

BakaBakaBerserker

Uni

October 23, 2015

目录

1	Basic	1
1.1	Attentions	1
1.2	Binary Search	1
1.3	Bitwise Op	1
1.4	Date	1
1.5	Fast IO	2
1.6	Rope	2
1.7	Trick	2
2	Data Structures	3
2.1	Binary Indexed Tree	3
2.2	Block	3
2.3	Chairman Tree	4
2.4	Interval Tree	4
2.5	Interval Tree 2D	5
2.6	Interval Tree Summary	6
2.7	Monotone	10
2.8	Partition Tree	10
2.9	RMQ	11
2.10	Shunting Yard	11
2.11	Union Find	12
3	String	12
3.1	AC	12
3.2	Hash	12
3.3	KMP	13
3.4	Manacher	13
3.5	Trie	14
4	Geometry	14
4.1	include	14
4.2	point	14
4.3	line	15
4.4	circle	15
4.5	polygon	17
4.6	polygons	19
4.7	circles	20
4.8	halfplane	21
4.9	point3	21
4.10	line3	22
4.11	plane	22
5	Number Theory	23
5.1	GCD	23
5.2	Prime	23
6	Graph Theory	23
6.1	Kruskal	23
7	DP	24
7.1	LCS	24
7.2	LIS	24

1 Basic

1.1 Attentions

```

1 1. 调试
2 输入输出格式? 调试信息? 文件函数? 初始化? 算术溢出? 数组大小?
3 左右端点范围? acos/asin/sqrt 函数定义域? 精度问题?
4 RE: 爆栈? 整数除以 0? 数组严重越界? 未 return 0?
5
6 2. Code::Blocks 更换终端
7 gnome-terminal -t $TITLE -x
8
9 3. 0 除以负数是负零, 输出时加 eps, 注意 sqrt(-0.00)
10
11 4. double 全局变量初始可能非零

```

1.2 Binary Search

```

1 // 二分查找, 区间 [l, r]
2 int bs(int k) {
3     int l = 0, r = n - 1, p;
4     while (l <= r) {
5         p = (l + r) >> 1;
6         if (a[p] == k)
7             return p;
8         if (a[p] < k)
9             l = p + 1;
10        else
11            r = p - 1;
12    }
13    return -1;
14 }
15
16 // 二分查找, 区间 [l, r]
17 int bs(int k) {
18     int l = 0, r = n, p;
19     while (l < r) {
20         p = (l + r) >> 1;
21         if (a[p] == k)
22             return p;
23         if (a[p] < k)
24             l = p + 1;
25         else
26             r = p;
27    }
28    return -1;
29 }
30
31 // lower_bound (第一个 >=) 区间 [l, r)
32 // STL 排序
33 bool cmp(const A &x, const A &y) {
34     return x.v < y.v;
35 }
36
37 int lb(int k) {
38     int cnt = r - 1, it, step;
39     while (cnt > 0) {
40         step = cnt / 2;
41         it = 1 + step;
42         if (a[it] < k) {
43             l = ++it;
44             cnt -= step + 1;
45         } else {
46             cnt = step;
47         }
48     }
49     return l;
50 }
51
52 // upper_bound (第一个 >) 区间 [l, r)
53 int ub(int k) {
54     int cnt = r - 1, it, step;
55     while (cnt > 0) {
56         step = cnt / 2;
57         it = 1 + step;
58         if (a[it] <= k) {
59             l = ++it;
60             cnt -= step + 1;
61         } else {
62             cnt = step;
63         }

```

```

    }
    return l;
}

// 二分查找答案 区间 [l, r)
void solve(int l, int r) {
    int m;
    while (r - l > 1) {
        m = (l + r) >> 1;
        ok(m) ? l = m : r = m;
    }
    return l;
}

```

```

// 二分迭代
double l = 0, r = 1, m;
for (int i = 0; i < 100; ++i) {
    m = (l + r) / 2.0;
    ok(m) ? l = m : r = m;
}

```

```

// 三分查找答案
double l = 0, r = INF, tmp, m1, m2;
while (l + eps < r) {
    tmp = (r - l) / 3.0;
    m1 = l + tmp;
    m2 = r - tmp;
    calc(m1) > calc(m2) + eps ? l = m1 : r = m2;
}
printf("Case #%d: %.2lf %.2lf\n", cas++, l, calc(l));

```

1.3 Bitwise Op

```

// 枚举长为 n 含 k 个 1 的 01 串
int n = 5, k = 3;
for (int s = (1 << k) - 1, u = 1 << n; s < u; ) {
    for (int i = 0; i < n; i++)
        printf("%d", (((s >> (n - 1 - i)) & 1) == 1));
    printf("\n");
    int b = s & -s;
    s = (s + b) | (((s ^ (s + b)) >> 2) / b);
}
// #include <bitset>
// 清零 reset()
// 计数 count()
// 翻转 flip()

```

1.4 Date

```

//日期函数
int days[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
struct date {
    int year, month, day;
};

//判闰年
inline int leap(int year) {
    return (year % 4 == 0 && year % 100 != 0) || year % 400 == 0;
}

//判合法性
inline int legal(date a) {
    if (a.month < 0 || a.month > 12) return 0;
    if (a.month == 2)
        return a.day > 0 && a.day <= 28 + leap(a.year);
    return a.day > 0 && a.day <= days[a.month - 1];
}

//比较日期大小
inline int datecmp(date a, date b) {
    if (a.year != b.year)
        return a.year - b.year;
    if (a.month != b.month)
        return a.month - b.month;
    return a.day - b.day;
}

//返回指定日期是星期几
int weekday(date a) {
    int tm = a.month >= 3 ? (a.month - 2) : (a.month + 10);
    ;
}

```

```

32     int ty = a.month >= 3 ? a.year : (a.year - 1);
33     return (ty + ty / 4 - ty / 100 + ty / 400 + (int)(2.6
    * tm - 0.2) + a.day) % 7;
34 }
35
36 //日期转天数偏移 (用于计算日期差?)
37 int date2int(date a) {
38     int ret = a.year * 365 + (a.year - 1) / 4 - (a.year -
    1) / 100 +
39         (a.year - 1) / 400, i; days[1] += leap(a.
    year);
40     for (i = 0; i < a.month - 1; ret += days[i++]);
41     days[1] = 28;
42     return ret + a.day;
43 }
44
45 //天数偏移转日期
46 date int2date(int a) {
47     date ret;
48     ret.year = a / 146097 * 400;
49     for (a %= 146097; a >= 365 + leap(ret.year);
    a -= 365 + leap(ret.year), ret.year++);
51     days[1] += leap(ret.year);
52     for (ret.month = 1; a >= days[ret.month - 1];
    a -= days[ret.month - 1], ret.month++);
53     days[1] = 28; ret.day = a + 1; return ret;
54 }
55 }

```

```

    if (in == '-') {
        neg = true;
        while ((in = getchar()) > '9' || in < '0')
            ;
    }
    num = in - '0';
    while (in = getchar(), in >= '0' && in <= '9')
        num *= 10, num += in - '0';
    if (neg)
        num = 0 - num;
}
inline void printf_(int num) {
    bool flag = false;
    if (num < 0) {
        putchar('-');
        num = -num;
    }
    int ans[10], top = 0;
    while (num != 0) {
        ans[top++] = num % 10;
        num /= 10;
    }
    if (top == 0)
        putchar('0');
    for (int i = top - 1; i >= 0; i--) {
        char ch = ans[i] + '0';
        putchar(ch);
    }
}
}

```

1.5 Fast IO

```

1 // 1. fread + 缓冲区法
2 struct FastIO {
3     static const int S = 131072;
4     int wpos;
5     char wbuf[S];
6     FastIO() : wpos(0) {
7     }
8     inline int xchar() {
9         static char buf[S];
10        static int len = 0, pos = 0;
11        if (pos == len)
12            pos = 0, len = fread(buf, 1, S, stdin);
13        if (pos == len)
14            return -1;
15        return buf[pos++];
16    }
17    inline int xuint() {
18        int s;
19        scanf("%d", &s);
20        return s;
21        int c = xchar(), x = 0;
22        while (c <= 32)
23            c = xchar();
24        for (; '0' <= c && c <= '9'; c = xchar())
25            x = x * 10 + c - '0';
26        return x;
27    }
28    inline int xint() {
29        int t;
30        scanf("%d", &t);
31        return t;
32        int s = 1, c = xchar(), x = 0;
33        while (c <= 32)
34            c = xchar();
35        if (c == '-')
36            s = -1, c = xchar();
37        for (; '0' <= c && c <= '9'; c = xchar())
38            x = x * 10 + c - '0';
39        return x * s;
40    }
41    ~FastIO() {
42        if (wpos)
43            fwrite(wbuf, 1, wpos, stdout), wpos = 0;
44    }
45 } io;
46
47 // 2. naive
48 inline void scanf_(int &num) {
49     char in;
50     bool neg = false;
51     while (((in = getchar()) > '9' || in < '0') && in !=
    '-');
52

```

1.6 Rope

```

#include <ext/rope>
using namespace __gnu_cxx;
int a[1000];
rope<int> x;
rope<int> x(a, a + n);
rope<int> a(x);

x->at(10);
x[10];
x->push_back(x); // 在末尾添加x
x->insert(pos, x); // 在pos插入x
x->erase(pos, x); // 从pos开始删除x个
x->replace(pos, x); // 从pos开始换成x
x->substr(pos, x); // 提取pos开始x个
// 一键可持久化, O(1) 的历史版本
his[0] = new rope<char>();
his[i] = new rope<char>(*his[i - 1]);
/*
题意: 设计数据结构支持插入删除反转字符串
分析: 由于rope的底层实现, insert, erase, get都是logn的
就是翻转不行, 不是自己手写的打不了标记啊!
答: 同时维护一正一反两个rope……反转即交换两个子串……
区间循环移位? 简单, 拆成多个子串连起来就好了……
区间a变b b变c c变d …… z变a? 呃……维护26个rope?
区间和? 滚蛋, 那是线段树的活
区间kth? sorry, 与数值有关的操作rope一概不支持……
*/

```

1.7 Trick

```

// C++ 扩栈
#pragma comment(linker, "/STACK:1024000000,1024000000")

// 读取空格分隔的数据 (个数不定)
gets(buf);
int v;
char *p = strtok(buf, " ");
while (p) {
    sscanf(p, "%d", &v); // v = atoi(p);
    p = strtok(NULL, " ");
}

// 读取空格分隔的字符串 (个数确定)
while (gets(s), s[0]) {
    sscanf(s, "%s%s", a, b);
    m[string(b)] = string(a);
}

// string 与 char* 互转
char a[42];

```

```

21 string b = string(a);
22 printf("%s", b.c_str());
23
24 // string 中查找字符
25 string s;
26 if (s.find('+') != string::npos) {
27     // s.find('+') 类型为 size_type
28 }
29
30 // string 中消除指定字符
31 string t = "";
32 for (int j = 0; j < (int)s.size(); ++j) {
33     if (s[j] != '.') {
34         t += s[j];
35     }
36 }
37 s = t;
38
39 // 优先队列结构体比较 (从小到大)
40 struct P {
41     int a, b;
42     bool operator<(const P &x) const {
43         return b > x.b;
44     }
45 }

```

2 Data Structures

2.1 Binary Indexed Tree

```

1 // 1. 单点
2 // 每次用之前 memset
3 int c[MX];
4
5 // 不判 0 可能操蛋
6 void U(int x, int v) {
7     if (x == 0) {
8         return;
9     }
10    for (; x <= n; x += (x & -x))
11        c[x] += v;
12 }
13
14 // [1,x]的和
15 int Q(int x) {
16     if (x == 0) {
17         return;
18     }
19     int r = 0;
20     for (; x > 0; x -= (x & -x))
21         r += c[x];
22     return r;
23 }
24
25 // 2. 二维
26 int c[MAXX][MAXY];
27
28 // 更新 (x,y)
29 void U(int x, int y, int v) {
30     for (; x <= MAXX; x += (x & -x))
31         for (int i = y; i <= MAXY; i += (i & -i))
32             c[x][i] += v;
33 }
34
35 // (1,1) 到 (x,y) 的和
36 int Q(int x, int y) {
37     int r = 0;
38     for (; x > 0; x -= (x & -x))
39         for (int i = y; i > 0; i -= (i & -i))
40             r += c[x][i];
41     return r;
42 }
43
44 // 3. 区间更新 [l,r]
45 // U(l-1, -c), U(r, c);
46 void U(int x, int v) {
47     for (; x > 0; x -= (x & -x))
48         b[x] += v;
49 }
50
51 // Q(x) 查询 a[x] 的值
52 int Q(int x) {

```

```

    int r = 0;
    for (; x <= n; x += (x & -x))
        r += b[x];
    return r;
}

/// 4. 区间更新、查询
// 更新[l,r]: U(r, c); if (l > 1) U(l-1, -c);
void U(int x, int v) {
    if (x == 0)
        return;
    for (int i = x; i > 0; i -= (i & -i))
        b[i] += v;
    for (int i = x; i <= n; i += (i & -i))
        c[i] += x * v;
}

// 查询[l,r]: Q(r) - Q(l-1);
int QB(int x) {
    int r = 0;
    for (; x <= n; x += (x & -x))
        r += b[x];
    return r;
}

int QC(int x) {
    int r = 0;
    for (; x > 0; x -= (x & -x))
        r += c[x];
    return r;
}

int Q(int x) {
    if (x)
        return QB(x) * x + QC(x-1);
    return 0;
}

// 5. 带重复的逆序对, 最大 10^9
#define MX 99999
typedef long long ll;
struct P {
    int v, w, i;
} p[MX];
int c[MX];
int x(P a, P b) {
    return a.v < b.v;
}
int y(P a, P b) {
    return a.i > b.i;
}
void U(int i) {
    for (; i < MX; i += i & -i)
        ++c[i];
}
int Q(int i) {
    int r = 0;
    for (; i > 0; i -= i & -i)
        r += c[i];
    return r;
}
int main() {
    int n, k;
    while (~scanf("%d", &n)) {
        ll a = 0;
        k = 1;
        memset(c, 0, sizeof(c));
        for (int i = 0; i < n; ++i)
            scanf("%d", &p[i].v), p[i].i = i;
        sort(p, p + n, x);
        p[0].w = k;
        for (int i = 1; i < n; ++i)
            p[i].w = p[i].v == p[i-1].v ? k : ++k;
        sort(p, p + n, y);
        for (int i = 0; i < n; ++i)
            a += Q(p[i].w-1), U(p[i].w);
        printf("%I64d\n", a);
    }
    return 0;
}

```

2.2 Block

```

1 // 桶的大小 p = sqrt(n), 块数 q = n / p + (n % p ? 1 : 0)
2 // 可以预处理 belong[i] = (i - 1) / p - 1
3 // 五维偏序: 分块 + bitset
4 l[i] = (i - 1) * p + 1, r[i] = i * p, r[q] = n;
5 bitset<50001> b[7][270];
6 bitset<50001> Ans[7];
7 int now[7];
8 struct node {
9     int x, y;
10 };
11 bool cmp(node a, node b) {
12     return a.x < b.x;
13 }
14 node a[7][50001];
15 int l[300], r[300];
16 int belong[50001];
17 int main() {
18     int t;
19     scanf("%d", &t);
20     for (int cas = 1; cas <= t; cas++) {
21         int n = read(), m = read();
22         for (int i = 1; i <= 5; i++)
23             for (int j = 1; j < 250; j++)
24                 b[i][j].reset();
25         for (int i = 1; i <= n; i++) {
26             for (int j = 1; j <= 5; j++) {
27                 a[j][i].x = read();
28                 a[j][i].y = i;
29             }
30         }
31         for (int i = 1; i <= 5; i++)
32             sort(a[i] + 1, a[i] + n + 1, cmp);
33         int block = sqrt(n);
34         int num = n / block;
35         if (n % block)
36             num++;
37         for (int i = 1; i <= num; i++)
38             l[i] = (i - 1) * block + 1, r[i] = i * block;
39         r[num] = n;
40         for (int i = 1; i <= n; i++)
41             belong[i] = (i - 1) / block + 1;
42         for (int i = 1; i <= 5; i++) {
43             for (int j = 1; j <= num; j++) {
44                 b[i][j] |= b[i][j - 1];
45                 for (int k = l[j]; k <= r[j]; k++)
46                     b[i][j][a[i][k].y] = 1;
47             }
48         }
49
50         int q = read();
51         int lastans = 0;
52         while (q--) {
53             for (int i = 1; i <= 5; i++)
54                 now[i] = read();
55             for (int i = 1; i <= 5; i++)
56                 now[i] ^= lastans;
57             for (int i = 1; i <= 5; i++)
58                 Ans[i].reset();
59             for (int i = 1; i <= 5; i++) {
60                 int L = 0, R = n;
61                 while (L <= R) {
62                     int mid = (L + R) / 2;
63                     if (now[i] >= a[i][mid].x)
64                         L = mid + 1;
65                     else
66                         R = mid - 1;
67                 }
68                 int p = L - 1;
69                 if (p == 0)
70                     continue;
71                 Ans[i] |= b[i][belong[p] - 1];
72                 for (int j = l[belong[p]]; j <= p; j++)
73                     Ans[i][a[i][j].y] = 1;
74             }
75             Ans[1] = Ans[1] & Ans[2] & Ans[3] & Ans[4] &
76                 Ans[5];
77             lastans = Ans[1].count();
78             printf("%d\n", lastans);
79         }
80     }

```

2.3 Chairman Tree

```

#include <cstdio>
#include <algorithm>
#define MX 100010
using namespace std;
struct Node {
    int l, r, v;
} t[MX * 20];
struct P {
    int v, i;
} p[MX];
bool cmp(P a, P b) {
    return a.v < b.v;
}
int n, cnt, pos[MX], root[MX];
void U(int v, int &x, int l = 1, int r = n) {
    t[cnt++] = t[x];
    x = cnt - 1;
    ++t[x].v;
    if (l == r) {
        return;
    }
    int m = (l + r) >> 1;
    if (v <= m) {
        U(v, t[x].l, l, m);
    } else {
        U(v, t[x].r, m + 1, r);
    }
}
int Q(int L, int R, int k, int l = 1, int r = n) {
    if (l == r) {
        return l;
    }
    int v = t[t[R].l].v - t[t[L].l].v;
    int m = (l + r) >> 1;
    if (k <= v) {
        return Q(t[L].l, t[R].l, k, l, m);
    }
    return Q(t[L].r, t[R].r, k - v, m + 1, r);
}
int main() {
    int m;
    t[0].l = t[0].r = t[0].v = root[0] = 0;
    while (~scanf("%d%d", &n, &m)) {
        for (int i = 1; i <= n; ++i) {
            scanf("%d", &p[i].v);
            p[i].i = i;
        }
        sort(p + 1, p + n + 1, cmp);
        for (int i = 1; i <= n; ++i) {
            pos[p[i].i] = i;
        }
        cnt = 1;
        for (int i = 1; i <= n; ++i) {
            root[i] = root[i - 1];
            U(pos[i], root[i]);
        }
        while (m--) {
            int l, r, k;
            scanf("%d%d%d", &l, &r, &k);
            printf("%d\n", p[Q(root[l - 1], root[r], k)].v);
        }
        return 0;
    }
}

```

2.4 Interval Tree

```

#include <cstdio>
#include <cstring>
#include <algorithm>

#define MX 1024000
#define ls l,m,n<<1 // lson
#define rs m+1,r,n<<1|1 // rson
#define lc n<<1 // lchild
#define rc n<<1|1 // rchild
using namespace std;

int num[MX], sum[MX << 2], ma[MX << 2], mi[MX << 2], add[
MX << 2];

```

```

13 int N, L, R, V, X;
14
15 void up(int n) {
16     sum[n] = sum[lc] + sum[rc];
17     ma[n] = max(ma[lc], ma[rc]);
18     mi[n] = min(mi[lc], mi[rc]);
19 }
20
21 void down(int n, int m) {
22     if (add[n]) {
23         add[lc] += add[n];
24         add[rc] += add[n];
25         sum[lc] += add[n] * (m - (m >> 1));
26         sum[rc] += add[n] * (m >> 1);
27         add[n] = 0;
28     }
29 }
30
31 void B(int l = 1, int r = N, int n = 1) {
32     add[n] = 0;
33     if (l == r) {
34         scanf("%d", &num[l]);
35         sum[n] = ma[n] = mi[n] = num[l];
36         return;
37     }
38     int m = (l + r) >> 1;
39     B(ls), B(rs), up(n);
40 }
41
42 // 单点更新 X 为 V
43 void U(int l = 1, int r = N, int n = 1) {
44     if (l == r) {
45         sum[n] = ma[n] = mi[n] = num[l] = V; // 赋值或增加
46         return;
47     }
48     down(n, r - l + 1);
49     int m = (l + r) >> 1;
50     if (X <= m) U(ls);
51     else U(rs);
52     up(n);
53 }
54
55 // 区间更新 [L,R] 为 V
56 void U(int l = 1, int r = N, int n = 1) {
57     if (L <= l && r <= R) {
58         add[n] += V, sum[n] += V * (r - l + 1);
59         return;
60     }
61     down(n, r - l + 1);
62     int m = (l + r) >> 1;
63     if (L <= m) U(ls);
64     if (m < R) U(rs);
65     up(n);
66 }
67
68 // 查询区间 [L,R]
69 int Q(int l = 1, int r = N, int n = 1) {
70     if (L <= l && r <= R) {
71         return sum[n];
72     }
73     down(n, r - l + 1);
74     int ans = 0, m = (l + r) >> 1;
75     if (L <= m) ans += Q(ls);
76     // ans = max(ans, Q(ls));
77     if (m < R) ans += Q(rs);
78     // ans = max(ans, Q(rs));
79     return ans;
80 }

```

2.5 Interval Tree 2D

```

1 #include <cstdio>
2 #include <algorithm>
3 #define maxn 510
4 #define it tree[p1][p2]
5 using namespace std;
6 struct Seg_Tree2D
7 {
8     int minv, maxv;
9     friend Seg_Tree2D operator + (const Seg_Tree2D &a,
10     const Seg_Tree2D &b)
11     {
12         Seg_Tree2D c;

```

```

12         c.minv = min(a.minv, b.minv);
13         c.maxv = max(a.maxv, b.maxv);
14         return c;
15     }
16 } tree[maxn << 2][maxn << 2];
17 int matrix[maxn][maxn], n, m;
18 void Build2(int p1, int p2, int l, int r, int a, int b)
19 {
20     if (a == b)
21     {
22         if (l == r)
23             it.minv = it.maxv = matrix[l][a];
24         else
25             it = tree[p1 << 1][p2] + tree[p1 << 1 | 1][p2];
26     }
27     else
28     {
29         int mid = (a + b) >> 1;
30         Build2(p1, p2 << 1, l, r, a, mid), Build2(p1, p2
31         << 1 | 1, l, r, mid + 1, b);
32         it = tree[p1][p2 << 1] + tree[p1][p2 << 1 | 1];
33     }
34 }
35 void Build1(int p1, int l, int r)
36 {
37     if (l == r)
38     {
39         Build2(p1, 1, l, r, 1, m);
40         return ;
41     }
42     int mid = (l + r) >> 1;
43     Build1(p1 << 1, l, mid), Build1(p1 << 1 | 1, mid + 1,
44     r);
45     Build2(p1, 1, l, r, 1, m);
46 }
47 void Modify2(int p1, int p2, int l, int r, int a, int b,
48 int y, int v)
49 {
50     if (a == b)
51     {
52         if (l == r)
53             it.minv = it.maxv = v;
54         else
55             it = tree[p1 << 1][p2] + tree[p1 << 1 | 1][p2];
56     }
57     else
58     {
59         int mid = (a + b) >> 1;
60         if (y <= mid)
61             Modify2(p1, p2 << 1, l, r, a, mid, y, v);
62         else
63             Modify2(p1, p2 << 1 | 1, l, r, mid + 1, b, y,
64             v);
65         it = tree[p1][p2 << 1] + tree[p1][p2 << 1 | 1];
66     }
67 }
68 void Modify1(int p1, int l, int r, int x, int y, int v)
69 {
70     if (l == r)
71     {
72         Modify2(p1, 1, l, r, 1, m, y, v);
73         return ;
74     }
75     int mid = (l + r) >> 1;
76     if (x <= mid)
77         Modify1(p1 << 1, l, mid, x, y, v);
78     else
79         Modify1(p1 << 1 | 1, mid + 1, r, x, y, v);
80     Modify2(p1, 1, l, r, 1, m, y, v);
81 }
82 Seg_Tree2D Query2(int p1, int p2, int l, int r, int a, int
83 b)
84 {
85     if (l == a && r == b)
86         return it;
87     int mid = (l + r) >> 1;
88     if (b <= mid)
89         return Query2(p1, p2 << 1, l, mid, a, b);
90     else if (mid < a)
91         return Query2(p1, p2 << 1 | 1, mid + 1, r, a, b);
92     return Query2(p1, p2 << 1, l, mid, a, mid) + Query2(p1

```

```

    , p2 << 1 | 1, mid + 1, r, mid + 1, b);
88 }
89 Seg_Tree2D Query1(int p1, int l, int r, int ax, int ay,
int bx, int by)
90 {
91     if (l == ax && r == bx)
92         return Query2(p1, 1, 1, m, ay, by);
93     int mid = (l + r) >> 1;
94     if (bx <= mid)
95         return Query1(p1 << 1, l, mid, ax, ay, bx, by);
96     else if (mid < ax)
97         return Query1(p1 << 1 | 1, mid + 1, r, ax, ay, bx,
by);
98     return Query1(p1 << 1, l, mid, ax, ay, mid, by) +
Query1(p1 << 1 | 1, mid + 1, r, mid + 1, ay, bx, by);
99 }
100 void read()
101 {
102     scanf("%d %d", &n, &m);
103     for (int i = 1; i <= n; i++)
104         for (int j = 1; j <= m; j++)
105             scanf("%d", &matrix[i][j]);
106     Build1(1, 1, n);
107 }
108 void Query()
109 {
110     char task[10];
111     int q;
112     scanf("%d", &q);
113     Seg_Tree2D ans;
114     for (int i = 1, a, b, c, d; i <= q; i++)
115     {
116         scanf("%s", task);
117         if (task[0] == 'c')
118         {
119             scanf("%d %d %d", &a, &b, &c);
120             Modify1(1, 1, n, a, b, c);
121         }
122         else
123         {
124             scanf("%d %d %d %d", &a, &b, &c, &d);
125             ans = Query1(1, 1, n, a, b, c, d);
126             printf("%d %d\n", ans.maxv, ans.minv);
127         }
128     }
129 }
130 int main()
131 {
132     read();
133     Query();
134     return 0;
135 }

```

```

} x[MX];
bool cmp(seg a, seg b) {
    return a.p < b.p;
}
double a, b, c, d, y[MX], sum[MX << 2];
int L, R, V, N, add[MX << 2];
void U(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R) {
        add[n] += V;
        // printf("#%d %d %d\n", l, r, add[n]);
        sum[n] = add[n] ? (y[r + 1] - y[l]) : (l == r ? 0
: sum[lc] + sum[rc]);
        return;
    }
    int m = (l + r) >> 1;
    if (L <= m)
        U(ls);
    if (m < R)
        U(rs);
    sum[n] = add[n] ? (y[r + 1] - y[l]) : (l == r ? 0 :
sum[lc] + sum[rc]);
}
int main() {
    int n, k, cas = 0;
    while (scanf("%d", &n) && n) {
        n *= 2;
        for (int i = 1; i <= n; i += 2) {
            scanf("%lf%lf%lf%lf", &a, &b, &c, &d);
            x[i] = seg(b, d, a, 1);
            x[i + 1] = seg(b, d, c, -1);
            y[i] = b;
            y[i + 1] = d;
        }
        sort(x + 1, x + n + 1, cmp);
        sort(y + 1, y + n + 1);
        N = unique(y + 1, y + n + 1) - y - 1;
        memset(sum, 0, sizeof(sum));
        memset(add, 0, sizeof(add));
        double ans = 0.0;
        for (int i = 1; i < n; ++i) {
            L = lower_bound(y + 1, y + N + 1, x[i].l) - y;
            R = lower_bound(y + 1, y + N + 1, x[i].r) - y
- 1;
            V = x[i].d;
            if (L <= R)
                U();
            ans += sum[1] * (x[i + 1].p - x[i].p);
            // printf("%.2lf\n", sum[1]);
        }
        printf("Test case #%d\nTotal explored area: %.2lf\n",
++cas, ans);
    }
    return 0;
}

```

2.6 Interval Tree Summary

```

1 // 1. LA 3787
2 // 单点赋值，查询区间哈希值。
3 ll b, p, c[MX << 2], s[MX];
4 void up() {
5     c[n] = (c[lc] * s[r - m] + c[rc]) % p;
6 }
7 ll Q(int l = 1, int r = N, int n = 1) {
8     if (x <= l && r <= y) {
9         return c[n] * s[y - r];
10    }
11    ... return ans % p;
12 }
13 void init() {
14     s[0] = 1;
15     for (int i = 1; i <= N; ++i)
16         s[i] = (s[i - 1] * b) % p;
17     memset(c, 0, sizeof(c));
18 }
19
20 // 2. 扫描线
21 struct seg {
22     double l, r, p;
23     int d;
24     seg() {
25     }
26     seg(double _l, double _r, double _p, int _d) {
27         l = _l, r = _r, p = _p, d = _d;
28     }

```

```

// 3. HDU 3308
// 最长连续递增序列
int N, X, V, num[MX], mm[MX << 2], lm[MX << 2], rm[MX <<
2];
void B(int l = 1, int r = N, int n = 1) {
    if (l == r) {
        scanf("%d", &num[l]);
        lm[n] = rm[n] = mm[n] = 1;
        return;
    }
    int m = (l + r) >> 1, k = r - l + 1;
    B(lson), B(rson);
    lm[n] = lm[lc];
    if (lm[lc] == k - (k >> 1) && num[m] < num[m + 1])
        lm[n] += lm[rc];
    rm[n] = rm[rc];
    if (rm[rc] == (k >> 1) && num[m] < num[m + 1])
        rm[n] += rm[lc];
    mm[n] = max(mm[lc], mm[rc]);
    if (num[m] < num[m + 1])
        mm[n] = max(mm[n], rm[lc] + lm[rc]);
}
void U(int l = 1, int r = N, int n = 1) {
    if (l == r) {
        num[l] = V;
        return;
    }
    int m = (l + r) >> 1, k = r - l + 1;

```

```

107     if (X <= m)
108         U(lson);
109     else
110         U(rson);
111     lm[n] = lm[lc];
112     if (lm[lc] == k - (k >> 1) && num[m] < num[m + 1])
113         lm[n] += lm[rc];
114     rm[n] = rm[rc];
115     if (rm[rc] == (k >> 1) && num[m] < num[m + 1])
116         rm[n] += rm[lc];
117     mm[n] = max(mm[lc], mm[rc]);
118     if (num[m] < num[m + 1])
119         mm[n] = max(mm[n], rm[lc] + lm[rc]);
120 }
121 int QL(int L, int R, int l = 1, int r = N, int n = 1) {
122     if (L <= l && r <= R)
123         return lm[n];
124     int m = (l + r) >> 1;
125     if (L > m)
126         return QL(L, R, rson);
127     if (R <= m)
128         return QL(L, R, lson);
129     int ans = QL(L, m, lson), k = m - L + 1;
130     if (ans == k && num[m] < num[m + 1])
131         ans += QL(m + 1, R, rson);
132     return ans;
133 }
134 int QR(int L, int R, int l = 1, int r = N, int n = 1) {
135     if (L <= l && r <= R)
136         return rm[n];
137     int m = (l + r) >> 1;
138     if (L > m)
139         return QR(L, R, rson);
140     if (R <= m)
141         return QR(L, R, lson);
142     int ans = QR(m + 1, R, rson), k = R - m;
143     if (ans == k && num[m] < num[m + 1])
144         ans += QR(L, m, lson);
145     return ans;
146 }
147 int Q(int L, int R, int l = 1, int r = N, int n = 1) {
148     if (L <= l && r <= R)
149         return mm[n];
150     int m = (l + r) >> 1;
151     if (L > m)
152         return Q(L, R, rson);
153     if (R <= m)
154         return Q(L, R, lson);
155     int ans = max(Q(L, m, lson), Q(m + 1, R, rson));
156     if (num[m] < num[m + 1])
157         ans = max(ans, QR(L, m, lson) + QL(m + 1, R, rson));
158     return ans;
159 }
160 int main() {
161     int t, m, L, R;
162     char o[9];
163     scanf("%d", &t);
164     while (t--) {
165         scanf("%d%d", &N, &m), B();
166         while (m--) {
167             scanf("%s", o);
168             if (o[0] == 'Q') {
169                 scanf("%d%d", &L, &R), printf("%d\n", Q(L
170                     + 1, R + 1));
171             } else {
172                 scanf("%d%d", &X, &V), ++X, U();
173             }
174         }
175         return 0;
176     }
177 }
178 // 4. POJ 3277
179 // 更新区间, 只保留最大值
180 int v, N, L, R, x[MX], y[MX], h[MX], p[MX], s[MX << 2];
181 void B(int l = 1, int r = N, int n = 1) {
182     if (l == r - 1)
183         return;
184     int m = (l + r) >> 1;
185     B(lson), B(rson);
186 }
187 void U(int L, int R, int l = 1, int r = N, int n = 1) {

```

```

    if (p[l] == L && p[r] == R) {
188         if (s[n] < h[v])
189             s[n] = h[v];
190         return;
191     }
192     int m = (l + r) >> 1;
193     if (R <= p[m])
194         U(L, R, lson);
195     else if (L >= p[m])
196         U(L, R, rson);
197     else
198         U(L, p[m], lson), U(p[m], R, rson);
199 }
200 }
201 ll Q(int l = 1, int r = N, int n = 1, int t = 0) {
202     if (s[n] < t)
203         s[n] = t;
204     if (l == r - 1)
205         return (ll)(p[r] - p[l]) * s[n];
206     int m = (l + r) >> 1;
207     return Q(lson, s[n]) + Q(rson, s[n]);
208 }
209 int main() {
210     int m;
211     scanf("%d", &m);
212     for (int i = 1; i <= m; ++i) {
213         scanf("%d%d", &x[i], &y[i], &h[i]);
214         p[++N] = x[i], p[++N] = y[i];
215     }
216     sort(p + 1, p + N + 1);
217     N = unique(p + 1, p + N + 1) - p - 1;
218     B();
219     for (v = 1; v <= m; ++v)
220         U(x[v], y[v]);
221     printf("%lld\n", Q());
222     return 0;
223 }
224
225 // 5. HDU 3333
226 // 区间不重复数字和
227 struct S {
228     int l, r, i;
229 } p[MX];
230 int x[MX], a[MX], f[MX];
231 ll c[MX], ans[MX];
232 int cmp(S a, S b) {
233     return a.r < b.r;
234 }
235 void U(int x, int v) {
236     for (; x < MX; x += (x & -x))
237         c[x] += v;
238 }
239 ll Q(int x) {
240     ll s = 0;
241     for (; x > 0; x -= (x & -x))
242         s += c[x];
243     return s;
244 }
245 int B(int v, int u) {
246     int l = 0, r = u - 1, m;
247     while (l <= r) {
248         m = (l + r) >> 1;
249         if (x[m] == v)
250             return m;
251         if (x[m] < v)
252             l = m + 1;
253         else
254             r = m - 1;
255     }
256     return -1;
257 }
258 int main() {
259     int t, n, q, l, r, cnt;
260     scanf("%d", &t);
261     while (t--) {
262         memset(c, 0, sizeof(c));
263         memset(f, 0, sizeof(f));
264         scanf("%d", &n), cnt = n;
265         for (int i = 1; i <= n; ++i)
266             scanf("%d", &a[i]), x[i - 1] = a[i];
267         sort(x, x + cnt);
268         cnt = unique(x, x + cnt) - x;
269         scanf("%d", &q);
270         for (int i = 0; i < q; ++i)

```



```

271         scanf("%d%d", &p[i].l, &p[i].r), p[i].i = i;
272     sort(p, p + q, cmp);
273     int k = 0, d;
274     for (int i = 1; i <= n; ++i) {
275         d = B(a[i], cnt);
276         if (f[d])
277             U(f[d], -a[i]);
278         U(i, a[i]);
279         f[d] = i;
280         for (; k < q; ++k) {
281             if (p[k].r == i)
282                 ans[p[k].i] = Q(p[k].r) - Q(p[k].l - 1);
283         }
284         else
285             break;
286     }
287     for (int i = 0; i < q; ++i)
288         printf("%I64d\n", ans[i]);
289 }
290 return 0;
291 }
292
293 // 6. POJ 2452
294 // 最大的 j-i 满足 a[i] 最小 a[j] 最大
295 int num[MX], ma[MX << 2], mi[MX << 2];
296 int L, R, N, ans;
297 inline void up(int n) {
298     ma[n] = num[ma[lc]] > num[ma[rc]] ? ma[lc] : ma[rc];
299     mi[n] = num[mi[lc]] < num[mi[rc]] ? mi[lc] : mi[rc];
300 }
301 void B(int l = 1, int r = N, int n = 1) {
302     if (l == r) {
303         scanf("%d", &num[l]);
304         ma[n] = mi[n] = l;
305         return;
306     }
307     int m = l + r >> 1;
308     B(lson), B(rson), up(n);
309 }
310 int QA(int l = 1, int r = N, int n = 1) {
311     if (L <= l && r <= R)
312         return ma[n];
313     int ans = 0, m = l + r >> 1, t;
314     if (L <= m)
315         t = QA(lson), ans = (num[ans] > num[t]) ? ans : t;
316     if (m < R)
317         t = QA(rson), ans = (num[ans] > num[t]) ? ans : t;
318     return ans;
319 }
320 int QB(int l = 1, int r = N, int n = 1) {
321     if (L <= l && r <= R)
322         return mi[n];
323     int ans = N + 1, m = l + r >> 1, t;
324     if (L <= m)
325         t = QB(lson), ans = (num[ans] < num[t]) ? ans : t;
326     if (m < R)
327         t = QB(rson), ans = (num[ans] < num[t]) ? ans : t;
328     return ans;
329 }
330 int S(int l, int r) {
331     if (l >= r)
332         return -1;
333     L = l, R = r;
334     int ans = -1, a = QA(), b = QB();
335     if (a > b)
336         ans = max(a - b, max(S(l, b), S(a, r)));
337     else
338         ans = max(S(a + 1, b - 1), max(S(l, a), S(b, r)));
339     return ans;
340 }
341 int main() {
342     while (~scanf("%d", &N)) {
343         B(), num[N + 1] = 111111;
344         printf("%d\n", S(1, N));
345     }
346     return 0;
347 }
348
349 // 7. SPOJ GSS3
350 // 区间最大子序列和
351 int num, sum[MX << 2], mm[MX << 2], lm[MX << 2], rm[MX << 2];

```

```

int L, R, N, x, v;
352 inline void up(int n) {
353     sum[n] = sum[lc] + sum[rc];
354     lm[n] = max(lm[lc], sum[lc] + lm[rc]);
355     rm[n] = max(rm[rc], sum[rc] + rm[lc]);
356     mm[n] = max(rm[lc] + lm[rc], max(mm[lc], mm[rc]));
357 }
358 void B(int l = 1, int r = N, int n = 1) {
359     if (l == r) {
360         scanf("%d", &sum[n]);
361         mm[n] = lm[n] = rm[n] = sum[n];
362         return;
363     }
364     int m = l + r >> 1;
365     B(lson), B(rson), up(n);
366 }
367 void U(int l = 1, int r = N, int n = 1) {
368     if (l == r) {
369         mm[n] = lm[n] = rm[n] = sum[n] = R;
370         return;
371     }
372     int m = l + r >> 1;
373     if (L <= m)
374         U(lson);
375     else
376         U(rson);
377     up(n);
378 }
379 int QL(int l = 1, int r = N, int n = 1) {
380     if (L <= l && r <= R)
381         return lm[n];
382     int m = l + r >> 1, ans;
383     if (L > m)
384         return QL(rson);
385     if (R <= m)
386         return QL(lson);
387     return max(QL(rson) + sum[lc], lm[lc]);
388 }
389 int QR(int l = 1, int r = N, int n = 1) {
390     if (L <= l && r <= R)
391         return rm[n];
392     int m = l + r >> 1, ans;
393     if (L > m)
394         return QR(rson);
395     if (R <= m)
396         return QR(lson);
397     return max(QR(lson) + sum[rc], rm[rc]);
398 }
399 int Q(int l = 1, int r = N, int n = 1) {
400     if (L <= l && r <= R)
401         return mm[n];
402     int m = l + r >> 1, ans;
403     if (L > m)
404         return Q(rson);
405     if (R <= m)
406         return Q(lson);
407     return max(QR(lson) + QL(rson), max(Q(lson), Q(rson)));
408 }
409 int main() {
410     int m, q;
411     scanf("%d", &N), B();
412     scanf("%d", &m);
413     while (m--) {
414         scanf("%d%d%d", &q, &L, &R);
415         if (q)
416             printf("%d\n", Q());
417         else
418             U();
419     }
420     return 0;
421 }
422
423 // 8. SPOJ GSS1
424 int N, sum[MX], mm[MX << 2], lm[MX << 2], rm[MX << 2];
425 int get(int l, int r) {
426     return sum[r] - sum[l - 1];
427 }
428 void B(int l = 1, int r = N, int n = 1) {
429     if (l == r) {
430         scanf("%d", &mm[n]);
431         lm[n] = rm[n] = mm[n];
432         sum[l] = sum[l - 1] + mm[n];
433     }

```

```

434     return;
435 }
436 int m = l + r >> 1;
437 B(lson), B(rson);
438 lm[n] = max(lm[lc], get(l, m) + lm[rc]);
439 rm[n] = max(rm[rc], get(m + 1, r) + rm[lc]);
440 mm[n] = max(rm[lc] + lm[rc], max(mm[lc], mm[rc]));
441 }
442 int QL(int L, int R, int l = 1, int r = N, int n = 1) {
443     if (L <= l && r <= R)
444         return lm[n];
445     int m = l + r >> 1, ans;
446     if (L > m)
447         return QL(L, R, rson);
448     if (R <= m)
449         return QL(L, R, lson);
450     return max(QL(m + 1, R, rson) + get(L, m), max(get(L,
451 m), QL(L, m, lson)));
452 }
453 int QR(int L, int R, int l = 1, int r = N, int n = 1) {
454     if (L <= l && r <= R)
455         return rm[n];
456     int m = l + r >> 1, ans;
457     if (L > m)
458         return QR(L, R, rson);
459     if (R <= m)
460         return QR(L, R, lson);
461     return max(QR(L, m, lson) + get(m + 1, R),
462 max(get(m + 1, R), QR(m + 1, R, rson)));
463 }
464 int Q(int L, int R, int l = 1, int r = N, int n = 1) {
465     if (L <= l && r <= R)
466         return mm[n];
467     int m = l + r >> 1, ans;
468     if (L > m)
469         return Q(L, R, rson);
470     if (R <= m)
471         return Q(L, R, lson);
472     return max(QR(L, m, lson) + QL(m + 1, R, rson),
473 max(QL(L, m, lson), Q(m + 1, R, rson)));
474 }
475 int main() {
476     int m, L, R;
477     scanf("%d", &N), B();
478     scanf("%d", &m);
479     while (m--)
480         scanf("%d%d", &L, &R), printf("%d\n", Q(L, R));
481     return 0;
482 }
483 // 9. 区间更新与赋值
484 ll sum[MX << 2], add[MX << 2], V;
485 bool se[MX << 2];
486 int L, R, N;
487 inline void up(int n) {
488     sum[n] = sum[lc] + sum[rc];
489 }
490 void B(int l = 1, int r = N, int n = 1) {
491     add[n] = 0;
492     se[n] = false;
493     if (l == r) {
494         sum[n] = 0;
495         return;
496     }
497     int m = (l + r) >> 1;
498     B(ls), B(rs), up(n);
499 }
500 void down(int n, int m) {
501     if (se[n]) {
502         se[lc] = se[rc] = se[n];
503         sum[lc] = sum[rc] = 0;
504         se[n] = false;
505         add[lc] = add[rc] = 0;
506     }
507     if (add[n]) {
508         add[lc] += add[n];
509         add[rc] += add[n];
510         sum[lc] += add[n] * (m - (m >> 1));
511         sum[rc] += add[n] * (m >> 1);
512         add[n] = 0;
513     }
514 }
515 void U(int l = 1, int r = N, int n = 1) {
516     if (L <= l && r <= R) {
517         add[n] += V, sum[n] += V * (r - l + 1);
518         return;
519     }
520     down(n, r - l + 1);
521     int m = (l + r) >> 1;
522     if (L <= m)
523         U(ls);
524     if (m < R)
525         U(rs);
526     up(n);
527 }
528 void S(int l = 1, int r = N, int n = 1) {
529     if (L <= l && r <= R) {
530         se[n] = true, add[n] = false, sum[n] = 0;
531         return;
532     }
533     down(n, r - l + 1);
534     int m = (l + r) >> 1;
535     if (L <= m)
536         S(ls);
537     if (m < R)
538         S(rs);
539     up(n);
540 }
541 ll Q(int l = 1, int r = N, int n = 1) {
542     if (L <= l && r <= R) {
543         return sum[n];
544     }
545     down(n, r - l + 1);
546     ll ans = 0, m = (l + r) >> 1;
547     if (L <= m)
548         ans += Q(ls);
549     if (m < R)
550         ans += Q(rs);
551     return ans;
552 }
553 int main() {
554     int t, m;
555     scanf("%d", &t);
556     while (t--) {
557         scanf("%d%d", &N, &m), B();
558         ll ans = 0;
559         int p = 0, q = 0;
560         while (m--) {
561             scanf("%d", &q);
562             L = 1, R = N, V = q - p;
563             U(), p = q;
564             scanf("%d%d", &L, &R);
565             ans += Q(), S();
566         }
567         printf("%lld\n", ans);
568     }
569     return 0;
570 }
571 // 10. UVA 1400
572 typedef long long ll;
573 typedef pair<int, int> seg;
574 ll sum[MX];
575 int pre[MX << 2], suf[MX << 2];
576 seg sub[MX << 2];
577 ll get(int l, int r) {
578     return sum[r] - sum[l - 1];
579 }
580 ll get(seg s) {
581     return get(s.first, s.second);
582 }
583 seg max(seg a, seg b) {
584     if (get(a) != get(b))
585         return get(a) > get(b) ? a : b;
586     return a < b ? a : b;
587 }
588 void pushUp(int l, int r, int n) {
589     ll v1 = get(l, pre[n << 1]), v2 = get(l, pre[n << 1 |
590 1]);
591     if (v1 == v2)
592         pre[n] = min(pre[n << 1], pre[n << 1 | 1]);
593     else
594         pre[n] = v1 > v2 ? pre[n << 1] : pre[n << 1 | 1];
595     v1 = get(suf[n << 1], r), v2 = get(suf[n << 1 | 1], r);
596     if (v1 == v2)

```

```

597     suf[n] = min(suf[n << 1], suf[n << 1 | 1]);
598     else
599         suf[n] = v1 > v2 ? suf[n << 1] : suf[n << 1 | 1];
600     sub[n] = max(make_pair(suf[n << 1], pre[n << 1 | 1]),
601                 max(sub[n << 1], sub[n << 1 | 1]));
602 }
603 void build(int l, int r, int n) {
604     if (l == r) {
605         pre[n] = suf[n] = l, sub[n] = make_pair(l, l);
606         return;
607     }
608     int m = (l + r) >> 1;
609     build(lson);
610     build(rson);
611     pushUp(l, r, n);
612 }
613 seg prefix(int L, int R, int l, int r, int n) {
614     if (pre[n] <= R)
615         return make_pair(l, pre[n]);
616     int m = (l + r) >> 1;
617     if (R <= m)
618         return prefix(L, R, lson);
619     seg ans = prefix(L, R, rson);
620     ans.first = l;
621     return max(ans, make_pair(l, pre[n << 1]));
622 }
623 seg suffix(int L, int R, int l, int r, int n) {
624     if (suf[n] >= L)
625         return make_pair(suf[n], r);
626     int m = (l + r) >> 1;
627     if (L > m)
628         return suffix(L, R, rson);
629     seg ans = suffix(L, R, lson);
630     ans.second = r;
631     return max(ans, make_pair(suf[n << 1 | 1], r));
632 }
633 seg query(int L, int R, int l, int r, int n) {
634     if (L <= l && r <= R)
635         return sub[n];
636     int m = (l + r) >> 1;
637     if (R <= m)
638         return query(L, R, lson);
639     if (L > m)
640         return query(L, R, rson);
641     return max(max(query(L, R, lson), query(L, R, rson)),
642               make_pair(suffix(L, R, lson).first, prefix(
643                   L, R, rson).second));
644 }
645 int main() {
646     int cas = 0, n, m, a, b;
647     while (~scanf("%d%d", &n, &m)) {
648         sum[0] = 0;
649         for (int i = 0; i < n; ++i)
650             scanf("%d", &a), sum[i + 1] = sum[i] + a;
651         build(1, n, 1);
652         printf("Case %d:\n", ++cas);
653         while (m--) {
654             scanf("%d%d", &a, &b);
655             seg ans = query(a, b, 1, n, 1);
656             printf("%d %d\n", ans.first, ans.second);
657         }
658     }
659     return 0;

```

2.7 Monotone

```

1 // 单调队列
2 // 利用双端队列实现，在队列中存数组下标。
3 // 滚动窗口：每次从头弹出不在窗口内的元素，从尾弹出小于新
  元素的元素，新元素插入到尾。
4 #include <cstdio>
5 #include <algorithm>
6 #include <deque>
7 #define MX 1000001
8 using namespace std;
9 typedef long long ll;
10 ll p[MX];
11 int main() {
12     int t, n, k, s;
13     scanf("%d", &t);
14     while (t--) {
15         scanf("%d%d%d", &n, &k, &s);

```

```

p[0] = s;
for (int i = 1; i < n; ++i)
    p[i] = (1LL * p[i - 1] * 1103515245 + 12345) %
        (2147483648LL);
ll ans = 0;
deque<int> q;
for (int i = 0; i < n; ++i) {
    while (!q.empty() && (i - q.front() >= k)) {
        q.pop_front();
    }
    while (!q.empty() && p[q.back()] <= p[i]) {
        q.pop_back();
    }
    q.push_back(i);
    if (i >= k - 1)
        ans += p[q.front()];
}
printf("%lld\n", ans);
}
return 0;
}

// 单调栈（最大面积）
// 同样存下标。
// 最大面积：正反扫两遍维护 l, r 数组，表示 h[i] 能向左/右
// 扩展的最大长度。
int h[MX], l[MX], r[MX];
stack<int> s;
int main() {
    int n;
    while (scanf("%d", &n) && n) {
        for (int i = 0; i < n; ++i) {
            scanf("%d", &h[i]);
        }
        while (!s.empty())
            s.pop();
        for (int i = 0; i < n; ++i) {
            while (!s.empty() && h[s.top()] >= h[i])
                s.pop();
            l[i] = (s.empty() ? 0 : (s.top() + 1));
            s.push(i);
        }
        while (!s.empty())
            s.pop();
        for (int i = n - 1; i >= 0; --i) {
            while (!s.empty() && h[s.top()] >= h[i])
                s.pop();
            r[i] = (s.empty() ? n : s.top());
            s.push(i);
        }
        ll a = 0;
        for (int i = 0; i < n; ++i)
            a = max(a, (ll)h[i] * (r[i] - l[i]));
        printf("%lld\n", a);
    }
    return 0;
}

```

2.8 Partition Tree

```

#include <cstdio>
#include <algorithm>
#define MX 100010
using namespace std;
int p[MX], s[20][MX], sum[20][MX];
void B(int l, int r, int n = 0) {
    int m = (l + r) >> 1;
    int L = l, R = m + 1;
    int x = p[m];
    sum[n][l] = 0;
    int cnt = m - l + 1;
    for (int i = l; i <= r; ++i) {
        if (s[n][i] < x) {
            --cnt;
        }
    }
    for (int i = l; i <= r; ++i) {
        if (i > l) {
            sum[n][i] = sum[n][i - 1];
        }
        if (L <= m && (s[n][i] < x || (s[n][i] == x && cnt
            -- > 0))) {
            s[n + 1][L++] = s[n][i];

```

```

23         sum[n][i]++;
24     } else {
25         s[n + 1][R++] = s[n][i];
26     }
27 }
28 if (l < m) {
29     B(l, m, n + 1);
30 }
31 if (m + 1 < r) {
32     B(m + 1, r, n + 1);
33 }
34 }
35 int Q(int L, int R, int k, int l, int r, int n = 0) {
36     if (l == r) {
37         return s[n][l];
38     }
39     int m = (l + r) >> 1;
40     int v = 0;
41     if (L > l) {
42         v = sum[n][L - 1];
43     }
44     int t = sum[n][R] - v;
45     if (t >= k) {
46         return Q(l + v, l + sum[n][R] - 1, k, l, m, n + 1);
47     }
48     return Q(m + 1 + L - 1 - v, m + 1 + R - 1 - sum[n][R],
49             k - t, m + 1, r, n + 1);
50 }
51 int main() {
52     int n, m;
53     while (~scanf("%d%d", &n, &m)) {
54         for (int i = 1; i <= n; ++i) {
55             scanf("%d", p + i);
56         }
57         for (int i = 0; i < 20; ++i) {
58             sum[i][0] = 0;
59         }
60         for (int i = 1; i <= n; ++i) {
61             s[0][i] = p[i];
62         }
63         sort(p + 1, p + n + 1);
64         B(1, n);
65         while (m--) {
66             int l, r, k;
67             scanf("%d%d%d", &l, &r, &k);
68             printf("%d\n", Q(l, r, k, 1, n));
69         }
70     }
71     return 0;
72 }

```

2.9 RMQ

```

1 // st 算法更快, 树状数组空间小。
2 // RMQ 初始化 查询 空间
3 // st算法 O(nlogn) O(1) O(nlogn)
4 // 树状数组 O(nlogn) O(logn) O(n)
5 // st 算法
6 int p[MX], d[MX][20];
7 void init(int n) {
8     for (int i = 0; i < n; ++i)
9         d[i][0] = p[i];
10    for (int j = 1; (1 << j) <= n; ++j)
11        for (int i = 0; i + (1 << j) - 1 < n; ++i)
12            d[i][j] = max(d[i][j - 1], d[i + (1 << (j - 1))]
13                        [j - 1]);
14 }
15 int rmq(int x, int y) {
16     if (x > y)
17         return 0;
18     int k = 0;
19     while ((1 << (k + 1)) <= y - x + 1)
20         ++k;
21     return max(d[x][k], d[y - (1 << k) + 1][k]);
22 }
23 // 树状数组
24 int p[MX], d[MX];
25 void init(int n) {
26     for (int i = 1; i < n; ++i) {
27         d[i] = p[i];

```

```

        for (int j = 1; j < (i & -i); j <= 1)
            d[i] = max(d[i], d[i - j]);
    }
}
int rmq(int x, int y) {
    if (x > y)
        return 0;
    int ans = p[y];
    while (1) {
        ans = max(ans, p[y]);
        if (x == y)
            break;
        for (y -= 1; y - x >= (y & -y); y -= (y & -y))
            ans = max(ans, d[y]);
    }
    return ans;
}

```

2.10 Shunting Yard

```

// 调度场算法: 中缀转后缀 (逆波兰)
// 算符优先级
int op_rank(char c) {
    switch (c) {
        case '^':
            return 6;
        case '*': case '/':
            return 5;
        case '+': case '-':
            return 4;
        case '>': case '<': case '=': case '#':
            return 3;
        case '.':
            return 2;
        case '|':
            return 1;
    }
    return 0;
}
void shunting_yard(char *p) {
    stack<char> s;
    queue<char> q;

    int len = strlen(p);
    for (int i = 0; i < len; ++i) {
        if (isdigit(p[i]) || isalpha(p[i])) {
            if (i && (isdigit(p[i - 1]) || isalpha(p[i - 1]))) {
                puts("Syntax Error!");
                return;
            }
            q.push(p[i]);
        } else if (op_rank(p[i])) {
            if (i && op_rank(p[i - 1])) {
                puts("Syntax Error!");
                return;
            }
            while (!s.empty()) {
                if (op_rank(p[i]) <= op_rank(s.top())) {
                    q.push(s.top());
                    s.pop();
                } else break;
            }
            s.push(p[i]);
        } else if (p[i] == '(') {
            s.push(p[i]);
        } else if (p[i] == ')') {
            while (!s.empty() && s.top() != '(') {
                q.push(s.top());
                s.pop();
            }
            if (!s.empty()) {
                s.pop();
            }
        } else {
            puts("Syntax Error!");
            return;
        }
    }
    while (!s.empty()) {
        q.push(s.top());
        s.pop();
    }
    while (!q.empty()) {
        puts(q.front());
        q.pop();
    }
}

```

```

63     if (s.top() == '(') {
64         puts("Syntax Error!");
65         return;
66     }
67     q.push(s.top());
68     s.pop();
69 }
70 while (!q.empty()) {
71     putchar(q.front());
72     q.pop();
73 }
74 puts("");
75 }
76 int main() {
77     char p[1024];
78     while (~scanf("%s", p)) {
79         shunting_yard(p);
80     }
81     return 0;
82 }

```

2.11 Union Find

```

1  int p[MX], q[MX], n;
2  void init() {
3      for (int i = 0; i < MX; ++i) {
4          p[i] = i;
5          q[i] = 1;
6      }
7  }
8  int F(int x) {
9      return x == p[x] ? x : (p[x] = F(p[x]));
10 }
11 // 同组
12 bool equal(int x, int y) {
13     return F(x) == F(y);
14 }
15 void uni(int x, int y) {
16     x = F(x), y = F(y);
17     if (x == y)
18         return;
19     —n;
20     p[x] = y;
21     q[y] += q[x];
22     q[x] = 0;
23 }

```

3 String

3.1 AC

```

1  #include <cstdio>
2  #include <algorithm>
3  #include <iostream>
4  #include <cstring>
5  #include <queue>
6  #include <string>
7  using namespace std;
8
9  struct Trie {
10     int next[500010][26], fail[500010], end[500010];
11     int root, L;
12     int newnode() {
13         for (int i = 0; i < 26; i++) {
14             next[L][i] = -1;
15         }
16         end[L++] = 0;
17         return L - 1;
18     }
19     void init() {
20         L = 0;
21         root = newnode();
22     }
23     void insert(char buf[]) {
24         int len = strlen(buf);
25         int now = root;
26         for (int i = 0; i < len; i++) {
27             if (next[now][buf[i] - 'a'] == -1) {
28                 next[now][buf[i] - 'a'] = newnode();
29             }
30             now = next[now][buf[i] - 'a'];

```

```

        }
        end[now]++;
    }
    void build() {
        queue<int> Q;
        fail[root] = root;
        for (int i = 0; i < 26; i++)
            if (next[root][i] == -1) {
                next[root][i] = root;
            } else {
                fail[next[root][i]] = root;
                Q.push(next[root][i]);
            }
        while (!Q.empty()) {
            int now = Q.front();
            Q.pop();
            for (int i = 0; i < 26; i++) {
                if (next[now][i] == -1) {
                    next[now][i] = next[fail[now]][i];
                } else {
                    fail[next[now][i]] = next[fail[now]][i];
                    Q.push(next[now][i]);
                }
            }
        }
    }
    int query(char buf[]) {
        int len = strlen(buf);
        int now = root;
        int res = 0;
        for (int i = 0; i < len; i++) {
            now = next[now][buf[i] - 'a'];
            int temp = now;
            while (temp != root) {
                res += end[temp];
                end[temp] = 0; //
                temp = fail[temp];
            }
        }
        return res;
    }
};
char buf[1000010];
Trie ac;
int main() {
    int T, m;
    scanf("%d", &T);
    while (T—) {
        scanf("%d", &m);
        ac.init();
        for (int i = 0; i < m; i++) {
            scanf("%s", buf);
            ac.insert(buf);
        }
        ac.build();
        scanf("%s", buf);
        printf("%d\n", ac.query(buf));
    }
    return 0;
}

```

3.2 Hash

```

// Hash (Rabin-Karp, RK), 与二分配合使用。应用广泛。(综合
// 白书和图灵白书学习)
// 题库: http://acm.hust.edu.cn/vjudge/contest/view.action
// ?cid=41757#overview
const ull B = 1000000007ULL; // 哈希基数, 1e8 + 7
const int mx_s_num = 105; // 字符串个数

char s[mx_s_num][mx]; // 注意, 一定要用gets(s[i] + 1), 从
// 下标1开始读
ull ha[mx_s_num][mx], bp[mx] = {1ULL}; // ha[i]从1开始,
// 一直到ha[i][n]
int len[mx_s_num]; // len[i] = strlen(s[i] + 1); 一定要是
// s[i] + 1, 否则n会是0

void init_hash(int s_num) { // 请在main()中完成len的求
// 取。
    int i, j;
    For(i, s_num) Forr(j, 1, len[i] + 1) ha[i][j] = ha[i][
// j - 1] * B + s[i][j];
}

```

```

13     int n = Max(len, s_num); // 调用#define的Max()
14     Forr(i, 1, n + 1) bp[i] = bp[i - 1] * B;
15 }
16
17 ull get_hash(char *s) { // 直接返回整个字符串的hash
18     ull ha = OULL;
19     for (int i = 0; s[i]; ++i)
20         ha = ha * B + s[i];
21     return ha;
22 }
23
24 ull get_hash(int *a, int n) { // 返回整个int数组的hash值
25     int i;
26     ull ha = OULL;
27     For(i, n) ha = ha * B + (ull)a[i];
28     return ha;
29 }
30
31 // 注意pos一定不能是0!!!
32 inline ull get_hash(ull *Ha, int pos,
33                     int l) { // 返回Ha[pos...pos+l-1]的
                               // 值, pos与l必须是正数
34     return Ha[pos + l - 1] - Ha[pos - 1] * bp[l];
35 }
36
37 inline ull merge_hash(ull ha1, ull ha2, int len2) { // 返
38     // 回s1+s2拼接后的hash值
39     return ha1 * bp[len2] + ha2;
40 }
41
42 bool contain(int ida, int idb) { // b是否为a的子串
43     // , ida和idb为字符串下
44     // 标, 若只有两个字符串, 使
45     // 用时传入参数(0,
46     // 1)、(1, 0)就行
47
48     if (len[ida] < len[idb])
49         return false;
50     ull hab = ha[idb][len[idb]];
51     for (int i = 1; i + len[idb] <= len[ida]; ++i)
52         if (get_hash(ha[ida], i, len[idb]) == hab)
53             return true;
54     return false;
55 }
56
57 int overlap(
58     int ida,
59     int idb) { // 求a后缀与b前缀的最长公共子串, ida和idb
60     // 为字符串下标, 若只有两个字符串, 使用时传入参数(0,1)、
61     // (1,
62     // 0)就行
63     int ans = 0, i;
64     Forr(i, 1, min(len[ida], len[idb]) +
65         1) if (get_hash(ha[ida], len[ida] - i +
66             1, i) ==
67             get_hash(ha[idb], 1, i)) ans = i;
68
69     // 可在if中加上 && strncmp(s[ida] + len[ida] - i + 1,
70     // s[idb] + 1, i) ==
71     // 0(不过这就失去意义了, 还不如双hash)
72     return ans;
73 }
74
75 #include <cstdio>
76 #include <cstring>
77 #include <algorithm>
78 #define MAXL 222
79 typedef unsigned long long ull;
80 using namespace std;
81 const ull B = 1000000007ULL;
82 char s[MAXL];
83 int len;
84 ull ha[MAXL], bp[MAXL] = {1ULL};
85 void hash() {
86     ha[0] = s[0];
87     for (int i = 1; i < len; ++i)
88         ha[i] = ha[i - 1] * B + s[i], bp[i] = bp[i - 1] *
89             B;
90 }
91
92 ull get(int pos, int l) {
93     return ha[pos + l - 1] - ha[pos - 1] * bp[l];
94 }
95
96 bool check(int p, int l) {
97     return get(p, min(l, len - p - 1)) == get(p + 1, min(l

```

```

86     , len - p - 1));
87 }
88 int main() {
89     int k;
90     scanf("%s", s, &k);
91     len = strlen(s);
92     if (k >= len) {
93         printf("%d\n", 2 * ((k + len) >> 1));
94         return 0;
95     }
96     hash();
97     for (int i = (len + k) >> 1; i >= k; --i) {
98         for (int j = 0; j <= len + k - 2 * i; ++j) {
99             if (check(j, i)) {
100                 printf("%d\n", 2 * i);
101                 return 0;
102             }
103         }
104     }
105 }

```

3.3 KMP

```

1 int m, n, p[10010];
2 char a[1000100], b[10010];
3 void init() {
4     p[0] = -1;
5     int i = 0, j = -1;
6     while (i < n) {
7         if (j == -1 || b[i] == b[j]) {
8             ++i, ++j, p[i] = j;
9         } else {
10             j = p[j];
11         }
12     }
13 }
14 int kmp() {
15     int i = 0, j = 0, ans = 0;
16     while (i < m) {
17         if (j == -1 || b[j] == a[i]) {
18             ++i, ++j;
19         } else {
20             j = p[j];
21         }
22         if (j == n) {
23             ++ans;
24         }
25     }
26     return ans;
27 }

```

3.4 Manacher

```

1 char s[MX];
2 int l[MX];
3
4 // len[i] 是 i/2 为中心的最大回文长度
5 void palindrome(char cs[], int len[], int n) {
6     for (int i = 0; i < n * 2; ++i) {
7         len[i] = 0;
8     }
9     for (int i = 0, j = 0, k; i < n * 2; i += k, j = max(j
10         - k, 0)) {
11         while (i - j >= 0 && i + j + 1 < n * 2 &&
12             cs[(i - j) / 2] == cs[(i + j + 1) / 2]) {
13             ++j;
14         }
15         len[i] = j;
16         for (k = 1; i - k >= 0 && j - k >= 0 && len[i - k]
17             != j - k; k++) {
18             len[i + k] = min(len[i - k], j - k);
19         }
20     }
21 }
22 int main() {
23     while (~scanf("%s", s)) {
24         int ans = 0, sl = strlen(s);
25         palindrome(s, l, sl);
26         for (int i = 0; i < sl * 2; ++i) {
27             ans = max(ans, l[i]);
28         }
29     }
30 }

```

```

28     printf("%d\n", ans);
29 }
30 return 0;
31 }

```

3.5 Trie

```

1  #define MAXW 100010
2  #define MAXL 12
3  const int MAXN = MAXW * MAXL;
4
5  struct node {
6      int next[26];
7      int cnt; // 附加信息
8  } t[MAXN];
9
10 int ts;
11
12 void clear() {
13     ts = 0;
14     memset(t, 0, sizeof(t));
15 }
16
17 int insert(char s[]) {
18     int len = strlen(s), p = 0;
19     for (int i = 0; i < len; ++i) {
20         if (!t[p].next[s[i] - 'a']) {
21             t[p].next[s[i] - 'a'] = ++ts;
22         }
23         ++t[p].cnt;
24         p = t[p].next[s[i] - 'a'];
25     }
26     return ++t[p].cnt;
27 }
28
29 int query(char s[]) {
30     int len = strlen(s), p = 0;
31     for (int i = 0; i < len; ++i) {
32         if (!t[p].next[s[i] - 'a']) {
33             return 0;
34         }
35         p = t[p].next[s[i] - 'a'];
36     }
37     return t[p].cnt;
38 }
39
40 int main() {
41     char s[MAXL];
42     int n;
43     scanf("%d", &n);
44     while (n--) {
45         scanf("%s", s);
46         insert(s);
47     }
48     scanf("%d", &n);
49     while (n--) {
50         scanf("%s", s);
51         printf("%d\n", query(s));
52     }
53     return 0;
54 }

```

4 Geometry

4.1 include

```

1  #include <vector>
2  #include <list>
3  #include <map>
4  #include <set>
5  #include <deque>
6  #include <queue>
7  #include <stack>
8  #include <bitset>
9  #include <algorithm>
10 #include <functional>
11 #include <numeric>
12 #include <utility>
13 #include <iostream>
14 #include <sstream>
15 #include <iomanip>

```

```

#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <cctype>
#include <string>
#include <cstring>
#include <stdio>
#include <cmath>
#include <stdlib>
#include <ctime>
#include <limits>
#include <complex>
#define mp make_pair
#define pb push_back
using namespace std;
const double eps = 1e-8;
const double pi = acos(-1.0);
const double inf = 1e20;
const int maxp = 1111;
int dblcmp(double d) {
    if (fabs(d) < eps)
        return 0;
    return d > eps ? 1 : -1;
}
inline double sqr(double x) {
    return x * x;
}

```

4.2 point

```

struct point {
    double x, y;
    point() {}
    point(double _x, double _y) : x(_x), y(_y){};
    void input() {
        scanf("%lf%lf", &x, &y);
    }
    void output() {
        printf("%.2f %.2f\n", x, y);
    }
    bool operator==(point a) const {
        return dblcmp(a.x - x) == 0 && dblcmp(a.y - y) == 0;
    }
    bool operator<(point a) const {
        return dblcmp(a.x - x) == 0 ? dblcmp(y - a.y) < 0 : x < a.x;
    }
    double len() {
        return hypot(x, y);
    }
    double len2() {
        return x * x + y * y;
    }
    double distance(point p) {
        return hypot(x - p.x, y - p.y);
    }
    point add(point p) {
        return point(x + p.x, y + p.y);
    }
    point sub(point p) {
        return point(x - p.x, y - p.y);
    }
    point mul(double b) {
        return point(x * b, y * b);
    }
    point div(double b) {
        return point(x / b, y / b);
    }
    double dot(point p) //点积
    {
        return x * p.x + y * p.y;
    }
    double det(point p) //叉积
    {
        return x * p.y - y * p.x;
    }
    double rad(point a, point b) {
        point p = *this;
        return fabs(
            atan2(fabs(a.sub(p).det(b.sub(p))), a.sub(p).
                dot(b.sub(p)))));
    }
}

```

```

51     }
52     point trunc(double r) {
53         double l = len();
54         if (!dblcmp(l))
55             return *this;
56         r /= l;
57         return point(x * r, y * r);
58     }
59     point rotleft() {
60         return point(-y, x);
61     }
62     point rotright() {
63         return point(y, -x);
64     }
65     point rotate(point p, double angle) //绕点p逆时针旋转
        angle角度
66     {
67         point v = this->sub(p);
68         double c = cos(angle), s = sin(angle);
69         return point(p.x + v.x * c - v.y * s, p.y + v.x *
70             s + v.y * c);
71     };

```

4.3 line

```

1 struct line {
2     point a, b;
3     line() {
4     }
5     line(point _a, point _b) {
6         a = _a;
7         b = _b;
8     }
9     bool operator==(line v) {
10         return (a == v.a) && (b == v.b);
11     }
12     //倾斜角angle
13     line(point p, double angle) {
14         a = p;
15         if (dblcmp(angle - pi / 2) == 0) {
16             b = a.add(point(0, 1));
17         } else {
18             b = a.add(point(1, tan(angle)));
19         }
20     }
21     // ax+by+c=0
22     line(double _a, double _b, double _c) {
23         if (dblcmp(_a) == 0) {
24             a = point(0, -_c / _b);
25             b = point(1, -_c / _b);
26         } else if (dblcmp(_b) == 0) {
27             a = point(-_c / _a, 0);
28             b = point(-_c / _a, 1);
29         } else {
30             a = point(0, -_c / _b);
31             b = point(1, (-_c - _a) / _b);
32         }
33     }
34     void input() {
35         a.input();
36         b.input();
37     }
38     void adjust() {
39         if (b < a)
40             swap(a, b);
41     }
42     double length() {
43         return a.distance(b);
44     }
45     double angle() //直线倾斜角 0<=angle<180
46     {
47         double k = atan2(b.y - a.y, b.x - a.x);
48         if (dblcmp(k) < 0)
49             k += pi;
50         if (dblcmp(k - pi) == 0)
51             k -= pi;
52         return k;
53     }
54     //点和线段关系
55     // 1 在逆时针
56     // 2 在顺时针
57     // 3 平行

```

```

int relation(point p) {
    int c = dblcmp(p.sub(a).det(b.sub(a)));
    if (c < 0)
        return 1;
    if (c > 0)
        return 2;
    return 3;
}
bool pointonseg(point p) {
    return dblcmp(p.sub(a).det(b.sub(a))) == 0 &&
        dblcmp(p.sub(a).dot(p.sub(b))) <= 0;
}
bool parallel(line v) {
    return dblcmp(b.sub(a).det(v.b.sub(v.a))) == 0;
}
// 2 规范相交
// 1 非规范相交
// 0 不相交
int segcrossseg(line v) {
    int d1 = dblcmp(b.sub(a).det(v.a.sub(a)));
    int d2 = dblcmp(b.sub(a).det(v.b.sub(a)));
    int d3 = dblcmp(v.b.sub(v.a).det(a.sub(v.a)));
    int d4 = dblcmp(v.b.sub(v.a).det(b.sub(v.a)));
    if ((d1 ^ d2) == -2 && (d3 ^ d4) == -2)
        return 2;
    return (d1 == 0 && dblcmp(v.a.sub(a).dot(v.a.sub(b
        ))) <= 0 ||
        d2 == 0 && dblcmp(v.b.sub(a).dot(v.b.sub(b
        ))) <= 0 ||
        d3 == 0 && dblcmp(a.sub(v.a).dot(a.sub(v.b
        ))) <= 0 ||
        d4 == 0 && dblcmp(b.sub(v.a).dot(b.sub(v.b
        ))) <= 0);
}
int linecrossseg(line v) /*this seg v line
{
    int d1 = dblcmp(b.sub(a).det(v.a.sub(a)));
    int d2 = dblcmp(b.sub(a).det(v.b.sub(a)));
    if ((d1 ^ d2) == -2)
        return 2;
    return (d1 == 0 || d2 == 0);
}
// 0 平行
// 1 重合
// 2 相交
int linecrossline(line v) {
    if ((*this).parallel(v)) {
        return v.relation(a) == 3;
    }
    return 2;
}
point crosspoint(line v) {
    double a1 = v.b.sub(v.a).det(a.sub(v.a));
    double a2 = v.b.sub(v.a).det(b.sub(v.a));
    return point((a.x * a2 - b.x * a1) / (a2 - a1),
        (a.y * a2 - b.y * a1) / (a2 - a1));
}
double dispointtoline(point p) {
    return fabs(p.sub(a).det(b.sub(a))) / length();
}
double dispointtoseg(point p) {
    if (dblcmp(p.sub(b).dot(a.sub(b))) < 0 ||
        dblcmp(p.sub(a).dot(b.sub(a))) < 0) {
        return min(p.distance(a), p.distance(b));
    }
    return dispointtoline(p);
}
point lineprog(point p) {
    return a.add(b.sub(a).mul(b.sub(a).dot(p.sub(a)) /
        b.sub(a).len2()));
}
point symmetrypoint(point p) {
    point q = lineprog(p);
    return point(2 * q.x - p.x, 2 * q.y - p.y);
}
};

```

4.4 circle

```

1 struct circle {
2     point p;
3     double r;
4     circle() {

```



```

5   }
6   circle(point _p, double _r) : p(_p), r(_r){};
7   circle(double x, double y, double _r) : p(point(x, y))
8   , r(_r){};
9   circle(point a, point b, point c) //三角形的外接圆
10  {
11      p = line(a.add(b).div(2), a.add(b).div(2).add(b.
12      sub(a).rotleft()))
13      .crosspoint(line(c.add(b).div(2),
14      c.add(b).div(2).add(b.sub
15      (c).rotleft())));
16      r = p.distance(a);
17  }
18  circle(point a, point b, point c, bool t) //三角形的
19  内切圆
20  {
21      line u, v;
22      double m = atan2(b.y - a.y, b.x - a.x), n = atan2(
23      c.y - a.y, c.x - a.x);
24      u.a = a;
25      u.b = u.a.add(point(cos((n + m) / 2), sin((n + m)
26      / 2)));
27      v.a = b;
28      m = atan2(a.y - b.y, a.x - b.x), n = atan2(c.y - b
29      .y, c.x - b.x);
30      v.b = v.a.add(point(cos((n + m) / 2), sin((n + m)
31      / 2)));
32      p = u.crosspoint(v);
33      r = line(a, b).dispointtoseg(p);
34  }
35  void input() {
36      p.input();
37      scanf("%lf", &r);
38  }
39  void output() {
40      printf("%.2lf %.2lf %.2lf\n", p.x, p.y, r);
41  }
42  bool operator==(circle v) {
43      return ((p == v.p) && dblcmp(r - v.r) == 0);
44  }
45  bool operator<(circle v) const {
46      return ((p < v.p) || (p == v.p) && dblcmp(r - v.r)
47      < 0);
48  }
49  double area() {
50      return pi * sqr(r);
51  }
52  double circumference() {
53      return 2 * pi * r;
54  }
55  // 0 圆外
56  // 1 圆上
57  // 2 圆内
58  int relation(point b) {
59      double dst = b.distance(p);
60      if (dblcmp(dst - r) < 0)
61          return 2;
62      if (dblcmp(dst - r) == 0)
63          return 1;
64      return 0;
65  }
66  int relationseg(line v) {
67      double dst = v.dispointtoseg(p);
68      if (dblcmp(dst - r) < 0)
69          return 2;
70      if (dblcmp(dst - r) == 0)
71          return 1;
72      return 0;
73  }
74  int relationline(line v) {
75      double dst = v.dispointtoline(p);
76      if (dblcmp(dst - r) < 0)
77          return 2;
78      if (dblcmp(dst - r) == 0)
79          return 1;
80      return 0;
81  }
82  //过a b两点 半径r的两个圆
83  int getcircle(point a, point b, double r, circle &c1,
84  circle &c2) {
85      circle x(a, r), y(b, r);
86      int t = x.pointcrosscircle(y, c1.p, c2.p);
87      if (!t)
88          return 0;
89      c1.r = c2.r = r;
90      return t;
91  }

```

```

92      return 0;
93      c1.r = c2.r = r;
94      return t;
95  }
96  //与直线u相切 过点q 半径r1的圆
97  int getcircle(line u, point q, double r1, circle &c1,
98  circle &c2) {
99      double dis = u.dispointtoline(q);
100     if (dblcmp(dis - r1 * 2) > 0)
101         return 0;
102     if (dblcmp(dis) == 0) {
103         c1.p = q.add(u.b.sub(u.a).rotleft().trunc(r1))
104         ;
105         c2.p = q.add(u.b.sub(u.a).rotright().trunc(r1)
106         );
107         c1.r = c2.r = r1;
108         return 2;
109     }
110     line u1 = line(u.a.add(u.b.sub(u.a).rotleft().
111     trunc(r1)),
112     u.b.add(u.b.sub(u.a).rotleft().
113     trunc(r1)));
114     line u2 = line(u.a.add(u.b.sub(u.a).rotright().
115     trunc(r1)),
116     u.b.add(u.b.sub(u.a).rotright().
117     trunc(r1)));
118     circle cc = circle(q, r1);
119     point p1, p2;
120     if (!cc.pointcrossline(u1, p1, p2))
121         cc.pointcrossline(u2, p1, p2);
122     c1 = circle(p1, r1);
123     if (p1 == p2) {
124         c2 = c1;
125         return 1;
126     }
127     c2 = circle(p2, r1);
128     return 2;
129 }
130 //同时与直线u,v相切 半径r1的圆
131 int getcircle(line u, line v, double r1, circle &c1,
132 circle &c2, circle &c3,
133 circle &c4) {
134     if (u.parallel(v))
135         return 0;
136     line u1 = line(u.a.add(u.b.sub(u.a).rotleft().
137     trunc(r1)),
138     u.b.add(u.b.sub(u.a).rotleft().
139     trunc(r1)));
140     line u2 = line(u.a.add(u.b.sub(u.a).rotright().
141     trunc(r1)),
142     u.b.add(u.b.sub(u.a).rotright().
143     trunc(r1)));
144     line v1 = line(v.a.add(v.b.sub(v.a).rotleft().
145     trunc(r1)),
146     v.b.add(v.b.sub(v.a).rotleft().
147     trunc(r1)));
148     line v2 = line(v.a.add(v.b.sub(v.a).rotright().
149     trunc(r1)),
150     v.b.add(v.b.sub(v.a).rotright().
151     trunc(r1)));
152     c1.r = c2.r = c3.r = c4.r = r1;
153     c1.p = u1.crosspoint(v1);
154     c2.p = u1.crosspoint(v2);
155     c3.p = u2.crosspoint(v1);
156     c4.p = u2.crosspoint(v2);
157     return 4;
158 }
159 //同时与不相交圆cx,cy相切 半径为r1的圆
160 int getcircle(circle cx, circle cy, double r1, circle
161 &c1, circle &c2) {
162     circle x(cx.p, r1 + cx.r), y(cy.p, r1 + cy.r);
163     int t = x.pointcrosscircle(y, c1.p, c2.p);
164     if (!t)
165         return 0;
166     c1.r = c2.r = r1;
167     return t;
168 }
169 int pointcrossline(line v, point &p1,
170 point &p2) //求与线段交要先判断
171 relationseg
172 {
173     if (!(*this).relationline(v))
174         return 0;
175 }

```

```

143 point a = v.lineprog(p);
144 double d = v.dispointtoline(p);
145 d = sqrt(r * r - d * d);
146 if (dblcmp(d) == 0) {
147     p1 = a;
148     p2 = a;
149     return 1;
150 }
151 p1 = a.sub(v.b.sub(v.a).trunc(d));
152 p2 = a.add(v.b.sub(v.a).trunc(d));
153 return 2;
154 }
155 // 5 相离
156 // 4 外切
157 // 3 相交
158 // 2 内切
159 // 1 内含
160 int relationcircle(circle v) {
161     double d = p.distance(v.p);
162     if (dblcmp(d - r - v.r) > 0)
163         return 5;
164     if (dblcmp(d - r - v.r) == 0)
165         return 4;
166     double l = fabs(r - v.r);
167     if (dblcmp(d - r - v.r) < 0 && dblcmp(d - l) > 0)
168         return 3;
169     if (dblcmp(d - l) == 0)
170         return 2;
171     if (dblcmp(d - l) < 0)
172         return 1;
173 }
174 int pointcrosscircle(circle v, point &p1, point &p2) {
175     int rel = relationcircle(v);
176     if (rel == 1 || rel == 5)
177         return 0;
178     double d = p.distance(v.p);
179     double l = (d + (sqr(r) - sqr(v.r)) / d) / 2;
180     double h = sqrt(sqr(r) - sqr(l));
181     p1 = p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).
182         rotleft().trunc(h)));
183     p2 = p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).
184         rotright().trunc(h)));
185     if (rel == 2 || rel == 4) {
186         return 1;
187     }
188     return 2;
189 }
190 //过一点做圆的切线 (先判断点和圆关系)
191 int tangentline(point q, line &u, line &v) {
192     int x = relation(q);
193     if (x == 2)
194         return 0;
195     if (x == 1) {
196         u = line(q, q.add(q.sub(p).rotleft()));
197         v = u;
198         return 1;
199     }
200     double d = p.distance(q);
201     double l = sqr(r) / d;
202     double h = sqrt(sqr(r) - sqr(l));
203     u = line(q, p.add(q.sub(p).trunc(l).add(q.sub(p).
204         rotleft().trunc(h))));
205     v = line(q, p.add(q.sub(p).trunc(l).add(q.sub(p).
206         rotright().trunc(h))));
207     return 2;
208 }
209 double areacircle(circle v) {
210     int rel = relationcircle(v);
211     if (rel >= 4)
212         return 0.0;
213     if (rel <= 2)
214         return min(area(), v.area());
215     double d = p.distance(v.p);
216     double hf = (r + v.r + d) / 2.0;
217     double ss = 2 * sqrt(hf * (hf - r) * (hf - v.r) *
218         (hf - d));
219     double a1 = acos((r * r + d * d - v.r * v.r) /
220         (2.0 * r * d));
221     a1 = a1 * r * r;
222     double a2 = acos((v.r * v.r + d * d - r * r) /
223         (2.0 * v.r * d));
224     a2 = a2 * v.r * v.r;
225     return a1 + a2 - ss;

```

```

219 }
220 double areatriangle(point a, point b) {
221     if (dblcmp(p.sub(a).det(p.sub(b)) == 0))
222         return 0.0;
223     point q[5];
224     int len = 0;
225     q[len++] = a;
226     line l(a, b);
227     point p1, p2;
228     if (pointcrossline(l, q[1], q[2]) == 2) {
229         if (dblcmp(a.sub(q[1]).dot(b.sub(q[1]))) < 0)
230             q[len++] = q[1];
231         if (dblcmp(a.sub(q[2]).dot(b.sub(q[2]))) < 0)
232             q[len++] = q[2];
233     }
234     q[len++] = b;
235     if (len == 4 && (dblcmp(q[0].sub(q[1]).dot(q[2].
236         sub(q[1]))) > 0))
237         swap(q[1], q[2]);
238     double res = 0;
239     int i;
240     for (i = 0; i < len - 1; i++) {
241         if (relation(q[i]) == 0 || relation(q[i + 1])
242             == 0) {
243             double arg = p.rad(q[i], q[i + 1]);
244             res += r * r * arg / 2.0;
245         } else {
246             res += fabs(q[i].sub(p).det(q[i + 1].sub(p)
247                 )) / 2.0;
248         }
249     }
250     return res;
251 }

```

4.5 polygon

```

1 struct polygon {
2     int n;
3     point p[maxp];
4     line l[maxp];
5     void input() {
6         n = 4;
7         for (int i = 0; i < n; i++) {
8             p[i].input();
9         }
10    }
11    void add(point q) {
12        p[n++] = q;
13    }
14    void getline() {
15        for (int i = 0; i < n; i++) {
16            l[i] = line(p[i], p[(i + 1) % n]);
17        }
18    }
19    struct cmp {
20        point p;
21        cmp(const point &p0) {
22            p = p0;
23        }
24        bool operator()(const point &aa, const point &bb)
25        {
26            point a = aa, b = bb;
27            int d = dblcmp(a.sub(p).det(b.sub(p)));
28            if (d == 0) {
29                return dblcmp(a.distance(p) - b.distance(p)
30                    ) < 0;
31            }
32            return d > 0;
33        }
34    };
35    void norm() {
36        point mi = p[0];
37        for (int i = 1; i < n; i++)
38            mi = min(mi, p[i]);
39        sort(p, p + n, cmp(mi));
40    }
41    void getconvex(polygon &convex) {
42        int i, j, k;
43        sort(p, p + n);
44        convex.n = n;
45        for (i = 0; i < min(n, 2); i++) {
46            convex.p[i] = p[i];

```

```

45     }
46     if (n <= 2)
47         return;
48     int &top = convex.n;
49     top = 1;
50     for (i = 2; i < n; i++) {
51         while (top &&
52             convex.p[top].sub(p[i]).det(convex.p[
53                 top - 1].sub(p[i])) <=
54                 0)
55             top--;
56         convex.p[++top] = p[i];
57     }
58     int temp = top;
59     convex.p[++top] = p[n - 2];
60     for (i = n - 3; i >= 0; i--) {
61         while (top != temp &&
62             convex.p[top].sub(p[i]).det(convex.p[
63                 top - 1].sub(p[i])) <=
64                 0)
65             top--;
66         convex.p[++top] = p[i];
67     }
68     bool isconvex() {
69         bool s[3];
70         memset(s, 0, sizeof(s));
71         int i, j, k;
72         for (i = 0; i < n; i++) {
73             j = (i + 1) % n;
74             k = (j + 1) % n;
75             s[dblcmp(p[j].sub(p[i]).det(p[k].sub(p[i]))) +
76                 1] = 1;
77             if (s[0] && s[2])
78                 return 0;
79         }
80         return 1;
81     }
82     // 3 点上
83     // 2 边上
84     // 1 内部
85     // 0 外部
86     int relationpoint(point q) {
87         int i, j;
88         for (i = 0; i < n; i++) {
89             if (p[i] == q)
90                 return 3;
91         }
92         getline();
93         for (i = 0; i < n; i++) {
94             if (l[i].pointonseg(q))
95                 return 2;
96         }
97         int cnt = 0;
98         for (i = 0; i < n; i++) {
99             j = (i + 1) % n;
100             int k = dblcmp(q.sub(p[j]).det(p[i].sub(p[j])))
101             );
102             int u = dblcmp(p[i].y - q.y);
103             int v = dblcmp(p[j].y - q.y);
104             if (k > 0 && u < 0 && v >= 0)
105                 cnt++;
106             if (k < 0 && v < 0 && u >= 0)
107                 cnt--;
108         }
109         return cnt != 0;
110     }
111     // 1 在多边形内长度为正
112     // 2 相交或与边平行
113     // 0 无任何交点
114     int relationline(line u) {
115         int i, j, k = 0;
116         getline();
117         for (i = 0; i < n; i++) {
118             if (l[i].segcrossseg(u) == 2)
119                 return 1;
120             if (l[i].segcrossseg(u) == 1)
121                 k = 1;
122         }
123         if (!k)
124             return 0;
125         vector<point> vp;
126         for (i = 0; i < n; i++) {
127             if (l[i].segcrossseg(u)) {
128                 if (l[i].parallel(u)) {
129                     vp.pb(u.a);
130                     vp.pb(u.b);
131                     vp.pb(l[i].a);
132                     vp.pb(l[i].b);
133                     continue;
134                 }
135                 vp.pb(l[i].crosspoint(u));
136             }
137         }
138         sort(vp.begin(), vp.end());
139         int sz = vp.size();
140         for (i = 0; i < sz - 1; i++) {
141             point mid = vp[i].add(vp[i + 1]).div(2);
142             if (relationpoint(mid) == 1)
143                 return 1;
144         }
145         return 2;
146     }
147     // 直线u切割凸多边形左侧
148     // 注意直线方向
149     void convexcute(line u, polygon &po) {
150         int i, j, k;
151         int &top = po.n;
152         top = 0;
153         for (i = 0; i < n; i++) {
154             int d1 = dblcmp(p[i].sub(u.a).det(u.b.sub(u.a)
155                 ));
156             int d2 = dblcmp(p[(i + 1) % n].sub(u.a).det(u.
157                 b.sub(u.a)));
158             if (d1 >= 0)
159                 po.p[top++] = p[i];
160             if (d1 * d2 < 0)
161                 po.p[top++] = u.crosspoint(line(p[i], p[(i
162                     + 1) % n]));
163         }
164     }
165     double getcircumference() {
166         double sum = 0;
167         int i;
168         for (i = 0; i < n; i++) {
169             sum += p[i].distance(p[(i + 1) % n]);
170         }
171         return sum;
172     }
173     double getarea() {
174         double sum = 0;
175         int i;
176         for (i = 0; i < n; i++) {
177             sum += p[i].det(p[(i + 1) % n]);
178         }
179         return fabs(sum) / 2;
180     }
181     bool getdir() // 1代表逆时针 0代表顺时针
182     {
183         double sum = 0;
184         int i;
185         for (i = 0; i < n; i++) {
186             sum += p[i].det(p[(i + 1) % n]);
187         }
188         if (dblcmp(sum) > 0)
189             return 1;
190         return 0;
191     }
192     point getbarycentre() {
193         point ret(0, 0);
194         double area = 0;
195         int i;
196         for (i = 1; i < n - 1; i++) {
197             double tmp = p[i].sub(p[0]).det(p[i + 1].sub(p
198                 [0]));
199             if (dblcmp(tmp) == 0)
200                 continue;
201             area += tmp;
202             ret.x += (p[0].x + p[i].x + p[i + 1].x) / 3 *
203             tmp;
204             ret.y += (p[0].y + p[i].y + p[i + 1].y) / 3 *
205             tmp;
206         }
207         if (dblcmp(area))
208             ret = ret.div(area);
209         return ret;
210     }

```

```

201 }
202 double areaintersection(polygon po) {
203 }
204 double areaunion(polygon po) {
205     return getarea() + po.getarea() - areaintersection
206     (po);
207 }
208 double areacircle(circle c) {
209     int i, j, k, l, m;
210     double ans = 0;
211     for (i = 0; i < n; i++) {
212         int j = (i + 1) % n;
213         if (dblcmp(p[j].sub(c.p).det(p[i].sub(c.p)))
214             >= 0) {
215             ans += c.areatriangle(p[i], p[j]);
216         } else {
217             ans -= c.areatriangle(p[i], p[j]);
218         }
219     }
220     return fabs(ans);
221 }
222 //多边形和圆关系
223 // 0 一部分在圆外
224 // 1 与圆某条边相切
225 // 2 完全在圆内
226 int relationcircle(circle c) {
227     getline();
228     int i, x = 2;
229     if (relationpoint(c.p) != 1)
230         return 0;
231     for (i = 0; i < n; i++) {
232         if (c.relationseg(l[i]) == 2)
233             return 0;
234         if (c.relationseg(l[i]) == 1)
235             x = 1;
236     }
237     return x;
238 }
239 void find(int st, point tri[], circle &c) {
240     if (!st) {
241         c = circle(point(0, 0), -2);
242     }
243     if (st == 1) {
244         c = circle(tri[0], 0);
245     }
246     if (st == 2) {
247         c = circle(tri[0].add(tri[1]).div(2),
248             tri[0].distance(tri[1]) / 2.0);
249     }
250     if (st == 3) {
251         c = circle(tri[0], tri[1], tri[2]);
252     }
253 }
254 void solve(int cur, int st, point tri[], circle &c) {
255     find(st, tri, c);
256     if (st == 3)
257         return;
258     int i;
259     for (i = 0; i < cur; i++) {
260         if (dblcmp(p[i].distance(c.p) - c.r) > 0) {
261             tri[st] = p[i];
262             solve(i, st + 1, tri, c);
263         }
264     }
265 }
266 circle mincircle() //点集最小圆覆盖
267 {
268     random_shuffle(p, p + n);
269     point tri[4];
270     circle c;
271     solve(n, 0, tri, c);
272     return c;
273 }
274 int circlecover(double r) //单位圆覆盖
275 {
276     int ans = 0, i, j;
277     vector<pair<double, int>> v;
278     for (i = 0; i < n; i++) {
279         v.clear();
280         for (j = 0; j < n; j++)
281             if (i != j) {
282                 point q = p[i].sub(p[j]);
283                 double d = q.len();

```

```

284                 if (dblcmp(d - 2 * r) <= 0) {
285                     double arg = atan2(q.y, q.x);
286                     if (dblcmp(arg) < 0)
287                         arg += 2 * pi;
288                     double t = acos(d / (2 * r));
289                     v.push_back(make_pair(arg - t + 2
290                         * pi, -1));
291                     v.push_back(make_pair(arg + t + 2
292                         * pi, 1));
293                 }
294             }
295     sort(v.begin(), v.end());
296     int cur = 0;
297     for (j = 0; j < v.size(); j++) {
298         if (v[j].second == -1)
299             ++cur;
300         else
301             --cur;
302         ans = max(ans, cur);
303     }
304     return ans + 1;
305 }
306 int pointinpolygon(point q) //点在凸多边形内部的判定
307 {
308     if (getdir())
309         reverse(p, p + n);
310     if (dblcmp(q.sub(p[0]).det(p[n - 1].sub(p[0]))) ==
311         0) {
312         if (line(p[n - 1], p[0]).pointonseg(q))
313             return n - 1;
314         return -1;
315     }
316     int low = 1, high = n - 2, mid;
317     while (low <= high) {
318         mid = (low + high) >> 1;
319         if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0])))
320             >= 0 &&
321             dblcmp(q.sub(p[0]).det(p[mid + 1].sub(p
322                 [0]))) < 0) {
323             polygon c;
324             c.p[0] = p[mid];
325             c.p[1] = p[mid + 1];
326             c.p[2] = p[0];
327             c.n = 3;
328             if (c.relationpoint(q))
329                 return mid;
330             return -1;
331         }
332         if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0])))
333             > 0) {
334             low = mid + 1;
335         } else {
336             high = mid - 1;
337         }
338     }
339     return -1;
340 }
341 };

```

4.6 polygons

```

1 struct polygons {
2     vector<polygon> p;
3     polygons() {
4         p.clear();
5     }
6     void clear() {
7         p.clear();
8     }
9     void push(polygon q) {
10         if (dblcmp(q.getarea()))
11             p.pb(q);
12     }
13     vector<pair<double, int>> > e;
14     void ins(point s, point t, point X, int i) {
15         double r = fabs(t.x - s.x) > eps ? (X.x - s.x) / (
16             t.x - s.x)
17             : (X.y - s.y) / (
18                 t.y - s.y);
19         r = min(r, 1.0);
20         r = max(r, 0.0);
21         e.pb(mp(r, i));

```

```

20 }
21 double polyareaunion() {
22     double ans = 0.0;
23     int c0, c1, c2, i, j, k, w;
24     for (i = 0; i < p.size(); i++) {
25         if (p[i].getdir() == 0)
26             reverse(p[i].p, p[i].p + p[i].n);
27     }
28     for (i = 0; i < p.size(); i++) {
29         for (k = 0; k < p[i].n; k++) {
30             point &s = p[i].p[k], &t = p[i].p[(k + 1)
31             % p[i].n];
32             if (!dblcmp(s.det(t)))
33                 continue;
34             e.clear();
35             e.pb(mp(0.0, 1));
36             e.pb(mp(1.0, -1));
37             for (j = 0; j < p.size(); j++)
38                 if (i != j) {
39                     for (w = 0; w < p[j].n; w++) {
40                         point a = p[j].p[w], b = p[j].
41                         p[(w + 1) % p[j].n],
42                         c = p[j].p[(w - 1 + p[j]
43                         ].n) % p[j].n];
44                         c0 = dblcmp(t.sub(s).det(c.sub
45                         (s)));
46                         c1 = dblcmp(t.sub(s).det(a.sub
47                         (s)));
48                         c2 = dblcmp(t.sub(s).det(b.sub
49                         (s)));
50                         if (c1 * c2 < 0)
51                             ins(s, t, line(s, t).
52                             crosspoint(line(a, b)),
53                             -c2);
54                         else if (!c1 && c0 * c2 < 0)
55                             ins(s, t, a, -c2);
56                         else if (!c1 && !c2) {
57                             int c3 = dblcmp(t.sub(s).
58                             det(
59                             p[j].p[(w + 2) % p[j].
60                             n].sub(s)));
61                             int dp = dblcmp(t.sub(s).
62                             dot(b.sub(a)));
63                             if (dp && c0)
64                                 ins(s, t, a, dp > 0
65                                 ? c0
66                                 * ((j
67                                 > i)
68                                 ^ (
69                                 c0 <
70                                 0))
71                                 : -(
72                                 c0 <
73                                 0));
74                             if (dp && c3)
75                                 ins(s, t, b,
76                                 dp > 0 ? -c3 * ((j
77                                 > i) ^ (c3 < 0))
78                                 : c3 < 0);
79                         }
80                     }
81                 }
82             sort(e.begin(), e.end());
83             int ct = 0;
84             double tot = 0.0, last;
85             for (j = 0; j < e.size(); j++) {
86                 if (ct == 2)
87                     tot += e[j].first - last;
88                 ct += e[j].second;
89                 last = e[j].first;
90             }
91             ans += s.det(t) * tot;
92         }
93     }
94     return fabs(ans) * 0.5;
95 }
96 };

```

4.7 circles

```

1 const int maxn = 500;
2 struct circles {
3     circle c[maxn];

```

```

4     double ans[maxn]; // ans[i]表示被覆盖了i次的面积
5     double pre[maxn];
6     int n;
7     circles() {
8     }
9     void add(circle cc) {
10         c[n++] = cc;
11     }
12     bool inner(circle x, circle y) {
13         if (x.relationcircle(y) != 1)
14             return 0;
15         return dblcmp(x.r - y.r) <= 0 ? 1 : 0;
16     }
17     void init_or() //圆的面积并去掉内含的圆
18     {
19         int i, j, k = 0;
20         bool mark[maxn] = {0};
21         for (i = 0; i < n; i++) {
22             for (j = 0; j < n; j++)
23                 if (i != j && !mark[j]) {
24                     if ((c[i] == c[j]) || inner(c[i], c[j]
25                     )))
26                         break;
27                 }
28                 if (j < n)
29                     mark[i] = 1;
30             }
31             for (i = 0; i < n; i++)
32                 if (!mark[i])
33                     c[k++] = c[i];
34             n = k;
35         }
36     void init_and() //圆的面积交去掉内含的圆
37     {
38         int i, j, k = 0;
39         bool mark[maxn] = {0};
40         for (i = 0; i < n; i++) {
41             for (j = 0; j < n; j++)
42                 if (i != j && !mark[j]) {
43                     if ((c[i] == c[j]) || inner(c[j], c[i]
44                     )))
45                         break;
46                 }
47                 if (j < n)
48                     mark[i] = 1;
49             }
50             for (i = 0; i < n; i++)
51                 if (!mark[i])
52                     c[k++] = c[i];
53             n = k;
54         }
55     double areaarc(double th, double r) {
56         return 0.5 * sqr(r) * (th - sin(th));
57     }
58     void getarea() {
59         int i, j, k;
60         memset(ans, 0, sizeof(ans));
61         vector<pair<double, int>> v;
62         for (i = 0; i < n; i++) {
63             v.clear();
64             v.push_back(make_pair(-pi, 1));
65             v.push_back(make_pair(pi, -1));
66             for (j = 0; j < n; j++)
67                 if (i != j) {
68                     point q = c[j].p.sub(c[i].p);
69                     double ab = q.len(), ac = c[i].r, bc =
70                     c[j].r;
71                     if (dblcmp(ab + ac - bc) <= 0) {
72                         v.push_back(make_pair(-pi, 1));
73                         v.push_back(make_pair(pi, -1));
74                         continue;
75                     }
76                     if (dblcmp(ab + bc - ac) <= 0)
77                         continue;
78                     if (dblcmp(ab - ac - bc) > 0)
79                         continue;
80                     double th = atan2(q.y, q.x),
81                     fai = acos((ac * ac + ab * ab -
82                     bc * bc) /
83                     (2.0 * ac * ab));
84                     double a0 = th - fai;
85                     if (dblcmp(a0 + pi) < 0)
86                         a0 += 2 * pi;
87                 }
88             }
89         }

```

```

83     double a1 = th + fai;
84     if (dblcmp(a1 - pi) > 0)
85         a1 -= 2 * pi;
86     if (dblcmp(a0 - a1) > 0) {
87         v.push_back(make_pair(a0, 1));
88         v.push_back(make_pair(pi, -1));
89         v.push_back(make_pair(-pi, 1));
90         v.push_back(make_pair(a1, -1));
91     } else {
92         v.push_back(make_pair(a0, 1));
93         v.push_back(make_pair(a1, -1));
94     }
95 }
96 sort(v.begin(), v.end());
97 int cur = 0;
98 for (j = 0; j < v.size(); j++) {
99     if (cur && dblcmp(v[j].first - pre[cur]))
100         {
101             ans[cur] += areaarc(v[j].first - pre[
102             cur], c[i].r);
103             ans[cur] +=
104             0.5 *
105             point(c[i].p.x + c[i].r * cos(pre[
106             cur]),
107             c[i].p.y + c[i].r * sin(pre[
108             cur]))
109             .det(point(c[i].p.x + c[i].r *
110             cos(v[j].first),
111             c[i].p.y + c[i].r *
112             sin(v[j].first)));
113         }
114     cur += v[j].second;
115     pre[cur] = v[j].first;
116 }
117 }
118 for (i = 1; i <= n; i++) {
119     ans[i] -= ans[i + 1];
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

4.8 halfplane

```

1 struct halfplane : public line {
2     double angle;
3     halfplane() {
4     }
5     //表示向量 a->b逆时针(左侧)的半平面
6     halfplane(point _a, point _b) {
7         a = _a;
8         b = _b;
9     }
10    halfplane(line v) {
11        a = v.a;
12        b = v.b;
13    }
14    void calcangle() {
15        angle = atan2(b.y - a.y, b.x - a.x);
16    }
17    bool operator<(const halfplane &b) const {
18        return angle < b.angle;
19    }
20 }
21 struct halfplanes {
22     int n;
23     halfplane hp[maxp];
24     point p[maxp];
25     int que[maxp];
26     int st, ed;
27     void push(halfplane tmp) {
28         hp[n++] = tmp;
29     }
30     void unique() {
31         int m = 1, i;
32         for (i = 1; i < n; i++) {
33             if (dblcmp(hp[i].angle - hp[i - 1].angle))
34                 hp[m++] = hp[i];
35             else if (dblcmp(hp[m - 1]
36                 .b.sub(hp[m - 1].a)
37                 .det(hp[i].a.sub(hp[m -
38                 1].a)) > 0))
39                 hp[m - 1] = hp[i];
40         }
41     }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

```

n = m;
}
bool halfplaneinsert() {
    int i;
    for (i = 0; i < n; i++)
        hp[i].calcangle();
    sort(hp, hp + n);
    unique();
    que[st = 0] = 0;
    que[ed = 1] = 1;
    p[1] = hp[0].crosspoint(hp[1]);
    for (i = 2; i < n; i++) {
        while (st < ed &&
            dblcmp((hp[i].b.sub(hp[i].a).det(p[ed].
            sub(hp[i].a)))) < 0)
            ed--;
        while (st < ed &&
            dblcmp((hp[i].b.sub(hp[i].a).det(p[st +
            1].sub(hp[i].a)))) <
            0)
            st++;
        que[++ed] = i;
        if (hp[i].parallel(hp[que[ed - 1]]))
            return false;
        p[ed] = hp[i].crosspoint(hp[que[ed - 1]]);
    }
    while (st < ed &&
        dblcmp(hp[que[st]]
            .b.sub(hp[que[st]].a)
            .det(p[ed].sub(hp[que[st]].a)))
            < 0)
        st++;
    while (st < ed &&
        dblcmp(hp[que[ed]]
            .b.sub(hp[que[ed]].a)
            .det(p[st + 1].sub(hp[que[ed]].a
            ))) < 0)
        ed--;
    st++;
    if (st + 1 >= ed)
        return false;
    return true;
}
void getconvex(polygon &con) {
    p[st] = hp[que[st]].crosspoint(hp[que[ed]]);
    con.n = ed - st + 1;
    int j = st, i = 0;
    for (; j <= ed; i++, j++) {
        con.p[i] = p[j];
    }
}
};

```

4.9 point3

```

1 struct point3 {
2     double x, y, z;
3     point3() {
4     }
5     point3(double _x, double _y, double _z) : x(_x), y(_y), z(_z) {}
6     void input() {
7         scanf("%lf%lf%lf", &x, &y, &z);
8     }
9     void output() {
10        printf("%.2lf %.2lf %.2lf\n", x, y, z);
11    }
12    bool operator==(point3 a) {
13        return dblcmp(a.x - x) == 0 && dblcmp(a.y - y) ==
14        0 &&
15        dblcmp(a.z - z) == 0;
16    }
17    bool operator<(point3 a) const {
18        return dblcmp(a.x - x) == 0
19        ? dblcmp(y - a.y) == 0 ? dblcmp(z - a.z)
20        : < 0 : y < a.y
21        : x < a.x;
22    }
23    double len() {
24        return sqrt(len2());
25    }
26    double len2() {
27        return x * x + y * y + z * z;
28    }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267
```

```

27 double distance(point3 p) {
28     return sqrt((p.x - x) * (p.x - x) + (p.y - y) * (p
        .y - y) +
29         (p.z - z) * (p.z - z));
30 }
31 point3 add(point3 p) {
32     return point3(x + p.x, y + p.y, z + p.z);
33 }
34 point3 sub(point3 p) {
35     return point3(x - p.x, y - p.y, z - p.z);
36 }
37 point3 mul(double d) {
38     return point3(x * d, y * d, z * d);
39 }
40 point3 div(double d) {
41     return point3(x / d, y / d, z / d);
42 }
43 double dot(point3 p) {
44     return x * p.x + y * p.y + z * p.z;
45 }
46 point3 det(point3 p) {
47     return point3(y * p.z - p.y * z, p.x * z - x * p.z
        , x * p.y - p.x * y);
48 }
49 double rad(point3 a, point3 b) {
50     point3 p = (*this);
51     return acos(a.sub(p).dot(b.sub(p)) / (a.distance(p)
        ) * b.distance(p));
52 }
53 point3 trunc(double r) {
54     r /= len();
55     return point3(x * r, y * r, z * r);
56 }
57 point3 rotate(point3 o, double r) {
58 }
59 };

```

4.10 line3

```

1 struct line3 {
2     point3 a, b;
3     line3() {
4     }
5     line3(point3 _a, point3 _b) {
6         a = _a;
7         b = _b;
8     }
9     bool operator==(line3 v) {
10         return (a == v.a) && (b == v.b);
11     }
12     void input() {
13         a.input();
14         b.input();
15     }
16     double length() {
17         return a.distance(b);
18     }
19     bool pointonseg(point3 p) {
20         return dblcmp(p.sub(a).det(p.sub(b)).len()) == 0
            &&
21             dblcmp(a.sub(p).dot(b.sub(p))) <= 0;
22     }
23     double dispointtoline(point3 p) {
24         return b.sub(a).det(p.sub(a)).len() / a.distance(b
            );
25     }
26     double dispointtoseg(point3 p) {
27         if (dblcmp(p.sub(b).dot(a.sub(b))) < 0 ||
28             dblcmp(p.sub(a).dot(b.sub(a))) < 0) {
29             return min(p.distance(a), p.distance(b));
30         }
31         return dispointtoline(p);
32     }
33     point3 lineprog(point3 p) {
34         return a.add(b.sub(a).trunc(b.sub(a).dot(p.sub(a))
            / b.distance(a));
35     }
36     point3 rotate(point3 p, double ang) // p绕此向量逆时
        针ang角度
37     {
38         if (dblcmp((p.sub(a).det(p.sub(b)).len())) == 0)
39             return p;
40         point3 f1 = b.sub(a).det(p.sub(a));

```

```

        point3 f2 = b.sub(a).det(f1);
        double len = fabs(a.sub(p).det(b.sub(p)).len() / a
            .distance(b));
        f1 = f1.trunc(len);
        f2 = f2.trunc(len);
        point3 h = p.add(f2);
        point3 pp = h.add(f1);
        return h.add((p.sub(h)).mul(cos(ang * 1.0)))
            .add((pp.sub(h)).mul(sin(ang * 1.0)));
    }
};

```

4.11 plane

```

1 struct plane {
2     point3 a, b, c, o;
3     plane() {
4     }
5     plane(point3 _a, point3 _b, point3 _c) {
6         a = _a;
7         b = _b;
8         c = _c;
9         o = pvec();
10    }
11    plane(double _a, double _b, double _c, double _d) {
12        // ax+by+cz+d=0
13        o = point3(_a, _b, _c);
14        if (dblcmp(_a) != 0) {
15            a = point3((-_d - _c - _b) / _a, 1, 1);
16        } else if (dblcmp(_b) != 0) {
17            a = point3(1, (-_d - _c - _a) / _b, 1);
18        } else if (dblcmp(_c) != 0) {
19            a = point3(1, 1, (-_d - _a - _b) / _c);
20        }
21    }
22    void input() {
23        a.input();
24        b.input();
25        c.input();
26        o = pvec();
27    }
28    point3 pvec() {
29        return b.sub(a).det(c.sub(a));
30    }
31    bool pointonplane(point3 p) //点是否在平面上
32    {
33        return dblcmp(p.sub(a).dot(o)) == 0;
34    }
35    // 0 不在
36    // 1 在边界上
37    // 2 在内部
38    int pointontriangle(point3 p) //点是否在空间三角形abc
        上
39    {
40        if (!pointonplane(p))
41            return 0;
42        double s = a.sub(b).det(c.sub(b)).len();
43        double s1 = p.sub(a).det(p.sub(b)).len();
44        double s2 = p.sub(a).det(p.sub(c)).len();
45        double s3 = p.sub(b).det(p.sub(c)).len();
46        if (dblcmp(s - s1 - s2 - s3))
47            return 0;
48        if (dblcmp(s1) && dblcmp(s2) && dblcmp(s3))
49            return 2;
50        return 1;
51    }
52    //判断两平面关系
53    // 0 相交
54    // 1 平行但不重合
55    // 2 重合
56    bool relationplane(plane f) {
57        if (dblcmp(o.det(f.o).len()))
58            return 0;
59        if (pointonplane(f.a))
60            return 2;
61        return 1;
62    }
63    double angleplane(plane f) //两平面夹角
64    {
65        return acos(o.dot(f.o) / (o.len() * f.o.len()));
66    }
67    double dispoint(point3 p) //点到平面距离
68    {

```

```

69     return fabs(p.sub(a).dot(o) / o.len());
70 }
71 point3 pttoplane(point3 p) //点到平面最近点
72 {
73     line3 u = line3(p, p.add(o));
74     crossline(u, p);
75     return p;
76 }
77 int crossline(line3 u, point3 &p) //平面和直线的交点
78 {
79     double x = o.dot(u.b.sub(a));
80     double y = o.dot(u.a.sub(a));
81     double d = x - y;
82     if (dblcmp(fabs(d)) == 0)
83         return 0;
84     p = u.a.mul(x).sub(u.b.mul(y)).div(d);
85     return 1;
86 }
87 int crossplane(plane f, line3 &u) //平面和平面的交线
88 {
89     point3 oo=o.det(f.o);
90     point3 v=o.det(oo);
91     double d=fabs(f.o.dot(v));
92     if (dblcmp(d)==0)return 0;
93     point3 q=a.add(v.mul(f.o.dot(f.a.sub(a))/d));
94     u=line3(q,q.add(oo));
95     return 1;
96 }
97 };

```

5 Number Theory

5.1 GCD

```

1 int gcd(int a, int b)
2 {
3     return (b ? gcd(b, a % b) : a);
4 }
5
6 int gcd(int a, int b)
7 {
8     int t = 1, c, d;
9     while(a != b)
10    {
11        if(a < b) swap(a, b);
12        if(!(a & 1))
13        {
14            a >>= 1;
15            c = 1;
16        }
17        else c = 0;
18        if(!(b & 1))
19        {
20            b >>= 1;
21            d = 1;
22        }
23        else d = 0;
24        if(c && d) t <<= 1;
25        else if(!c && !d) a -= b;
26    }
27    return t * a;
28 }

```

5.2 Prime

```

1 // TODO
2 bool prime[MAXN];
3 void findprime() {
4     for (int i = 0; i < MAXN; i++)
5         prime[i] = true;
6
7     prime[0] = false;
8     prime[1] = false;
9
10    for (int i = 2; i < MAXN; i++)
11        if (prime[i])
12            for (int j = i * i; j < MAXN; j += i)
13                prime[j] = false;
14 }
15 int main() {
16     int i;

```

```

findprime();
for (i = 3; i < 100; i++)
    if (prime[i] == true)
        printf("%d\n", i);
return 0;
}

void get_prime() {
    int cnt = 0;
    for (int i = 2; i < N; i++) {
        if (!tag[i])
            p[cnt++] = i;
        for (int j = 0; j < cnt && p[j] * i < N; j++) {
            tag[i * p[j]] = 1;
            if (i % p[j] == 0)
                break;
        }
    }
}

// 欧拉函数
int phi[3000010];
int euler() {
    int i, j;
    for (i = 0; i < 3000010; i++)
        phi[i] = 0;
    for (i = 2; i < 3000010; i++) {
        if (!phi[i])
            for (j = i; j < 3000010; j += i) {
                if (!phi[j])
                    phi[j] = j;
                phi[j] = phi[j] / i * (i - 1);
            }
    }
}
}

```

6 Graph Theory

6.1 Kruskal

```

#include <stdio>
#include <cstring>
#include <algorithm>
using namespace std;
const int mx = int(1e4) + 5;
const int mxm = int(1e5) + 5;

struct edge {
    int from, to, cost;
    void read() {
        scanf("%d%d%d", &from, &to, &cost);
    }
    bool operator<(const edge &b) const {
        return cost < b.cost;
    }
} e[mxm];

int fa[mx], n, m;
int find(int x) {
    return ~fa[x] ? fa[x] = find(fa[x]) : x;
}

int Kruskal() {
    sort(e, e + m);
    memset(fa, -1, sizeof(fa));
    int sum = 0, cnt = 0;
    for (int i = 0; i < m; ++i) {
        int fau = find(e[i].from), fav = find(e[i].to);
        if (fau != fav)
            fa[fau] = fav, sum += e[i].cost, ++cnt;
    }
    return cnt == n - 1 ? sum : -1;
}

#include <iostream>
#include <algorithm>
#define MAXN 11111
using namespace std;
int w[MAXN], p[MAXN], r[MAXN], u[MAXN], v[MAXN];
int cmp(int i, int j) {
    return w[i] < w[j];
}

```



```

43 int find(int x) {
44     return p[x] == x ? x : p[x] = find(p[x]);
45 }
46 void kruskal(int n, int m) {
47     int ans = 0;
48     for (int i = 0; i <= n; ++i)
49         p[i] = i;
50     for (int i = 0; i <= m; ++i)
51         r[i] = i;
52     sort(r, r + m, cmp);
53     for (int i = 0; i < m; ++i) {
54         int e = r[i], x = find(u[e]), y = find(v[e]);
55         if (x != y) {
56             ans += w[e];
57             p[x] = y;
58         }
59     }
60     cout << ans << endl;
61 }
62 int main() {
63     int n;
64     while (cin >> n && n) {
65         int m = 0;
66         for (int i = 1; i <= n; ++i) {
67             for (int j = 1; j <= n; ++j) {
68                 cin >> w[m];
69                 u[m] = i;
70                 v[m] = j;
71                 ++m;
72             }
73         }
74         kruskal(n, m);
75     }
76     return 0;
77 }

```

```

1 int lis[MAXN];
2 int lis() {
3     int n, i, j, x, len = 0;
4     for (i = 1; i <= n; ++i) {
5         scanf("%d", &x); // 或者 x = num[i]
6         j = lower_bound(lis + 1, lis + len + 1, x) - lis;
7         // LDS: j = lower_bound(lds + 1, lds + len + 1, x,
8             greater<int>()) - lds;
9         lis[j] = x;
10        len = max(len, j);
11    }
12    return len;
13 }
14 // 打印路径: 使用 pos 数组, 从后往前扫, (len—)的第一次出
15 // 现就是答案
16 // LIS[i] = max{1, LIS[k] + 1} (0 < k < i, arr[i] > arr[k])
17 /*
18 Longest Not-decrease Sequence:
19 bool cmp(int a, int b) {
20     return a <= b;
21 }
22 j = lower_bound(lis + 1, lis + len + 1, x, cmp) - lis;
23 */

```

7 DP

7.1 LCS

```

1 const int MX = 1000;
2
3 char a[MX], b[MX];
4 int dp[MX][MX], path[MX][MX];
5
6 int Lcs(char x[], char y[]) {
7     int i, j, len1 = strlen(x + 1), len2 = strlen(y + 1);
8     memset(dp, 0, sizeof(dp));
9     for (i = 1; i <= len1; ++i)
10         for (j = 1; j <= len2; ++j) {
11             if (x[i] == y[j])
12                 dp[i][j] = dp[i - 1][j - 1] + 1, path[i][j]
13                     = 1;
14             else if (dp[i - 1][j] >= dp[i][j - 1])
15                 dp[i][j] = dp[i - 1][j], path[i][j] = 2;
16             else
17                 dp[i][j] = dp[i][j - 1], path[i][j] = 3;
18         }
19     return dp[len1][len2];
20 }
21
22 void PrintLcs(int i, int j) {
23     if (i == 0 || j == 0) return;
24     if (path[i][j] == 1) {
25         PrintLcs(i - 1, j - 1);
26         putchar(a[i]);
27     } else if (path[i][j] == 2) PrintLcs(i - 1, j);
28     else PrintLcs(i, j - 1);
29 }
30
31 int main() {
32     while (gets(a + 1)) {
33         gets(b + 1);
34         printf("%d\n", Lcs(a, b));
35         PrintLcs(strlen(a + 1), strlen(b + 1));
36         putchar(10);
37     }
38     return 0;
39 }

```

7.2 LIS