

Apocalypse
Standard Code Library

version 3.141

May, 2016

Contents

1	二维几何	5
1.1	Naive Tips	5
1.2	几何公式	5
1.3	点类	8
1.4	基本操作	10
1.5	球面	13
1.6	半平面交	14
1.7	最小圆覆盖	16
2	图论	19
2.1	Dijkstra	19
2.2	最大流	19
2.2.1	iSAP	19
2.2.2	Dinic	21
2.3	上下界流	23
2.3.1	上下界无源汇可行流	23
2.3.2	上下界最大流	25
2.3.3	上下界最小流	28
2.3.4	上下界有源汇可行流	30
2.4	费用流	34
2.4.1	负费用路	34
2.4.2	ZKW	35
2.5	强联通分量	37
2.5.1	递归	37
2.5.2	手写栈	38
2.6	最近公共祖先	40
2.7	KM	43
2.7.1	邻接阵	43
2.7.2	链表	44
3	数据结构	47
3.1	KD 树	47
3.2	Splay 树	50
3.3	区间第 k 大	53
3.3.1	动态	53
3.3.2	树套树 treap	56
3.4	Treap	60

3.5	线段树	62
3.6	KMP	62
3.7	扩展 KMP	63
3.8	Manacher	63
3.9	AC 自动机	64
3.10	后缀数组	65
3.10.1	倍增	65
3.10.2	DC3	66
4	杂	69
4.1	$m^2 \log n$ 求线性递推第 n 项	69
4.2	FFT	69
4.3	线性筛莫比乌斯	71
4.4	中国剩余定理	71
4.5	Pollard's Rho+Miller-Rabbin	72
4.6	素数判定 (long long 内确定性算法)	74
4.7	求前 P 个数的逆元	74
4.8	Lucas 快速取 mod	75
4.9	快速幂	75
4.10	广义离散对数 (不需要互质)	76
4.11	n 次剩余	77
4.12	二次剩余	81
4.13	长方体表面两点最短距离	83
4.14	字符串的最小表示	84
4.14.1	min	84
4.14.2	min_1	84
4.15	牛顿迭代开根号	85
4.16	求某年某月某日星期几	85
4.17	日期类解决两个日期之间差多少天	85
4.18	多项式求根 (求导二分)	87
4.19	有多少个点在多边形内	89
4.20	斜线下格点统计	89
4.21	杂知识	89
4.22	补充公式	91
4.23	Language Reference	92
4.23.1	C++ Tips	92
4.23.2	Java Reference	92
4.24	vimrc	96

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. $\text{atan2}(0,0)=0$, atan2 的值域为 $[-\pi, \pi]$
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

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

$$7. \text{ 外接圆半径 } R = abc/(4S) = a/(2 \sin(A)) = b/(2 \sin(B)) = c/(2 \sin(C))$$

四边形

$D1, D2$ 为对角线, M 为对角线中点连线, 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 $S1 = rl/2 = r^2 A/2$
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 $V = (A_1 + A_2 + \sqrt{A_1 A_2})h/3$, A_1, A_2 为上下底面积, h 为高
2. 正棱台侧面积 $S = (p_1 + p_2)l/2$, p_1, p_2 为上下底面周长, l 为斜高
3. 正棱台全面积 $T = S + A_1 + A_2$

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 $l = \sqrt{h^2 + (r_1 - r_2)^2}$
2. 侧面积 $S = \pi(r_1 + r_2)l$
3. 全面积 $T = \pi r_1(l + r_1) + \pi r_2(l + r_2)$
4. 体积 $V = \pi(r_1^2 + r_2^2 + r_1 r_2)h/3$

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 $T = \pi(2rh + r_1^2 + r_2^2)$
3. 体积 $V = \pi h(3(r_1^2 + r_2^2) + h^2)/6$

球扇形

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

1.3 点类

```

1 #include <cmath>
2 #include <cstdio>
3 #include <vector>
4 #include <cstring>
5 #include <iostream>
6 #include <algorithm>
7 #define foreach(e,x) for(__typeof(x.begin()) e=x.begin();e!=x.end();++e)
8 using namespace std;
9
10 const double PI = acos(-1.);
11 const double EPS = 1e-8;
12 inline int sign(double a) {
13     return a < -EPS ? -1 : a > EPS;
14 }
15
16 struct Point {
17     double x, y;
18     Point() {}
19 }
20 Point(double _x, double _y) :
21     x(_x), y(_y) {}
22
23 Point operator+(const Point&p) const {
24     return Point(x + p.x, y + p.y);
25 }
26 Point operator-(const Point&p) const {
27     return Point(x - p.x, y - p.y);
28 }
29 Point operator*(double d) const {
30     return Point(x * d, y * d);
31 }
32 Point operator/(double d) const {
33     return Point(x / d, y / d);
34 }
35 bool operator<(const Point&p) const {
36     int c = sign(x - p.x);
37     if (c)
38         return c == -1;
39     return sign(y - p.y) == -1;
40 }
41 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 $\overrightarrow{q_1q_2}$ 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) *
4         cross(q1, q2, p2) < 0;
5  }

```

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

```

1 int insidePolygon(Point q, vector<Point> &p)
2 {
3     int n = p.size();
4     for(int i = 0; i < n; ++i) {
5         if (onSegment(q, p[i], p[(i + 1) % n])) return false;
6     }
7     Point q2;
8     double offsite = LIM;
9     for( ; ; ) {
10        int flag = true;
11        int rnd = rand() % 10000;
12        q2.x = cos(rnd) * offsite;
13        q2.y = sin(rnd) * offsite;
14        for(int i = 0; i < n; ++i) {
15            if (onSegment(p[i], q, q2)) {
16                flag = false;
17                break;
18            }
19        }
20        if (flag) break;
21    }
22    int cnt = 0;
23    for(int i = 0; i < n; ++i) {
24        cnt += isIntersect(p[i], p[(i + 1) % n], q, q2);
25    }
26    return cnt & 1;
27 }

```

判断直线 $l1l2$ 是否与圆相交, 相切返回 true

```

1 int isIntersectLineToCircle(Point c, double r, Point l1, Point l2)
2 {
3     return (distPointToLine(c, l1, l2) - r) <= 0;
4 }

```

判断圆与线段是否有公共点, 线段在圆内部返回 true

```

1 int isIntersectSegmentToCircle(Point c, double r, Point p1, Point p2)
2 {
3     if ((distPointToLine(c, p1, p2) - r) > 0) return false;
4     if (sign(c.distTo(p1) - r) <= 0 || sign(c.distTo(p2) - r) <= 0) return true;
5     Point c2 = (p2 - p1).rot90() + c;
6     return crossOp(c, c2, p1) * crossOp(c, c2, p2) <= 0;
7 }

```

判断圆与圆是否相交, 外切或内切返回 true

```

1 int isIntersectCircleToCircle(Point c1, double r1, Point c2, double r2)

```

```

2 {
3     double dis = c1.distTo(c2);
4     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 }

```

圆外一点引圆的切线

```

1 void cutpoint(point c, point r, point sp, point &rp1, point &rp2)
2 {
3     point c2;
4     double r2;
5     c2 = (c + sp) / 2.0;
6     r2 = (c2 - sp).abs();
7     intersectionCircleToCircle(point c, double r, point c2, double r2, point &rp1,
  point &rp2);
8 }

```

1.5 球面

计算圆心角 lat 表示纬度, $-90 \leq w \leq 90$, lng 表示经度

返回两点所在大圆劣弧对应圆心角, $0 \leq angle \leq \pi$

```

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)-
75         len(z))/2);
76     cir.c = couple(t3*t2, t1*t3)/(t1*t2);
77 }
78 inline circle mi()
79 {
80     sort(a + 1, a + 1 + n);
81     n = unique(a + 1, a + 1 + n) - a - 1;
82     if(n == 1)
83     {
84         cir.c = a[1];
85         cir.r = 0;
86         return cir;
87     }
88     random_shuffle(a + 1, a + 1 + n);
89     p2c(1, 2);
90     for(int i = 3; i <= n; i++)
91         if(!inside(a[i]))
92         {
93             p2c(1, i);
94             for(int j = 2; j < i; j++)
95                 if(!inside(a[j]))
96                 {
97                     p2c(i, j);
98                     for(int k = 1; k < j; k++)
99                         if(!inside(a[k]))
100                             p3c(i, j, k);
101                 }
102         }
103     return cir;
104 }

```

Chapter 2

图论

2.1 Dijkstra

求 s 到其他点的最短路

```
1 int used[MAX_N], dis[MAX_N];
2 void dijkstra(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 }
```

2.2 最大流

2.2.1 iSAP

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 }

```

2.2.2 Dinic

```

1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4  using namespace std;
5
6  const long long inf = (long long)1e17 + 10;
7  struct edge{
8      int v, w, n;
9  } e[60200];
10 int en[5010], lv[5010], q[5010], cur[5010];
11 int n, m, sou, tar, esize;
12 void addedge(int u, int v, int w){
13     e[esize].v = v;
14     e[esize].w = w;
15     e[esize].n = en[u];
16     en[u] = esize++;
17 }
18 bool bfs()
19 {
20     memset(lv, -1, sizeof(lv));
21     int head, tail;
22     lv[tar] = head = tail = 0;
23     q[tail++] = tar;
24     while(head < tail){
25         int u = q[head++];
26         for (int t = en[u]; t != -1; t = e[t].n){
27             int v = e[t].v;
28             if (lv[v] == -1 && e[t].w > 0){
29                 lv[v] = lv[u] + 1;

```

```

30         if (v != sou) q[tail++] = v;
31     }
32 }
33 }
34 return lv[sou] != -1;
35 }
36 long long dfs(int u, long long maxf)
37 {
38     if (maxf == 0) return 0;
39     if (u == tar) return maxf;
40     long long ret = 0, res, flow;
41     for (int &t = cur[u]; t != -1; t = e[t].n){
42         int v = e[t].v;
43         if (e[t].w > 0 && lv[v] + 1 == lv[u]){
44             //res = min(maxf - ret, e[t].w);
45             res = maxf - ret;
46             if (e[t].w < res) res = e[t].w;
47             flow = dfs(v, res);
48             if (flow > 0){
49                 e[t].w -= flow;
50                 e[t^1].w += flow;
51                 ret += flow;
52                 if (maxf == ret) return ret;
53             }
54             //else lv[v] = -1;
55         }
56     }
57     return ret;
58 }
59 int main()
60 {
61     while(scanf("%d%d", &n, &m) != EOF)
62     {
63         int x, y, z;
64         sou = 1, tar = n;
65         esize = 0;
66         memset(en, -1, sizeof(en));
67         for (int i = 0; i < m; i++){
68             scanf("%d%d%d", &x, &y, &z);
69             addedge(x, y, z);
70             addedge(y, x, z);
71         }
72         //dinic();
73         long long maxflow = 0;
74         while(bfs()){
75             for (int i = 1; i <= n; i++) cur[i] = en[i];
76             maxflow = maxflow + dfs(sou, inf);
77         }
78         printf("%lld\n", maxflow);

```

```

79     }
80     return 0;
81 }

```

2.3 上下界流

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

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

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

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

2.3.1 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[l] = 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 }

```

2.3.2 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,
98                     tot);
99             }
100         }
101         cnt = 0;
102         for (int i = S; i <= T; i++)
103             if (flow[i] > 0) makelist(DS, i, flow[i], 0), makelist(i, DS, 0, 0), cnt
104                 += flow[i];
105             else makelist(i, DT, abs(flow[i]), 0), makelist(DT, i, 0, 0);
106         makelist(T, S, 1 << 30, 0); makelist(S, T, 0, 0);
107         ans = 0;
108         for (; check(DS, DT);) dinic(DS, DS, DT);
109         if (ans != cnt)
110         {
111             printf("-1\n\n");
112             continue;
113         }
114         else
115         {
116             v[1] = v[l - 1] = 0;
117             for (; check(S, T);) dinic(S, S, T);
118             printf("%d\n", ans);
119             for (int i = 3; i <= l; i += 2)
120                 if (what[i]) out[what[i]] += v[i];
121             for (int i = 1; i <= tot; i++) printf("%d\n", out[i]);

```

```

120         printf("\n");
121     }
122 }
123 }

```

2.3.3 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 }

```

2.3.4 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         }
97         bool ok = true;
98         for (int i = 1; i <= n && ok; i++)
99             for (int j = 1; j <= m; j++)
100                 if (down[i][j] > up[i][j])
101                 {
102                     ok = false;
103                     break;
104                 }
105         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                                     ,
121                                     makelist(j + n, i, 0);
122             }
123     DS = T + 1; DT = DS + 1;
124     int cnt = 0;
125     for (int i = S; i <= T; i++)
126         if (flow[i] > 0) makelist(DS, i, flow[i]), makelist(i, DS, 0), cnt +=
127             flow[i];
128         else if (flow[i] < 0) makelist(i, DT, abs(flow[i])), makelist(DT, i, 0);
129     makelist(T, S, 1 << 30); makelist(S, T, 0);
130     ans = 0;
131     for (; check();) dinic(DS);
132     if (ans != cnt)
133     {
134         printf("IMPOSSIBLE\n");
135         if (uu != test) printf("\n");
136         continue;
137     }
138     for (int i = 1; i <= n; i++)
139         for (int x = first[i]; x; x = next[x])
140             if (where[x] >= n + 1 && where[x] <= n + m)
141                 down[i][where[x] - n] += v[x ^ 1];
142     for (int i = 1; i <= n; i++)
143         for (int j = 1; j <= m; j++)
144             {
145                 printf("%d", down[i][j]);
146                 if (j != m) printf("_");
147                 else printf("\n");
148             }
149     if (uu != test) printf("\n");

```

2.4 费用流

2.4.1 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 }

```

2.4.2 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 }

```

2.5 强联通分量

2.5.1 递归

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

```

1 int totID, totCol;
2 int col[MAX_N], low[MAX_N], dfn[MAX_N];
3 int top, stack[MAX_N], instack[MAX_N];

```

```

4
5 int tarjan(int u)
6 {
7     low[u] = dfn[u] = ++ totID;
8     instack[u] = true; stack[++ top] = u;
9
10    int v;
11    foreach(it, adj[u]) {
12        v = it->first;
13        if (dfn[v] == -1)
14            low[u] = min(low[u], tarjan(v));
15        else if (instack[v])
16            low[u] = min(low[u], dfn[v]);
17    }
18
19    if (low[u] == dfn[u]) {
20        do {
21            v = stack[top--];
22            instack[v] = false;
23            col[v] = totCol;
24        } while (v != u);
25        ++ totCol;
26    }
27    return low[u];
28 }
29
30 void solve()
31 {
32     totID = totCol = top = 0;
33     fill(dfn, dfn + N, 0);
34     for(int i = 0; i < N; ++ i)
35         if (! dfn[i])
36             tarjan(i);
37 }

```

2.5.2 手写栈

```

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[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 }

```

2.6 最近公共祖先

```

1 #include<iostream>
2 #include<cmath>
3 #include<cstdio>

```



```

4  #include<cstdlib>
5  #include<cstring>
6  #include<string>
7  using namespace std;
8  const int maxn=20000;
9  int pre[maxn];
10 int other[maxn];
11 int last[maxn];
12 int father[maxn];
13 int v[maxn];
14 int deep[maxn];
15 int que[maxn];
16 int f[maxn][105];
17 int n,root,l;
18 void add_edge(int a,int b)
19 {
20     pre[++l]=last[a]; last[a]=l; other[l]=b; father[b]=a;
21 }
22 void init()
23 {
24     int a,b;
25     memset(pre,0,sizeof(pre));
26     memset(other,0,sizeof(other));
27     memset(last,0,sizeof(last));
28     memset(father,0,sizeof(father));
29     memset(v,0,sizeof(v));
30     memset(deep,0,sizeof(deep));
31     memset(f,0,sizeof(f));
32     memset(que,0,sizeof(que));
33     scanf("%d",&n);
34     for (int i=1;i<n;i++)
35     {
36         scanf("%d%d",&a,&b);
37         add_edge(a,b);
38         v[b]=1;
39     }
40     for (int i=1;i<=n;i++)
41         if (!v[i]) root=i;
42 }
43 void bfs()
44 {
45     deep[root]=1;
46     int l=0,r=1; que[1]=root;
47     while (l!=r)
48     {
49         int x=que[++l];
50         for (int p=last[x]; p!=0; p=pre[p])
51         {
52             que[++r]=other[p];

```

```

53         deep[other[p]] = deep[x] + 1;
54     }
55 }
56 }
57 void prepare()
58 {
59     for (int i=1; i<=n; i++)
60         f[i][0] = father[i];
61     for (int i=1; i<=100; i++)
62     {
63         for (int j=1; j<=n; j++)
64             f[j][i] = f[f[j][i-1]][i-1];
65     }
66 }
67 int lca(int x, int y)
68 {
69     if (x==y) return x;
70     if (deep[x]<deep[y]) swap(x,y);
71     int t=deep[x]-deep[y];
72     for (int i=0; t!=0; i++, t=t>>1)
73         if ((t & 1)==1) x=f[x][i];
74     if (x==y) return x;
75     for (int i=0; x!=y; i++)
76     {
77         if ((f[x][i]!=f[y][i]) || ((f[x][i]==f[y][i]) && (i==0)))
78         {
79             x=f[x][i];
80             y=f[y][i];
81             i++;
82         }
83         else i--;
84     }
85     return x;
86 }
87 void solve()
88 {
89     int x,y;
90     bfs();
91     scanf("%d%d",&x,&y);
92     printf("%d\n",lca(x,y));
93 }
94 int main()
95 {
96
97     int pp;
98     scanf("%d",&pp);
99     for (int i=1; i<=pp; i++)
100     {
101         l=0;

```

```

102         init();
103         prepare();
104         solve();
105     }
106 }

```

2.7 KM

2.7.1 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[
64             i][j];
65     km();
66     for (int i = 1; i <= n; i++)
67         for (int j = 1; j <= m; j++)
68             value[i][j] = -value[i][j];
69     km();
70 }
71
72 /*hint 500 * 500 1.5s
73 can solve problems whose n != m
74 must be complete graph, or should change some values of matrix to satisfy the
75 condition*/

```

2.7.2 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

```

Chapter 3

数据结构

3.1 KD 树

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 = 2000000000000000000LL;
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 }

```

3.2 Splay 树

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(1, 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 }

```

3.3 区间第 k 大

3.3.1 动态

```

1  #include<iostream>
2  #include<cstdio>
3  #include<cstring>
4  #include<cstdlib>
5  using namespace std;
6  const int max_n=200000+10;
7  const int tree_size=1000000+10;
8  int n,m,tot;
9  int a[max_n],ll[max_n],rr[max_n],root[max_n],lef[tree_size],rig[tree_size],key[
    tree_size],s[tree_size];
10 void l_rotate(int &t)
11 {
12     int k=rig[t]; rig[t]=lef[k]; lef[k]=t;
13     s[k]=s[t]; s[t]=s[lef[t]]+s[rig[t]]+1;
14     t=k;
15 }
16 void r_rotate(int &t)
17 {
18     int k=lef[t]; lef[t]=rig[k]; rig[k]=t;
19     s[k]=s[t]; s[t]=s[lef[t]]+s[rig[t]]+1;
20     t=k;
21 }
22 void maintain(int &t,int flag)
23 {
24     if (flag)
25     {
26         if (s[lef[lef[t]]]>s[rig[t]]) r_rotate(t);
27         else if (s[rig[lef[t]]]>s[rig[t]]) l_rotate(lef[t]),r_rotate(t);
28         else return;
29     } else
30     {

```

```

31         if (s[rig[rig[t]]] > s[lef[t]]) l_rotate(t);
32         else if (s[lef[rig[t]]] > s[lef[t]]) r_rotate(rig[t], l_rotate(t));
33         else return;
34     }
35     maintain(lef[t], 1); maintain(rig[t], 0);
36     maintain(t, 1); maintain(t, 0);
37 }
38 void insert(int &t, int x)
39 {
40     if (t == 0)
41     {
42         t = ++tot;
43         lef[t] = rig[t] = 0;
44         s[t] = 1; key[t] = x;
45         return;
46     }
47     ++s[t];
48     if (x < key[t]) insert(lef[t], x); else insert(rig[t], x);
49     maintain(t, x < key[t]);
50 }
51 int Delete(int &t, int x)
52 {
53     s[t]--;
54     if (key[t] == x || x < key[t] && lef[t] == 0 || x > key[t] && rig[t] == 0)
55     {
56         int tmp = key[t];
57         if (lef[t] == 0 || rig[t] == 0) t = lef[t] + rig[t];
58         else key[t] = Delete(lef[t], x + 1);
59         return tmp;
60     }
61     if (x < key[t]) return Delete(lef[t], x);
62     return Delete(rig[t], x);
63 }
64 int rank(int t, int x)
65 {
66     if (t == 0) return 0;
67     if (x <= key[t]) return rank(lef[t], x);
68     return s[lef[t]] + 1 + rank(rig[t], x);
69 }
70 void build(int t, int l, int r)
71 {
72     ll[t] = l; rr[t] = r; root[t] = 0;
73     if (l == r) return;
74     int mid = l + r >> 1;
75     build(t + t, l, mid);
76     build(t + t + 1, mid + 1, r);
77 }
78 void ins(int t, int pos, int x)
79 {

```

```

80     insert(root[t],x);
81     if (ll[t]==rr[t]) return;
82     int mid=ll[t]+rr[t]>>1;
83     if (pos<=mid) ins(t+t,pos,x);
84     else ins(t+t+1,pos,x);
85 }
86 void del(int t,int pos,int x)
87 {
88     Delete(root[t],x);
89     if (ll[t]==rr[t]) return;
90     int mid=ll[t]+rr[t]>>1;
91     if (pos<=mid) del(t+t,pos,x);
92     else del(t+t+1,pos,x);
93 }
94 int get_kth(int t,int l,int r,int x)
95 {
96     if (l<=ll[t] && r>=rr[t]) return rank(root[t],x);
97     int ans=0;
98     int mid=ll[t]+rr[t]>>1;
99     if (l<=mid) ans+=get_kth(t+t,l,r,x);
100    if (r>mid) ans+=get_kth(t+t+1,l,r,x);
101    return ans;
102 }
103 void init()
104 {
105     scanf("%d%d",&n,&m); tot=0;
106     for (int i=1;i<=n;i++)scanf("%d",&a[i]);
107     build(1,1,n);
108     for (int i=1;i<=n;i++)
109         ins(1,i,a[i]);
110 }
111 int query(int l,int r,int k)
112 {
113     int p=-1,q=1000000000+1,mid;
114     while (p+1!=q)
115     {
116         mid=(p+q)>>1;
117         if (get_kth(1,l,r,mid)<k) p=mid; else q=mid;
118     }
119     return p;
120 }
121
122 void solve()
123 {
124     char str[5];
125     int st,en,k,delta;
126     for (int i=1;i<=m;i++)
127     {
128         scanf("%s",str);

```

```

129         if (str[0] == 'Q')
130         {
131             scanf("%d%d%d", &st, &en, &k);
132             printf("%d\n", query(st, en, k));
133         } else
134         {
135             scanf("%d%d", &k, &delta);
136             del(1, k, a[k]);
137             a[k] = delta;
138             ins(1, k, a[k]);
139         }
140     }
141 }
142 int main()
143 {
144     int M;
145     for (scanf("%d", &M); M; --M)
146     {
147         init();
148         solve();
149     }
150     // system("pause");
151     return 0;
152 }

```

3.3.2 树套树 treap

```

1  #include <cstring>
2  #include <iostream>
3  #include <algorithm>
4  #include <cstdio>
5  using namespace std;
6  int Scan() //输入外挂
7  {
8      int res = 0, ch, flag = 0;
9      if ((ch = getchar()) == '-')
10         flag = 1;
11     else if (ch >= '0' && ch <= '9')
12         res = ch - '0';
13     while ((ch = getchar()) >= '0' && ch <= '9')
14         res = res * 10 + ch - '0';
15     return flag ? -res : res;
16 }
17
18 #define N 400010
19 #define M 400010
20 #define INF 1000000000
21

```



```

22  int  ctrl[M];
23  int  cnt,n,m;
24  int  P[M],Q[M],a[N],b[N],K[M];
25
26  struct  treap
27  {
28      int  key,wei,cnt,size,ch[2];
29  }T[N * 15];
30
31  int  tree[N << 1],nodecnt,root;
32
33  void  init()
34  {
35      T[0].size = 0;
36      T[0].wei = -INF;
37      nodecnt = root = 0;
38  }
39
40  int  ID(int l,int r)
41  {
42      return l + r | l != r;
43  }
44
45  void  update(int x)
46  {
47      T[x].size = T[T[x].ch[0]].size + T[T[x].ch[1]].size + T[x].cnt;
48  }
49
50  void  rotate(int &x,int t)
51  {
52      int y = T[x].ch[t];
53      T[x].ch[t] = T[y].ch[!t];
54      T[y].ch[!t] = x;
55      update(x);
56      update(y);
57      x = y;
58  }
59
60  void  insert(int &x,int t)
61  {
62      if (!x)
63      {
64          x = ++ nodecnt;
65          T[x].key = t;
66          T[x].wei = rand();
67          T[x].cnt = 1;
68          T[x].ch[0] = T[x].ch[1] = 0;
69      }
70      else if (T[x].key == t)

```

```

71     T[x].cnt ++;
72     else
73     {
74         int k = T[x].key < t;
75         insert(T[x].ch[k], t);
76         if (T[x].wei < T[T[x].ch[k]].wei)
77             rotate(x, k);
78     }
79     update(x);
80 }
81
82 void erase(int &x, int t)
83 {
84     if (T[x].key == t)
85     {
86         if (T[x].cnt == 1)
87         {
88             if (!T[x].ch[0] && !T[x].ch[1])
89             {
90                 x = 0;
91                 return;
92             }
93             rotate(x, T[T[x].ch[0]].wei < T[T[x].ch[1]].wei);
94             erase(x, t);
95         }
96         else T[x].cnt--;
97     }
98     else
99         erase(T[x].ch[T[x].key < t], t);
100    update(x);
101 }
102
103 int select(int x, int t)
104 {
105     if (!x) return 0;
106     if (T[x].key > t) return select(T[x].ch[0], t);
107     return T[x].cnt + T[T[x].ch[0]].size + select(T[x].ch[1], t);
108 }
109
110 void treeins(int l, int r, int i, int x)
111 {
112     insert(tree[ID(l, r)], x);
113     if (l == r) return;
114     int m = l + r >> 1;
115     if (i <= m) treeins(l, m, i, x);
116     else treeins(m + 1, r, i, x);
117 }
118
119 void treedel(int l, int r, int i, int x)

```

```

120 {
121     erase(tree[ID(1,r)],x);
122     if (l == r) return;
123     int m = l + r >> 1;
124     if (i <= m) treedel(1,m,i,x);
125     else treedel(m+1,r,i,x);
126 }
127
128 int query(int l,int r,int x,int y,int t)
129 {
130     if (l == r) return l;
131     int m = l + r >> 1;
132     int ans = select(tree[ID(1,m)],y) - select(tree[ID(1,m)],x);
133     if (ans >= t) return query(1,m,x,y,t);
134     return query(m+1,r,x,y,t - ans);
135 }
136
137 int main()
138 {
139
140     while (~scanf("%d",&n))
141     {
142         memset(tree,0,sizeof tree);
143         init();
144         cnt = 0;
145         for (int i = 1;i <= n;i ++)
146             {a[i]=Scan();b[++ cnt] = a[i];}
147         m=Scan();
148         for (int i = 1;i <= m;i ++)
149         {
150             ctrl[i]=Scan();P[i]=Scan();Q[i]=Scan();
151             //scanf("%s%d%d",&ctrl[i],&P[i],&Q[i]);
152             if (ctrl[i] == 2)
153                 {K[i]=Scan();} //scanf("%d",&K[i]);
154             else
155                 b[++ cnt] = Q[i];
156         }
157         sort(b+1,b+1+cnt);
158         cnt = unique(b+1,b+1+cnt) - b - 1;
159         for (int i = 1;i <= n;i ++)
160         {
161             a[i] = lower_bound(b+1,b+1+cnt,a[i]) - b;
162             treeins(1,cnt,a[i],i);
163         }
164         for(int i = 1;i <= m;i ++)
165         {
166             if (ctrl[i] == 2)
167             {
168                 int id = query(1,cnt,P[i] - 1,Q[i],K[i]);

```

```

169         printf("%d\n", b[id]);
170     }
171     else
172     {
173         treedel(1, cnt, a[P[i]], P[i]);
174         a[P[i]] = lower_bound(b + 1, b + 1 + cnt, Q[i]) - b;
175         treeins(1, cnt, a[P[i]], P[i]);
176     }
177 }
178 }
179 return 0;
180 }

```

3.4 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 };

```

3.5 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    };

```

3.6 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 }

```

3.7 扩展 KMP

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

```

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 }

```

3.8 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[
9             i]]) {
10            len[i] += 1;
11        }
12        if (q + len[i] - 1 > r) {

```

```

12         j = i;
13     }
14 }
15 }

```

3.9 AC 自动机

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 }

```

3.10 后缀数组

3.10.1 倍增

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 }

```

3.10.2 DC3

```

1  //DC3 待排序的字符串放在 r 数组中,从 r[0]到 r[n-1],长度为 n,且最大值小于 m。
2  //约定除 r[n-1]外所有的 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[0]到 sa[n-1]。
4  //r必须开长度乘 3
5  #define maxn 10000
6  #define F(x) ((x)/3+((x)%3==1?0:tb))
7  #define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
8  int wa[maxn],wb[maxn],wv[maxn],wss[maxn];
9  int s[maxn*3],sa[maxn*3];
10 int c0(int *r,int a,int b)
11 {
12     return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];
13 }
14 int c12(int k,int *r,int a,int b)
15 {
16     if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
17     else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
18 }
19 void sort(int *r,int *a,int *b,int n,int m)
20 {
21     int i;
22     for(i=0;i<n;i++) wv[i]=r[a[i]];
23     for(i=0;i<m;i++) wss[i]=0;
24     for(i=0;i<n;i++) wss[wv[i]]++;
25     for(i=1;i<m;i++) wss[i]+=wss[i-1];
26     for(i=n-1;i>=0;i--) b[--wss[wv[i]]]=a[i];
27 }
28 void dc3(int *r,int *sa,int n,int m)
29 {
30     int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
31     r[n]=r[n+1]=0;
32     for(i=0;i<n;i++)
33         if(i%3!=0) wa[tbc++]=i;
34     sort(r+2,wa,wb,tbc,m);
35     sort(r+1,wb,wa,tbc,m);
36     sort(r,wa,wb,tbc,m);
37     for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
38         rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
39     if(p<tbc) dc3(rn,san,tbc,p);

```

```

40     else for ( i=0;i<tbc ; i++) san [rn [ i ]]= i ;
41     for ( i=0;i<tbc ; i++)
42         if (san [ i]<tb) wb [ ta++]=san [ i ]*3;
43     if (n%3==1) wb [ ta++]=n-1;
44     sort ( r ,wb ,wa ,ta ,m);
45     for ( i=0;i<tbc ; i++)
46         wv [wb [ i ]]=G (san [ i ])= i ;
47     for ( i=0,j=0,p=0;i<ta && j<tbc ; p++)
48         sa [p]=c12 (wb [ j ]%3,r ,wa [ i ] ,wb [ j ] ) ?wa [ i++]:wb [ j++];
49     for (; i<ta ; p++) sa [p]=wa [ i++];
50     for (; j<tbc ; p++) sa [p]=wb [ j++];
51 }
52 int main () {
53     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 4

杂

4.1 $m^2 \log n$ 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  $O(m^2 \log(n))$ .
3 //  $a[n] = \sum(c[m-i] * a[n-i])$ ,  $i = 1 \dots m$ 
4 // i.e.  $a[m] = \sum(c[i] * a[i])$ ,  $i = 0 \dots m-1$ 
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 }
```

4.2 FFT

```
1 #include <complex>
2 #include <algorithm>
```

```

3  #include <cmath>
4  #include <vector>
5
6  using namespace std;
7
8  typedef complex <double> Complex;
9  typedef vector <int> Polynomial;
10
11 const double PI = acos(-1.);
12 const int N = 1 << 17;
13
14 void FFT(Complex* P, int n, int oper) {
15     for (int i = 1, j = 0; i < n - 1; i++) {
16         for (int s = n; j ^= s >>= 1, ~j & s;) {
17             if (i < j) {
18                 swap(P[i], P[j]);
19             }
20         }
21         Complex unit_p0;
22         for (int d = 0; (1 << d) < n; d++) {
23             int m = 1 << d, m2 = m * 2;
24             double p0 = PI / m * oper;
25             unit_p0 = Complex(cos(p0), sin(p0));
26             for (int i = 0; i < n; i += m2) {
27                 Complex unit = 1;
28                 for (int j = 0; j < m; j++) {
29                     Complex &P1 = P[i + j + m], &P2 = P[i + j];
30                     Complex t = unit * P1;
31                     P1 = P2 - t;
32                     P2 = P2 + t;
33                     unit = unit * unit_p0;
34                 }
35             }
36         }
37     }
38
39     Complex A[N], B[N];
40
41     Polynomial operator * (const Polynomial &u, const Polynomial &v)
42     {
43         int n=1,p=u.size(),q=v.size(),r=p+q-1,i;
44         while (n<=p+q-2) n<<=1;
45         for (i=0;i<n;++i) A[i]=i<p?u[i]:0;
46         for (i=0;i<n;++i) B[i]=i<q?v[i]:0;
47         FFT(A,n,1);
48         FFT(B,n,1);
49         for (i=0;i<n;++i) A[i]*=B[i];
50         FFT(A,n,-1);
51         Polynomial w(p+q-1);

```

```

52     for (i=0;i<r;++i)
53         w[i]=(int)(A[i].real()/n+0.5);
54     return w;
55 }

```

4.3 线性筛莫比乌斯

```

1  void prepare()
2  {
3      mu[1] = 1;
4      for (int i = 2 ; i <= 50000; i++)
5      {
6          if (!mark[i])
7          {
8              pr[++tot] = i;
9              mu[i] = -1;
10         }
11         for (int j = 1; j <= tot && i * pr[j] <= 50000; j++)
12         {
13             mark[i * pr[j]] = 1;
14             if (i % pr[j] == 0)
15             {
16                 mu[i * pr[j]] = 0;
17                 break;
18             }
19             else mu[i*pr[j]] = -mu[i];
20         }
21     }
22     for (int i = 1 ; i <= 50000; i++)
23         sum[i] = sum[i - 1] + mu[i];
24 }

```

4.4 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 }

```

4.5 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 }

```

4.6 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 }

```

4.7 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 }

```

4.8 Lucas 快速取 mod

附加移位乘法

```

1
2 long long fast_mod(long long a , long long b , long long mod)
3 {
4     long long ans = 1;
5     a %= mod;
6     while (b)
7     {
8         if (b & 1)
9         {
10            ans = ans * a % mod;
11            b --;
12        }
13        b >>= 1;
14        a = a * a % mod;
15    }
16    return ans;
17 }
18 long long lucas(long long n , long long m , long long mod)
19 {
20     if (m == 0) return 1;
21     return C(n % mod , m % mod , mod) * lucas(n / mod , m / mod, mod) % mod;
22 }

```

4.9 快速幂

```

1
2
3 #include<iostream>
4 #include<cstring>
5 #include<cstdio>
6 using namespace std;
7 const int N=55;
8 const int mod=2015;
9 struct Mat {
10     int mat[N][N];
11 };
12 int n,m;
13 Mat operator * (Mat a, Mat b) {
14     Mat c;

```

```

15     memset(c.mat, 0, sizeof(c.mat));
16     int i, j, k;
17     for(k = 0; k < n; ++k) {
18         for(i = 0; i < n; ++i) {
19             for(j = 0; j < n; ++j) {
20                 c.mat[i][j] = (c.mat[i][j] + a.mat[i][k] * b.mat[k][j]) % mod;
21             }
22         }
23     }
24     return c;
25 }
26 Mat operator ^ (Mat a, int k) {
27     Mat c;
28     int i, j;
29     for(i = 0; i < n; ++i)
30         for(j = 0; j < n; ++j)
31             c.mat[i][j] = (i == j);
32
33     for(; k; k >>= 1) {
34         if(k & 1) c = c * a;
35
36         a = a * a;
37     }
38     return c;
39 }

```

4.10 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 //  $a^x = b \pmod m$ 
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 }

```

4.11 n 次剩余

```

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_ordinroot(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 }

```

4.12 二次剩余

```

1  /*
2    $a*x^2+b*x+c==0 \pmod{P}$ 
3   求  $0..P-1$  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 }

```

4.13 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 }

```

4.14 字符串的最小表示

4.14.1 min

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

```

1  int minCycle(char *a)
2  {
3      int n = strlen(a);
4      for(int i = 0; i < n; ++i) {
5          a[i + n] = a[i];
6      }
7      a[n + n] = 0;
8      int i = 0, j = 1, k = 0;
9      do {
10         for(k = 0; a[i + k] == a[j + k]; ++k);
11         if (a[i + k] > a[j + k]) i = i + k + 1;
12         else j = j + k + 1;
13         j += i == j;
14         if (i > j) swap(i, j);
15     } while(j < n);
16     return i;
17 }

```

4.14.2 min_1

```

1  struct cyc_string
2  {
3      int n, offset;
4      char str[max_length];
5      char & operator [] (int x)
6      {return str[((offset + x) % n)];}
7      cyc_string(){offset = 0;}
8  };
9  void minimum_circular_representation(cyc_string & a)
10 {
11     int i = 0, j = 1, dlt = 0, n = a.n;
12     while(i < n and j < n and dlt < n)
13     {
14         if(a[i + dlt] == a[j + dlt]) dlt++;

```

```

15     else
16     {
17         if(a[i + dlt] > a[j + dlt]) i += dlt + 1; else j += dlt + 1;
18         dlt = 0;
19     }
20 }
21 a.offset = min(i, j);
22 }
23 int main()
24 {return 0;}

```

4.15 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 }

```

4.16 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 }

```

4.17 日期类解决两个日期之间差多少天

```

1 //日期类，构造函数参数可以是年月日，或是以公元元年一月一日作为第一天的第几天数。
   totalday 计算是第几天，whatday 计算是星期几。
2 //特判了1752年9月3日到9月13日，日历中没有这些日期。
3 //由于1700年2月有29日，所以之前的星期都会出错。（所以之后的天数也是错的）
4 #include<cstdio>
5 using namespace std;
6
7 bool isleap(int y)
8 {
9     if (y % 400 == 0 || (y % 100 != 0 && y % 4 == 0)) return true;
10    return false;
11 }
12
13 char Week[8][12] = {"", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "
   Saturday", "Sunday"};
14 int dayofmonth[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
15 struct Date{
16     int y, m, d;
17     Date(){}
18     Date(int totday){
19         //
20         if (totday > 639785) totday += 11; //特判1752年9月
21         //
22         y = totday / 366;
23         totday %= 366;
24         totday += y - (y/4 - y/100 + y/400);
25         y++;
26         for (int &year = y;; year++){
27             int del = 365 + isleap(year);
28             if (totday > del) totday -= del;
29             else break;
30         }
31         m = 1;
32         for (int &month = m; month < 12; month++){
33             int del = dayofmonth[month] + (month==2&&isleap(y));
34             if (totday > del) totday -= del;
35             else break;
36         }
37         d = totday;
38     }
39     Date(int _y, int _m, int _d): y(_y), m(_m), d(_d){}
40     int totalday(){
41         int leap = y/4 - y/100 + y/400;
42         if (isleap(y)) leap--;
43         int ret = (y-1) * 365 + leap;
44         for (int i = 1; i < m; i++) ret += dayofmonth[i];
45         if (isleap(y) && m > 2) ret++;
46         ret += d;
47         //

```

```

48         if (y > 1752) ret -= 11;           //特判 1752年9月
49         else if (y == 1752 && m > 9) ret -= 11;
50         else if (y == 1752 && m == 9 && d >= 14) ret -= 11;
51         //
52         return ret;
53     }
54     int whatday() {
55         int st = Date(1752, 9, 2).totalday(),
56             sum = totalday();
57         int del = sum - st;
58         if (del >= 0) {
59             del = (del % 7 + 3) % 7;
60             if (del == 0) del = 7;
61         }
62         else {
63             del = -del;
64             del %= 7;
65             del = 3 - del;
66             if (del <= 0) del += 7;
67         }
68         return del;
69     }
70 };
71
72 int main()
73 {
74     int y, m, d;
75     while (scanf("%d%d%d", &y, &m, &d))
76     {
77         Date ans = Date(y, m, d);
78         printf("%d-%02d-%02d%s\n", ans.y, ans.m, ans.d, Week[ans.whatday()]);
79     }
80 }

```

4.18 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("%.6lf\n", x[i]);
61     return 0;

```



```
62 }
```

4.19 有多少个点在多边形内

```
1 //rn中的标号必须逆时针给出。一开始要旋转坐标,保证同一个x值上只有一个点。正向减点,
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onit的方法,不要多算了线段上的点。
4 int ans=0,z,onit=0,low=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]; low+=on[i][j]-1;}
8     else ans+=num[i][j];
9 }
10 //ans-low+1 is inside. 只会多算一次正向上的点(除去最左和最右的点)。Low
    只算了除开最左边的点,但会多算最右边的点,所以要再加上1.
11 printf("%d\n",ans-low+1+onit);
```

4.20 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 }
```

4.21 杂知识

牛顿迭代

$x_1 = x_0 - \text{func}(x_0) / \text{func1}(x_0)$; 进行牛顿迭代计算

我们要求 $f(x)=0$ 的解。 $\text{func}(x)$ 为原方程, func1 为原方程的导数方程

图同构 Hash

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

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 $n = \prod p_i^{a_i}$, 则 $x^2 + y^2 = n$ 有整数解的充要条件是 n 中不存在形如 $p_i \mod 4 = 3$ 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 即是答案最小下届网络流。

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

4.22 补充公式

Catalan Number

通项形式：

$$C_n = \frac{2n!}{(n+1)!n!}$$

递推形式：

$$C_0 = 1, C_{n+1} = \frac{2(2n+1)}{n+2} C_n$$

Stirling Number

第一类：

$$\begin{bmatrix} n \\ k \end{bmatrix} = (n-1) \begin{bmatrix} n-1 \\ k \end{bmatrix} + \begin{bmatrix} n-1 \\ k-1 \end{bmatrix}, \sum_{k=1}^N \begin{bmatrix} n \\ k \end{bmatrix} = n!$$

第二类：

$$\begin{aligned} \binom{n}{k} &= k \binom{n-1}{k} + \binom{n-1}{k-1} \\ k=2, \binom{n}{k} &= 2^{(n-1)} - 1 \end{aligned}$$

错位排序

$$f(n) = (n-1) * (f(n-1) + f(n-2))$$

贝尔三角形

第一行第一个数是贝尔数，最后一个数是斯特林数

贝尔数是集合划分

贝尔三角形：

$$1, 3, 10, 37, \dots$$

$$f[i][0] = f[i-1][i-1]$$

$$f[i][j] = f[i][j-1] + f[i-1][j-1]$$

欧拉函数

pi 是 x 的质因数

$$\varphi(x) = x \prod (1 - 1/p_i)$$

欧拉定理

$$a^{\varphi(n)} \equiv 1(mod\ n)$$

费马小定理（欧拉定理特例）

$$a^{-1} \equiv a^{\varphi(n)-1}(mod\ n)$$

积性函数性质

$$\sum_{d|n} \varphi(d) = n$$

莫比乌斯函数

定义：

$$\mu(x) = \begin{cases} 1 & n=1 \\ (-1)^k & n = p_1p_2...p_n \\ 0 & \text{其余情况} \end{cases}$$

求和性质：

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

莫比乌斯反演：(f(n) 为积性，则 g(n) 也是)

$$g(n) = \sum_{d|n} f(d)$$

$$f(n) = \sum_{d|n} \mu(d)g(\frac{n}{d})$$

4.23 Language Reference

4.23.1 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

4.23.2 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); //  $a^b \bmod c$ ;
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); //  $(1 - 1/2^{\text{certainty}})$ .
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*

4.24 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>
```