114             } else
115                 t = t->ch[0];
```

```
116             }
117         }
118
119         //make range[l,r] root->ch[1]->ch[0]
120         //make range[x+1,x] to add something after position x
121         void getRng(int l, int r) {
122             r += 2;
123             Node *p = getKth(l);
124             Node *q = getKth(r);
125             splay(p); splay(q, p);
126         }
127
128         void addRng(int l, int r, int x) {
129             getRng(l, r);
130             root->ch[1]->ch[0]->appAdd(x);
131         }
132
133         void revRng(int l, int r) {
134             getRng(l, r);
135             root->ch[1]->ch[0]->appRev();
136         }
137
138         int maxvRng(int l, int r) {
139             getRng(l, r);
140             return root->ch[1]->ch[0]->maxv;
141         }
142     };
143
144     void initNull()
145     {
146         curNode = nodePool;
147         null = curNode ++;
148         null->size = 0;
149         null->maxv = - INF;
150     }
```

4.3.2 shytangyuan

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <cstring>
4  #include <ctime>
5
6  using namespace std;
7
8  struct {
9          int L,R,father,key,size;
10 } f[100001];
```

```
11 int root,n,Q;
12
13 inline void zig(int now){
14     int x=f[now].father,y=f[x].father;
15     if (y)
16        if (f[y].L==x) f[y].L=now;
17        else f[y].R=now;
18     f[now].father=y;
19     f[x].father=now;
20     f[x].L=f[now].R;
21     f[f[x].L].father=x;
22     f[now].R=x;
23     f[x].size=f[f[x].L].size+f[f[x].R].size+1;
24     f[now].size=f[f[now].L].size+f[f[now].R].size+1;
25 }
26
27 inline void zag(int now){
28     int x=f[now].father,y=f[x].father;
29     if (y)
30        if (f[y].L==x) f[y].L=now;
31        else f[y].R=now;
32     f[now].father=y;
33     f[x].father=now;
34     f[x].R=f[now].L;
35     f[f[x].R].father=x;
36     f[now].L=x;
37     f[x].size=f[f[x].L].size+f[f[x].R].size+1;
38     f[now].size=f[f[now].L].size+f[f[now].R].size+1;
39 }
40
41 inline void splay(int now){
42     int x=f[now].father,y=f[x].father;
43     while (x)
44     {
45           if (!y)
46               if (f[x].L==now) zig(now);
47               else zag(now);
48           else if (f[y].L==x)
49                   if (f[x].L==now) zig(x),zig(now);
50                   else zag(now),zig(now);
51                else if (f[x].L==now) zig(now),zag(now);
52                else zag(x),zag(now);
53           x=f[now].father;y=f[x].father;
54     }
55     root=now;
56 }
57
58 inline void insert(int now,int k){
59     if (!root)
```

```
60      {
61          root=k;
62          return;
63      }
64      if (f[k].key<=f[now].key)
65          if (!f[now].L) f[now].L=k,f[k].father=now,splay(k);
66          else insert(f[now].L,k);
67      else if (!f[now].R) f[now].R=k,f[k].father=now,splay(k);
68           else insert(f[now].R,k);
69  }
70
71  inline void del(int now){
72      splay(now);
73      int LL=f[now].L,RR=f[now].R;
74      f[now].L=f[now].R=f[now].key=f[now].father=f[now].size=0;
75      f[LL].father=f[RR].father=0;
76      if (!LL && !RR) root=0;
77      else if (!LL) root=RR;
78      else if (!RR) root=LL;
79      else
80      {
81          root=RR;
82          while (f[RR].L) RR=f[RR].L;
83          f[RR].L=LL;
84          f[LL].father=RR;
85          splay(LL);
86      }
87  }
88
89  int findkth(int now,int k){
90      if (k==f[f[now].L].size+1) return(now);
91
92  int main(){
93      scanf("%d",&n);
94      root=0;
95      for (int i=1;i<=n;i++)
96          scanf("%d",&f[i].key),f[i].size=1,insert(root,i);
97      scanf("%d",&Q);
98      for (;Q--;)
99      {
100         int type;
101         scanf("%d",&type);
102         if (type==1)
103         {
104             int x,y;
105             scanf("%d%d",&x,&y);
106             del(x);
107             f[x].key=y;f[x].size=1;
108             insert(root,x);
```

```
109             }
110             else
111             {
112                 int x;
113                 scanf("%d",&x);
114                 printf("%d\n",f[findkth(root,x)].key);
115             }
116         }
117     }
```

4.4 动态树

根据需求修改 Node 中的 relax 和 update 函数，修改 access，以及 Node 的构造函数，注意初始化内存池和 null 节点

```
1   struct Node
2   {
3       Node *ch[2], *p;
4       int isroot;
5       bool dir();
6       void set(Node*, bool);
7       void update();
8       void relax();
9   } *null;
10
11  void rot(Node *t)
12  {
13      Node *p = t->p; bool d = t->dir();
14      p->relax(); t->relax();
15      p->set(t->ch[! d], d);
16      if (p->isroot) t->p = p->p, swap(p->isroot, t->isroot);
17      else p->p->set(t, p->dir());
18      t->set(p, ! d);
19      p->update();
20  }
21
22  void Splay(Node *t)
23  {
24      for(t->relax(); ! t->isroot; ) {
25          if (t->p->isroot) rot(t);
26          else t->dir() == t->p->dir() ? (rot(t->p), rot(t)) : (rot(t), rot(t));
27      }
28      t->update();
29  }
30
31  void Access(Node *t)
32  {
33      for(Node *s = null; t != null; s = t, t = t->p) {
34          Splay(t);
35          t->ch[1]->isroot = true;
36          s->isroot = false;
```

```
37          t->ch[1] = s;
38          t->update();
39      }
40  }
41  bool Node::dir()
42  {
43      return this == p->ch[1];
44  }
45  void Node::set(Node *t, bool _d)
46  {
47      ch[_d] = t; t->p = this;
48  }
49  void Node::Update()
50  {
51
52  }
53  void Node::Relax()
54  {
55      if (this == Null) return;
56
57  }
```

4.5 二叉堆

双射堆，ind[v] 表示标号为 v 的节点在堆中的位置

```
1   const int MAX_V = 100000 + 10;
2   struct Heap
3   {
4       int tot;
5       int a[MAX_V], h[MAX_V], ind[MAX_V];
6       void exchange(int i, int j) {
7           swap(h[i], h[j]);
8           swap(ind[h[i]], ind[h[j]]);
9       }
10      inline int val(int x) {
11          return a[h[x]];
12      }
13      void fixUp(int x) {
14          if (x / 2 && val(x / 2) < val(x))
15              exchange(x, x / 2), fixUp(x / 2);
16      }
17      void fixDown(int x) {
18          int p = x * 2; if (p > tot) return;
19          if (p < tot && val(p + 1) > val(p)) ++ p;
20          if (val(p) > val(x))
21              exchange(p, x), fixDown(p);
22      }
23      void Update(int i, int x) {
```

```
24          a[i] = x;
25          fixUp(ind[i]);
26          fixDown(ind[i]);
27      }
28      int top() {
29          return h[1];
30      }
31      void pop() {
32          exchange(1, tot);
33          -- tot;
34          fixDown(1);
35      }
36      void insert(int i, int x) {
37          ++ tot;
38          h[tot] = i;
39          ind[i] = tot;
40          a[i] = x;
41          fixUp(tot);
42      }
43  } H;
```

4.6 左偏树

没写 delete 操作，注意初始化内存池和 null 节点

```
1   struct Node
2   {
3       int dis, val;
4       Node *ch[2];
5   } *null;
6
7   Node* merge(Node *u, Node *v)
8   {
9       if (u == null) return v;
10      if (v == null) return u;
11      if (u->val < v->val) swap(u, v);
12      u->ch[1] = merge(u->ch[1], v);
13      if (u->ch[1]->dis > u->ch[0]->dis)
14          swap(u->ch[1], u->ch[0]);
15      u->dis = u->ch[1]->dis + 1;
16      return u;
17  }
18
19  Node* newNode(int w)
20  {
21      Node *t = totNode ++;
22      t->ch[0] = t->ch[1] = null;
23      t->val = w; t->dis = 0;
24      return t;
```

```
25 }
```

4.7 Treap

包含 build, insert 和 erase，执行时注意初始化内存池和 null 节点

```
1  struct Node *null;
2
3  struct Node
4  {
5      int key, val, size;
6      Node *ch[2];
7      Node() {
8          key = INT_MAX;
9          val = size = 0;
10     }
11     Node(int _val) {
12         size = 1;
13         val = _val;
14         key = bigRand();
15         ch[0] = ch[1] = null;
16     }
17     int bigRand() {
18         return rand() * RAND_MAX + rand();
19     }
20     void update() {
21         size = ch[0]->size + ch[1]->size + 1;
22     }
23 };
24
25 struct Treap
26 {
27     Node *root;
28     Treap() {
29         root = null;
30     }
31     void rot(Node *&t, int d) {
32         Node *p = t->ch[d]; t->ch[d] = p->ch[!d]; p->ch[!d] = t;
33         t->update(); p->update();
34         t = p;
35     }
36
37     void insert(Node *&t, int x) {
38         if (t == null) {
39             t = new Node(x);
40             return;
41         }
42         int dir = x >= t->val;
43         insert(t->ch[dir], x);
```

```
44          if (t->ch[dir]->key < t->key)
45              rot(t, dir);
46          else
47              t->update();
48      }
49
50      void erase(Node *&t, int x) {
51          if (t == null)
52              return;
53          if (t->val == x) {
54              int dir = t->ch[1]->key < t->ch[0]->key;
55              if (t->ch[dir] == null) {
56                  delete t;
57                  t = null;
58                  return;
59              }
60              rot(t, dir);
61              erase(t->ch[! dir], x);
62              t->update();
63              return;
64          }
65          bool dir = x > t->val;
66          erase(t->ch[dir], x);
67          t->update();
68      }
69
70      void insert(int x) {
71          insert(root, x);
72      }
73
74      void erase(int x) {
75          erase(root, x);
76      }
77  };
```

4.8 线段树

包含建树和区间操作样例，没有写具体操作

```
1   struct Tree
2   {
3       int l, r;
4       Tree *ch[2];
5       Tree() {}
6       Tree(int _l, int _r, int *sqn) {
7           l = _l; r = _r;
8           if (l + 1 == r)
9               return;
10          int mid = l + r >> 1;
```

```
11          ch[0] = new Tree(l, mid, sqn);
12          ch[1] = new Tree(mid, r, sqn);
13      }
14
15      void insert(int p, int x) {
16          if (p < l || p >= r)
17              return;
18          //some operations
19          if (l + 1 == r)
20              return;
21          ch[0]->insert(p, x);
22          ch[1]->insert(p, x);
23      }
24
25      int query(int _l, int _r, int x) {
26          if (_r <= l || _l >= r)
27              return 0;
28          if (_l <= l && _r >= r)
29              // return information in [l, r)
30          //merge ch[0]->query(_l, _r, x), ch[1]->query(_l, _r, x) and return
31      }
32  };
```

4.9 轻重链剖分

包含 BFS 剖分过程，线段树部分视题目而定

```
1   struct Tree()
2   {
3
4   };
5
6   int father[MAX_N], size[MAX_N], depth[MAX_N];
7   int bfsOrd[MAX_N], pathId[MAX_N], ordInPath[MAX_N], sqn[MAX_N];
8   Tree *root[MAX_N];
9
10  void doBfs(int s)
11  {
12      int *que = bfsOrd;
13      int qh = 0, qt = 0;
14      father[s] = -1; depth[s] = 0;
15
16      for(que[qt ++] = s; qh < qt; ) {
17          int u = que[qh ++];
18          foreach(iter, adj[u]) {
19              int v = *iter;
20              if (v == father[u])
21                  continue;
22              father[v] = u;
```

```
23              depth[v] = depth[u] + 1;
24              que[qt ++] = v;
25          }
26      }
27  }
28
29  void doSplit()
30  {
31      for(int i = N - 1; i >= 0; -- i) {
32          int u = bfsOrd[i];
33          size[u] = 1;
34          foreach(iter, adj[u]) {
35              int v = *iter;
36              if (v == father[u])
37                  continue;
38              size[u] += size[v];
39          }
40      }
41
42      memset(pathId, -1, sizeof pathId);
43      for(int i = 0; i < N; ++ i) {
44          int top = bfsOrd[i];
45          if (pathId[top] != -1)
46              continue;
47
48          int cnt = 0;
49          for(int u = top; u != -1; ) {
50              sqn[cnt] = val[u];
51              ordInPath[u] = cnt;
52              pathId[u] = top;
53              ++ cnt;
54
55              int next = -1;
56              foreach(iter, adj[u]) {
57                  int v = *iter;
58                  if (v == father[u])
59                      continue;
60                  if (next < 0 || size[next] < size[v])
61                      next = v;
62              }
63              u = next;
64          }
65
66          root[top] = new Tree(0, cnt, sqn);
67      }
68  }
69
70  void prepare()
71  {
```

```
72      doBfs(0);
73      doSplit();
74  }
```

4.10 KMP

```
1   vector<int> KMP()
2   {
3       vector<int> ans;
4       nxt[0] = -1;
5       nxt[1] = 0;
6       for(int i = 2; i <= m; i++)
7       {
8           nxt[i] = nxt[i - 1];
9           while(nxt[i] >= 0 and st[i] != st[nxt[i] + 1])
10              nxt[i] = nxt[nxt[i]];
11          nxt[i]++;
12      }
13      for(int i = 1, p = 1; i <= n; i++)
14      {
15          while(p and str1[i] != st[p])
16              p = nxt[p - 1] + 1;
17          p++;
18          if(p == m + 1) p = nxt[m] + 1, ans.push_back(i - m);
19      }
20      return ans;
21  }
```

4.11 扩展 KMP

传入字符串 s 和长度 N，next[i]=LCP(s, s[i..N-1])

```
1   void z(char *s, int *next, int N)
2   {
3       int j = 0, k = 1;
4       while (j + 1 < N && s[j] == s[j + 1]) ++ j;
5       next[0] = N - 1; next[1] = j;
6       for(int i = 2; i < N; ++ i) {
7           int far = k + next[k] - 1, L = next[i - k];
8           if (L < far - i + 1) next[i] = L;
9           else {
10              j = max(0, far - i + 1);
11              while (i + j < N && s[j] == s[i + j]) ++ j;
12              next[i] = j; k = i;
13          }
14      }
15  }
```

4.12 Manacher

```
1  void manacher (char str[], int len[], int n) {
2      len[0] = 1;
3      for (int i = 1, j = 0; i < (n << 1) - 1; ++ i) {
4          int p = i >> 1,
5          q = i - p,
6          r = ((j + 1) >> 1) + len[j] - 1;
7          len[i] = r < q? 0: min(r - q + 1, len[(j << 1) - i]);
8          while (p - len[i] > -1 and q + len[i] < n and str[p - len[i]] == str[q + len[
               i]]) {
9              len[i] += 1;
10         }
11         if (q + len[i] - 1 > r) {
12             j = i;
13         }
14     }
15 }
```

4.13 AC 自动机

4.13.1 Logic_IU

包含建 trie 和构造自动机的过程

```
1
2  struct acNode
3  {
4      int id;
5      acNode *ch[26], *fail;
6  } *totNode, *root, nodePool[MAX_V];
7
8  acNode* newNode()
9  {
10     acNode *now = totNode ++;
11     now->id = 0; now->fail = 0;
12     memset(now->ch, 0, sizeof now->ch);
13     return now;
14 }
15
16 void acInsert(char *c, int id)
17 {
18     acNode *cur = root;
19     while (*c) {
20         int p = *c - 'A'; //change the index
21         if (! cur->ch[p]) cur->ch[p] = newNode();
22         cur = cur->ch[p];
23         ++ c;
24     }
25     cur->id = id;
```

```
26 }
27
28 void getFail()
29 {
30     acNode *cur;
31     queue<acNode*> Q;
32     for(int i = 0; i < 26; ++ i)
33         if (root->ch[i]) {
34             root->ch[i]->fail = root;
35             Q.push(root->ch[i]);
36         } else root->ch[i] = root;
37     while (! Q.empty()) {
38         cur = Q.front(); Q.pop();
39         for(int i = 0; i < 26; ++ i)
40             if (cur->ch[i]) {
41                 cur->ch[i]->fail = cur->fail->ch[i];
42                 Q.push(cur->ch[i]);
43             } else cur->ch[i] = cur->fail->ch[i];
44     }
45 }
```

4.13.2 shytangyuan

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <cstring>
4  #include <ctime>
5
6  using namespace std;
7
8  int a[100001][27],fail[100001],last[100001],c[100001],l,father[100001],type[100001];
9  char can[1001];
10
11 inline void maketrie(){
12     memset(a,0,sizeof(a));
13     memset(type,0,sizeof(type));
14     memset(last,0,sizeof(last));
15     int n=strlen(can),now=0;
16     l=0;
17     for (int i=0;i<n;i++)
18     {
19         if (!a[now][can[i]-'A']) a[now][can[i]-'A']=++l,type[l]=can[i]-'A',father[l]=
               now;
20         now=a[now][can[i]-'A'];
21     }
22     last[now]=1;
23 }
24
```

```
25 inline void makefail(){
26     memset(fail,255,sizeof(fail));
27     fail[0]=0;
28     int k=0;
29     for (int i=0;i<=25;i++)
30        if (a[0][i]) fail[a[0][i]]=0,c[++k]=a[0][i];
31     for (int l=1;l<=k;l++)
32     {
33         int m=c[l];
34         if (fail[m]==-1)
35         {
36            int p=father[m];
37            while (p && !a[fail[p]][type[m]]) p=fail[p];
38            fail[m]=a[fail[p]][type[m]];
39            last[m]+=last[fail[m]];
40         }
41         for (int i=0;i<=25;i++)
42             if (a[m][i]) c[++k]=a[m][i];
43     }
44 }
45
46 int main(){
47     scanf("%s",can);
48     maketrie();
49     makefail();
50 }
```

4.14 后缀数组

4.14.1 Logic_IU

对于串 a 求 SA，长度为 N，M 为元素值范围，height[i]=LCP(suf[rank[i]],suf[rank[i]-1])

```
1  const int MAX_N = 1000000 + 10;
2
3  int rank[MAX_N], height[MAX_N];
4
5  int cmp(int *x, int a, int b, int d)
6  {
7      return x[a] == x[b] && x[a + d] == x[b + d];
8  }
9
10 void doubling(int *a, int N, int M)
11 {
12     static int sRank[MAX_N], tmpA[MAX_N], tmpB[MAX_N];
13     int *x = tmpA, *y = tmpB;
14     for(int i = 0; i < M; ++ i) sRank[i] = 0;
15     for(int i = 0; i < N; ++ i) ++ sRank[x[i] = a[i]];
16     for(int i = 1; i < M; ++ i) sRank[i] += sRank[i - 1];
17     for(int i = N - 1; i >= 0; -- i) sa[-- sRank[x[i]]] = i;
```

```
18
19      for(int d = 1, p = 0; p < N; M = p, d <<= 1) {
20          p = 0; for(int i = N - d; i < N; ++ i) y[p ++] = i;
21          for(int i = 0; i < N; ++ i)
22              if (sa[i] >= d) y[p ++] = sa[i] - d;
23          for(int i = 0; i < M; ++ i) sRank[i] = 0;
24          for(int i = 0; i < N; ++ i) ++ sRank[x[i]];
25          for(int i = 1; i < M; ++ i) sRank[i] += sRank[i - 1];
26          for(int i = N - 1; i >= 0; -- i) sa[-- sRank[x[y[i]]]] = y[i];
27          swap(x, y); x[sa[0]] = 0; p = 1;
28          for(int i = 1; i < N; ++ i)
29              x[sa[i]] = cmp(y, sa[i], sa[i - 1], d) ? p - 1 : p ++;
30      }
31  }
32
33  void calcHeight()
34  {
35      for(int i = 0; i < N; ++ i) rank[sa[i]] = i;
36      int cur = 0;
37      for(int i = 0; i < N; ++ i)
38          if (rank[i]) {
39              if (cur) cur --;
40              for( ; a[i + cur] == a[sa[rank[i] - 1] + cur]; ++ cur);
41              height[rank[i]] = cur;
42          }
43  }
```

4.14.2 shytangyuan

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <cstring>
4  #include <ctime>
5
6  using namespace std;
7
8  int test,n,SA[100001],c[100001],Rank[100001],tmp[100001],H[100001],f[100001];
9  char can[50001];
10
11 int main(){
12     //freopen("1.txt","r",stdin);
13     //freopen("2.txt","w",stdout);
14     scanf("%d",&test);
15     for (test;test;test--)
16     {
17              scanf("%s\n",&can);
18              n=strlen(can);
19              memset(f,0,sizeof(f));
```

```
20                  for (int i=1;i<=n;i++) f[i]=int (can[i-1]);
21                  memset(c,0,sizeof(c));
22                  for (int i=1;i<=n;i++) c[f[i]]++;
23                  for (int i=1;i<=1000;i++) c[i]+=c[i-1];
24                  for (int i=n;i;i--) SA[c[f[i]]--]=i;
25                  Rank[SA[1]]=1;
26                  for (int i=2;i<=n;i++)
27                      if (f[SA[i]]==f[SA[i-1]]) Rank[SA[i]]=Rank[SA[i-1]];
28                      else Rank[SA[i]]=Rank[SA[i-1]]+1;
29                  for (int L=1;L<=n;L+=L)
30                  {
31                       if (Rank[SA[n]]==n) break;
32                       memset(c,0,sizeof(c));
33                       for (int i=1;i<=n;i++) c[Rank[L+i]]++;
34                       for (int i=1;i<=n;i++) c[i]+=c[i-1];
35                       for (int i=n;i;i--) tmp[c[Rank[L+i]]--]=i;
36                       memset(c,0,sizeof(c));
37                       for (int i=1;i<=n;i++) c[Rank[i]]++;
38                       for (int i=1;i<=n;i++) c[i]+=c[i-1];
39                       for (int i=n;i;i--) SA[c[Rank[tmp[i]]]--]=tmp[i];
40                       tmp[SA[1]]=1;
41                       for (int i=2;i<=n;i++)
42                           if ((Rank[SA[i]]==Rank[SA[i-1]])&&(Rank[SA[i]+L]==Rank[SA[i-1]+L
                                ]))
43                               tmp[SA[i]]=tmp[SA[i-1]];
44                           else tmp[SA[i]]=tmp[SA[i-1]]+1;
45                       for (int i=1;i<=n;i++) Rank[i]=tmp[i];
46                  }
47      int p=0;
48      for (int i=1;i<=n;i++)
49      {
50          int j=SA[Rank[i]-1];
51          p-=1;
52          if (p<0) p=0;
53          while ((f[i+p]==f[j+p])) p++;
54          H[i]=p;
55      }
56      int ans=0;
57      for (int i=1;i<=n;i++)
58          ans+=n-SA[i]+1-H[i];
59      printf("%d\n",ans);
60      }
61  }
```

4.14.3 DC3

```
1  //DC3 待排序的字符串放在r 数组中从,r到[0]r[n长度为-1],n且最大值小于,。m
2  //约定除r[n外所有的-1]r[i都大于]0, r[n。-1]=0
3  //函数结束后结果放在,sa 数组中从,sa到[0]sa[n。-1]
4  //必须开长度乘r3
```

```
#define maxn 10000
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int wa[maxn],wb[maxn],wv[maxn],wss[maxn];
int s[maxn*3],sa[maxn*3];
int c0(int *r,int a,int b)
{
    return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];
}
int c12(int k,int *r,int a,int b)
{
    if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
    else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
}
void sort(int *r,int *a,int *b,int n,int m)
{
    int i;
    for(i=0;i<n;i++) wv[i]=r[a[i]];
    for(i=0;i<m;i++) wss[i]=0;
    for(i=0;i<n;i++) wss[wv[i]]++;
    for(i=1;i<m;i++) wss[i]+=wss[i-1];
    for(i=n-1;i>=0;i--) b[--wss[wv[i]]]=a[i];
}
void dc3(int *r,int *sa,int n,int m)
{
    int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
    r[n]=r[n+1]=0;
    for(i=0;i<n;i++)
        if(i%3!=0) wa[tbc++]=i;
    sort(r+2,wa,wb,tbc,m);
    sort(r+1,wb,wa,tbc,m);
    sort(r,wa,wb,tbc,m);
    for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
        rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
    if (p<tbc) dc3(rn,san,tbc,p);
    else for (i=0;i<tbc;i++) san[rn[i]]=i;
    for (i=0;i<tbc;i++)
        if(san[i]<tb) wb[ta++]=san[i]*3;
    if(n%3==1) wb[ta++]=n-1;
    sort(r,wb,wa,ta,m);
    for(i=0;i<tbc;i++)
        wv[wb[i]=G(san[i])]=i;
    for(i=0,j=0,p=0;i<ta && j<tbc;p++)
        sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
    for(;i<ta;p++) sa[p]=wa[i++];
    for(;j<tbc;p++) sa[p]=wb[j++];
}
int main(){
    int n,m=0;
```

```
54      scanf("%d",&n);
55      for (int i=0;i<n;i++) scanf("%d",&s[i]),s[i]++,m=max(s[i]+1,m);
56      printf("%d\n",m);
57      s[n++]=0;
58      dc3(s,sa,n,m);
59      for (int i=0;i<n;i++) printf("%d ",sa[i]);printf("\n");
60  }
```

Chapter 5

杂

5.1 m^2logn 求线性递推第 n 项

```
1  // given first m a[i] and coef c[i] (0-based),
2  // calc a[n] mod p in O(m*m*log(n)).
3  // a[n] = sum(c[m-i]*a[n-i]), i = 1...m
4  // i.e. a[m] = sum(c[i]*a[i]), i = 0...m-1
5  int linear_recurrence(LL n, int m, int a[], int c[], int p) {
6      LL v[M] = {1 % p}, u[M<<1], msk = !!n;
7      for(LL i = n; i > 1; i >>= 1) msk <<= 1;
8      for(LL x = 0; msk; msk >>= 1, x <<= 1) {
9          fill_n(u, m<<1, 0);
10         int b = !!(n & msk); x |= b;
11         if(x < m) u[x] = 1 % p;
12         else {
13             for(int i = 0; i < m; ++i)
14                 for(int j = 0, t = i+b; j < m; ++j, ++t)
15                     u[t] = (u[t]+v[i]*v[j]) % p;
16             for(int i = (m<<1)-1; i >= m; --i)
17                 for(int j = 0, t = i-m; j < m; ++j, ++t)
18                     u[t] = (u[t]+c[j]*u[i]) % p;
19         }
20         copy(u, u+m, v);
21     }
22     int an = 0;
23     for(int i = 0; i < m; ++i) an = (an+v[i]*a[i]) % p;
24     return an;
25 }
```

5.2 FFT

```
1 #include<cstdio>
2 #include<cmath>
3 #include<cstring>
```

```
4  #include<algorithm>
5  using namespace std;
6  double pi = 2 * acos(0.0);
7  const int pw2lim = 65536;
8  struct recmap
9  {
10     int y, next;
11 } map[100011];
12 struct recq
13 {
14     int p, v;
15 } st[50011];
16 int idx[50001], ii[5000000], li1, n, K, x, y, z, ans[50001], l, l2, ele, siz[50001],
       q[50001], cl, dis[50001], fa[50001];
17 int L, go[2 * pw2lim], d, nrec, rec[50001], v, p;
18 bool f[50001], isprime[50001];
19 int mnpw2(int x)
20 {
21     int rtn = 1;
22     while(x)
23     {
24         x >>= 1;
25         rtn <<= 1;
26     }
27     return rtn;
28 }
29 struct vector
30 {
31     int siz, *a;
32     int & operator [] (int x)
33     {
34         return a[x];
35     }
36     vector()
37     {
38         a = ii + li1;
39     }
40 };
41 struct C
42 {
43     double real, imag;
44     C(const double & _real, const double & _imag) : real(_real), imag(_imag){}
45     C(){}
46     C(const double & x){real = x; imag = 0;}
47     void print(char c)
48     {
49         printf("(%f, %f)%c", real, imag, c);
50     }
```

```
51 } a[pw2lim], b[pw2lim], res1[pw2lim], res2[pw2lim], res[pw2lim], tmp[pw2lim], unit[
       pw2lim], temp;
52 const C operator * (const C & a, const C & b)
53 {
54     return C(a.real * b.real - a.imag * b.imag, a.real * b.imag + a.imag * b.real);
55 }
56 const C operator + (const C & a, const C & b)
57 {
58     return C(a.real + b.real, a.imag + b.imag);
59 }
60 const C operator - (const C & a, const C & b)
61 {
62     return C(a.real - b.real, a.imag - b.imag);
63 }
64 void build(int x, int y)
65 {
66     map[++l].y = y;
67     map[l].next = idx[x];
68     idx[x] = l;
69 }
70 void dft(C * sour, C * dest)
71 {
72     for(int i = 0; i < L; i++) tmp[go[i + L]] = sour[i];
73     for(int l = 1; l < L; l <<= 1)
74     {
75         l2 = l << 1;
76         ele = pw2lim / l2;
77         for(int i = 0; i < L; i += l2)
78             for(int j = 0; j < l; j++)
79             {
80                 temp = tmp[i + l + j] * unit[ele * j];
81                 tmp[i + l + j] = tmp[i + j] - temp;
82                 tmp[i + j] = tmp[i + j] + temp;
83             }
84     }
85     for(int i = 0; i < L; i++) dest[i] = tmp[i];
86 }
87 void fft(vector p1, vector p2)
88 {
89     L = mnpw2(p1.siz + p2.siz - 1);
90     for(int i = 0; i < p1.siz; i++) a[i] = p1[i];
91     for(int i = p1.siz; i < L; i++) a[i] = 0;
92     for(int i = 0; i < p2.siz; i++) b[i] = p2[i];
93     for(int i = p2.siz; i < L; i++) b[i] = 0;
94     dft(a, res1);
95     dft(b, res2);
96     for(int i = 0; i < L; i++) res[i] = res1[i] * res2[i];
97     dft(res, res1);
```

```
98      for(int i = 1; i <= nrec and rec[i] < p1.siz + p2.siz - 1; i++) ans[i] += (int)(
            res1[L - rec[i]].real / L + 0.5);
99      //ans[i](0<=i<L) = res1[(L - i) % L].
100 }
101 vector dvcq(int v)
102 {
103     if(siz[v] == 2)
104     {
105         vector vec;
106         vec.siz = 2;
107         vec[0] = 0;
108         vec[1] = 1;
109         li1 += vec.siz;
110         return vec;
111     }
112     int u=v, sum=1;
113     for(int p=idx[v]; p;)
114     {
115         if(siz[y=map[p].y] > siz[u]/2)
116         {
117             siz[y] += siz[u] -= siz[y];
118             u = y;
119             p = idx[y];
120         }else p = map[p].next;
121     }
122     int bak, biz;
123     vector p1, p2;
124     for(int p = idx[u]; p; p = map[p].next)
125     {
126         sum += siz[map[p].y];
127         if(sum >= siz[u]/2)
128         {
129             biz = siz[u];
130             bak = idx[u];
131             idx[u] = map[p].next;
132             siz[u] -= sum-1;
133             p1 = dvcq(u);
134             idx[u] = bak;
135             bak = map[p].next;
136             map[p].next = 0;
137             siz[u] = sum;
138             p2 = dvcq(u);
139             siz[u] = biz;
140             map[p].next = bak;
141             break;
142         }
143     }
144     fft(p1, p2);
145     vector vec;
```

```
146         fa[v] = 0;
147         q[cl=1] = v;
148         dis[v] = 0;
149         vec[0] = 0;
150         vec.siz = 0;
151         for(int op = 1; op <= cl; op++)
152             for(int p = idx[u = q[op]], y; p; p = map[p].next)
153                 if((y=map[p].y) != fa[u])
154                 {
155                     vec[dis[y] = dis[fa[q[++cl] = y] = u] + 1] ++;
156                     vec.siz = max(vec.siz, dis[y]);
157                 }
158         vec.siz++;
159         li1 += vec.siz;
160         return vec;
161     }
162     int main()
163     {
164         go[1] = 0;
165         for(int i = 2; i <= pw2lim; i <<= 1)
166         {
167             for(int j = 0; j < i / 2; j++)
168             {
169                 go[i + j] = go[i / 2 + j] * 2;
170             }
171             for(int j = i / 2; j < i; j++)
172             {
173                 go[i + j] = go[j] * 2 + 1;
174             }
175         }
176         unit[0] = 1;
177         unit[32768] = -1;
178         unit[16384] = C(0, 1);
179         for(int i = 8192; i >= 1; i /= 2)
180         {
181             unit[i] = C((unit[0].real + unit[i * 2].real) / 2, (unit[0].imag + unit[i *
                    2].imag) / 2);
182             double len = sqrt(unit[i].imag * unit[i].imag + unit[i].real * unit[i].real);
183             unit[i].imag *= 1/len; unit[i].real *= 1/len;
184         }
185         for(int i = 1; i <= 65536; i++)
186         {
187             if(i - (i & -i))
188             {
189                 unit[i] = C(1, 0);
190                 for(int x = i; x; x -= x & -x)
191                 {
192                     unit[i] = unit[i] * unit[x & -x];
193                 }
```

```
194         }
195     }//求单位复根
196     memset(isprime, true, sizeof(isprime));
197     nrec = 0; rec[0] = 0x7fffffff;
198     for(int i = 2; i <= 50000; i++)
199     {
200         if(isprime[i]) rec[++nrec] = i;
201         for(int j = 1; j <= nrec and i * rec[j] <= 50000 and i % rec[j - 1]; j++)
202             isprime[i * rec[j]] = false;
203     }
204     scanf("%d", &n);
205     memset(idx, 0, sizeof(idx));
206     l = 1;
207     for(int i = 1; i < n; i++)
208     {
209         scanf("%d%d", &x, &y);
210         build(x, y);
211         build(y, x);
212     }
213     memset(siz, 0, sizeof(siz));
214     memset(f, true, sizeof(f));
215     f[1] = false;
216     st[cl = 1].v = 1;
217     st[1].p = idx[1];
218     while(cl)
219     {
220         v = st[cl].v;
221         st[cl].p = map[p = st[cl].p].next;
222         if(p)
223         {
224             if(f[map[p].y])
225             {
226                 st[++cl].v = map[p].y;
227                 st[cl].p = idx[map[p].y];
228                 f[map[p].y] = false;
229             }
230         }else
231         {
232             siz[v]++;
233             siz[st[cl - 1].v] += siz[v];
234             cl--;
235         }
236     }
237     li1 = 0;
238     memset(ans, 0, sizeof(ans));
239     dvcq(1);
240     long long tot = 0;
241     for(int i = 1; i <= nrec; i++) tot += ans[i];;
242     printf("%lf\n", (double)tot / ((long long)n * (n - 1) / 2));
```

```
243     fclose(stdin);
244     fclose(stdout);
245     return 0;
246 }
```

5.3 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

```
1  LL x, y;
2  void exGcd(LL a, LL b)
3  {
4      if (b == 0) {
5          x = 1;
6          y = 0;
7          return;
8      }
9      exGcd(b, a % b);
10     LL k = y;
11     y = x - a / b * y;
12     x = k;
13 }
14
15 LL inversion(LL a, LL b)
16 {
17     exGcd(a, b);
18     return (x % b + b) % b;
19 }
20
21 LL CRT(vector<LL> m, vector<LL> a)
22 {
23     int N = m.size();
24     LL M = 1, ret = 0;
25     for(int i = 0; i < N; ++ i)
26         M *= m[i];
27
28     for(int i = 0; i < N; ++ i) {
29         ret = (ret + (M / m[i]) * a[i] % M * inversion(M / m[i], m[i])) % M;
30     }
31     return ret;
32 }
```

5.4 Pollard's Rho+Miller-Rabbin

大数分解和素性判断

```
1  typedef long long LL;
2
3  LL modMul(LL a, LL b, LL P)
```

```
4  {
5      LL ret = 0;
6      for( ; a; a >>= 1) {
7          if (a & 1) {
8              ret += b;
9              if (ret >= P) ret -= P;
10         }
11         b <<= 1;
12         if (b >= P) b -= P;
13     }
14     return ret;
15 }
16
17 LL modPow(LL a, LL u, LL P)
18 {
19     LL ret = 1;
20     for( ; u; u >>= 1, a = modMul(a, a, P))
21         if (u & 1) ret = modMul(ret, a, P);
22     return ret;
23 }
24
25 int millerRabin(LL N)
26 {
27     if (N == 2) return true;
28     LL t = 0, u = N - 1, x, y, a;
29     for( ; ! (u & 1); ++ t, u >>= 1) ;
30     for(int k = 0; k < 10; ++ k) {
31         a = rand() % (N - 2) + 2;
32         x = modPow(a, u, N);
33         for(int i = 0; i < t; ++ i, x = y) {
34             y = modMul(x, x, N);
35             if (y == 1 && x > 1 && x < N - 1) return false;
36         }
37         if (x != 1) return false;
38     }
39     return true;
40 }
41
42 LL gcd(LL a, LL b)
43 {
44     return ! b ? a : gcd(b, a % b);
45 }
46
47 LL pollardRho(LL N)
48 {
49     LL i = 1, x = rand() % N;
50     LL y = x, k = 2, d = 1;
51     do {
52         d = gcd(x - y + N, N);
```

```
53          if (d != 1 && d != N) return d;
54          if (++ i == k) y = x, k <<= 1;
55          x = (modMul(x, x, N) - 1 + N) % N;
56      } while (y != x);
57      return N;
58  }
59
60  void getFactor(LL N)
61  {
62      if (N < 2) return;
63      if (millerRabin(N)) {
64          //do some operations
65          return;
66      }
67      LL x = pollardRho(N);
68      getFactor(x);
69      getFactor(N / x);
70  }
```

5.5 素数判定 (long long 内确定性算法)

```
1   int strong_pseudo_primetest(long long n,int base) {
2       long long n2=n-1,res;
3       int s; s=0;
4       while(n2%2==0) n2>>=1,s++;
5       res=powmod(base,n2,n);
6       if((res==1)||(res==n-1)) return 1;
7       s--;
8       while(s>=0) {
9           res=mulmod(res,res,n);
10          if(res==n-1) return 1;
11          s--;
12      }
13      return 0; // n is not a strong pseudo prime
14  }
15  int isprime(long long n) {
16      if(n<2) return 0;
17      if(n<4) return 1;
18      if(strong_pseudo_primetest(n,2)==0) return 0;
19      if(strong_pseudo_primetest(n,3)==0) return 0;
20      if(n<1373653LL) return 1;
21      if(strong_pseudo_primetest(n,5)==0) return 0;
22      if(n<25326001LL) return 1;
23      if(strong_pseudo_primetest(n,7)==0) return 0;
24      if(n==3215031751LL) return 0;
25      if(n<25000000000LL) return 1;
26      if(strong_pseudo_primetest(n,11)==0) return 0;
27      if(n<2152302898747LL) return 1;
```

```
28      if(strong_pseudo_primetest(n,13)==0) return 0;
29      if(n<3474749660383LL) return 1;
30      if(strong_pseudo_primetest(n,17)==0) return 0;
31      if(n<341550071728321LL) return 1;
32      if(strong_pseudo_primetest(n,19)==0) return 0;
33      if(strong_pseudo_primetest(n,23)==0) return 0;
34      if(strong_pseudo_primetest(n,29)==0) return 0;
35      if(strong_pseudo_primetest(n,31)==0) return 0;
36      if(strong_pseudo_primetest(n,37)==0) return 0;
37      return 1;
38  }
```

5.6 求前 P 个数的逆元

```
1  void solve (int m) {
2      int inv[m];
3      inv[1] = 1;
4      for (int i = 2; i < m; ++ i) {
5          inv[i] = ((long long)(m - m / i) * inv[m % i]) % m;
6      }
7  }
```

5.7 广义离散对数 (不需要互质)

```
1  void extendedGcd (int a, int b, long long &x, long long y) {
2      if (b) {
3          extendedGcd(b, a % b, y, x);
4          y -= a / b * x;
5      } else {
6          x = a;
7          y = 0;
8      }
9  }
10 int inverse (int a, int m) {
11     long long x, y;
12     extendedGcd(a, m, x, y);
13     return (x % m + m) % m;
14 }
15 // a ^ x = b (mod m)
16 int solve (int a, int b, int m) {
17     int tmp = 1 % m, c;
18     map<int, int> s;
19     if (tmp == b) {
20         return 0;
21     }
22     for (int i = 1; i <= 50; ++ i) {
23         tmp = ((long long)tmp * a) % m;
24         if (tmp == b) {
25             return i;
```

```
26          }
27      }
28      int x_0 = 0, d = 1 % m;
29      while (true) {
30          tmp = gcd(a, m);
31          if (tmp == 1) {
32              break;
33          }
34          x_0 ++;
35          d = ((long long)d * (a / tmp)) % m;
36          if (b % tmp) {
37              return -1;
38          }
39          b /= tmp;
40          m /= tmp;
41      }
42      b = ((long long)b * inverse(d, m)) % m;
43      c = int(ceil(sqrt(m)));
44      s.clear();
45      tmp = b;
46      int tmpInv = intverse(a, m);
47      for (int i = 0; i != c; ++ i) {
48          if (s.find(tmp) == s.end()) {
49              s[tmp] = i;
50          }
51          tmp = ((long long)tmp * tmpInv) % m;
52      }
53      tmp = 1;
54      for (int i = 0; i != c; ++ i) {
55          tmp = ((long long)tmp * a) % m;
56      }
57      int ans = 1;
58      for (int i = 0; i != c; ++ i) {
59          if (s.find(ans) != s.end()) {
60              return x_0 + i * c + s.find(ans)->second;
61          }
62          ans = ((long long)ans * tmp) % m;
63      }
64      return -1;
65  }
```

5.8 n 次剰余

```
1  const int LimitSave=100000;
2  long long P,K,A;
3  vector<long long>ans;
4  struct tp{
5      long long expo,res;
6  }data[LimitSave+100];
```

```
7  long long _mod(long long a, long long mo) {
8      a=a%mo;
9      if (a<0) a+=mo;
10     return a;
11 }
12 long long powers(long long a , long long K , long long modular) {
13     long long res;
14     res=1;
15     while (K!=0) {
16         if (K & 1) res=_mod(res*a,modular);
17         K=K>>1;
18         a=_mod(a*a , modular);
19     }
20     return res;
21 }
22 long long get_originroot(long long p) {
23     long long primes[100];
24     long long tot,i,tp,j;
25     i=2; tp=P-1; tot=0;
26     while (i*i<=P-1) {
27         if (_mod(tp,i)==0) {
28             tot++;
29             primes[tot]=i;
30             while (_mod(tp,i)==0) tp/=i;
31         }
32         i++;
33     }
34     if (tp!=1) {tot++; primes[tot]=tp;}
35     i=2;
36     bool ok;
37     while (1) {
38         ok=true;
39         foru(j,1,tot) {
40             if (powers(i, (P-1)/primes[j] , P)==1) {
41                 150
42                 ok=false;
43                 break;
44             }
45         }
46         if (ok) break;
47         i++;
48     }
49     return i;
50 }
51 bool
52 euclid_extend(long long A ,long long B ,long long C ,long long &x, long
53 long &y, long long
54 &gcdnum) {
55     long long t;
```

```
56      if (A==0) {
57          gcdnum = B;
58          if (_mod(C , B) ==0) {
59              x=0; y=C/B;
60              return true;
61          }
62          else return false;
63      }
64      else if (euclid_extend(_mod(B , A) , A , C , y , t , gcdnum)) {
65          x = t - int(B / A) * y;
66          return true;
67      }
68      else return false;
69  }
70  long long Division(long long A, long long B, long long modular) {
71      long long gcdnum,K,Y;
72      euclid_extend(modular, B,A,K,Y,gcdnum);
73      Y=_mod(Y,modular);
74      if (Y<0) Y+=modular;
75      return Y;
76  }
77  bool Binary_Search(long long key, long long &position) {
78      long long start,stop;
79      start=1; stop=LimitSave;
80      bool flag=true;
81      while (start<=stop) {
82          position=(start+stop)/2;
83          if (data[position].res==key) return true;
84          else
85              if (data[position].res<key) start=position+1;
86              else stop=position-1;
87      }
88      return false;
89  }
90  bool compareab(const tp &a, const tp &b) {
91      return a.res<b.res;
92  }
93  long long get_log(long long root, long long A, long long modular) {
94      long long i,j,times,XD,XT,position;
95      if (modular-1<LimitSave) {
96          long long now=1;
97          foru(i,0,modular-1) {
98              if (now==A) {
99                  return i;
100             }
101             now=_mod(now * root , modular);
102         }
103     }
104     data[1].expo=0; data[1].res=1;
```

```
105     foru(i,1,LimitSave-1) {
106         data[i+1].expo=i;
107         data[i+1].res=_mod(data[i].res*root,modular);
108     }
109     sort(data+1,data+LimitSave+1,compareab);
110     times=powers(root,LimitSave,modular);
111     j=0;
112     XD=1;
113     while (1) {
114         XT=Division(A,XD,modular);
115         if (Binary_Search(XT,position)) {
116             return j+data[position].expo;
117         }
118         j=j+LimitSave;
119         XD=_mod(XD*times,modular);
120     }
121 }
122 void work_ans() {
123     ans.clear();
124     if (A==0) {
125         ans.push_back(0);
126         return;
127     }
128     long long root,logs,delta,deltapower,now,gcdnum, i,x,y;
129     root=get_originroot(P);
130     logs=get_log(root,A,P);
131     if (euclid_extend(K,P-1,logs,x,y,gcdnum)) {
132         delta=(P-1)/gcdnum;
133         x=_mod(x,delta);
134         if (x<0) x+=delta;
135         now=powers(root,x,P);
136         deltapower=powers(root,delta,P);
137         while (x<P-1) {
138             ans.push_back(now);
139             now=_mod(now*deltapower,P);
140             x=x+delta;
141         }
142     }
143     if (ans.size()>1)
144         sort(ans.begin(),ans.end());
145 }
146 int main(){
147     int i,j,k,test,cases=0;
148     scanf("%d",&test);
149     prepare();
150     while (test) {
151         test--;
152         cin>>P>>K>>A;
153         A=A % P;
```

```
154             //x^K mod P = A
155             cases++;
156             printf("Case #%d:\n",cases);
157             work_ans();
158         }
159         return 0;
160     }
```

5.9 二次剩余

```
1   /*
2       a*x^2+b*x+c==0 (mod P)求
3         0..P-1 的根
4     */
5   #include <cstdio>
6   #include <cstdlib>
7   #include <ctime>
8   #define sqr(x) ((x)*(x))
9   int pDiv2,P,a,b,c,Pb,d;
10  inline int calc(int x,int Time)
11  {
12      if (!Time) return 1;
13      int tmp=calc(x,Time/2);
14      tmp=(long long)tmp*tmp%P;
15      if (Time&1) tmp=(long long)tmp*x%P;
16      return tmp;
17  }
18  inline int rev(int x)
19  {
20      if (!x) return 0;
21      return calc(x,P-2);
22  }
23  inline void Compute()
24  {
25      while (1)
26      {
27          b=rand()%(P-2)+2;
28          if (calc(b,pDiv2)+1==P) return;
29      }
30  }
31  int main()
32  {
33      srand(time(0)^312314);
34      int T;
35      for (scanf("%d",&T);T;--T)
36      {
37          scanf("%d%d%d%d",&a,&b,&c,&P);
38          if (P==2)
39          {
```

```
40              int cnt=0;
41              for (int i=0;i<2;++i)
42                  if ((a*i*i+b*i+c)%P==0) ++cnt;
43              printf("%d",cnt);
44              for (int i=0;i<2;++i)
45                  if ((a*i*i+b*i+c)%P==0) printf(" %d",i);
46              puts("");
47          }else
48          {
49              int delta=(long long)b*rev(a)*rev(2)%P;
50              a=(long long)c*rev(a)%P-sqr( (long long)delta )%P;
51              a%=P;a+=P;a%=P;
52              a=P-a;a%=P;
53              pDiv2=P/2;
54              if (calc(a,pDiv2)+1==P) puts("0");
55              else
56              {
57                  int t=0,h=pDiv2;
58                  while (!(h%2)) ++t,h/=2;
59                  int root=calc(a,h/2);
60                  if (t>0)
61                  {
62                      Compute();
63                      Pb=calc(b,h);
64                  }
65                  for (int i=1;i<=t;++i)
66                  {
67                      d=(long long)root*root*a%P;
68                      for (int j=1;j<=t-i;++j)
69                          d=(long long)d*d%P;
70                      if (d+1==P)
71                          root=(long long)root*Pb%P;
72                      Pb=(long long)Pb*Pb%P;
73                  }
74                  root=(long long)a*root%P;
75                  int root1=P-root;
76                  root-=delta;
77                  root%=P;
78                  if (root<0) root+=P;
79                  root1-=delta;
80                  root1%=P;
81                  if (root1<0) root1+=P;
82                  if (root>root1)
83                  {
84                      t=root;root=root1;root1=t;
85                  }
86                  if (root==root1) printf("1 %d\n",root);
87                  else printf("2 %d %d\n",root,root1);
88              }
```

```
89          }
90      }
91      return 0;
92  }
```

5.10 长方体表面两点最短距离

返回最短距离的平方

```
1  #include<cstdio>
2  #include<iostream>
3  #include<algorithm>
4
5  using namespace std;
6
7  int r;
8  void turn(int i, int j, int x, int y, int z, int x0, int y0, int L, int W, int H)
9  {
10     if (z == 0) {
11         int R = x * x + y * y;
12         if (R < r) r = R;
13     } else {
14         if (i >= 0 && i < 2)
15             turn(i + 1, j, x0 + L + z, y, x0 + L - x, x0 + L, y0, H, W, L);
16         if (j >= 0 && j < 2)
17             turn(i, j + 1, x, y0 + W + z, y0 + W - y, x0, y0 + W, L, H, W);
18         if (i <= 0 && i > -2)
19             turn(i - 1, j, x0 - z, y, x - x0, x0 - H, y0, H, W, L);
20         if (j <= 0 && j > -2)
21             turn(i, j - 1, x, y0 - z, y - y0, x0, y0 - H, L, H, W);
22     }
23 }
24
25 int main()
26 {
27     int L, H, W, x1, y1, z1, x2, y2, z2;
28     cin >> L >> W >> H >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;
29     if (z1 != 0 && z1 != H) {
30         if (y1 == 0 || y1 == W)
31             swap(y1, z1), swap(y2, z2), swap(W, H);
32         else
33             swap(x1, z1), swap(x2, z2), swap(L, H);
34     }
35     if (z1 == H) z1 = 0, z2 = H - z2;
36     r = 0x3fffffff;
37     turn(0, 0, x2 - x1, y2 - y1, z2, -x1, -y1, L, W, H);
38     cout << r << endl;
39     return 0;
40 }
```

5.11 字符串的最小表示

5.11.1 Logic_IU

传入字符串 s，返回 i，表示以 i 开始的循环串字典序最小，但不保证 i 在同样字典序最小的循环串里起始位置最小

```
int minCycle(char *a)
{
    int n = strlen(a);
    for(int i = 0; i < n; ++ i) {
        a[i + n] = a[i];
    }
    a[n + n] = 0;
    int i = 0, j = 1, k = 0;
    do {
        for(k = 0; a[i + k] == a[j + k]; ++ k);
        if (a[i + k] > a[j + k]) i = i + k + 1;
        else j = j + k + 1;
        j += i == j;
        if (i > j) swap(i, j);
    } while(j < n);
    return i;
}
```

5.11.2 tEJtM

```
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);
}
int main()
```

```
24 {return 0;}
```

5.12 牛顿迭代开根号

速度慢，精度有保证

```
1  typedef unsigned long long ull;
2  ull sqrtll(ull n)
3  {
4      if (n == 0) return 0;
5      ull x = 1ull << ((63 - __builtin_clzll(n)) >> 1);
6      ull xx = -1;
7      for( ; ; ) {
8          ull nx = (x + n / x) >> 1;
9          if (nx == xx)
10             return min(x, nx);
11         xx = x;
12         x = nx;
13     }
14 }
```

5.13 求某年某月某日星期几

```
1  int whatday(int d, int m, int y)
2  {
3      int ans;
4      if (m == 1 || m == 2) {
5          m += 12; y --;
6      }
7      if ((y < 1752) || (y == 1752 && m < 9) || (y == 1752 && m == 9 && d < 3))
8          ans = (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 + 5) % 7;
9      else ans = (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 - y / 100 + y / 400) % 7;
10     return ans;
11 }
```

5.14 A*

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <cstring>
4  #include <ctime>
5
6  using namespace std;
7
8  struct {
9          int pos,tot;
10 } w[2000001];
11
```

```
12 const int inf=120234234;
13 int n,m,l,len,first[5001],c[2000001],dist[5001],where[200001],next[200001],v[200001],
       f[5001],lenn[5001];
14 bool b[5001];
15
16 inline void makelist(int x,int y,int z){
17     where[++l]=y;
18     v[l]=z;
19     next[l]=first[x];
20     first[x]=l;
21 }
22
23 inline void spfa(){
24     memset(b,false,sizeof(false));
25     memset(f,127,sizeof(f));
26     f[n]=0;
27     c[1]=n;
28     for (int k=1,l=1;l<=k;l++)
29      {
30              int m=c[l];
31              b[m]=false;
32              for (int x=first[m];x;x=next[x])
33                  if (f[m]+v[x]<f[where[x]])
34                  {
35                      f[where[x]]=f[m]+v[x];
36                      if (!b[where[x]])
37                      {
38                          b[where[x]]=true;
39                          c[++k]=where[x];
40                      }
41                  }
42      }
43 }
44
45 inline void insect(int re,int uu){
46     w[++len].pos=re;
47     w[len].tot=uu;
48     int now=len;
49     while (now!=1)
50       if (w[now].tot+f[w[now].pos]<w[now>>1].tot+f[w[now>>1].pos])
51         {
52             int k=w[now].tot;
53             w[now].tot=w[now>>1].tot;
54             w[now>>1].tot=k;
55             k=w[now].pos;
56             w[now].pos=w[now>>1].pos;
57             w[now>>1].pos=k;
58             now=now>>1;
59         }
```

```
60        else break;
61  }
62
63  inline void delete1(){
64      w[1].pos=w[len].pos;
65      w[1].tot=w[len].tot;
66      w[len].pos=inf;
67      w[len].tot=inf;
68      len--;
69      int now=1;
70      while ((now<<1)<=len)
71        if ((now<<1)==len)
72         if (w[now<<1].tot+f[w[now<<1].pos]<w[now].tot+f[w[now].pos])
73        {
74              int k=w[now].tot;
75              w[now].tot=w[now<<1].tot;
76              w[now<<1].tot=k;
77              k=w[now].pos;
78              w[now].pos=w[now<<1].pos;
79              w[now<<1].pos=k;
80              now=now<<1;
81        }
82        else break;
83        else
84        if ((w[now<<1].tot+f[w[now<<1].pos]<w[(now<<1)+1].tot+f[w[(now<<1)+1].pos]))
85        if (w[now<<1].tot+f[w[now<<1].pos]<w[now].tot+f[w[now].pos])
86        {
87              int k=w[now].tot;
88              w[now].tot=w[now<<1].tot;
89              w[now<<1].tot=k;
90              k=w[now].pos;
91              w[now].pos=w[now<<1].pos;
92              w[now<<1].pos=k;
93              now=now<<1;
94        }
95        else break;
96        else
97        if (w[(now<<1)+1].tot+f[w[(now<<1)+1].pos]<w[now].tot+f[w[now].pos])
98        {
99              int k=w[now].tot;
100             w[now].tot=w[(now<<1)+1].tot;
101             w[(now<<1)+1].tot=k;
102             k=w[now].pos;
103             w[now].pos=w[(now<<1)+1].pos;
104             w[(now<<1)+1].pos=k;
105             now=(now<<1)+1;
106       }
107       else break;
108 }
```

```
109
110 inline void spfa_ans(){
111     memset(dist,127,sizeof(dist));
112     memset(lenn,0,sizeof(lenn));
113     memset(w,127,sizeof(w));
114     c[1]=1;
115     len=1;
116     w[1].pos=1;
117     w[1].tot=0;
118     for (int k=1,l=1;l<=k;)
119       {
120             int m=w[1].pos,flow=w[1].tot;
121             delete1();
122             dist[m]=inf;
123             lenn[m]++;
124             if (lenn[m]>1000)
125             {
126                printf("%d\n",-1);
127                return;
128             }
129             if ((m==n)&&(lenn[m]==2))
130              {
131                   printf("%d\n",flow);
132                   return;
133              }
134             for (int x=first[m];x;x=next[x])
135                   {
136                      dist[where[x]]=flow+v[x];
137                      insect(where[x],dist[where[x]]);
138                   }
139       }
140 }
141
142 int main(){
143     scanf("%d%d",&n,&m);
144     l=0;
145     for (int i=1;i<=m;i++)
146       {
147             int x,y,z;
148             scanf("%d%d%d",&x,&y,&z);
149             makelist(x,y,z);
150             makelist(y,x,z);
151       }
152     spfa();
153     spfa_ans();
154 }
```

5.15 Dancing Links

```
#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()
{
    //freopen("1.in","r",stdin);
    //freopen("1.out","w",stdout);
    memset(a,0,sizeof(a));
    scanf("%d%d",&m,&n);
    for (int i=1;i<=n;++i){
        int x,y;scanf("%d",&x);
```

```
50          for (int j=1;j<=x;++j){
51              scanf("%d",&y);a[i][y]=1;
52          }
53      }
54      for (int i=1;i<=m;++i)head[i]=n*m+i; head[0]=0;
55      for (int i=1;i<=m;++i)r[head[i]]=head[i+1];
56      for (int i=1;i<=m;++i)l[head[i]]=head[i-1];
57      r[head[0]]=head[1];l[head[1]]=head[0];
58      l[head[0]]=head[m];r[head[m]]=head[0];
59      for (int i=1;i<=n;++i){
60          int pre=0,first=0;
61          for (int j=1;j<=m;++j)if (a[i][j]){
62              if (pre)l[getid(i,j)]=getid(i,pre),r[getid(i,pre)]=getid(i,j);
63              pre=j;if (!first)first=j;
64          }
65          if (first){
66              l[getid(i,first)]=getid(i,pre);r[getid(i,pre)]=getid(i,first);
67          }
68      }
69      for (int j=1;j<=m;++j){
70          int pre=0,first=0;
71          for (int i=1;i<=n;++i)if (a[i][j]){
72              if (pre)u[getid(i,j)]=getid(pre,j),d[getid(pre,j)]=getid(i,j);
73              pre=i;if (!first)first=i;
74          }
75          if (pre){
76              u[getid(first,j)]=head[j];d[head[j]]=getid(first,j);
77              u[head[j]]=getid(pre,j);d[getid(pre,j)]=head[j];
78          }
79      }
80      for (int i=1;i<=n;++i)
81          for (int j=1;j<=m;++j)if (a[i][j])c[getid(i,j)]=head[j];
82      memset(s,0,sizeof(s));
83      for (int i=1;i<=n;++i)
84          for (int j=1;j<=m;++j)if (a[i][j])++s[j];
85      ans=1<<30;
86      dfs(0);
87      if (ans==1<<30)printf("-1\n");
88      else printf("%d\n",ans);
89      system("pause");for (;;);
90      return 0;
91  }
```

5.16 弦图判定

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <cstring>
4  #include <ctime>
```

```
5  #include <cmath>
6  #include <iostream>
7  #include <algorithm>
8
9  using namespace std;
10
11 int n, m, first[1001], l, next[2000001], where[2000001],f[1001], a[1001], c[1001], L
       [1001], R[1001],
12 v[1001], idx[1001], pos[1001];
13 bool b[1001][1001];
14
15 int read(){
16     char ch;
17     for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
18     int cnt = 0;
19     for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
20     return(cnt);
21 }
22
23 inline void makelist(int x, int y){
24     where[++l] = y;
25     next[l] = first[x];
26     first[x] = l;
27 }
28
29 bool cmp(const int &x, const int &y){
30     return(idx[x] < idx[y]);
31 }
32
33 int main(){
34    //freopen("1015.in", "r", stdin);
35    // freopen("1015.out", "w", stdout);
36     for (;;)
37     {
38         n = read(); m = read();
39         if (!n && !m) return 0;
40         memset(first, 0, sizeof(first)); l = 0;
41         memset(b, false, sizeof(b));
42         for (int i = 1; i <= m; i++)
43         {
44             int x = read(), y = read();
45             if (x != y && !b[x][y])
46             {
47                 b[x][y] = true; b[y][x] = true;
48                 makelist(x, y); makelist(y, x);
49             }
50         }
51         memset(f, 0, sizeof(f));
52         memset(L, 0, sizeof(L));
```

```
53          memset(R, 255, sizeof(R));
54          L[0] = 1; R[0] = n;
55          for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
56          memset(idx, 0, sizeof(idx));
57          memset(v, 0, sizeof(v));
58          for (int i = n; i; --i)
59          {
60              int now = c[i];
61              R[f[now]]--;
62              if (R[f[now]] < L[f[now]]) R[f[now]] = -1;
63              idx[now] = i; v[i] = now;
64              for (int x = first[now]; x; x = next[x])
65                  if (!idx[where[x]])
66                  {
67                      swap(c[pos[where[x]]], c[R[f[where[x]]]]);
68                      pos[c[pos[where[x]]]] = pos[where[x]];
69                      pos[where[x]] = R[f[where[x]]];
70                      L[f[where[x]] + 1] = R[f[where[x]]]--;
71                      if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
72                      if (R[f[where[x]] + 1] == -1)
73                          R[f[where[x]] + 1] = L[f[where[x]] + 1];
74                      ++f[where[x]];
75                  }
76          }
77          bool ok = true;
78          for (int i = 1; i <= n && ok; i++)
79          {
80              int cnt = 0;
81              for (int x = first[v[i]]; x; x = next[x])
82                  if (idx[where[x]] > i) c[++cnt] = where[x];
83              sort(c + 1, c + cnt + 1, cmp);
84              bool can = true;
85              for (int j = 2; j <= cnt; j++)
86                  if (!b[c[1]][c[j]])
87                  {
88                      ok = false;
89                      break;
90                  }
91          }
92          if (ok) printf("Perfect\n");
93          else printf("Imperfect\n");
94          printf("\n");
95      }
96  }
```

5.17 弦图求团数

```
1  #include <cstdio>
2  #include <cstdlib>
```

```
3  #include <cstring>
4  #include <ctime>
5  #include <cmath>
6  #include <iostream>
7  #include <algorithm>
8
9  using namespace std;
10
11 int n, m, first[100001], next[2000001], where[2000001], l, L[100001], R[100001], c
       [100001], f[100001],
12 pos[100001], idx[100001], v[100001], ans;
13
14 inline void makelist(int x, int y){
15     where[++l] = y;
16     next[l] = first[x];
17     first[x] = l;
18 }
19
20 int read(){
21     char ch;
22     for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
23     int cnt = 0;
24     for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
25     return(cnt);
26 }
27
28 int main(){
29     freopen("1006.in", "r", stdin);
30     freopen("1006.out", "w", stdout);
31     memset(first, 0, sizeof(first)); l = 0;
32     n = read(); m = read();
33     for (int i = 1; i <= m; i++)
34     {
35         int x, y;
36         x = read(); y = read();
37         makelist(x, y); makelist(y, x);
38     }
39     memset(L, 0, sizeof(L));
40     memset(R, 255, sizeof(R));
41     memset(f, 0, sizeof(f));
42     memset(idx, 0, sizeof(idx));
43     for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
44     L[0] = 1; R[0] = n; ans = 0;
45     for (int i = n; i; --i)
46     {
47         int now = c[i], cnt = 1;
48         idx[now] = i; v[i] = now;
49         if (--R[f[now]] < L[f[now]]) R[f[now]] = -1;
50         for (int x = first[now]; x; x = next[x])
```

```
51              if (!idx[where[x]])
52              {
53                  swap(c[pos[where[x]]], c[R[f[where[x]]]]);
54                  pos[c[pos[where[x]]]] = pos[where[x]];
55                  pos[where[x]] = R[f[where[x]]];
56                  L[f[where[x]] + 1] = R[f[where[x]]]--;
57                  if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
58                  if (R[f[where[x]] + 1] == -1) R[f[where[x]] + 1] = L[f[where[x]] +
                        1];
59                  ++f[where[x]];
60              }
61              else ++cnt;
62          ans = max(ans, cnt);
63      }
64      printf("%d\n", ans);
65  }
```

5.18 有根树的同构

```
1   //http://acm.sdut.edu.cn/judgeonline/showproblem?problem_id=1861      11
2   #include <cstdio>
3   #include <cstdlib>
4   #include <cstring>
5   #include <ctime>
6
7   using namespace std;
8
9   const int mm=1051697,p=4773737;
10  int m,n,first[101],where[10001],next[10001],l,hash[10001],size[10001],pos[10001];
11  long long f[10001],rt[10001];
12  bool in[10001];
13
14
15  inline void makelist(int x,int y){
16      where[++l]=y;
17      next[l]=first[x];
18      first[x]=l;
19  }
20
21
22  inline void hashwork(int now){
23      int a[1001],v[1001],tot=0;
24      size[now]=1;
25      for (int x=first[now];x;x=next[x])
26      {
27          hashwork(where[x]);
28          a[++tot]=f[where[x]];
29          v[tot]=size[where[x]];
30          size[now]+=size[where[x]];
```

```
31      }
32      a[++tot]=size[now];
33      v[tot]=1;
34      int len=0;
35      for (int i=1;i<=tot;i++)
36          for (int j=i+1;j<=tot;j++)
37              if (a[j]<a[i])
38              {
39                  int u=a[i];a[i]=a[j];a[j]=u;
40                  u=v[i];v[i]=v[j];v[j]=u;
41              }
42      f[now]=1;
43      for (int i=1;i<=tot;i++)
44          {
45                  f[now]=((f[now]*a[i])%p*rt[len])%p;
46                  len+=v[i];
47          }
48  }
49
50  int main(){
51      //freopen("1.txt","r",stdin);
52      //freopen("2.txt","w",stdout);
53      scanf("%d%d",&n,&m);
54      rt[0]=1;
55      for (int i=1;i<=100;i++)
56          rt[i]=(rt[i-1]*mm)%p;
57      for (int i=1;i<=n;i++)
58      {
59          memset(first,0,sizeof(first));
60          memset(in,false,sizeof(in));
61          l=0;
62          for (int j=1;j<m;j++)
63          {
64              int x,y;
65              scanf("%d%d",&x,&y);
66              makelist(x,y);
67              in[y]=true;
68          }
69          int root=0;
70          for (int j=1;j<=m;j++)
71          if (!in[j])
72          {
73              root=j;
74              break;
75          }
76          memset(size,0,sizeof(size));
77          memset(f,0,sizeof(f));
78          hashwork(root);
79          hash[i]=f[root];
```

```
80      }
81      for (int i=1;i<=n;i++) pos[i]=i;
82      memset(in,false,sizeof(in));
83      for (int i=1;i<=n;i++)
84       if (!in[i])
85       {
86                   printf("%d",i);
87                   for (int j=i+1;j<=n;j++)
88                   if (hash[j]==hash[i])
89                   {
90                        in[j]=true;
91                        printf("=%d",j);
92                   }
93                   printf("\n");
94       }
95  }
```

5.19 极大团搜索算法

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

```
1  void init(){
2      int i, j;
3      for (i=1; i<=n; ++i) for (j=1; j<=n; ++j) scanf("%d", &g[i][j]);
4  }
5  void dfs(int size){
6      int i, j, k;
7      if (len[size]==0) {
8          if (size>ans) {
9              ans=size; found=true;
10         }
11         return;
12     }
13     for (k=0; k<len[size] && !found; ++k) {
14         if (size+len[size]-k<=ans) break;
15         i=list[size][k];
16         if (size+mc[i]<=ans) break;
17         for (j=k+1, len[size+1]=0; j<len[size]; ++j)
18             if (g[i][list[size][j]]) list[size+1][len[size+1]++]=list[size][j];
19         dfs(size+1);
20     }
21 }
22 void work(){
23     int i, j;
24     mc[n]=ans=1;
```

```
25      for (i=n-1; i; --i) {
26          found=false;
27          len[1]=0;
28          for (j=i+1; j<=n; ++j) if (g[i][j]) list[1][len[1]++]=j;
29          dfs(1);
30          mc[i]=ans;
31      }
32  }
33  void print(){
34      printf("%d\n", ans);
35  }
```

5.20 极大团的计数

Bool g[][] 为图的邻接矩阵, 图点的标号由 1 至 n。

```
1   void dfs(int size){
2       int i, j, k, t, cnt, best = 0;
3       bool bb;
4       if (ne[size]==ce[size]){
5           if (ce[size]==0) ++ans;
6           return;
7       }
8       for (t=0, i=1; i<=ne[size]; ++i) {
9           for (cnt=0, j=ne[size]+1; j<=ce[size]; ++j)
10              if (!g[list[size][i]][list[size][j]]) ++cnt;
11          if (t==0 || cnt<best) t=i, best=cnt;
12      }
13      if (t && best<=0) return;
14      for (k=ne[size]+1; k<=ce[size]; ++k) {
15          if (t>0){
16              for (i=k; i<=ce[size]; ++i) if (!g[list[size][t]][list[size][i]])
17                  break;
18              swap(list[size][k], list[size][i]);
19          }
20          i=list[size][k];
21          ne[size+1]=ce[size+1]=0;
22          for (j=1; j<k; ++j)if (g[i][list[size][j]])
23              list[size+1][++ne[size+1]]=list[size][j];
24          for (ce[size+1]=ne[size+1], j=k+1; j<=ce[size]; ++j)
25              if (g[i][list[size][j]]) list[size+1][++ce[size+1]]=list[size][j];
26          dfs(size+1);
27          ++ne[size];
28          --best;
29          for (j=k+1, cnt=0; j<=ce[size]; ++j) if (!g[i][list[size][j]]) ++cnt;
30          if (t==0 || cnt<best) t=k, best=cnt;
31          if (t && best<=0) break;
32      }
```

```
33 }
34 int work(){
35     int i;
36     ne[0]=0; ce[0]=0;
37     for (i=1; i<=n; ++i) list[0][++ce[0]]=i;
38     ans=0;
39     dfs(0);
40     return 0;
41 }
```

5.21 多项式求根 (求导二分)

```
1  const double error=1e-12;
2  const double infi=1e+12;
3  double a[10],x[10];
4  int n;
5  int sign(double x) {
6      return (x<-error)?(-1):(x>error);
7  }
8  double f(double a[],int n,double x) {
9      double tmp=1,sum=0;
10     for (int i=0;i<=n;i++) {
11         sum=sum+a[i]*tmp;
12         tmp=tmp*x;
13     }
14     return sum;
15 }
16 double binary(double l,double r,double a[],int n) {
17     int sl=sign(f(a,n,l)),sr=sign(f(a,n,r));
18     if (sl==0) return l;
19     if (sr==0) return r;
20     if (sl*sr>0) return infi;
21     while (r-l>error) {
22         double mid=(l+r)/2;
23         int ss=sign(f(a,n,mid));
24         if (ss==0) return mid;
25         if (ss*sl>0) l=mid; else r=mid;
26     }
27     return l;
28 }
29 void solve(int n,double a[],double x[],int &nx) {
30     if (n==1) {
31         x[1]=-a[0]/a[1];
32         nx=1;
33         return;
34     }
35     double da[10],dx[10];
36     int ndx;
```

```
37      for (int i=n;i>=1;i--) da[i-1]=a[i]*i;
38      solve(n-1,da,dx,ndx);
39      nx=0;
40      if (ndx==0) {
41          double tmp=binary(-infi,infi,a,n);
42          if (tmp<infi) x[++nx]=tmp;
43          return;
44      }
45      double tmp;
46      tmp=binary(-infi,dx[1],a,n);
47      if (tmp<infi) x[++nx]=tmp;
48      for (int i=1;i<=ndx-1;i++) {
49          tmp=binary(dx[i],dx[i+1],a,n);
50          if (tmp<infi) x[++nx]=tmp;
51      }
52      tmp=binary(dx[ndx],infi,a,n);
53      if (tmp<infi) x[++nx]=tmp;
54  }
55  int main() {
56      scanf("%d",&n);
57      for (int i=n;i>=0;i--) scanf("%lf",&a[i]);
58      int nx;
59      solve(n,a,x,nx);
60      for (int i=1;i<=nx;i++) printf("%0.6lf\n",x[i]);
61      return 0;
62  }
```

5.22 有多少个点在多边形内

```
1  //中的标号必须逆时针给出。一开始要旋转坐标rn保证同一个,值上只有一个点。正向减点x,
2  //反向加点。num[i][j]=num[j][i严格在这根线下方的点。]= on[i][j]=on[j][i严格]=
3  //在线段上的点包括两个端点。若有回边的话注意计算,的方法onit不要多算了线段上的点。,
4  int ans=0,z,onit=0,lows=0;
5  rep(z,t) {
6      i=rn[z]; j=rn[z+1]; onit+=on[i][j]-1;
7      if (a[j].x>a[i].x){ans-=num[i][j];lows+=on[i][j]-1;}
8      else ans+=num[i][j];
9  }
10 //ans-lows+1 is inside. 只会多算一次正向上的点除去最左和最右的点()。只算了除开最左边的点Lows但会
       多算最右边的点所以要再加上,,1.
11 printf("%d\n",ans-lows+1+ onit);
```

5.23 斜线下格点统计

```
1  LL solve(LL n, LL a, LL b, LL m){
2      //计算for (int i=0;i<n;++i) s+=floor((a+b*i)/m)
3      //n,m,a,b>0
4      //printf("%lld %lld %lld %lld\n", n, a, b, m);
5      if(b == 0){
```

```
6           return n * (a / m);
7       }
8       if(a >= m){
9           return n * (a / m) + solve(n, a % m, b, m);
10      }
11      if(b >= m){
12          return (n - 1) * n / 2 * (b / m) + solve(n, a, b % m, m);
13      }
14      LL q = (a + b * n) / m;
15      return solve(q, (a + b * n) % m, m, b);
16  }
```

5.24 杂知识

牛顿迭代

x1=x0-func(x0)/func1(x0); 进行牛顿迭代计算
我们要求 f(x)=0 的解。func(x) 为原方程,func1 为原方程的导数方程

图同构 Hash

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

枚举点 a, 迭代 K 次后求得的 $F_k(a)$ 就是 a 点所对应的 hash 值。
其中 K、A、B、C、D、P 为 hash 参数, 可自选。

圆上有整点的充要条件

设正整数 n 的质因数分解为 $n = \Pi p_i^{a_i}$, 则 $x^2 + y^2 = n$ 有整数解的充要条件是 n 中不存在形如 $p_i \mod 4 = 3$ 且指数 a_i 为奇数的质因数 p_i

Pick 定理

简单多边形, 不自交。(严格在多边形内部的整点数 *2 + 在边上的整点数 − 2)/2 = 面积

图定理

定理 1: 最小覆盖数 = 最大匹配数
定理 2: 最大独立集 S 与最小覆盖集 T 互补。
算法:
1. 做最大匹配, 没有匹配的空闲点 $\in S$
2. 如果 $u \in S$ 那么 u 的临点必然属于 T
3. 如果一对匹配的点中有一个属于 T 那么另外一个属于 S
4. 还不能确定的, 把左子图的放入 S, 右子图放入 T
算法结束

梅森素数

p 是素数且 2^p-1 的是素数,n 不超过 258 的全部梅森素数终于确定! 是：
n=2,3,5,7,13,17,19,31,61,89,107,127

上下界网络流

有上下界网络流, 求可行流部分, 增广的流量不是实际流量。若要求实际流量应该强算一遍源点出去的流量。
求最小下届网络流:
方法一: 加 t-s 的无穷大流, 求可行流, 然后把边反向后 (减去下届网络流), 在残留网络中从汇到源做最大流。
方法二: 在求可行流的时候, 不加从汇到源的无穷大边, 得到最大流 X, 加上从汇到源无穷大边后, 再求最大流得到 Y。
那么 Y 即是答案最小下界网络流。
原因: 感觉上是在第一遍已经把内部都消耗光了, 第二遍是必须的流量。

平面图定理

平面图一定存在一个度小于等于 5 的点, 且可以四染色
(欧拉公式) 设 G 是连通的平面图,n,m,r 分别是其顶点数、边数和面数,n-m+r=2
极大平面图 $m \leq 3n-6$

Fibonacci 相关结论

gcd(F[n],F[m])=F[gcd(n,m)]
Fibonacci 质数 (和前面所有的 Fibonacci 数互质), 下标为质数或 4
定理: 如果 a 是 b 的倍数, 那么 F[a] 是 F[b] 的倍数。

二次剩余

p 为奇素数, 若 (a,p)=1, a 为 p 的二次剩余必要充分条件为 $a^{(p-1)/2} \mod p = 1$.(否则为 $p-1$)
p 为奇素数, $x^b = a(\mod p)$,a 为 p 的 b 次剩余的必要充分条件为若 $a^{(p-1)/(p-1,b)} \mod p = 1$.

5.25 Language Reference

5.25.1 C++ Tips

1. 开栈的命令 #pragma comment(linker, "/STACK:16777216"), 交 C++
2. ios::sync_with_stdio(false);
3. %o 八进制 %x 十六进制

5.25.2 Java Reference

```
1 import java.io.*;
2 import java.math.*;
3 import java.util.*;
4
5 public class Main {
6     final static int MOD = (int)1e9 + 7;
```

```
7
8       public void run() {
9           try {
10              int n = reader.nextInt();
11              String[] map = new String[n];
12              for (int i = 0; i < n; ++ i) {
13                  map[i] = reader.next();
14              }
15              writer.println(10 % MOD);
16          } catch (IOException ex) {
17          }
18          writer.close();
19      }
20
21      InputReader reader;
22      PrintWriter writer;
23
24      Main() {
25          reader = new InputReader();
26          writer = new PrintWriter(System.out);
27      }
28
29      public static void main(String[] args) {
30          new Main().run();
31      }
32
33      void debug(Object... os) {
34          System.err.println(Arrays.deepToString(os));
35      }
36  }
37
38  class InputReader {
39      BufferedReader reader;
40      StringTokenizer tokenizer;
41
42      InputReader() {
43          reader = new BufferedReader(new InputStreamReader(System.in));
44          tokenizer = new StringTokenizer("");
45      }
46
47      String next() throws IOException {
48          while (!tokenizer.hasMoreTokens()) {
49              tokenizer = new StringTokenizer(reader.readLine());
50          }
51          return tokenizer.nextToken();
52      }
53
54      Integer nextInt() throws IOException {
55          return Integer.parseInt(next());
```

```
56      }
57  }
58  //------------------------------------------------------------
59  import java.util.*;
60  import java.math.*;
61  import java.io.*;
62
63  public class Main {
64
65      Scanner cin;
66
67      void solve() {
68          BigInteger a, b, c;
69          a = cin.nextBigInteger();
70          b = cin.nextBigInteger();
71          c = a.add(b);
72          System.out.println(a + " + " + b + " = " + c);
73      }
74
75      void run() {
76          cin = new Scanner(new BufferedInputStream(System.in));
77          int tmp = cin.nextInt();
78          int testcase = 0;
79          while(cin.hasNextBigInteger()) {
80              ++ testcase;
81              if (testcase > 1)
82                  System.out.println();
83              System.out.println("Case " + testcase + ":");
84              solve();
85          }
86      }
87
88      public static void main(String[] args) {
89          new Main().run();
90      }
91  }
92  //Arrays
93  int a[]=new int[10];
94  Arrays.fill(a,0);
95  Arrays.sort(a);
96  //String
97  String s;
98  s.charAt(int i);
99  s.compareTo(String b);
100 s.compareToIgnoreCase();
101 s.contains(String b);
102 s.length();
103 s.substring(int l,int len);
104 //BigInteger
```

```
105 BigInteger a;
106 a.abs();
107 a.add(b);
108 a.bitLength();
109 a.subtract(b);
110 a.divide(b);
111 a.remainder(b);
112 a.divideAndRemainder(b);
113 a.modPow(b,c);//a^b mod c;
114 a.pow(int);
115 a.multiply(b);
116 a.compareTo(b);
117 a.gcd(b);
118 a.intValue();
119 a.longValue();
120 a.isProbablePrime(int certainty);//(1 - 1/2^certainty).
121 a.nextProbablePrime();
122 a.shiftLeft(int);
123 a.valueOf();
124 //BigDecimal
125 static int ROUND_CEILING,ROUND_DOWN,ROUND_FLOOR,
126             ROUND_HALF_DOWN,ROUND_HALF_EVEN,ROUND_HALF_UP,ROUND_UP;
127 a.divide(BigDecimal b,int scale,int round_mode);
128 a.doubleValue();
129 a.movePointLeft(int i);
130 a.pow(int);
131 a.setScale(int scale,int round_mode);
132 a.stripTrailingZeros();
133 //StringBuilder
134 StringBuilder sb=new StringBuilder();
135 sb.append(elem);
136 out.println(sb);
137 //StringTokenizer
138 StringTokenizer st=new StringTokenizer(in.readLine());
139 st.countTokens();
140 st.hasMoreTokens();
141 st.nextToken();
142 //Vector
143 a.add(elem);
144 a.add(index,elem);
145 a.clear();
146 a.elementAt(index);
147 a.isEmpty();
148 a.remove(index);
149 a.set(index,elem);
150 a.size();
151 //Queue
152 a.add(elem);
153 a.peek();//front
```

```
154 a.poll();//pop
155 //Integer Double Long
```

5.26 vimrc

```
1  set nu ai ci si mouse=a ts=4 sts=4 sw=4
2
3  nmap <C-A> ggVG
4  vmap <C-C> "+y
5
6  nmap<F3> :vs %<.in <CR>
7  nmap<F8> :!./%<<%<.in <CR>
8  nmap<F9> :make %< <CR>
9
10 nmap<F4> :!gedit %<CR>
11 nmap<F5> :!./%<<CR>
12 nmap<F6> :!java %<<%<.in <CR>
13 nmap<F10> :!javac %<CR>
```