Apocalypse

Standard Code Library

version 3.141

May, 2016

Contents

Chapter 1

二维几何

1.1 Naive Tips

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

1.2 几何公式

高维球

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

三角形

1. 半周长 $P = (a+b+c)/2$
2. 面积 $S = aH_a/2 = ab\sin(C)/2 = \sqrt{P(P-a)(P-b)(P-c)}$
3. 中线 $M_a = \sqrt{2(b^2+c^2)-a^2}/2 = \sqrt{b^2+c^2+2bc\cos(A)}/2$
4. 角平分线 $T_a = \sqrt{bc((b+c)^2-a^2)}/(b+c) = 2bc\cos(A/2)/(b+c)$
5. 高线 $H_a = b\sin(C) = c\sin(B) = \sqrt{b^2-((a^2+b^2-c^2)/(2a))^2}$

6. 内切圆半径

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

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

四边形

$D1, D2$ 为对角线，M 对角线中点连线，A 为对角线夹角

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

正 n 边形

R 为外接圆半径,r 为内切圆半径

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

圆

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

棱柱

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

棱锥

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

棱台

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

圆柱

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

圆锥

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

圆台

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

球

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

球台

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

球扇形

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

1.3 点类

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

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

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

```
        return x * p.x + y * p.y;
    }
    double det(const Point&p) const {
        return x * p.y - y * p.x;
    }
    double alpha() const {
        return atan2(y, x);
    }
    double distTo(const Point&p) const {
        double dx = x - p.x, dy = y - p.y;
        return hypot(dx, dy);
    }
    double alphaTo(const Point&p) const {
        double dx = x - p.x, dy = y - p.y;
        return atan2(dy, dx);
    }
    //clockwise
    Point rotAlpha(const double &alpha, const Point &o = Point(0, 0)) const {
        double nx = cos(alpha) * (x - o.x) + sin(alpha) * (y - o.y);
        double ny = -sin(alpha) * (x - o.x) + cos(alpha) * (y - o.y);
        return Point(nx, ny) + o;
    }
    Point rot90() const {
        return Point(-y, x);
    }
    Point unit() {
        return *this / abs();
    }
    void read() {
        scanf("%lf%lf", &x, &y);
    }
    double abs() {
        return hypot(x, y);
    }
    double abs2() {
        return x * x + y * y;
    }
    void write() {
        cout << "(" << x << "," << y << ")" << endl;
    }
};

#define cross(p1,p2,p3) ((p2.x-p1.x)*(p3.y-p1.y)-(p3.x-p1.x)*(p2.y-p1.y))
#define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))

Point isSS(Point p1, Point p2, Point q1, Point q2) {
    double a1 = cross(q1,q2,p1), a2 = -cross(q1,q2,p2);
    return (p1 * a2 + p2 * a1) / (a1 + a2);
}
```

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

1.4 基本操作

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

```
vector<Point> convexCut(const vector<Point>&ps, Point q1, Point q2) {
    vector<Point> qs;
    int n = ps.size();
    for (int i = 0; i < n; ++i) {
        Point p1 = ps[i], p2 = ps[(i + 1) % n];
        int d1 = crossOp(q1,q2,p1), d2 = crossOp(q1,q2,p2);
        if (d1 >= 0)
            qs.push_back(p1);
        if (d1 * d2 < 0)
            qs.push_back(isSS(p1, p2, q1, q2));
    }
    return qs;
}
```

返回 ps 的有向面积

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

返回点集 ps 组成的凸包

```
vector<Point> convexHull(vector<Point> ps) {
    int n = ps.size();
    if (n <= 1)
        return ps;
    sort(ps.begin(), ps.end());
    vector<Point> qs;
    for (int i = 0; i < n; qs.push_back(ps[i++])) {
        while (qs.size() > 1 && crossOp(qs[qs.size()-2],qs.back(),ps[i]) <= 0)
            qs.pop_back();
    }
    for (int i = n - 2, t = qs.size(); i >= 0; qs.push_back(ps[i--])) {
        while ((int)qs.size() > t && crossOp(qs[(int)qs.size()-2],qs.back(),ps[i]) <=
            0)
            qs.pop_back();
```

```
    }
    qs.pop_back();
    return qs;
}
```

返回凸包 ps 的直径

```
double convexDiameter(const vector<Point>&ps) {
    int n = ps.size();
    int is = 0, js = 0;
    for (int i = 1; i < n; ++i) {
        if (ps[i].x > ps[is].x)
            is = i;
        if (ps[i].x < ps[js].x)
            js = i;
    }
    double maxd = ps[is].distTo(ps[js]);
    int i = is, j = js;
    do {
        if ((ps[(i + 1) % n] - ps[i]).det(ps[(j + 1) % n] - ps[j]) >= 0)
            (++j) %= n;
        else
            (++i) %= n;
        maxd = max(maxd, ps[i].distTo(ps[j]));
    } while (i != is || j != js);
    return maxd;
}
```

判断点 p 在线段 q1q2 上，端点重合返回 true

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

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

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

判断直线 p1p2 和 q1q2 是否平行

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

返回点 p 到直线 uv 的距离

```
double distPointToLine(Point p, Point u, Point v)
{
```

```
    return abs((u - p).det(v - p)) / u.distTo(v);
}
```

判断点 q 是否在简单多边形 p 内部，边界返回 false

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

判断直线 l1l2 是否与圆相交，相切返回 true

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

判断圆与线段是否有公共点，线段在圆内部返回 true

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

判断圆与圆是否相交，外切或内切返回 true

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

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

求直线与圆的两个交点

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

求圆与圆的两个交点

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

圆外一点引圆的切线

```
void cutpoint(point c , point r , point sp ,point &rp1 ,point &rp2)
{
    point c2;
    double r2;
    c2 = (c + sp) / 2.0;
    r2 = (c2 - sp).abs();
    intersectionCircleToCircle(point c,double r,point c2 , double r2,point &rp1 ,
        point &rp2);
}
```

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,$0 \leq angle \leq \pi$

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

计算直线距离,r 为球半径

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

计算球面距离,r 为球半径

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

1.6 半平面交

```
struct Border {
    Point p1, p2;
    double alpha;
    void setAlpha() {
        alpha = atan2(p2.y - p1.y, p2.x - p1.x);
    }
    void read() {
        p1.read();
        p2.read();
        setAlpha();
    }
};

int n;
const int MAX_N_BORDER = 20000 + 10;
Border border[MAX_N_BORDER];

bool operator<(const Border&a, const Border&b) {
    int c = sign(a.alpha - b.alpha);
    if (c != 0)
        return c == 1;
    return crossOp(b.p1,b.p2,a.p1) >= 0;
}

bool operator==(const Border&a, const Border&b) {
    return sign(a.alpha - b.alpha) == 0;
}

const double LARGE = 10000;

void add(double x, double y, double nx, double ny) {
    border[n].p1 = Point(x, y);
```

```
        border[n].p2 = Point(nx, ny);
        border[n].setAlpha();
        n++;
}

Point isBorder(const Border&a, const Border&b) {
        return isSS(a.p1, a.p2, b.p1, b.p2);
}

Border que[MAX_N_BORDER];
int qh, qt;

bool check(const Border&a, const Border&b, const Border&me) {
        Point is = isBorder(a, b);
        return crossOp(me.p1,me.p2,is) > 0;
}

void convexIntersection() {
        qh = qt = 0;
        sort(border, border + n);
        n = unique(border, border + n) - border;
        for (int i = 0; i < n; ++i) {
                Border cur = border[i];
                while (qh + 1 < qt && !check(que[qt - 2], que[qt - 1], cur))
                        --qt;
                while (qh + 1 < qt && !check(que[qh], que[qh + 1], cur))
                        ++qh;
                que[qt++] = cur;
        }
        while (qh + 1 < qt && !check(que[qt - 2], que[qt - 1], que[qh]))
                --qt;
        while (qh + 1 < qt && !check(que[qh], que[qh + 1], que[qt - 1]))
                ++qh;
}

void calcArea() {
        static Point ps[MAX_N_BORDER];
        int cnt = 0;

        if (qt - qh <= 2) {
                puts("0.0");
                return;
        }

        for (int i = qh; i < qt; ++i) {
                int next = i + 1 == qt ? qh : i + 1;
                ps[cnt++] = isBorder(que[i], que[next]);
        }

```

```
    double area = 0;
    for (int i = 0; i < cnt; ++i) {
        area += ps[i].det(ps[(i + 1) % cnt]);
    }
    area /= 2;
    area = fabsl(area);
    cout.setf(ios::fixed);
    cout.precision(1);
    cout << area << endl;
}

void halfPlaneIntersection()
{
    cin >> n;
    for (int i = 0; i < n; ++i) {
        border[i].read();
    }
    add(0, 0, LARGE, 0);
    add(LARGE, 0, LARGE, LARGE);
    add(LARGE, LARGE, 0, LARGE);
    add(0, LARGE, 0, 0);

    convexIntersection();
    calcArea();
}
```

1.7 最小圆覆盖

```
#include<cmath>
#include<cstdio>
#include<algorithm>
using namespace std;
const double eps=1e-6;
struct couple
{
    double x, y;
    couple(){}
    couple(const double &xx, const double &yy)
    {
        x = xx; y = yy;
    }
} a[100001];
int n;
bool operator < (const couple & a, const couple & b)
{
    return a.x < b.x - eps or (abs(a.x - b.x) < eps and a.y < b.y - eps);
}
bool operator == (const couple & a, const couple & b)
```

```
{
    return !(a < b) and !(b < a);
}
inline couple operator - (const couple &a, const couple &b)
{
    return couple(a.x-b.x, a.y-b.y);
}
inline couple operator + (const couple &a, const couple &b)
{
    return couple(a.x+b.x, a.y+b.y);
}
inline couple operator * (const couple &a, const double &b)
{
    return couple(a.x*b, a.y*b);
}
inline couple operator / (const couple &a, const double &b)
{
    return a*(1/b);
}
inline double operator * (const couple &a, const couple &b)
{
    return a.x*b.y-a.y*b.x;
}
inline double len(const couple &a)
{
    return a.x*a.x+a.y*a.y;
}
inline double di2(const couple &a, const couple &b)
{
    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
}
inline double dis(const couple &a, const couple &b)
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
struct circle
{
    double r; couple c;
} cir;
inline bool inside(const couple & x)
{
    return di2(x, cir.c) < cir.r*cir.r+eps;
}
inline void p2c(int x, int y)
{
    cir.c.x = (a[x].x+a[y].x)/2;
    cir.c.y = (a[x].y+a[y].y)/2;
    cir.r = dis(cir.c, a[x]);
}
```

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

Chapter 2

图论

2.1 Dijkstra

求 s 到其他点的最短路

```
int used[MAX_N], dis[MAX_N];
void dijstra(int s) {
    fill(dis, dis + N, INF); dis[s] = 0;
    priority_queue<pair<int, int> > que;
    que.push(make_pair(-dis[s], s));
    while (!que.empty()) {
        int u = que.top().second; que.pop();
        if (used[u]) continue;
        used[u] = true;
        foreach(e, E[u])
            if (dis[u] + e->w < dis[e->t]) {
                dis[e->t] = dis[u] + e->w;
                que.push(make_pair(-dis[e->t], e->t));
            }
    }
}
```

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

```
struct Edge
{
    int t, c;
    Edge *n, *r;
} *E[MAX_V], edges[MAX_M], *totEdge;

Edge* makeEdge(int s, int t, int c)
```

```
{
    Edge *e = totEdge++;
    e->t = t; e->c = c; e->n = E[s];
    return E[s] = e;
}

void addEdge(int s, int t, int c)
{
    Edge *p = makeEdge(s, t, c), *q = makeEdge(t, s, 0);
    p->r = q; q->r = p;
}

int maxflow()
{
    static int  cnt     [MAX_V];
    static int  h       [MAX_V];
    static int  que     [MAX_V];
    static int  aug     [MAX_V];
    static Edge *cur    [MAX_V];
    static Edge *prev   [MAX_V];
    fill(h, h + cntN, cntN);
    memset(cnt, 0, sizeof cnt);
    int qt = 0, qh = 0; h[T] = 0;
    for(que[qt++] = T; qh < qt; ) {
        int u = que[qh++];
        ++cnt[h[u]];
        for(Edge *e = E[u]; e; e = e->n)
            if (e->r->c && h[e->t] == cntN) {
                h[e->t] = h[u] + 1;
                que[qt++] = e->t;
            }
    }
    memcpy(cur, E, sizeof E);
    aug[S] = INF; Edge *e;
    int flow = 0, u = S;
    while (h[S] < cntN) {
        for(e = cur[u]; e; e = e->n)
            if (e->c && h[e->t] + 1 == h[u])
                break;
        if (e) {
            int v = e->t;
            cur[u] = prev[v] = e;
            aug[v] = min(aug[u], e->c);
            if ((u = v) == T) {
                int by = aug[T];
                while (u != S) {
                    Edge *p = prev[u];
                    p->c -= by;
                    p->r->c += by;
```

```
                        u = p->r->t;
                    }
                    flow += by;
                }
        } else {
            if (!--cnt[h[u]]) return flow;
            h[u] = cntN;
            for(e = E[u]; e; e = e->n)
                if (e->c && h[u] > h[e->t] + 1)
                    h[u] = h[e->t] + 1, cur[u] = e;
            ++cnt[h[u]];
            if (u != S) u = prev[u]->r->t;
        }
    }
    return flow;
}
```

2.2.2 Dinic

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;

const long long inf = (long long)1e17 + 10;
struct edge{
    int v, w, n;
} e[60200];
int en[5010], lv[5010], q[5010], cur[5010];
int n, m, sou, tar, esize;
void addedge(int u, int v, int w){
    e[esize].v = v;
    e[esize].w = w;
    e[esize].n = en[u];
    en[u] = esize++;
}
bool bfs()
{
    memset(lv, -1, sizeof(lv));
    int head, tail;
    lv[tar] = head = tail = 0;
    q[tail++] = tar;
    while(head < tail){
        int u = q[head++];
        for (int t = en[u]; t != -1; t = e[t].n){
            int v = e[t].v;
            if (lv[v] == -1 && e[t^1].w > 0){
                lv[v] = lv[u] + 1;
```

```
                    if (v != sou) q[tail++] = v;
                }
            }
        }
        return lv[sou] != -1;
}
long long dfs(int u, long long maxf)
{
        if (maxf == 0) return 0;
        if (u == tar) return maxf;
        long long ret = 0, res, flow;
        for (int &t = cur[u]; t != -1; t = e[t].n){
            int v = e[t].v;
            if (e[t].w > 0 && lv[v] + 1 == lv[u]){
                //res = min(maxf - ret, e[t].w);
                res = maxf - ret;
                if (e[t].w < res) res = e[t].w;
                flow = dfs(v, res);
                if (flow > 0){
                    e[t].w -= flow;
                    e[t^1].w += flow;
                    ret += flow;
                    if (maxf == ret) return ret;
                }
                //else lv[v] = -1;
            }
        }
        return ret;
}
int main()
{
        while(scanf("%d %d", &n, &m) != EOF)
        {
            int x, y, z;
            sou = 1, tar = n;
            esize = 0;
            memset(en, -1, sizeof(en));
            for (int i = 0; i < m; i++){
                scanf("%d %d %d", &x, &y, &z);
                addedge(x, y, z);
                addedge(y, x, z);
            }
            //dinic();
            long long maxflow = 0;
            while(bfs()){
                for (int i = 1; i <= n; i++) cur[i] = en[i];
                maxflow = maxflow + dfs(sou, inf);
            }
            printf("%lld\n", maxflow);
```

```
    }
    return 0;
}
```

2.3 上下界流

上下界无源汇可行流: 不用添 T—>S，判断是否流量平衡
上下界无源汇可行流: 添 T—>S（下界 0，上界 oo），判断是否流量平衡
上下界最小流: 不添 T—>S 先流一遍，再添 T—>S（下界 0，上界 oo）在残图上流一遍，答案为 S—>T 的流量值
上下界最大流: 添 T—>S（下界 0，上界 oo）流一遍，再在残图上流一遍 S 到 T 的最大流，答案为前者的 S—>T 的值 + 残图中 S—>T 的最大流

2.3.1 上下界无源汇可行流

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

using namespace std;

struct {
        int x, y, down, up, what;
} a[100001];

int S, T, DS, DT, n, m, out[100001], what[100001], first[501], pre[501], way[501],
    len, dist[501], c[501], ans, flow[201], where[100001], next[100001], v[100001], l,
     cnt;

inline void makelist(int x, int y, int z, int q){
    where[++l] = y;
    v[l] = z;
    what[l] = q;
    next[l] = first[x];
    first[x] = l;
}

bool check(){
    memset(dist, 0, sizeof(dist));
    dist[DS] = 1; c[1] = DS;
    for (int k = 1, l = 1; l <= k; l++)
    {
        int m = c[l];
        if (m == DT)
```

```
        {
            len = dist[m] + 1;
            return(true);
        }
        for (int x = first[m]; x; x = next[x])
            if (v[x] && !dist[where[x]]) dist[where[x]] = dist[m] + 1, c[++k] = where[x];
    }
    return(false);
}

inline void dinic(int now){
    if (now == DT)
    {
        int Minflow = 1 << 30;
        for (; now != DS; now = pre[now]) Minflow = min(Minflow, v[way[now]]);
        ans += Minflow;
        for (now = DT; now != DS; now = pre[now])
        {
            v[way[now]] -= Minflow;
            v[way[now] ^ 1] += Minflow;
            if (!v[way[now]]) len = dist[now];
        }
        return;
    }
    for (int x = first[now]; x; x = next[x])
        if (v[x] && dist[now] + 1 == dist[where[x]])
        {
            pre[where[x]] = now;
            way[where[x]] = x;
            dinic(where[x]);
            if (dist[now] >= len) return;
            len = dist[DT] + 1;
        }
    dist[now] = -1;
}

int main(){
  //  freopen("194.in", "r", stdin);
  //  freopen("194.out", "w", stdout);
    scanf("%d%d", &n, &m);
    memset(flow, 0, sizeof(flow));
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d%d%d", &a[i].x, &a[i].y, &a[i].down, &a[i].up);
        flow[a[i].y] += a[i].down;
        flow[a[i].x] -= a[i].down;
    }
    cnt = 0;
```

```
    memset(first, 0, sizeof(first)); l = 1;
    S = 1; T = n; DS = 0; DT = n + 1; cnt = 0;
    for (int i = 1; i <= n; i++)
        if (flow[i] > 0) makelist(DS, i, flow[i], 0), makelist(i, DS, 0, 0), cnt +=
            flow[i];
                        else makelist(i, DT, abs(flow[i]), 0), makelist(DT, i, 0, 0);
//      makelist(T, S, 1 << 30, 0); makelist(S, T, 0, 0);
    for (int i = 1; i <= m; i++) makelist(a[i].x, a[i].y, a[i].up - a[i].down, i),
                                 makelist(a[i].y, a[i].x, 0, i);
    ans = 0;
    for (; check();) dinic(DS);
    if (ans != cnt) printf("NO\n");
    else
    {
        printf("YES\n");
        for (int i = 3; i <= l; i += 2)
            if (what[i]) out[what[i]] = v[i];
        for (int i = 1; i <= m; i++) printf("%d\n", a[i].down + out[i]);
    }
}
```

2.3.2 上下界最大流

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

using namespace std;

int n, m, S, T, DS, DT, a[1001], first[1501], next[100001], where[100001], v[100001],
    what[100001],
l, c[1501], dist[1501], len, pre[1501], way[1501], flow[1501], out[100001], tot, cnt,
    ans;

inline void makelist(int x, int y, int z, int q){
    where[++l] = y;
    v[l] = z;
    what[l] = q;
    next[l] = first[x];
    first[x] = l;
}

bool check(int S, int T){
    memset(dist, 0, sizeof(dist));
```

```
    c[1] = S; dist[S] = 1;
    for (int k = 1, l = 1; l <= k; l++)
    {
        int m = c[l];
        if (m == T)
        {
            len = dist[m] + 1;
            return(true);
        }
        for (int x = first[m]; x; x = next[x])
            if (v[x] && !dist[where[x]])
            {
                dist[where[x]] = dist[m] + 1;
                c[++k] = where[x];
            }
    }
    return(false);
}

inline void dinic(int now, int S, int T){
    if (now == T)
    {
        int Minflow = 1 << 30;
        for (; now != S; now = pre[now]) Minflow = min(Minflow, v[way[now]]);
        ans += Minflow;
        for (now = T; now != S; now = pre[now])
        {
            v[way[now]] -= Minflow;
            v[way[now] ^ 1] += Minflow;
            if (!v[way[now]]) len = dist[now];
        }
        return;
    }
    for (int x = first[now]; x; x = next[x])
        if (v[x] && dist[where[x]] == dist[now] + 1)
        {
            pre[where[x]] = now;
            way[where[x]] = x;
            dinic(where[x], S, T);
            if (dist[now] >= len) return;
            len = dist[T] + 1;
        }
    dist[now] = -1;
}

int main(){
//    freopen("3229.in", "r", stdin);
//    freopen("3229.out", "w", stdout);
    for (;;)
```

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

```
            printf("\n");
        }
    }
}
```

2.3.3 上下界最小流

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

using namespace std;

struct {
        int x, y, down, up;
} a[10001];
int out[100001], what[100001], cnt, S, T, DS, DT, l, ans, n, m, flow[101], first
    [201], next[100001], where[100001], v[100001], dist[201], c[201], pre[201], way
    [201], len;

int read(){
    char ch;
    for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
    int cnt = 0;
    for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
    return(cnt);
}

inline void makelist(int x, int y, int z, int q){
    where[++l] = y;
    v[l] = z;
    what[l] = q;
    next[l] = first[x];
    first[x] = l;
}

inline void makemap(){
    memset(first, 0, sizeof(first)); l = 1;
    S = 1, T = n, DS = 0, DT = n + 1; cnt = 0;
    for (int i = 1; i <= n; i++)
        if (flow[i] > 0) makelist(DS, i, flow[i], 0), makelist(i, DS, 0, 0), cnt +=
            flow[i];
        else makelist(i, DT, abs(flow[i]), 0), makelist(DT, i, 0, 0);
    for (int i = 1; i <= m; i++) makelist(a[i].x, a[i].y, a[i].up - a[i].down, i),
```

```
                makelist(a[i].y, a[i].x, 0, i);
}

bool check(){
    memset(dist, 0, sizeof(dist));
    dist[DS] = 1; c[1] = DS;
    for (int k = 1, l = 1; l <= k; l++)
    {
        int m = c[l];
        if (m == DT)
        {
            len = dist[m] + 1;
            return(true);
        }
        for (int x = first[m]; x; x = next[x])
            if (v[x] && !dist[where[x]]) dist[where[x]] = dist[m] + 1, c[++k] = where[x];
    }
    return(false);
}

inline void dinic(int now){
    if (now == DT)
    {
        int Minflow = 1 << 30;
        for (; now != DS; now = pre[now]) Minflow = min(Minflow, v[way[now]]);
        ans += Minflow;
        for (now = DT; now != DS; now = pre[now])
        {
            v[way[now]] -= Minflow;
            v[way[now] ^ 1] += Minflow;
            if (!v[way[now]]) len = dist[now];
        }
        return;
    }
    for (int x = first[now]; x; x = next[x])
        if (dist[where[x]] == dist[now] + 1 && v[x])
        {
            pre[where[x]] = now;
            way[where[x]] = x;
            dinic(where[x]);
            if (dist[now] >= len) return;
            len = dist[DT] + 1;
        }
    dist[now] = -1;
}

inline void network(){
    for (; check();) dinic(DS);
```

```
}

int main(){
    // freopen("176.in", "r", stdin);
    // freopen("176.out", "w", stdout);
    scanf("%d%d", &n, &m);
    memset(flow, 0, sizeof(flow));
    for (int i = 1; i <= m; i++)
    {
        a[i].x = read(), a[i].y = read(), a[i].up = read();
        int status = read();
        if (status) a[i].down = a[i].up;
        else a[i].down = 0;
        flow[a[i].y] += a[i].down;
        flow[a[i].x] -= a[i].down;
    }
    makemap();
    ans = 0;
    network();
    makelist(T, S, 1 << 30, 0); makelist(S, T, 0, 0);
    network();
    if (ans != cnt)
    {
        printf("Impossible\n");
        return 0;
    }
    printf("%d\n", v[l]);
    for (int i = 3; i <= l; i += 2)
        if (what[i]) out[what[i]] = v[i];
    for (int i = 1; i <= m; i++)
    {
        printf("%d", a[i].down + out[i]);
        if (i != m) printf(" ");
        else printf("\n");
    }
}
```

2.3.4 上下界有源汇可行流

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

using namespace std;
```

```cpp

int test, n, m, Q, first[501], a1[201], a2[201], flow[501], next[100001], where[100001], v[100001], len,
l, dist[501], c[501], up[201][201], down[201][201], S, T, DS, DT, ans, out[201][201], pre[501], way[501];

inline void makelist(int x, int y, int z){
    where[++l] = y;
    v[l] = z;
    next[l] = first[x];
    first[x] = l;
}

bool check(){
    memset(dist, 0, sizeof(dist));
    dist[DS] = 1; c[1] = DS;
    for (int k = 1, l = 1; l <= k; l++)
    {
        int m = c[l];
        if (m == DT)
        {
            len = dist[m] + 1;
            return(true);
        }
        for (int x = first[m]; x; x = next[x])
            if (v[x] && !dist[where[x]])
            {
                dist[where[x]] = dist[m] + 1;
                c[++k] = where[x];
            }
    }
    return(false);
}

inline void dinic(int now){
    if (now == DT)
    {
        int Minflow = 1 << 30;
        for (; now != DS; now = pre[now]) Minflow = min(Minflow, v[way[now]]);
        ans += Minflow;
        for (now = DT; now != DS; now = pre[now])
        {
            v[way[now]] -= Minflow;
            v[way[now] ^ 1] += Minflow;
            if (!v[way[now]]) len = dist[now];
        }
        return;
    }
    for (int x = first[now]; x; x = next[x])
```

```
        if (v[x] && dist[now] + 1 == dist[where[x]])
        {
            pre[where[x]] = now;
            way[where[x]] = x;
            dinic(where[x]);
            if (dist[now] >= len) return;
            len = dist[DT] + 1;
        }
    dist[now] = -1;
}

int main(){
//  freopen("2396.in", "r", stdin);
//  freopen("2396.out", "w", stdout);
    scanf("%d", &test);
    for (int uu = 1; uu <= test; uu++)
    {
        scanf("%d%d", &n, &m);
        for (int i = 1; i <= n; i++) scanf("%d", &a1[i]);
        for (int i = 1; i <= m; i++) scanf("%d", &a2[i]);
        memset(up, 127, sizeof(up));
        memset(down, 0, sizeof(down));
        scanf("%d", &Q);
        for (int i = 1; i <= Q; i++)
        {
            int x, y, z;
            char str[2];
            scanf("%d%d%s%d", &x, &y, str, &z);
            int L1, L2, R1, R2;
            if (x == 0) L1 = 1, R1 = n;
            else L1 = R1 = x;
            if (y == 0) L2 = 1, R2 = m;
            else L2 = R2 = y;
            for (int j = L1; j <= R1; j++)
                for (int k = L2; k <= R2; k++)
                    if (str[0] == '>') down[j][k] = max(down[j][k], z + 1);
                    else if (str[0] == '<') up[j][k] = min(up[j][k], z - 1);
                    else down[j][k] = max(down[j][k], z), up[j][k] = min(up[j][k], z);
        }
        bool ok = true;
        for (int i = 1; i <= n && ok; i++)
            for (int j = 1; j <= m; j++)
                if (down[i][j] > up[i][j])
                {
                    ok = false;
                    break;
                }
        if (!ok)
```

```
        {
            printf("IMPOSSIBLE\n");
            if (uu != test) printf("\n");
            continue;
        }
        memset(flow, 0, sizeof(flow));
        memset(first, 0, sizeof(first)); l = 1;
        S = 0; T = n + m + 1;
        for (int i = 1; i <= n; i++) flow[S] -= a1[i], flow[i] += a1[i];
        for (int i = 1; i <= m; i++) flow[i + n] -= a2[i], flow[T] += a2[i];
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= m; j++)
            {
                flow[i] -= down[i][j]; flow[j + n] += down[i][j];
                if (down[i][j] != up[i][j]) makelist(i, j + n, up[i][j] - down[i][j]),
                                            makelist(j + n, i, 0);
            }
        DS = T + 1; DT = DS + 1;
        int cnt = 0;
        for (int i = S; i <= T; i++)
            if (flow[i] > 0) makelist(DS, i, flow[i]), makelist(i, DS, 0), cnt += flow[i];
            else if (flow[i] < 0) makelist(i, DT, abs(flow[i])), makelist(DT, i, 0);
        makelist(T, S, 1 << 30); makelist(S, T, 0);
        ans = 0;
        for (; check();) dinic(DS);
        if (ans != cnt)
        {
            printf("IMPOSSIBLE\n");
            if (uu != test) printf("\n");
            continue;
        }
        for (int i = 1; i <= n; i++)
            for (int x = first[i]; x; x = next[x])
                if (where[x] >= n + 1 && where[x] <= n + m)
                    down[i][where[x] - n] += v[x ^ 1];
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= m; j++)
            {
                printf("%d", down[i][j]);
                if (j != m) printf(" ");
                else printf("\n");
            }
        if (uu != test) printf("\n");
    }
}
```

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

```
int flow, cost;

struct Edge
{
    int t, c, w;
    Edge *n, *r;
} *totEdge, edges[MAX_M], *E[MAX_V];

Edge* makeEdge(int s, int t, int c, int w)
{
    Edge *e = totEdge++;
    e->t = t; e->c = c; e->w = w; e->n = E[s];
    return E[s] = e;
}

void addEdge(int s, int t, int c, int w)
{
    Edge *st = makeEdge(s, t, c, w), *ts = makeEdge(t, s, 0, -w);
    st->r = ts; ts->r = st;
}

int SPFA()
{
    static int que[MAX_V];
    static int aug[MAX_V];
    static int in[MAX_V];
    static int dist[MAX_V];
    static Edge *prev[MAX_V];
    int qh = 0, qt = 0;

    int u, v;
    fill(dist, dist + cntN, INF); dist[S] = 0;
    fill(in, in + cntN, 0); in[S] = true;
    que[qt++] = S; aug[S] = INF;
    for( ; qh != qt; ) {
        u = que[qh]; qh = (qh + 1) % MAX_N;
        for(Edge *e = E[u]; e; e = e->n) {
            if (!e->c) continue;
            v = e->t;
            if (dist[v] > dist[u] + e->w) {
                dist[v] = dist[u] + e->w;
                aug[v] = min(aug[u], e->c);
                prev[v] = e;
                if (!in[v]) {
```

```
                        in[v] = true;
                        if (qh != qt && dist[v] <= dist[que[qh]]) {
                            qh = (qh - 1 + MAX_N) % MAX_N;
                            que[qh] = v;
                        } else {
                            que[qt] = v;
                            qt = (qt + 1) % MAX_N;
                        }
                    }
                }
            }
            in[u] = false;
        }

        if (dist[T] == INF) return false;
        cost += dist[T] * aug[T];
        flow += aug[T];
        for(u = T; u != S; ) {
            prev[u]->c -= aug[T];
            prev[u]->r->c += aug[T];
            u = prev[u]->r->t;
        }
        return true;
}

int minCostFlow()
{
    flow = cost = 0;
    while(SPFA());
    return cost;
}
```

2.4.2 ZKW

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

using namespace std;

int n, m, S, T, slk[1001], dist[1001], first[1001], l, c[1000001], next[1000001],
    where[1000001], ll[1000001], v[1000001];
bool b[1001];
long long ans1, ans2;
```

```
inline void makelist(int x, int y, int z, int p){
    where[++l] = y;
    ll[l] = z;
    v[l] = p;
    next[l] = first[x];
    first[x] = l;
}

inline void spfa(){
    memset(dist, 127, sizeof(dist));
    memset(b, false, sizeof(b));
    dist[T] = 0; c[1] = T;
    for (int k = 1, l = 1; l <= k; l++)
    {
        int m = c[l];
        b[m] = false;
        for (int x = first[m]; x; x = next[x])
            if (ll[x ^ 1] && dist[m] - v[x] < dist[where[x]])
            {
                dist[where[x]] = dist[m] - v[x];
                if (!b[where[x]]) b[where[x]] = true, c[++k] = where[x];
            }
    }
}

int zkw_work(int now, int cap){
    b[now] = true;
    if (now == T)
    {
        ans1 += cap;
        ans2 += (long long)cap * dist[S];
        return(cap);
    }
    int Left = cap;
    for (int x = first[now]; x; x = next[x])
        if (ll[x] && !b[where[x]])
           if (dist[now] == dist[where[x]] + v[x])
           {
               int use = zkw_work(where[x], min(Left, ll[x]));
               ll[x] -= use; ll[x ^ 1] += use;
               Left -= use;
               if (!Left) return(cap);
           }
           else slk[where[x]] = min(slk[where[x]], dist[where[x]] + v[x] - dist[now]);
    return(cap - Left);
}

```

```
bool relax(){
    int Min = 1 << 30;
    for (int i = 0; i <= T; i++)
        if (!b[i]) Min = min(Min, slk[i]);
    if (Min == 1 << 30) return(false);
    for (int i = 0; i <= T; i++)
        if (b[i]) dist[i] += Min;
    return(true);
}

inline void zkw(){
    ans1 = ans2 = 0;
    spfa();  //hint memset(dist, 0, sizeof(dist)); if all values of edges are
        nonnegative
    for (;;)
    {
        memset(slk, 127, sizeof(slk));
        for (;;)
        {
            memset(b, false, sizeof(b));
            if (!zkw_work(S, 1 << 30)) break;
        }
        if (!relax()) break;
    }
    printf("%I64d %I64d\n", ans1, ans2);
}

int main(){
    scanf("%d%d", &n, &m);
    S = 1; T = n;
    memset(first, 0, sizeof(first)); l = 1;
    for (int i = 1; i <= m; i++)
    {
        int x, y, z, q;
        scanf("%d%d%d%d", &x, &y, &z, &q);
        makelist(x, y, z, q); makelist(y, x, 0, -q);
    }
    zkw();
}
```

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

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

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

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

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

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

2.5.2 手写栈

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

using namespace std;

```

```
int n, m, first[10001], father[10001], dfn[10001], low[10001], c[10001], pos[10001], todo[10001],
cnt, len, next[2000001], where[2000001], l, kuai, Max, color[10001], number;
bool b[10001];

int read(){
    char ch;
    for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
    int cnt = 0;
    for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
    return(cnt);
}

inline void makelist(int x, int y){
    where[++l] = y;
    next[l] = first[x];
    first[x] = l;
}

inline void tarjan(int S){
    int now = S; todo[now] = first[now];
    for (;;)
    {
        if (!now) return;
        if (first[now] == todo[now])
        {
            b[now] = true;
            dfn[now] = low[now] = ++cnt;
            c[++len] = now; pos[now] = len;
        }
        int x = todo[now];
        if (!x)
        {
            if (father[now])
                low[father[now]] = min(low[father[now]], low[now]);
            int delta = -1;
            if (father[now]) ++delta;
            for (int x = first[now]; x; x = next[x])
                if (father[where[x]] == now)
                    if (low[where[x]] >= dfn[now]) ++delta;
            Max = max(Max, delta);
            if (low[now] == dfn[now])
            {
               ++number;
               for (int i = pos[now]; i <= len; i++) color[c[i]] = number;
               len = pos[now] - 1;
            }
            now = father[now];
            continue;
```

```
        }
        todo[now] = next[todo[now]];
        if (father[now] != where[x])
            if (!b[where[x]])
            {
                father[where[x]] = now;
                now = where[x];
                todo[now] = first[now];
                continue;
            }
            else if (!color[where[x]]) low[now] = min(low[now], dfn[where[x]]);
    }
}

int main(){
   // freopen("2117.in", "r", stdin);
   // freopen("2117.out", "w", stdout);
    for (;;)
    {
        n = read(); m = read();
        if (!n && !m) return 0;
        memset(first, 0, sizeof(first));
        l = 0;
        for (int i = 1; i <= m; i++)
        {
            int x = read() + 1, y = read() + 1;
            makelist(x, y);
            makelist(y, x);
        }
        memset(dfn, 0, sizeof(dfn));
        memset(low, 0, sizeof(low));
        memset(color, 0, sizeof(color));
        memset(b, false, sizeof(b));
        memset(father, 0, sizeof(father));
        cnt = 0; len = 0;
        Max = - (1 << 30);
        kuai = 0; number = 0;
        for (int i = 1; i <= n; i++)
            if (!b[i]) tarjan(i), ++kuai;
        printf("%d\n", kuai + Max);
    }
}
```

2.6 最近公共祖先

```
#include<iostream>
#include<cmath>
#include<cstdio>
```

```
#include<cstdlib>
#include<cstring>
#include<string>
using namespace std;
const int maxn=20000;
int pre[maxn];
int other[maxn];
int last[maxn];
int father[maxn];
int v[maxn];
int deep[maxn];
int que[maxn];
int f[maxn][105];
int n,root,l;
void add_edge(int a,int b)
{
    pre[++l]=last[a];last[a]=l;other[l]=b;father[b]=a;
}
void init()
{
    int a,b;
    memset(pre,0,sizeof(pre));
    memset(other,0,sizeof(other));
    memset(last,0,sizeof(last));
    memset(father,0,sizeof(father));
    memset(v,0,sizeof(v));
    memset(deep,0,sizeof(deep));
    memset(f,0,sizeof(f));
    memset(que,0,sizeof(que));
    scanf("%d",&n);
    for (int i=1;i<n;i++)
    {
        scanf("%d%d",&a,&b);
        add_edge(a,b);
        v[b]=1;
    }
    for (int i=1;i<=n;i++)
    if (!v[i]) root=i;
}
void bfs()
{
    deep[root]=1;
    int l=0,r=1;que[1]=root;
    while (l!=r)
    {
        int x=que[++l];
        for (int p=last[x];p!=0;p=pre[p])
        {
            que[++r]=other[p];
```

```
            deep[other[p]]=deep[x]+1;
        }
    }
}
void prepare()
{
    for (int i=1;i<=n;i++)
    f[i][0]=father[i];
    for (int i=1;i<=100;i++)
    {
        for (int j=1;j<=n;j++)
        f[j][i]=f[f[j][i-1]][i-1];
    }
}
int lca(int x,int y)
{
    if (x==y) return x;
    if (deep[x]<deep[y]) swap(x,y);
    int t=deep[x]-deep[y];
    for (int i=0;t!=0;i++,t=t>>1)
    if ((t & 1)==1) x=f[x][i];
    if (x==y) return x;
    for (int i=0;x!=y;)
    {
        if ((f[x][i]!=f[y][i]) || ((f[x][i]==f[y][i]) && (i==0)))
        {
            x=f[x][i];
            y=f[y][i];
            i++;
        }
        else i--;
    }
    return x;
}
void solve()
{
    int x,y;
    bfs();
    scanf("%d%d",&x,&y);
    printf("%d\n",lca(x,y));
}
int main()
{

    int pp;
    scanf("%d",&pp);
    for (int i=1;i<=pp;i++)
    {
        l=0;
```

```
        init();
        prepare();
        solve();
    }
}
```

2.7 KM

2.7.1 邻接阵

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

using namespace std;

const int oo = 1 << 30;
int ans, value[501][501], n, m, L[501], R[501], v[501];
bool bx[501], by[501];


bool find(int now){
    bx[now] = true;
    for (int i = 1; i <= m; i++)
        if (!by[i] && L[now] + R[i] == value[now][i])
        {
            by[i] = true;
            if (!v[i] || find(v[i]))
            {
                v[i] = now;
                return(true);
            }
        }
    return(false);
}

inline void km(){
    memset(L, 0, sizeof(L));
    memset(R, 0, sizeof(R));
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            L[i] = max(L[i], value[i][j]);
    ans = 0;
    memset(v, 0, sizeof(v));
```

```
    for (int i = 1; i <= min(n, m); i++)
        for (;;)
        {
            memset(bx, false, sizeof(bx));
            memset(by, false, sizeof(by));
            if (find(i)) break;
            int Min = oo;
            for (int j = 1; j <= n; j++)
                if (bx[j])
                    for (int k = 1; k <= m; k++)
                        if (!by[k])
                            Min = min(Min, L[j] + R[k] - value[j][k]);
            for (int j = 1; j <= n; j++) if (bx[j]) L[j] -= Min;
            for (int j = 1; j <= m; j++) if (by[j]) R[j] += Min;
        }
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            if (v[j] == i) ans += value[i][j];
    printf("%d\n", abs(ans));
}

int main(){
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++) scanf("%d", &value[i][j]), value[i][j] = -value[i][j];
    km();
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            value[i][j] = -value[i][j];
    km();
}

/*hint 500 * 500 1.5s
can solve problems whose n != m
must be complete graph, or should change some values of matrix to satisfy the condition*/
```

2.7.2 链表

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

```

```
using namespace std;

const int oo = 1 << 30;
int ans, first[501], l, where[250001], next[250001], value[250001], n, m, L[501], R
    [501], v[501];
bool bx[501], by[501];

inline void makelist(int x, int y, int z){
    where[++l] = y;
    value[l] = z;
    next[l] = first[x];
    first[x] = l;
}

bool find(int now){
    bx[now] = true;
    for (int x = first[now]; x; x = next[x])
        if (!by[where[x]] && L[now] + R[where[x]] == value[x])
        {
            by[where[x]] = true;
            if (!v[where[x]] || find(v[where[x]]))
            {
                v[where[x]] = now;
                return(true);
            }
        }
    return(false);
}

inline void km(){
    memset(L, 0, sizeof(L));
    memset(R, 0, sizeof(R));
    for (int i = 1; i <= n; i++)
        for (int x = first[i]; x; x = next[x])
            L[i] = max(L[i], value[x]);
    ans = 0;
    memset(v, 0, sizeof(v));
    for (int i = 1; i <= min(n, m); i++)
        for (;;)
        {
            memset(bx, false, sizeof(bx));
            memset(by, false, sizeof(by));
            if (find(i)) break;
            int Min = oo;
            for (int j = 1; j <= n; j++)
                if (bx[j])
                    for (int x = first[j]; x; x = next[x])
                        if (!by[where[x]])
                            Min = min(Min, L[j] + R[where[x]] - value[x]);
```

```
            for (int j = 1; j <= n; j++) if (bx[j]) L[j] -= Min;
            for (int j = 1; j <= m; j++) if (by[j]) R[j] += Min;
        }
    for (int i = 1; i <= n; i++)
        for (int x = first[i]; x; x = next[x])
            if (v[where[x]] == i) ans += value[x];
    printf("%d\n", abs(ans));
}

int main(){
    scanf("%d%d", &n, &m);
    memset(first, 0, sizeof(first)); l = 0;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
        {
            int x;
            scanf("%d", &x);
            makelist(i, j, -x);
        }
    km();
    for (int i = 1; i <= l; i++) value[i] = -value[i];
    km();
}

//hint 500 * 500 2.2s
//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。

```
const int MAX_N = 100000 + 10;
const int MAX_NODE = 200000 + 10;
const LL INF = 2000000000000000020LL;

int N;

struct Point
{
    int x, y, id;
};

LL dis(const Point &a, const Point &b)
{
    return 1LL * (a.x - b.x) * (a.x - b.x) + 1LL * (a.y - b.y) * (a.y - b.y);
}

struct Node
{
    Point p;
    int maxX, minX, maxY, minY;
    int l, r, d;
    Node *ch[2];
};

LL ret;
LL ans[MAX_N];
Node *root;
Point p[MAX_N], queryPoint;
Node *totNode, nodePool[MAX_NODE];

int cmpx(const Point &a, const Point &b)
```

```
{
    return a.x < b.x;
}
int cmpy(const Point &a, const Point &b)
{
    return a.y < b.y;
}

Node* newNode(int l, int r, Point p, int deep)
{
    Node *t = totNode ++;
    t->l = l; t->r = r;
    t->p = p; t->d = deep;
    t->maxX = t->minX = p.x;
    t->maxY = t->minY = p.y;
    return t;
}

void updateInfo(Node *t, Node *p)
{
    t->maxX = max(t->maxX, p->maxX);
    t->maxY = max(t->maxY, p->maxY);
    t->minX = min(t->minX, p->minX);
    t->minY = min(t->minY, p->minY);
}

Node* build(int l, int r, int deep)
{
    if (l == r) return NULL;
    if (deep & 1) sort(p + l, p + r, cmpx);
    else sort(p + l, p + r, cmpy);
    int mid = (l + r) >> 1;
    Node *t = newNode(l, r, p[mid], deep & 1);
    if (l + 1 == r) return t;
    t->ch[0] = build(l, mid, deep + 1);
    t->ch[1] = build(mid + 1, r, deep + 1);
    if (t->ch[0]) updateInfo(t, t->ch[0]);
    if (t->ch[1]) updateInfo(t, t->ch[1]);
    return t;
}

void updateAns(Point p)
{
    ret = min(ret, dis(p, queryPoint));
}

LL calc(Node *t, LL d)
{
    LL tmp;
```

```
    if (d) {
        if (queryPoint.x >= t->minX && queryPoint.x <= t->maxX) tmp = 0;
        else tmp = min(abs(queryPoint.x - t->maxX), abs(queryPoint.x - t->minX));
    } else {
        if (queryPoint.y >= t->minY && queryPoint.y <= t->maxY) tmp = 0;
        else tmp = min(abs(queryPoint.y - t->maxY), abs(queryPoint.y - t->minY));
    }
    return tmp * tmp;
}

void query(Node *t)
{
    if (t == NULL) return;
    if (t->p.id != queryPoint.id) updateAns(t->p);
    if (t->l + 1 == t->r) return;
    LL dl = t->ch[0] ? calc(t->ch[0], t->d) : INF;
    LL dr = t->ch[1] ? calc(t->ch[1], t->d) : INF;
    if (dl < dr) {
        query(t->ch[0]);
        if (ret > dr) query(t->ch[1]);
    } else {
        query(t->ch[1]);
        if (ret > dl) query(t->ch[0]);
    }
}

void solve()
{
    scanf("%d", &N);
    for(int i = 0; i < N; ++i) {
        scanf("%d%d", &p[i].x, &p[i].y);
        p[i].id = i;
    }
    totNode = nodePool;
    root = build(0, N, 1);

    for(int i = 0; i < N; ++i) {
        queryPoint = p[i];
        ret = INF;
        query(root);
        ans[p[i].id] = ret;
    }
    for(int i = 0; i < N; ++i)
        printf("%I64d\n", ans[i]);
}

int main()
{
    int T; scanf("%d", &T);
```

```
    for( ; T--; )
        solve();
    return 0;
}
```

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

```
const int MAX_NODE = 50000 + 10;
const int INF = 2000000000;

struct Node *null;

struct Node
{
    int rev, add;
    int val, maxv, size;
    Node *ch[2], *p;

    void set(Node *t, int _d) {
        ch[_d] = t;
        t->p = this;
    }
    int dir() {
        return this == p->ch[1];
    }
    void update() {
        maxv = max(max(ch[0]->maxv, ch[1]->maxv), val);
        size = ch[0]->size + ch[1]->size + 1;
    }
    void relax() {
        if (add) {
            ch[0]->appAdd(add);
            ch[1]->appAdd(add);
            add = 0;
        }
        if (rev) {
            ch[0]->appRev();
            ch[1]->appRev();
            rev = false;
        }
    }
    void appAdd(int x) {
        if (this == null) return;
        add += x;
        val += x;
        maxv += x;
    }
```

```
    void appRev() {
        if (this == null) return;
        rev ^= true;
        swap(ch[0], ch[1]);
    }
};

Node nodePool[MAX_NODE], *curNode;

Node *newNode(int val = 0)
{
    Node *t = curNode++;
    t->maxv = t->val = val;
    t->rev = t->add = 0;
    t->size = 1;
    t->ch[0] = t->ch[1] = t->p = null;
    return t;
}

struct Splay
{
    Node *root;

    Splay() {
        root = newNode();
        root->set(newNode(), 0);
        root->update();
    }

    Splay(int *a, int N) { //sequence is 1-based
        root = build(a, 0, N + 1);
    }

    Node* build(int *a, int l, int r) {
        if (l > r) return null;
        int mid = l + r >> 1;
        Node *t = newNode(a[mid]);
        t->set(build(a, l, mid - 1), 0);
        t->set(build(a, mid + 1, r), 1);
        t->update();
        return t;
    }

    void rot(Node *t)
    {
        Node *p = t->p; int d = t->dir();
        p->relax(); t->relax();
        if (p == root) root = t;
        p->set(t->ch[!d], d);
```

```
        p->p->set(t, p->dir());
        t->set(p, !d);
        p->update();
    }

    void splay(Node *t, Node *f = null)
    {
        for(t->relax(); t->p != f; ) {
            if (t->p->p == f) rot(t);
            else t->dir() == t->p->dir() ? (rot(t->p), rot(t)) : (rot(t), rot(t));
        }
        t->update();
    }

    Node* getKth(int k) {
        Node *t = root;
        int tmp;
        for( ; ; ) {
            t->relax();
            tmp = t->ch[0]->size + 1;
            if (tmp == k) return t;
            if (tmp < k) {
                k -= tmp;
                t = t->ch[1];
            } else
                t = t->ch[0];
        }
    }

    //make range[l,r] root->ch[1]->ch[0]
    //make range[x+1,x] to add something after position x
    void getRng(int l, int r) {
        r += 2;
        Node *p = getKth(l);
        Node *q = getKth(r);
        splay(p); splay(q, p);
    }

    void addRng(int l, int r, int x) {
        getRng(l, r);
        root->ch[1]->ch[0]->appAdd(x);
    }

    void revRng(int l, int r) {
        getRng(l, r);
        root->ch[1]->ch[0]->appRev();
    }

    int maxvRng(int l, int r) {
```

```
        getRng(l, r);
        return root->ch[1]->ch[0]->maxv;
    }
};

void initNull()
{
    curNode = nodePool;
    null = curNode ++;
    null->size = 0;
    null->maxv = - INF;
}
```

3.3 区间第 k 大

3.3.1 动态

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<cstdlib>
using namespace std;
const int max_n=200000+10;
const int tree_size=1000000+10;
int n,m,tot;
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];
void l_rotate(int &t)
{
    int k=rig[t];rig[t]=lef[k];lef[k]=t;
    s[k]=s[t];s[t]=s[lef[t]]+s[rig[t]]+1;
    t=k;
}
void r_rotate(int &t)
{
    int k=lef[t];lef[t]=rig[k];rig[k]=t;
    s[k]=s[t];s[t]=s[lef[t]]+s[rig[t]]+1;
    t=k;
}
void maintain(int &t,int flag)
{
    if (flag)
    {
        if (s[lef[lef[t]]]>s[rig[t]]) r_rotate(t);
        else if (s[rig[lef[t]]]>s[rig[t]]) l_rotate(lef[t]),r_rotate(t);
        else return;
    } else
    {
```

```
        if (s[rig[rig[t]]]>s[lef[t]]) l_rotate(t);
        else if (s[lef[rig[t]]]>s[lef[t]]) r_rotate(rig[t]),l_rotate(t);
        else return;
    }
    maintain(lef[t],1);maintain(rig[t],0);
    maintain(t,1);maintain(t,0);
}
void insert(int &t,int x)
{
    if (t==0)
    {
        t=++tot;
        lef[t]=rig[t]=0;
        s[t]=1;key[t]=x;
        return;
    }
    ++s[t];
    if (x < key[t]) insert(lef[t],x);else insert(rig[t],x);
    maintain(t,x<key[t]);
}
int Delete(int &t,int x)
{
    s[t]--;
    if (key[t]==x || x <key[t] && lef[t]==0 || x>key[t] && rig[t]==0)
    {
        int tmp=key[t];
        if (lef[t]==0 || rig[t]==0) t=lef[t]+rig[t];
        else key[t]=Delete(lef[t],x+1);
        return tmp;
    }
    if (x<key[t]) return Delete(lef[t],x);
    return Delete(rig[t],x);
}
int rank(int t,int x)
{
    if (t==0) return 0;
    if (x<=key[t]) return rank(lef[t],x);
    return s[lef[t]]+1+rank(rig[t],x);
}
void build(int t,int l,int r)
{
    ll[t]=l;rr[t]=r;root[t]=0;
    if (l==r) return ;
    int mid=l+r >> 1;
    build(t+t,l,mid);
    build(t+t+1,mid+1,r);
}
void ins(int t,int pos,int x)
{
```

```
    insert(root[t],x);
    if (ll[t]==rr[t]) return;
    int mid=ll[t]+rr[t] >> 1;
    if (pos <=mid) ins(t+t,pos,x);
    else ins(t+t+1,pos,x);
}
void del(int t,int pos,int x)
{
    Delete(root[t],x);
    if (ll[t]==rr[t]) return ;
    int mid=ll[t]+rr[t] >> 1;
    if (pos <=mid ) del(t+t,pos,x);
    else del(t+t+1,pos,x);
}
int get_kth(int t,int l,int r,int x)
{
    if (l<=ll[t] && r>=rr[t]) return rank(root[t],x);
    int ans=0;
    int mid=ll[t]+rr[t] >> 1;
    if (l<=mid) ans+=get_kth(t+t,l,r,x);
    if (r>mid ) ans+=get_kth(t+t+1,l,r,x);
    return ans;
}
void init()
{
    scanf("%d%d",&n,&m);tot=0;
    for (int i=1;i<=n;i++)scanf("%d",&a[i]);
    build(1,1,n);
    for (int i=1;i<=n;i++)
    ins(1,i,a[i]);
}
int query(int l,int r,int k)
{
    int p=-1,q=1000000000+1,mid;
    while (p+1!=q)
    {
        mid=(p+q) >> 1;
        if (get_kth(1,l,r,mid)<k) p=mid;else q=mid;
    }
    return p;
}

void solve()
{
    char str[5];
    int st,en,k,delta;
    for (int i=1;i<=m;i++)
    {
        scanf("%s",str);
```

```
        if (str[0]=='Q')
        {
            scanf("%d%d%d",&st,&en,&k);
            printf("%d\n",query(st,en,k));
        }else
        {
            scanf("%d%d",&k,&delta);
            del(1,k,a[k]);
            a[k]=delta;
            ins(1,k,a[k]);
        }
    }
}
int main()
{
    int M;
    for (scanf("%d",&M);M;--M)
    {
        init();
        solve();
    }
//  system("pause");
    return 0;
}
```

3.3.2 树套树 treap

```
#include<cstring>
#include<iostream>
#include<algorithm>
#include<cstdio>
using namespace std;
int Scan()      //输入外挂
{
    int res=0,ch,flag=0;
    if((ch=getchar())=='-')
        flag=1;
    else if(ch>='0'&&ch<='9')
        res=ch-'0';
    while((ch=getchar())>='0'&&ch<='9')
        res=res*10+ch-'0';
    return flag?-res:res;
}

#define N 400010
#define M 400010
#define INF 1000000000

```

```
int ctrl[M];
int cnt,n,m;
int P[M],Q[M],a[N],b[N],K[M];

struct treap
{
    int key,wei,cnt,size,ch[2];
}T[N * 15];

int tree[N << 1],nodecnt,root;

void init()
{
    T[0].size = 0;
    T[0].wei = -INF;
    nodecnt = root = 0;
}

int ID(int l,int r)
{
     return l + r | l != r;
}

void update(int x)
{
    T[x].size = T[T[x].ch[0]].size + T[T[x].ch[1]].size + T[x].cnt;
}

void rotate(int &x,int t)
{
    int y = T[x].ch[t];
    T[x].ch[t] = T[y].ch[!t];
    T[y].ch[!t] = x;
    update(x);
    update(y);
    x = y;
}

void insert(int &x,int t)
{
    if (!x)
    {
        x = ++ nodecnt;
        T[x].key = t;
        T[x].wei = rand();
        T[x].cnt = 1;
        T[x].ch[0] = T[x].ch[1] = 0;
    }
    else if (T[x].key == t)
```

```
        T[x].cnt ++;
    else
    {
        int k = T[x].key < t;
        insert(T[x].ch[k],t);
        if (T[x].wei < T[T[x].ch[k]].wei)
            rotate(x,k);
    }
    update(x);
}

void erase(int &x,int t)
{
    if (T[x].key == t)
    {
        if (T[x].cnt == 1)
        {
            if (!T[x].ch[0] && !T[x].ch[1])
            {
                x = 0;
                return;
            }
            rotate(x,T[T[x].ch[0]].wei < T[T[x].ch[1]].wei);
            erase(x,t);
        }
        else T[x].cnt--;
    }
    else
        erase(T[x].ch[T[x].key < t],t);
    update(x);
}

int select(int x,int t)
{
    if (!x) return 0;
    if (T[x].key > t) return select(T[x].ch[0],t);
    return T[x].cnt + T[T[x].ch[0]].size + select(T[x].ch[1],t);
}

void treeins(int l,int r,int i,int x)
{
    insert(tree[ID(l,r)],x);
    if (l == r) return;
    int m = l + r >> 1;
    if (i <= m) treeins(l,m,i,x);
    else treeins(m + 1,r,i,x);
}

void treedel(int l,int r,int i,int x)
```

```
{
    erase(tree[ID(l,r)],x);
    if (l == r) return;
    int m = l + r >> 1;
    if (i <= m) treedel(l,m,i,x);
    else treedel(m + 1,r,i,x);
}

int query(int l,int r,int x,int y,int t)
{
    if (l == r) return l;
    int m = l + r >> 1;
    int ans = select(tree[ID(l,m)],y) - select(tree[ID(l,m)],x);
    if (ans >= t) return query(l,m,x,y,t);
    return query(m + 1,r,x,y,t - ans);
}

int main()
{

    while (~scanf("%d",&n))
    {
        memset(tree,0,sizeof tree);
        init();
        cnt = 0;
        for (int i = 1;i <= n;i ++)
            {a[i]=Scan();b[++ cnt] = a[i];}
         m=Scan();
        for (int i = 1;i <= m;i ++)
        {
            ctrl[i]=Scan();P[i]=Scan();Q[i]=Scan();
            //scanf("%s%d%d",ctrl[i],&P[i],&Q[i]);
            if (ctrl[i] == 2)
                {K[i]=Scan();}//scanf("%d",&K[i]);
            else
                b[++ cnt] = Q[i];
        }
        sort(b + 1,b + 1 + cnt);
        cnt = unique(b + 1,b + 1 + cnt) - b - 1;
        for (int i = 1;i <= n;i ++)
        {
            a[i] = lower_bound(b + 1,b + 1 + cnt,a[i]) - b;
            treeins(1,cnt,a[i],i);
        }
        for(int i = 1;i <= m;i ++)
        {
            if (ctrl[i] == 2)
            {
                int id = query(1,cnt,P[i] - 1,Q[i],K[i]);
```

```
                printf("%d\n",b[id]);
            }
            else
            {
                treedel(1,cnt,a[P[i]],P[i]);
                a[P[i]] = lower_bound(b + 1,b + 1 + cnt,Q[i]) - b;
                treeins(1,cnt,a[P[i]],P[i]);
            }
        }
    }
    return 0;
}
```

3.4 Treap

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

```
struct Node *null;

struct Node
{
    int key, val, size;
    Node *ch[2];
    Node() {
        key = INT_MAX;
        val = size = 0;
    }
    Node(int _val) {
        size = 1;
        val = _val;
        key = bigRand();
        ch[0] = ch[1] = null;
    }
    int bigRand() {
        return rand() * RAND_MAX + rand();
    }
    void update() {
        size = ch[0]->size + ch[1]->size + 1;
    }
};

struct Treap
{
    Node *root;
    Treap() {
        root = null;
    }
    void rot(Node *&t, int d) {
        Node *p = t->ch[d]; t->ch[d] = p->ch[!d]; p->ch[!d] = t;
```

```
        t->update(); p->update();
        t = p;
    }

    void insert(Node *&t, int x) {
        if (t == null) {
            t = new Node(x);
            return;
        }
        int dir = x >= t->val;
        insert(t->ch[dir], x);
        if (t->ch[dir]->key < t->key)
            rot(t, dir);
        else
            t->update();
    }

    void erase(Node *&t, int x) {
        if (t == null)
            return;
        if (t->val == x) {
            int dir = t->ch[1]->key < t->ch[0]->key;
            if (t->ch[dir] == null) {
                delete t;
                t = null;
                return;
            }
            rot(t, dir);
            erase(t->ch[!dir], x);
            t->update();
            return;
        }
        bool dir = x > t->val;
        erase(t->ch[dir], x);
        t->update();
    }

    void insert(int x) {
        insert(root, x);
    }

    void erase(int x) {
        erase(root, x);
    }
};
```

3.5 线段树

包含建树和区间操作样例，没有写具体操作

```
struct Tree
{
    int l, r;
    Tree *ch[2];
    Tree() {}
    Tree(int _l, int _r, int *sqn) {
        l = _l; r = _r;
        if (l + 1 == r)
            return;
        int mid = l + r >> 1;
        ch[0] = new Tree(l, mid, sqn);
        ch[1] = new Tree(mid, r, sqn);
    }

    void insert(int p, int x) {
        if (p < l || p >= r)
            return;
        //some operations
        if (l + 1 == r)
            return;
        ch[0]->insert(p, x);
        ch[1]->insert(p, x);
    }

    int query(int _l, int _r, int x) {
        if (_r <= l || _l >= r)
            return 0;
        if (_l <= l && _r >= r)
            // return information in [l, r)
        //merge ch[0]->query(_l, _r, x), ch[1]->query(_l, _r, x) and return
    }
};
```

3.6 KMP

```
vector<int> KMP()
{
    vector<int> ans;
    nxt[0] = -1;
    nxt[1] = 0;
    for(int i = 2; i <= m; i++)
    {
        nxt[i] = nxt[i - 1];
        while(nxt[i] >= 0 and st[i] != st[nxt[i] + 1])
```

```
                nxt[i] = nxt[nxt[i]];
            nxt[i]++;
        }
        for(int i = 1, p = 1; i <= n; i++)
        {
            while(p and str1[i] != st[p])
                p = nxt[p - 1] + 1;
            p++;
            if(p == m + 1) p = nxt[m] + 1, ans.push_back(i - m);
        }
        return ans;
}
```

3.7 扩展 KMP

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

```
void z(char *s, int *next, int N)
{
    int j = 0, k = 1;
    while (j + 1 < N && s[j] == s[j + 1]) ++j;
    next[0] = N - 1; next[1] = j;
    for(int i = 2; i < N; ++i) {
        int far = k + next[k] - 1, L = next[i - k];
        if (L < far - i + 1) next[i] = L;
        else {
            j = max(0, far - i + 1);
            while (i + j < N && s[j] == s[i + j]) ++j;
            next[i] = j; k = i;
        }
    }
}
```

3.8 Manacher

```
void manacher (char str[], int len[], int n) {
    len[0] = 1;
    for (int i = 1, j = 0; i < (n << 1) - 1; ++i) {
        int p = i >> 1,
        q = i - p,
        r = ((j + 1) >> 1) + len[j] - 1;
        len[i] = r < q? 0: min(r - q + 1, len[(j << 1) - i]);
        while (p - len[i] > -1 and q + len[i] < n and str[p - len[i]] == str[q + len[
            i]]) {
            len[i] += 1;
        }
        if (q + len[i] - 1 > r) {
```

```
                j = i;
            }
        }
}
```

3.9 AC 自动机

包含建 trie 和构造自动机的过程

```

struct acNode
{
    int id;
    acNode *ch[26], *fail;
} *totNode, *root, nodePool[MAX_V];

acNode* newNode()
{
    acNode *now = totNode ++;
    now->id = 0; now->fail = 0;
    memset(now->ch, 0, sizeof now->ch);
    return now;
}

void acInsert(char *c, int id)
{
    acNode *cur = root;
    while (*c) {
        int p = *c - 'A'; //change the index
        if (!cur->ch[p]) cur->ch[p] = newNode();
        cur = cur->ch[p];
        ++ c;
    }
    cur->id = id;
}

void getFail()
{
    acNode *cur;
    queue<acNode*> Q;
    for(int i = 0; i < 26; ++ i)
        if (root->ch[i]) {
            root->ch[i]->fail = root;
            Q.push(root->ch[i]);
        } else root->ch[i] = root;
    while (!Q.empty()) {
        cur = Q.front(); Q.pop();
        for(int i = 0; i < 26; ++ i)
            if (cur->ch[i]) {
```

```
                        cur->ch[i]->fail = cur->fail->ch[i];
                        Q.push(cur->ch[i]);
                } else cur->ch[i] = cur->fail->ch[i];
        }
}
```

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])

```
const int MAX_N = 1000000 + 10;

int rank[MAX_N], height[MAX_N];

int cmp(int *x, int a, int b, int d)
{
    return x[a] == x[b] && x[a + d] == x[b + d];
}

void doubling(int *a, int N, int M)
{
    static int sRank[MAX_N], tmpA[MAX_N], tmpB[MAX_N];
    int *x = tmpA, *y = tmpB;
    for(int i = 0; i < M; ++i) sRank[i] = 0;
    for(int i = 0; i < N; ++i) ++sRank[x[i] = a[i]];
    for(int i = 1; i < M; ++i) sRank[i] += sRank[i - 1];
    for(int i = N - 1; i >= 0; --i) sa[--sRank[x[i]]] = i;

    for(int d = 1, p = 0; p < N; M = p, d <<= 1) {
        p = 0; for(int i = N - d; i < N; ++i) y[p++] = i;
        for(int i = 0; i < N; ++i)
            if (sa[i] >= d) y[p++] = sa[i] - d;
        for(int i = 0; i < M; ++i) sRank[i] = 0;
        for(int i = 0; i < N; ++i) ++sRank[x[i]];
        for(int i = 1; i < M; ++i) sRank[i] += sRank[i - 1];
        for(int i = N - 1; i >= 0; --i) sa[--sRank[x[y[i]]]] = y[i];
        swap(x, y); x[sa[0]] = 0; p = 1;
        for(int i = 1; i < N; ++i)
            x[sa[i]] = cmp(y, sa[i], sa[i - 1], d) ? p - 1 : p++;
    }
}

void calcHeight()
{
    for(int i = 0; i < N; ++i) rank[sa[i]] = i;
    int cur = 0;
    for(int i = 0; i < N; ++i)
```

```
        if (rank[i]) {
            if (cur) cur--;
            for( ; a[i + cur] == a[sa[rank[i] - 1] + cur]; ++cur);
            height[rank[i]] = cur;
        }
}
```

3.10.2 DC3

```
//DC3 待排序的字符串放在r 数组中,从r[0]到r[n-1],长度为n,且最大值小于m。
//约定除r[n-1]外所有的r[i]都大于0, r[n-1]=0。
//函数结束后,结果放在sa 数组中,从sa[0]到sa[n-1]。
//r必须开长度乘3
#define maxn 10000
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int wa[maxn],wb[maxn],wv[maxn],wss[maxn];
int s[maxn*3],sa[maxn*3];
int c0(int *r,int a,int b)
{
    return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];
}
int c12(int k,int *r,int a,int b)
{
    if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
    else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
}
void sort(int *r,int *a,int *b,int n,int m)
{
    int i;
    for(i=0;i<n;i++) wv[i]=r[a[i]];
    for(i=0;i<m;i++) wss[i]=0;
    for(i=0;i<n;i++) wss[wv[i]]++;
    for(i=1;i<m;i++) wss[i]+=wss[i-1];
    for(i=n-1;i>=0;i--) b[--wss[wv[i]]]=a[i];
}
void dc3(int *r,int *sa,int n,int m)
{
    int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
    r[n]=r[n+1]=0;
    for(i=0;i<n;i++)
        if(i%3!=0) wa[tbc++]=i;
    sort(r+2,wa,wb,tbc,m);
    sort(r+1,wb,wa,tbc,m);
    sort(r,wa,wb,tbc,m);
    for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
        rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
    if (p<tbc) dc3(rn,san,tbc,p);
```

```
    else for (i=0;i<tbc;i++) san[rn[i]]=i;
    for (i=0;i<tbc;i++)
        if(san[i]<tb) wb[ta++]=san[i]*3;
    if(n%3==1) wb[ta++]=n-1;
    sort(r,wb,wa,ta,m);
    for(i=0;i<tbc;i++)
        wv[wb[i]=G(san[i])]=i;
    for(i=0,j=0,p=0;i<ta && j<tbc;p++)
        sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
    for(;i<ta;p++) sa[p]=wa[i++];
    for(;j<tbc;p++) sa[p]=wb[j++];
}
int main(){
    int n,m=0;
    scanf("%d",&n);
    for (int i=0;i<n;i++) scanf("%d",&s[i]),s[i]++,m=max(s[i]+1,m);
    printf("%d\n",m);
    s[n++]=0;
    dc3(s,sa,n,m);
    for (int i=0;i<n;i++) printf("%d ",sa[i]);printf("\n");
}
```

Chapter 4

杂

4.1 m^2logn 求线性递推第 n 项

```
// given first m a[i] and coef c[i] (0-based),
// calc a[n] mod p in O(m*m*log(n)).
// a[n] = sum(c[m-i]*a[n-i]), i = 1...m
// i.e. a[m] = sum(c[i]*a[i]), i = 0...m-1
int linear_recurrence(LL n, int m, int a[], int c[], int p) {
    LL v[M] = {1 % p}, u[M<<1], msk = !!n;
    for(LL i = n; i > 1; i >>= 1) msk <<= 1;
    for(LL x = 0; msk; msk >>= 1, x <<= 1) {
        fill_n(u, m<<1, 0);
        int b = !!(n & msk); x |= b;
        if(x < m) u[x] = 1 % p;
        else {
            for(int i = 0; i < m; ++i)
                for(int j = 0, t = i+b; j < m; ++j, ++t)
                    u[t] = (u[t]+v[i]*v[j]) % p;
            for(int i = (m<<1)-1; i >= m; --i)
                for(int j = 0, t = i-m; j < m; ++j, ++t)
                    u[t] = (u[t]+c[j]*u[i]) % p;
        }
        copy(u, u+m, v);
    }
    int an = 0;
    for(int i = 0; i < m; ++i) an = (an+v[i]*a[i]) % p;
    return an;
}
```

4.2 FFT

```
#include <complex>
#include <algorithm>
```

```
#include <cmath>
#include <vector>

using namespace std;

typedef complex <double> Complex;
typedef vector <int> Polynomial;

const double PI = acos(-1.);
const int N = 1 << 17;

void FFT(Complex* P, int n, int oper) {
    for (int i = 1, j = 0; i < n - 1; i++) {
        for (int s = n; j ^= s >>= 1, ~j & s;);
        if (i < j) {
            swap(P[i], P[j]);
        }
    }
    Complex unit_p0;
    for (int d = 0; (1 << d) < n; d++) {
        int m = 1 << d, m2 = m * 2;
        double p0 = PI / m * oper;
        unit_p0 = Complex(cos(p0), sin(p0));
        for (int i = 0; i < n; i += m2) {
            Complex unit = 1;
            for (int j = 0; j < m; j++) {
                Complex &P1 = P[i + j + m], &P2 = P[i + j];
                Complex t = unit * P1;
                P1 = P2 - t;
                P2 = P2 + t;
                unit = unit * unit_p0;
            }
        }
    }
}

Complex A[N], B[N];

Polynomial operator * (const Polynomial &u, const Polynomial &v)
{
    int n=1,p=u.size(),q=v.size(),r=p+q-1,i;
    while (n<=p+q-2) n<<=1;
    for (i=0;i<n;++i) A[i]=i<p?u[i]:0;
    for (i=0;i<n;++i) B[i]=i<q?v[i]:0;
    FFT(A,n,1);
    FFT(B,n,1);
    for (i=0;i<n;++i) A[i]*=B[i];
    FFT(A,n,-1);
    Polynomial w(p+q-1);
```

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

4.3 线性筛莫比乌斯

```
void prepare()
{
    mu[1] = 1;
    for (int i = 2 ; i <= 50000; i++)
    {
        if (!mark[i])
        {
            pr[++tot] = i;
            mu[i] = -1;
        }
        for (int j = 1; j <= tot && i * pr[j] <= 50000; j ++)
        {
            mark[i * pr[j]] = 1;
            if ( i %pr[j] == 0)
            {
                mu[i * pr[j]] = 0;
                break;
            }
            else mu[i*pr[j]] = -mu[i];
        }
    }
    for (int i = 1 ; i <= 50000; i ++)
        sum[i] = sum[i - 1] + mu[i];
}
```

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

```
LL x, y;
void exGcd(LL a, LL b)
{
    if (b == 0) {
        x = 1;
        y = 0;
        return;
    }
    exGcd(b, a % b);
    LL k = y;
    y = x - a / b * y;
```

```
    x = k;
}

LL inversion(LL a, LL b)
{
    exGcd(a, b);
    return (x % b + b) % b;
}

LL CRT(vector<LL> m, vector<LL> a)
{
    int N = m.size();
    LL M = 1, ret = 0;
    for(int i = 0; i < N; ++i)
        M *= m[i];

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

4.5 Pollard's Rho+Miller-Rabbin

大数分解和素性判断

```
typedef long long LL;

LL modMul(LL a, LL b, LL P)
{
    LL ret = 0;
    for( ; a; a >>= 1) {
        if (a & 1) {
            ret += b;
            if (ret >= P) ret -= P;
        }
        b <<= 1;
        if (b >= P) b -= P;
    }
    return ret;
}

LL modPow(LL a, LL u, LL P)
{
    LL ret = 1;
    for( ; u; u >>= 1, a = modMul(a, a, P))
        if (u & 1) ret = modMul(ret, a, P);
    return ret;
}
```

```

int millerRabin(LL N)
{
    if (N == 2) return true;
    LL t = 0, u = N - 1, x, y, a;
    for( ; !(u & 1); ++t, u >>= 1) ;
    for(int k = 0; k < 10; ++k) {
        a = rand() % (N - 2) + 2;
        x = modPow(a, u, N);
        for(int i = 0; i < t; ++i, x = y) {
            y = modMul(x, x, N);
            if (y == 1 && x > 1 && x < N - 1) return false;
        }
        if (x != 1) return false;
    }
    return true;
}

LL gcd(LL a, LL b)
{
    return !b ? a : gcd(b, a % b);
}

LL pollardRho(LL N)
{
    LL i = 1, x = rand() % N;
    LL y = x, k = 2, d = 1;
    do {
        d = gcd(x - y + N, N);
        if (d != 1 && d != N) return d;
        if (++i == k) y = x, k <<= 1;
        x = (modMul(x, x, N) - 1 + N) % N;
    } while (y != x);
    return N;
}

void getFactor(LL N)
{
    if (N < 2) return;
    if (millerRabin(N)) {
        //do some operations
        return;
    }
    LL x = pollardRho(N);
    getFactor(x);
    getFactor(N / x);
}
```

4.6 素数判定 (long long 内确定性算法)

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

4.7 求前 P 个数的逆元

```
void solve (int m) {
    int inv[m];
    inv[1] = 1;
    for (int i = 2; i < m; ++i) {
```

```
            inv[i] = ((long long)(m - m / i) * inv[m % i]) % m;
        }
}
```

4.8 Lucas 快速取 mod

附加移位乘法

```

long long fast_mod(long long a , long long b , long long mod)
{
    long long ans = 1;
    a %= mod;
    while (b)
    {
        if (b & 1)
        {
            ans = ans * a % mod;
            b --;
        }
        b >>= 1;
        a = a * a % mod;
    }
    return ans;
}
long long lucas(long long n , long long m , long long mod)
{
    if (m == 0) return 1;
    return C(n % mod , m % mod , mod) * lucas(n / mod , m / mod, mod) % mod;
}
```

4.9 快速幂

```


#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;
const int N=55;
const int mod=2015;
struct Mat {
    int mat[N][N];
};
int n,m;
Mat operator * (Mat a, Mat b) {
    Mat c;
```

```
    memset(c.mat, 0, sizeof(c.mat));
    int i, j, k;
    for(k = 0; k < n; ++k) {
        for(i = 0; i < n; ++i) {
            for(j = 0; j < n; ++j) {
                c.mat[i][j] = (c.mat[i][j]+a.mat[i][k] * b.mat[k][j])%mod;
            }
        }
    }
    return c;
}
Mat operator ^ (Mat a, int k) {
    Mat c;
    int i, j;
    for(i = 0; i < n; ++i)
        for(j = 0; j < n; ++j)
            c.mat[i][j] = (i == j);

    for(; k; k >>= 1) {
        if(k&1) c = c*a;

        a = a*a;
    }
    return c;
}
```

4.10 广义离散对数 (不需要互质)

```
void extendedGcd (int a, int b, long long &x, long long y) {
    if (b) {
        extendedGcd(b, a % b, y, x);
        y -= a / b * x;
    } else {
        x = a;
        y = 0;
    }
}
int inverse (int a, int m) {
    long long x, y;
    extendedGcd(a, m, x, y);
    return (x % m + m) % m;
}
// a ^ x = b (mod m)
int solve (int a, int b, int m) {
    int tmp = 1 % m, c;
    map<int, int> s;
    if (tmp == b) {
        return 0;
```

```
    }
    for (int i = 1; i <= 50; ++i) {
        tmp = ((long long)tmp * a) % m;
        if (tmp == b) {
            return i;
        }
    }
    int x_0 = 0, d = 1 % m;
    while (true) {
        tmp = gcd(a, m);
        if (tmp == 1) {
            break;
        }
        x_0 ++;
        d = ((long long)d * (a / tmp)) % m;
        if (b % tmp) {
            return -1;
        }
        b /= tmp;
        m /= tmp;
    }
    b = ((long long)b * inverse(d, m)) % m;
    c = int(ceil(sqrt(m)));
    s.clear();
    tmp = b;
    int tmpInv = intverse(a, m);
    for (int i = 0; i != c; ++i) {
        if (s.find(tmp) == s.end()) {
            s[tmp] = i;
        }
        tmp = ((long long)tmp * tmpInv) % m;
    }
    tmp = 1;
    for (int i = 0; i != c; ++i) {
        tmp = ((long long)tmp * a) % m;
    }
    int ans = 1;
    for (int i = 0; i != c; ++i) {
        if (s.find(ans) != s.end()) {
            return x_0 + i * c + s.find(ans)->second;
        }
        ans = ((long long)ans * tmp) % m;
    }
    return -1;
}
```

4.11 n 次剰余

```
const int LimitSave=100000;
long long P,K,A;
vector<long long>ans;
struct tp{
    long long expo,res;
}data[LimitSave+100];
long long _mod(long long a, long long mo) {
    a=a%mo;
    if (a<0) a+=mo;
    return a;
}
long long powers(long long a , long long K , long long modular) {
    long long res;
    res=1;
    while (K!=0) {
        if (K & 1) res=_mod(res*a,modular);
        K=K>>1;
        a=_mod(a*a , modular);
    }
    return res;
}
long long get_originroot(long long p) {
    long long primes[100];
    long long tot,i,tp,j;
    i=2; tp=P-1; tot=0;
    while (i*i<=P-1) {
        if (_mod(tp,i)==0) {
            tot++;
            primes[tot]=i;
            while (_mod(tp,i)==0) tp/=i;
        }
        i++;
    }
    if (tp!=1) {tot++; primes[tot]=tp;}
    i=2;
    bool ok;
    while (1) {
        ok=true;
        foru(j,1,tot) {
            if (powers(i, (P-1)/primes[j] , P)==1) {
                150
                ok=false;
                break;
            }
        }
        if (ok) break;
        i++;
    }
    return i;
```

```
}
bool
euclid_extend(long long A ,long long B ,long long C ,long long &x, long
long &y, long long
&gcdnum) {
    long long t;
    if (A==0) {
        gcdnum = B;
        if (_mod(C , B) ==0) {
            x=0; y=C/B;
            return true;
        }
        else return false;
    }
    else if (euclid_extend(_mod(B , A) , A , C , y , t , gcdnum)) {
        x = t - int(B / A) * y;
        return true;
    }
    else return false;
}
long long Division(long long A, long long B, long long modular) {
    long long gcdnum,K,Y;
    euclid_extend(modular, B,A,K,Y,gcdnum);
    Y=_mod(Y,modular);
    if (Y<0) Y+=modular;
    return Y;
}
bool Binary_Search(long long key, long long &position) {
    long long start,stop;
    start=1; stop=LimitSave;
    bool flag=true;
    while (start<=stop) {
        position=(start+stop)/2;
        if (data[position].res==key) return true;
        else
            if (data[position].res<key) start=position+1;
            else stop=position-1;
    }
    return false;
}
bool compareab(const tp &a, const tp &b) {
    return a.res<b.res;
}
long long get_log(long long root, long long A, long long modular) {
    long long i,j,times,XD,XT,position;
    if (modular-1<LimitSave) {
        long long now=1;
        foru(i,0,modular-1) {
            if (now==A) {
```

```
                    return i;
                }
                now=_mod(now * root , modular);
            }
        }
        data[1].expo=0; data[1].res=1;
        foru(i,1,LimitSave-1) {
            data[i+1].expo=i;
            data[i+1].res=_mod(data[i].res*root,modular);
        }
        sort(data+1,data+LimitSave+1,compareab);
        times=powers(root,LimitSave,modular);
        j=0;
        XD=1;
        while (1) {
            XT=Division(A,XD,modular);
            if (Binary_Search(XT,position)) {
                return j+data[position].expo;
            }
            j=j+LimitSave;
            XD=_mod(XD*times,modular);
        }
}
void work_ans() {
        ans.clear();
        if (A==0) {
            ans.push_back(0);
            return;
        }
        long long root,logs,delta,deltapower,now,gcdnum, i,x,y;
        root=get_originroot(P);
        logs=get_log(root,A,P);
        if (euclid_extend(K,P-1,logs,x,y,gcdnum)) {
            delta=(P-1)/gcdnum;
            x=_mod(x,delta);
            if (x<0) x+=delta;
            now=powers(root,x,P);
            deltapower=powers(root,delta,P);
            while (x<P-1) {
                ans.push_back(now);
                now=_mod(now*deltapower,P);
                x=x+delta;
            }
        }
        if (ans.size()>1)
            sort(ans.begin(),ans.end());
}
int main(){
        int i,j,k,test,cases=0;
```

```
    scanf("%d",&test);
    prepare();
    while (test) {
        test--;
        cin>>P>>K>>A;
        A=A % P;
        //x^K mod P = A
        cases++;
        printf("Case #%d:\n",cases);
        work_ans();
    }
    return 0;
}
```

4.12 二次剩余

```
/*
   a*x^2+b*x+c==0 (mod P)
   求 0..P-1 的根
 */
#include <cstdio>
#include <cstdlib>
#include <ctime>
#define sqr(x) ((x)*(x))
int pDiv2,P,a,b,c,Pb,d;
inline int calc(int x,int Time)
{
    if (!Time) return 1;
    int tmp=calc(x,Time/2);
    tmp=(long long)tmp*tmp%P;
    if (Time&1) tmp=(long long)tmp*x%P;
    return tmp;
}
inline int rev(int x)
{
    if (!x) return 0;
    return calc(x,P-2);
}
inline void Compute()
{
    while (1)
    {
        b=rand()%(P-2)+2;
        if (calc(b,pDiv2)+1==P) return;
    }
}
int main()
{
```

```
    srand(time(0)^312314);
    int T;
    for (scanf("%d",&T);T;--T)
    {
        scanf("%d%d%d%d",&a,&b,&c,&P);
        if (P==2)
        {
            int cnt=0;
            for (int i=0;i<2;++i)
                if ((a*i*i+b*i+c)%P==0) ++cnt;
            printf("%d",cnt);
            for (int i=0;i<2;++i)
                if ((a*i*i+b*i+c)%P==0) printf(" %d",i);
            puts("");
        }else
        {
            int delta=(long long)b*rev(a)*rev(2)%P;
            a=(long long)c*rev(a)%P-sqr( (long long)delta )%P;
            a%=P;a+=P;a%=P;
            a=P-a;a%=P;
            pDiv2=P/2;
            if (calc(a,pDiv2)+1==P) puts("0");
            else
            {
                int t=0,h=pDiv2;
                while (!(h%2)) ++t,h/=2;
                int root=calc(a,h/2);
                if (t>0)
                {
                    Compute();
                    Pb=calc(b,h);
                }
                for (int i=1;i<=t;++i)
                {
                    d=(long long)root*root*a%P;
                    for (int j=1;j<=t-i;++j)
                        d=(long long)d*d%P;
                    if (d+1==P)
                        root=(long long)root*Pb%P;
                    Pb=(long long)Pb*Pb%P;
                }
                root=(long long)a*root%P;
                int root1=P-root;
                root-=delta;
                root%=P;
                if (root<0) root+=P;
                root1-=delta;
                root1%=P;
                if (root1<0) root1+=P;
```

```
                    if (root>root1)
                    {
                        t=root;root=root1;root1=t;
                    }
                    if (root==root1) printf("1 %d\n",root);
                    else printf("2 %d %d\n",root,root1);
                }
            }
        }
    return 0;
}
```

4.13 长方体表面两点最短距离

返回最短距离的平方

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

using namespace std;

int r;
void turn(int i, int j, int x, int y, int z, int x0, int y0, int L, int W, int H)
{
    if (z == 0) {
        int R = x * x + y * y;
        if (R < r) r = R;
    } else {
        if (i >= 0 && i < 2)
            turn(i + 1, j, x0 + L + z, y, x0 + L - x, x0 + L, y0, H, W, L);
        if (j >= 0 && j < 2)
            turn(i, j + 1, x, y0 + W + z, y0 + W - y, x0, y0 + W, L, H, W);
        if (i <= 0 && i > -2)
            turn(i - 1, j, x0 - z, y, x - x0, x0 - H, y0, H, W, L);
        if (j <= 0 && j > -2)
            turn(i, j - 1, x, y0 - z, y - y0, x0, y0 - H, L, H, W);
    }
}

int main()
{
    int L, H, W, x1, y1, z1, x2, y2, z2;
    cin >> L >> W >> H >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;
    if (z1 != 0 && z1 != H) {
        if (y1 == 0 || y1 == W)
            swap(y1, z1), swap(y2, z2), swap(W, H);
        else
            swap(x1, z1), swap(x2, z2), swap(L, H);
```

```
    }
    if (z1 == H) z1 = 0, z2 = H - z2;
    r = 0x3fffffff;
    turn(0, 0, x2 - x1, y2 - y1, z2, -x1, -y1, L, W, H);
    cout << r << endl;
    return 0;
}
```

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

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

4.14.2 min_1

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

```
        else
        {
            if(a[i + dlt] > a[j + dlt]) i += dlt + 1; else j += dlt + 1;
            dlt = 0;
        }
    }
    a.offset = min(i, j);
}
int main()
{return 0;}
```

4.15 牛顿迭代开根号

速度慢，精度有保证

```
typedef unsigned long long ull;
ull sqrtll(ull n)
{
    if (n == 0) return 0;
    ull x = 1ull << ((63 - __builtin_clzll(n)) >> 1);
    ull xx = -1;
    for( ; ; ) {
        ull nx = (x + n / x) >> 1;
        if (nx == xx)
            return min(x, nx);
        xx = x;
        x = nx;
    }
}
```

4.16 求某年某月某日星期几

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

4.17 日期类解决两个日期之间差多少天

```
//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。
//特判了1752年9月3日到9月13日，日历中没有这些日期。
//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)
#include<cstdio>
using namespace std;

bool isleap(int y)
{
    if (y % 400 == 0 || (y % 100 != 0 && y % 4 == 0)) return true;
    return false;
}

char Week[8][12] = {"", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};
int dayofmonth[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
struct Date{
    int y, m, d;
    Date(){}
    Date(int totday){
        //
        if (totday > 639785) totday += 11;  //特判1752年9月
        //
        y = totday / 366;
        totday %= 366;
        totday += y - (y/4 - y/100 + y/400);
        y++;
        for (int &year = y;; year++){
            int del = 365 + isleap(year);
            if (totday > del) totday -= del;
            else break;
        }
        m = 1;
        for (int &month = m; month < 12; month++){
            int del = dayofmonth[month] + (month==2&&isleap(y));
            if (totday > del) totday -= del;
            else break;
        }
        d = totday;
    }
    Date(int _y, int _m, int _d): y(_y), m(_m), d(_d){}
    int totalday(){
        int leap = y/4 - y/100 + y/400;
        if (isleap(y)) leap --;
        int ret = (y-1) * 365 + leap;
        for (int i = 1; i < m; i++) ret += dayofmonth[i];
        if (isleap(y) && m > 2) ret++;
        ret += d;
        //
```

```
        if (y > 1752) ret -= 11;          //特判1752年9月
        else if (y == 1752 && m > 9) ret -= 11;
        else if (y == 1752 && m == 9 && d >= 14) ret -= 11;
        //
        return ret;
    }
    int whatday(){
        int st = Date(1752, 9, 2).totalday(),
            sum = totalday();
        int del = sum - st;
        if (del >= 0){
            del = (del % 7 + 3) % 7;
            if (del == 0) del = 7;
        }
        else{
            del = -del;
            del %= 7;
            del = 3 - del;
            if (del <= 0) del += 7;
        }
        return del;
    }
};

int main()
{
    int y, m, d;
    while(scanf("%d %d %d", &y, &m, &d))
    {
        Date ans = Date(y, m, d);
        printf("%d-%02d-%02d %s\n", ans.y, ans.m, ans.d, Week[ans.whatday()]);
    }
}
```

4.18 多项式求根 (求导二分)

```
const double error=1e-12;
const double infi=1e+12;
double a[10],x[10];
int n;
int sign(double x) {
    return (x<-error)?(-1):(x>error);
}
double f(double a[],int n,double x) {
    double tmp=1,sum=0;
    for (int i=0;i<=n;i++) {
        sum=sum+a[i]*tmp;
        tmp=tmp*x;
```

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

```
}
```

4.19 有多少个点在多边形内

```
//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,
//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
//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。
int ans=0,z,onit=0,lows=0;
rep(z,t) {
    i=rn[z]; j=rn[z+1]; onit+=on[i][j]-1;
    if (a[j].x>a[i].x){ans-=num[i][j];lows+=on[i][j]-1;}
    else ans+=num[i][j];
}
//ans-lows+1 is inside. 只会多算一次正向上的点(除去最左和最右的点)。Lows
    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.
printf("%d\n",ans-lows+1+ onit);
```

4.20 斜线下格点统计

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

4.21 杂知识

牛顿迭代

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

图同构 Hash

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

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

圆上有整点的充要条件

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

Pick 定理

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

图定理

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

梅森素数

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

上下界网络流

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

平面图定理

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

Fibonacci 相关结论

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

二次剩余

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

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：

$$\binom{n}{k} = k\binom{n-1}{k} + \binom{n-1}{k-1}$$

$$k = 2, \binom{n}{k} = 2^{(n-1)} - 1$$

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：

$$1, 3, 10, 37......$$

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

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

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：

$$\mu(x) = \begin{cases} 1 & \text{n=1} \\ (-1)^k & n = p_1 p_2 ... p_n \\ 0 & \text{其余情况} \end{cases}$$

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

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

public class Main {
    final static int MOD = (int)1e9 + 7;

```

```
    public void run() {
        try {
            int n = reader.nextInt();
            String[] map = new String[n];
            for (int i = 0; i < n; ++i) {
                map[i] = reader.next();
            }
            writer.println(10 % MOD);
        } catch (IOException ex) {
        }
        writer.close();
    }

    InputReader reader;
    PrintWriter writer;

    Main() {
        reader = new InputReader();
        writer = new PrintWriter(System.out);
    }

    public static void main(String[] args) {
        new Main().run();
    }

    void debug(Object... os) {
        System.err.println(Arrays.deepToString(os));
    }
}

class InputReader {
    BufferedReader reader;
    StringTokenizer tokenizer;

    InputReader() {
        reader = new BufferedReader(new InputStreamReader(System.in));
        tokenizer = new StringTokenizer("");
    }

    String next() throws IOException {
        while (!tokenizer.hasMoreTokens()) {
            tokenizer = new StringTokenizer(reader.readLine());
        }
        return tokenizer.nextToken();
    }

    Integer nextInt() throws IOException {
        return Integer.parseInt(next());
    }
```

```java
}
//-------------------------------------------------------------
import java.util.*;
import java.math.*;
import java.io.*;

public class Main {

    Scanner cin;

    void solve() {
        BigInteger a, b, c;
        a = cin.nextBigInteger();
        b = cin.nextBigInteger();
        c = a.add(b);
        System.out.println(a + " + " + b + " = " + c);
    }

    void run() {
        cin = new Scanner(new BufferedInputStream(System.in));
        int tmp = cin.nextInt();
        int testcase = 0;
        while(cin.hasNextBigInteger()) {
            ++testcase;
            if (testcase > 1)
                System.out.println();
            System.out.println("Case " + testcase + ":");
            solve();
        }
    }

    public static void main(String[] args) {
        new Main().run();
    }
}
//Arrays
int a[]=new int[10];
Arrays.fill(a,0);
Arrays.sort(a);
//String
String s;
s.charAt(int i);
s.compareTo(String b);
s.compareToIgnoreCase();
s.contains(String b);
s.length();
s.substring(int l,int len);
//BigInteger
BigInteger a;
```

```
a.abs();
a.add(b);
a.bitLength();
a.subtract(b);
a.divide(b);
a.remainder(b);
a.divideAndRemainder(b);
a.modPow(b,c);//a^b mod c;
a.pow(int);
a.multiply(b);
a.compareTo(b);
a.gcd(b);
a.intValue();
a.longValue();
a.isProbablePrime(int certainty);//(1 - 1/2^certainty).
a.nextProbablePrime();
a.shiftLeft(int);
a.valueOf();
//BigDecimal
static int ROUND_CEILING,ROUND_DOWN,ROUND_FLOOR,
           ROUND_HALF_DOWN,ROUND_HALF_EVEN,ROUND_HALF_UP,ROUND_UP;
a.divide(BigDecimal b,int scale,int round_mode);
a.doubleValue();
a.movePointLeft(int i);
a.pow(int);
a.setScale(int scale,int round_mode);
a.stripTrailingZeros();
//StringBuilder
StringBuilder sb=new StringBuilder();
sb.append(elem);
out.println(sb);
//StringTokenizer
StringTokenizer st=new StringTokenizer(in.readLine());
st.countTokens();
st.hasMoreTokens();
st.nextToken();
//Vector
a.add(elem);
a.add(index,elem);
a.clear();
a.elementAt(index);
a.isEmpty();
a.remove(index);
a.set(index,elem);
a.size();
//Queue
a.add(elem);
a.peek();//front
a.poll();//pop
```

```
//Integer Double Long
```

4.24 vimrc

```
set nu ai ci si mouse=a ts=4 sts=4 sw=4

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

nmap<F3>␣:␣vs␣%<.in␣<CR>
nmap<F8>␣:␣!./%<␣<␣%<.in␣<CR>
nmap<F9>␣:␣make␣%<␣<CR>

nmap<F4>␣:␣!gedit␣%␣<CR>
nmap<F5>␣:␣!./%<␣<CR>
nmap<F6>␣:␣!java␣%<␣<␣%<.in␣<CR>
nmap<F10>␣:␣!javac␣%␣<CR>
```