BakaBakaBerserker

Uni

October 23, 2015

目录

1 Basic

1.1 Attentions

```
1. 调试
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？
左右端点范围？acos/asin/sqrt 函数定义域？精度问题？
RE：爆栈？整数除以 0？数组严重越界？未 return 0？

2. Code::Blocks 更换终端
gnome-terminal -t $TITLE -x

3. 0 除以负数是负零，输出时加 eps，注意 sqrt(-0.00)

4. double 全局变量初始可能非零
```

1.2 Binary Search

```
// 二分查找，区间 [l, r]
int bs(int k) {
    int l = 0, r = n - 1, p;
    while (l <= r) {
        p = (l + r) >> 1;
        if (a[p] == k)
            return p;
        if (a[p] < k)
            l = p + 1;
        else
            r = p - 1;
    }
    return -1;
}

// 二分查找，区间 [l, r)
int bs(int k) {
    int l = 0, r = n, p;
    while (l < r) {
        p = (l + r) >> 1;
        if (a[p] == k)
            return p;
        if (a[p] < k)
            l = p + 1;
        else
            r = p;
    }
    return -1;
}

// lower_bound（第一个 >=）区间 [l, r)
// STL 排序
bool cmp(const A &x, const A &y) {
    return x.v < y.v;
}

int lb(int k) {
    int cnt = r - l, it, step;
    while (cnt > 0) {
        step = cnt / 2;
        it = l + step;
        if (a[it] < k) {
            l = ++it;
            cnt -= step + 1;
        } else {
            cnt = step;
        }
    }
    return l;
}

// upper_bound（第一个 >）区间 [l, r)
int ub(int k) {
    int cnt = r - l, it, step;
    while (cnt > 0) {
        step = cnt / 2;
        it = l + step;
        if (a[it] <= k) {
            l = ++it;
            cnt -= step + 1;
        } else {
            cnt = step;
        }
    }
    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);
```

```
    int ty = a.month >= 3 ? a.year : (a.year - 1);
    return (ty + ty / 4 - ty / 100 + ty / 400 + (int)(2.6
    * tm - 0.2) + a.day) % 7;
}

//日期转天数偏移（用于计算日期差？）
int date2int(date a) {
    int ret = a.year * 365 + (a.year - 1) / 4 - (a.year -
    1) / 100 +
              (a.year - 1) / 400, i; days[1] += leap(a.
              year);
    for (i = 0; i < a.month - 1; ret += days[i++]);
    days[1] = 28;
    return ret + a.day;
}

//天数偏移转日期
date int2date(int a) {
    date ret;
    ret.year = a / 146097 * 400;
    for (a %= 146097; a >= 365 + leap(ret.year);
            a -= 365 + leap(ret.year), ret.year++);
    days[1] += leap(ret.year);
    for (ret.month = 1; a >= days[ret.month - 1];
            a -= days[ret.month - 1], ret.month++);
    days[1] = 28; ret.day = a + 1; return ret;
}
```

1.5 Fast IO

```
// 1. fread + 缓冲区法
struct FastIO {
    static const int S = 131072;
    int wpos;
    char wbuf[S];
    FastIO() : wpos(0) {
    }
    inline int xchar() {
        static char buf[S];
        static int len = 0, pos = 0;
        if (pos == len)
            pos = 0, len = fread(buf, 1, S, stdin);
        if (pos == len)
            return -1;
        return buf[pos++];
    }
    inline int xuint() {
        int s;
        scanf("%d", &s);
        return s;
        int c = xchar(), x = 0;
        while (c <= 32)
            c = xchar();
        for (; '0' <= c && c <= '9'; c = xchar())
            x = x * 10 + c - '0';
        return x;
    }
    inline int xint() {
        int t;
        scanf("%d", &t);
        return t;
        int s = 1, c = xchar(), x = 0;
        while (c <= 32)
            c = xchar();
        if (c == '-')
            s = -1, c = xchar();
        for (; '0' <= c && c <= '9'; c = xchar())
            x = x * 10 + c - '0';
        return x * s;
    }
    ~FastIO() {
        if (wpos)
            fwrite(wbuf, 1, wpos, stdout), wpos = 0;
    }
} io;

// 2. naive
inline void scanf_(int &num) {
    char in;
    bool neg = false;
    while (((in = getchar()) > '9' || in < '0') && in !=
    '-')
        ;
    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);
    }
}
```

1.6 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……
区间循环位移？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];
```

```
string b = string(a);
printf("%s", b.c_str());

// string 中寻找字符
string s;
if (s.find('+') != string::npos) {
    // s.find('+') 类型为 size_type
}

// string 中消除指定字符
string t = "";
for (int j = 0; j < (int)s.size(); ++j) {
    if (s[j] != '.') {
        t += s[j];
    }
}
s = t;

// 优先队列结构体比较（从小到大）
struct P {
    int a, b;
    bool operator<(const P &x) const {
        return b > x.b;
    }
}
```

2 Data Structures

2.1 Binary Indexed Tree

```
// 1. 单点
// 每次用之前 memset
int c[MX];

// 不判 0 可能操蛋
void U(int x, int v) {
    if (x == 0) {
        return;
    }
    for (; x <= n; x += (x & -x))
        c[x] += v;
}

// [1,x]的和
int Q(int x) {
    if (x == 0) {
        return;
    }
    int r = 0;
    for (; x > 0; x -= (x & -x))
        r += c[x];
    return r;
}

// 2. 二维
int c[MAXX][MAXY];

// 更新 (x,y)
void U(int x, int y, int v) {
    for (; x <= MAXX; x += (x & -x))
        for (int i = y; i <= MAXY; i += (i & -i))
            c[x][i] += v;
}

// (1,1) 到 (x,y) 的和
int Q(int x, int y) {
    int r = 0;
    for (; x > 0; x -= (x & -x))
        for (int i = y; i > 0; i -= (i & -i))
            r += c[x][i];
    return r;
}

// 3. 区间更新 [l,r]
// U(l - 1, -c), U(r, c);
void U(int x, int v) {
    for (; x > 0; x -= (x & -x))
        b[x] += v;
}

// Q(x) 查询 a[x] 的值
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

```
// 桶的大小 p = sqrt(n), 块数 q = n / p + (n % p ? 1 : 0)
// 可以预处理 belong[i] = (i - 1) / p - 1
// 五维偏序: 分块 + bitset
l[i] = (i - 1) * p + 1, r[i] = i * p, r[q] = n;
bitset<50001> b[7][270];
bitset<50001> Ans[7];
int now[7];
struct node {
    int x, y;
};
bool cmp(node a, node b) {
    return a.x < b.x;
}
node a[7][50001];
int l[300], r[300];
int belong[50001];
int main() {
    int t;
    scanf("%d", &t);
    for (int cas = 1; cas <= t; cas++) {
        int n = read(), m = read();
        for (int i = 1; i <= 5; i++)
            for (int j = 1; j < 250; j++)
                b[i][j].reset();
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= 5; j++) {
                a[j][i].x = read();
                a[j][i].y = i;
            }
        }
        for (int i = 1; i <= 5; i++)
            sort(a[i] + 1, a[i] + n + 1, cmp);
        int block = sqrt(n);
        int num = n / block;
        if (n % block)
            num++;
        for (int i = 1; i <= num; i++)
            l[i] = (i - 1) * block + 1, r[i] = i * block;
        r[num] = n;
        for (int i = 1; i <= n; i++)
            belong[i] = (i - 1) / block + 1;
        for (int i = 1; i <= 5; i++) {
            for (int j = 1; j <= num; j++) {
                b[i][j] |= b[i][j - 1];
                for (int k = l[j]; k <= r[j]; k++)
                    b[i][j][a[i][k].y] = 1;
            }
        }

        int q = read();
        int lastans = 0;
        while (q--) {
            for (int i = 1; i <= 5; i++)
                now[i] = read();
            for (int i = 1; i <= 5; i++)
                now[i] ^= lastans;
            for (int i = 1; i <= 5; i++)
                Ans[i].reset();
            for (int i = 1; i <= 5; i++) {
                int L = 0, R = n;
                while (L <= R) {
                    int mid = (L + R) / 2;
                    if (now[i] >= a[i][mid].x)
                        L = mid + 1;
                    else
                        R = mid - 1;
                }
                int p = L - 1;
                if (p == 0)
                    continue;
                Ans[i] |= b[i][belong[p] - 1];
                for (int j = l[belong[p]]; j <= p; j++)
                    Ans[i][a[i][j].y] = 1;
            }
            Ans[1] = Ans[1] & Ans[2] & Ans[3] & Ans[4] &
            Ans[5];
            lastans = Ans[1].count();
            printf("%d\n", lastans);
        }
    }
}
```

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];
```

```
int N, L, R, V, X;

void up(int n) {
    sum[n] = sum[lc] + sum[rc];
    ma[n] = max(ma[lc], ma[rc]);
    mi[n] = min(mi[lc], mi[rc]);
}

void down(int n, int m) {
    if (add[n]) {
        add[lc] += add[n];
        add[rc] += add[n];
        sum[lc] += add[n] * (m - (m >> 1));
        sum[rc] += add[n] * (m >> 1);
        add[n] = 0;
    }
}

void B(int l = 1, int r = N, int n = 1) {
    add[n] = 0;
    if (l == r) {
        scanf("%d", &num[l]);
        sum[n] = ma[n] = mi[n] = num[l];
        return;
    }
    int m = (l + r) >> 1;
    B(ls), B(rs), up(n);
}

// 单点更新 X 为 V
void U(int l = 1, int r = N, int n = 1) {
    if (l == r) {
        sum[n] = ma[n] = mi[n] = num[l] = V; // 赋值或增加
        return;
    }
    down(n, r - l + 1);
    int m = (l + r) >> 1;
    if (X <= m) U(ls);
    else U(rs);
    up(n);
}

// 区间更新 [L,R] 为 V
void U(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R) {
        add[n] += V, sum[n] += V * (r - l + 1);
        return;
    }
    down(n, r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) U(ls);
    if (m < R) U(rs);
    up(n);
}

// 查询区间 [L,R]
int Q(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R) {
        return sum[n];
    }
    down(n, r - l + 1);
    int ans = 0, m = (l + r) >> 1;
    if (L <= m) ans += Q(ls);
    // ans = max(ans, Q(ls));
    if (m < R) ans += Q(rs);
    // ans = max(ans, Q(rs));
    return ans;
}
```

2.5 Interval Tree 2D

```
#include <cstdio>
#include <algorithm>
#define maxn 510
#define it tree[p1][p2]
using namespace std;
struct Seg_Tree2D
{
    int minv, maxv;
    friend Seg_Tree2D operator + (const Seg_Tree2D &a, const Seg_Tree2D &b)
    {
        Seg_Tree2D c;
        c.minv = min(a.minv, b.minv);
        c.maxv = max(a.maxv, b.maxv);
        return c;
    }
} tree[maxn << 2][maxn << 2];
int matrix[maxn][maxn], n, m;
void Build2(int p1, int p2, int l, int r, int a, int b)
{
    if (a == b)
    {
        if (l == r)
            it.minv = it.maxv = matrix[l][a];
        else
            it = tree[p1 << 1][p2] + tree[p1 << 1 | 1][p2];
    }
    else
    {
        int mid = (a + b) >> 1;
        Build2(p1, p2 << 1, l, r, a, mid), Build2(p1, p2 << 1 | 1, l, r, mid + 1, b);
        it = tree[p1][p2 << 1] + tree[p1][p2 << 1 | 1];
    }
}
void Build1(int p1, int l, int r)
{
    if (l == r)
    {
        Build2(p1, 1, l, r, 1, m);
        return ;
    }
    int mid = (l + r) >> 1;
    Build1(p1 << 1, l, mid), Build1(p1 << 1 | 1, mid + 1, r);
    Build2(p1, 1, l, r, 1, m);
}
void Modify2(int p1, int p2, int l, int r, int a, int b, int y, int v)
{
    if (a == b)
    {
        if (l == r)
            it.minv = it.maxv = v;
        else
            it = tree[p1 << 1][p2] + tree[p1 << 1 | 1][p2];
    }
    else
    {
        int mid = (a + b) >> 1;
        if (y <= mid)
            Modify2(p1, p2 << 1, l, r, a, mid, y, v);
        else
            Modify2(p1, p2 << 1 | 1, l, r, mid + 1, b, y, v);
        it = tree[p1][p2 << 1] + tree[p1][p2 << 1 | 1];
    }
}
void Modify1(int p1, int l, int r, int x, int y, int v)
{
    if (l == r)
    {
        Modify2(p1, 1, l, r, 1, m, y, v);
        return ;
    }
    int mid = (l + r) >> 1;
    if (x <= mid)
        Modify1(p1 << 1, l, mid, x, y, v);
    else
        Modify1(p1 << 1 | 1, mid + 1, r, x, y, v);
    Modify2(p1, 1, l, r, 1, m, y, v);
}
Seg_Tree2D Query2(int p1, int p2, int l, int r, int a, int b)
{
    if (l == a && r == b)
        return it;
    int mid = (l + r) >> 1;
    if (b <= mid)
        return Query2(p1, p2 << 1, l, mid, a, b);
    else if (mid < a)
        return Query2(p1, p2 << 1 | 1, mid + 1, r, a, b);
    return Query2(p1, p2 << 1, l, mid, a, mid) + Query2(p1
```

```
    , p2 << 1 | 1, mid + 1, r, mid + 1, b);
}
Seg_Tree2D Query1(int p1, int l, int r, int ax, int ay, int bx, int by)
{
    if (l == ax && r == bx)
        return Query2(p1, 1, 1, m, ay, by);
    int mid = (l + r) >> 1;
    if (bx <= mid)
        return Query1(p1 << 1, l, mid, ax, ay, bx, by);
    else if (mid < ax)
        return Query1(p1 << 1 | 1, mid + 1, r, ax, ay, bx, by);
    return Query1(p1 << 1, l, mid, ax, ay, mid, by) + Query1(p1 << 1 | 1, mid + 1, r, mid + 1, ay, bx, by);
}
void read()
{
    scanf("%d %d", &n, &m);
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            scanf("%d", &matrix[i][j]);
    Build1(1, 1, n);
}
void Query()
{
    char task[10];
    int q;
    scanf("%d", &q);
    Seg_Tree2D ans;
    for (int i = 1, a, b, c, d; i <= q; i++)
    {
        scanf("%s", task);
        if (task[0] == 'c')
        {
            scanf("%d %d %d", &a, &b, &c);
            Modify1(1, 1, n, a, b, c);
        }
        else
        {
            scanf("%d %d %d %d", &a, &b, &c, &d);
            ans = Query1(1, 1, n, a, b, c, d);
            printf("%d %d\n", ans.maxv, ans.minv);
        }
    }
}
int main()
{
    read();
    Query();
    return 0;
}
```

2.6 Interval Tree Summary

```
// 1. LA 3787
// 单点赋值，查询区间哈希值。
ll b, p, c[MX << 2], s[MX];
void up() {
    c[n] = (c[lc] * s[r - m] + c[rc]) % p;
}
ll Q(int l = 1, int r = N, int n = 1) {
    if (x <= l && r <= y) {
        return c[n] * s[y - r];
    }
    ... return ans % p;
}
void init() {
    s[0] = 1;
    for (int i = 1; i <= N; ++i)
        s[i] = (s[i - 1] * b) % p;
    memset(c, 0, sizeof(c));
}

// 2. 扫描线
struct seg {
    double l, r, p;
    int d;
    seg() {
    }
    seg(double _l, double _r, double _p, int _d) {
        l = _l, r = _r, p = _p, d = _d;
    }
} 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\n", ++cas, ans);
    }
    return 0;
}

// 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;
```

```
    if (X <= m)
        U(lson);
    else
        U(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]);
}
int QL(int L, int R, int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R)
        return lm[n];
    int m = (l + r) >> 1;
    if (L > m)
        return QL(L, R, rson);
    if (R <= m)
        return QL(L, R, lson);
    int ans = QL(L, m, lson), k = m - L + 1;
    if (ans == k && num[m] < num[m + 1])
        ans += QL(m + 1, R, rson);
    return ans;
}
int QR(int L, int R, int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R)
        return rm[n];
    int m = (l + r) >> 1;
    if (L > m)
        return QR(L, R, rson);
    if (R <= m)
        return QR(L, R, lson);
    int ans = QR(m + 1, R, rson), k = R - m;
    if (ans == k && num[m] < num[m + 1])
        ans += QR(L, m, lson);
    return ans;
}
int Q(int L, int R, int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R)
        return mm[n];
    int m = (l + r) >> 1;
    if (L > m)
        return Q(L, R, rson);
    if (R <= m)
        return Q(L, R, lson);
    int ans = max(Q(L, m, lson), Q(m + 1, R, rson));
    if (num[m] < num[m + 1])
        ans = max(ans, QR(L, m, lson) + QL(m + 1, R, rson)
        );
    return ans;
}
int main() {
    int t, m, L, R;
    char o[9];
    scanf("%d", &t);
    while (t--) {
        scanf("%d%d", &N, &m), B();
        while (m--) {
            scanf("%s", o);
            if (o[0] == 'Q') {
                scanf("%d%d", &L, &R), printf("%d\n", Q(L
                + 1, R + 1));
            } else {
                scanf("%d%d", &X, &V), ++X, U();
            }
        }
    }
    return 0;
}

// 4. POJ 3277
// 更新区间，只报留最大值
int v, N, L, R, x[MX], y[MX], h[MX], p[MX], s[MX << 2];
void B(int l = 1, int r = N, int n = 1) {
    if (l == r - 1)
        return;
    int m = (l + r) >> 1;
    B(lson), B(rson);
}
void U(int L, int R, int l = 1, int r = N, int n = 1) {
    if (p[l] == L && p[r] == R) {
        if (s[n] < h[v])
            s[n] = h[v];
        return;
    }
    int m = (l + r) >> 1;
    if (R <= p[m])
        U(L, R, lson);
    else if (L >= p[m])
        U(L, R, rson);
    else
        U(L, p[m], lson), U(p[m], R, rson);
}
ll Q(int l = 1, int r = N, int n = 1, int t = 0) {
    if (s[n] < t)
        s[n] = t;
    if (l == r - 1)
        return (ll)(p[r] - p[l]) * s[n];
    int m = (l + r) >> 1;
    return Q(lson, s[n]) + Q(rson, s[n]);
}
int main() {
    int m;
    scanf("%d", &m);
    for (int i = 1; i <= m; ++i) {
        scanf("%d%d%d", &x[i], &y[i], &h[i]);
        p[++N] = x[i], p[++N] = y[i];
    }
    sort(p + 1, p + N + 1);
    N = unique(p + 1, p + N + 1) - p - 1;
    B();
    for (v = 1; v <= m; ++v)
        U(x[v], y[v]);
    printf("%lld\n", Q());
    return 0;
}

// 5. HDU 3333
// 区间不重复数字和
struct S {
    int l, r, i;
} p[MX];
int x[MX], a[MX], f[MX];
ll c[MX], ans[MX];
int cmp(S a, S b) {
    return a.r < b.r;
}
void U(int x, int v) {
    for (; x < MX; x += (x & -x))
        c[x] += v;
}
ll Q(int x) {
    ll s = 0;
    for (; x > 0; x -= (x & -x))
        s += c[x];
    return s;
}
int B(int v, int u) {
    int l = 0, r = u - 1, m;
    while (l <= r) {
        m = (l + r) >> 1;
        if (x[m] == v)
            return m;
        if (x[m] < v)
            l = m + 1;
        else
            r = m - 1;
    }
    return -1;
}
int main() {
    int t, n, q, l, r, cnt;
    scanf("%d", &t);
    while (t--) {
        memset(c, 0, sizeof(c));
        memset(f, 0, sizeof(f));
        scanf("%d", &n), cnt = n;
        for (int i = 1; i <= n; ++i)
            scanf("%d", &a[i]), x[i - 1] = a[i];
        sort(x, x + cnt);
        cnt = unique(x, x + cnt) - x;
        scanf("%d", &q);
        for (int i = 0; i < q; ++i)
```

```
            scanf("%d%d", &p[i].l, &p[i].r), p[i].i = i;
        sort(p, p + q, cmp);
        int k = 0, d;
        for (int i = 1; i <= n; ++i) {
            d = B(a[i], cnt);
            if (f[d])
                U(f[d], -a[i]);
            U(i, a[i]);
            f[d] = i;
            for (; k < q; ++k) {
                if (p[k].r == i)
                    ans[p[k].i] = Q(p[k].r) - Q(p[k].l - 1);
                else
                    break;
            }
        }
        for (int i = 0; i < q; ++i)
            printf("%I64d\n", ans[i]);
    }
    return 0;
}

// 6. POJ 2452
// 最大的 j-i 满足 a[i] 最小 a[j] 最大
int num[MX], ma[MX << 2], mi[MX << 2];
int L, R, N, ans;
inline void up(int n) {
    ma[n] = num[ma[lc]] > num[ma[rc]] ? ma[lc] : ma[rc];
    mi[n] = num[mi[lc]] < num[mi[rc]] ? mi[lc] : mi[rc];
}
void B(int l = 1, int r = N, int n = 1) {
    if (l == r) {
        scanf("%d", &num[l]);
        ma[n] = mi[n] = l;
        return;
    }
    int m = l + r >> 1;
    B(lson), B(rson), up(n);
}
int QA(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R)
        return ma[n];
    int ans = 0, m = l + r >> 1, t;
    if (L <= m)
        t = QA(lson), ans = (num[ans] > num[t]) ? ans : t;
    if (m < R)
        t = QA(rson), ans = (num[ans] > num[t]) ? ans : t;
    return ans;
}
int QB(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R)
        return mi[n];
    int ans = N + 1, m = l + r >> 1, t;
    if (L <= m)
        t = QB(lson), ans = (num[ans] < num[t]) ? ans : t;
    if (m < R)
        t = QB(rson), ans = (num[ans] < num[t]) ? ans : t;
    return ans;
}
int S(int l, int r) {
    if (l >= r)
        return -1;
    L = l, R = r;
    int ans = -1, a = QA(), b = QB();
    if (a > b)
        ans = max(a - b, max(S(l, b), S(a, r)));
    else
        ans = max(S(a + 1, b - 1), max(S(l, a), S(b, r)));
    return ans;
}
int main() {
    while (~scanf("%d", &N)) {
        B(), num[N + 1] = 111111;
        printf("%d\n", S(1, N));
    }
    return 0;
}

// 7. SPOJ GSS3
// 区间最大子序列和
int num, sum[MX << 2], mm[MX << 2], lm[MX << 2], rm[MX << 2];
int L, R, N, x, v;
inline void up(int n) {
    sum[n] = sum[lc] + sum[rc];
    lm[n] = max(lm[lc], sum[lc] + lm[rc]);
    rm[n] = max(rm[rc], sum[rc] + rm[lc]);
    mm[n] = max(rm[lc] + lm[rc], max(mm[lc], mm[rc]));
}
void B(int l = 1, int r = N, int n = 1) {
    if (l == r) {
        scanf("%d", &sum[n]);
        mm[n] = lm[n] = rm[n] = sum[n];
        return;
    }
    int m = l + r >> 1;
    B(lson), B(rson), up(n);
}
void U(int l = 1, int r = N, int n = 1) {
    if (l == r) {
        mm[n] = lm[n] = rm[n] = sum[n] = R;
        return;
    }
    int m = l + r >> 1;
    if (L <= m)
        U(lson);
    else
        U(rson);
    up(n);
}
int QL(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R)
        return lm[n];
    int m = l + r >> 1, ans;
    if (L > m)
        return QL(rson);
    if (R <= m)
        return QL(lson);
    return max(QL(rson) + sum[lc], lm[lc]);
}
int QR(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R)
        return rm[n];
    int m = l + r >> 1, ans;
    if (L > m)
        return QR(rson);
    if (R <= m)
        return QR(lson);
    return max(QR(lson) + sum[rc], rm[rc]);
}
int Q(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R)
        return mm[n];
    int m = l + r >> 1, ans;
    if (L > m)
        return Q(rson);
    if (R <= m)
        return Q(lson);
    return max(QR(lson) + QL(rson), max(Q(lson), Q(rson)))
    ;
}
int main() {
    int m, q;
    scanf("%d", &N), B();
    scanf("%d", &m);
    while (m--) {
        scanf("%d%d%d", &q, &L, &R);
        if (q)
            printf("%d\n", Q());
        else
            U();
    }
    return 0;
}

// 8. SPOJ GSS1
int N, sum[MX], mm[MX << 2], lm[MX << 2], rm[MX << 2];
int get(int l, int r) {
    return sum[r] - sum[l - 1];
}
void B(int l = 1, int r = N, int n = 1) {
    if (l == r) {
        scanf("%d", &mm[n]);
        lm[n] = rm[n] = mm[n];
        sum[l] = sum[l - 1] + mm[n];
```

```
        return;
    }
    int m = l + r >> 1;
    B(lson), B(rson);
    lm[n] = max(lm[lc], get(l, m) + lm[rc]);
    rm[n] = max(rm[rc], get(m + 1, r) + rm[lc]);
    mm[n] = max(rm[lc] + lm[rc], max(mm[lc], mm[rc]));
}
int QL(int L, int R, int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R)
        return lm[n];
    int m = l + r >> 1, ans;
    if (L > m)
        return QL(L, R, rson);
    if (R <= m)
        return QL(L, R, lson);
    return max(QL(m + 1, R, rson) + get(L, m), max(get(L, m), QL(L, m, lson)));
}
int QR(int L, int R, int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R)
        return rm[n];
    int m = l + r >> 1, ans;
    if (L > m)
        return QR(L, R, rson);
    if (R <= m)
        return QR(L, R, lson);
    return max(QR(L, m, lson) + get(m + 1, R),
               max(get(m + 1, R), QR(m + 1, R, rson)));
}
int Q(int L, int R, int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R)
        return mm[n];
    int m = l + r >> 1, ans;
    if (L > m)
        return Q(L, R, rson);
    if (R <= m)
        return Q(L, R, lson);
    return max(QR(L, m, lson) + QL(m + 1, R, rson),
               max(Q(L, m, lson), Q(m + 1, R, rson)));
}
int main() {
    int m, L, R;
    scanf("%d", &N), B();
    scanf("%d", &m);
    while (m--)
        scanf("%d%d", &L, &R), printf("%d\n", Q(L, R));
    return 0;
}

// 9. 区间更新与赋值
ll sum[MX << 2], add[MX << 2], V;
bool se[MX << 2];
int L, R, N;
inline void up(int n) {
    sum[n] = sum[lc] + sum[rc];
}
void B(int l = 1, int r = N, int n = 1) {
    add[n] = 0;
    se[n] = false;
    if (l == r) {
        sum[n] = 0;
        return;
    }
    int m = (l + r) >> 1;
    B(ls), B(rs), up(n);
}
void down(int n, int m) {
    if (se[n]) {
        se[lc] = se[rc] = se[n];
        sum[lc] = sum[rc] = 0;
        se[n] = false;
        add[lc] = add[rc] = 0;
    }
    if (add[n]) {
        add[lc] += add[n];
        add[rc] += add[n];
        sum[lc] += add[n] * (m - (m >> 1));
        sum[rc] += add[n] * (m >> 1);
        add[n] = 0;
    }
}
void U(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R) {
        add[n] += V, sum[n] += V * (r - l + 1);
        return;
    }
    down(n, r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m)
        U(ls);
    if (m < R)
        U(rs);
    up(n);
}
void S(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R) {
        se[n] = true, add[n] = false, sum[n] = 0;
        return;
    }
    down(n, r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m)
        S(ls);
    if (m < R)
        S(rs);
    up(n);
}
ll Q(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R) {
        return sum[n];
    }
    down(n, r - l + 1);
    ll ans = 0, m = (l + r) >> 1;
    if (L <= m)
        ans += Q(ls);
    if (m < R)
        ans += Q(rs);
    return ans;
}
int main() {
    int t, m;
    scanf("%d", &t);
    while (t--) {
        scanf("%d%d", &N, &m), B();
        ll ans = 0;
        int p = 0, q = 0;
        while (m--) {
            scanf("%d", &q);
            L = 1, R = N, V = q - p;
            U(), p = q;
            scanf("%d%d", &L, &R);
            ans += Q(), S();
        }
        printf("%lld\n", ans);
    }
    return 0;
}

// 10. UVA 1400
typedef long long ll;
typedef pair<int, int> seg;
ll sum[MX];
int pre[MX << 2], suf[MX << 2];
seg sub[MX << 2];
ll get(int l, int r) {
    return sum[r] - sum[l - 1];
}
ll get(seg s) {
    return get(s.first, s.second);
}
seg max(seg a, seg b) {
    if (get(a) != get(b))
        return get(a) > get(b) ? a : b;
    return a < b ? a : b;
}
void pushUp(int l, int r, int n) {
    ll v1 = get(l, pre[n << 1]), v2 = get(l, pre[n << 1 | 1]);
    if (v1 == v2)
        pre[n] = min(pre[n << 1], pre[n << 1 | 1]);
    else
        pre[n] = v1 > v2 ? pre[n << 1] : pre[n << 1 | 1];
    v1 = get(suf[n << 1], r), v2 = get(suf[n << 1 | 1], r);
    if (v1 == v2)
```

```
        suf[n] = min(suf[n << 1], suf[n << 1 | 1]);
    else
        suf[n] = v1 > v2 ? suf[n << 1] : suf[n << 1 | 1];
    sub[n] = max(make_pair(suf[n << 1], pre[n << 1 | 1]),
                 max(sub[n << 1], sub[n << 1 | 1]));
}
void build(int l, int r, int n) {
    if (l == r) {
        pre[n] = suf[n] = l, sub[n] = make_pair(l, l);
        return;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    pushUp(l, r, n);
}
seg prefix(int L, int R, int l, int r, int n) {
    if (pre[n] <= R)
        return make_pair(l, pre[n]);
    int m = (l + r) >> 1;
    if (R <= m)
        return prefix(L, R, lson);
    seg ans = prefix(L, R, rson);
    ans.first = l;
    return max(ans, make_pair(l, pre[n << 1]));
}
seg suffix(int L, int R, int l, int r, int n) {
    if (suf[n] >= L)
        return make_pair(suf[n], r);
    int m = (l + r) >> 1;
    if (L > m)
        return suffix(L, R, rson);
    seg ans = suffix(L, R, lson);
    ans.second = r;
    return max(ans, make_pair(suf[n << 1 | 1], r));
}
seg query(int L, int R, int l, int r, int n) {
    if (L <= l && r <= R)
        return sub[n];
    int m = (l + r) >> 1;
    if (R <= m)
        return query(L, R, lson);
    if (L > m)
        return query(L, R, rson);
    return max(max(query(L, R, lson), query(L, R, rson)),
               make_pair(suffix(L, R, lson).first, prefix(
                L, R, rson).second));
}
int main() {
    int cas = 0, n, m, a, b;
    while (~scanf("%d%d", &n, &m)) {
        sum[0] = 0;
        for (int i = 0; i < n; ++i)
            scanf("%d", &a), sum[i + 1] = sum[i] + a;
        build(1, n, 1);
        printf("Case %d:\n", ++cas);
        while (m--) {
            scanf("%d%d", &a, &b);
            seg ans = query(a, b, 1, n, 1);
            printf("%d %d\n", ans.first, ans.second);
        }
    }
    return 0;
}
```

2.7 Monotone

```
// 单调队列
// 利用双端队列实现，在队列中存数组下标。
// 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。
#include <cstdio>
#include <algorithm>
#include <deque>
#define MX 1000001
using namespace std;
typedef long long ll;
ll p[MX];
int main() {
    int t, n, k, s;
    scanf("%d", &t);
    while (t--) {
        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];
```

```
            sum[n][i]++;
        } else {
            s[n + 1][R++] = s[n][i];
        }
    }
    if (l < m) {
        B(l, m, n + 1);
    }
    if (m + 1 < r) {
        B(m + 1, r, n + 1);
    }
}
int Q(int L, int R, int k, int l, int r, int n = 0) {
    if (l == r) {
        return s[n][l];
    }
    int m = (l + r) >> 1;
    int v = 0;
    if (L > l) {
        v = sum[n][L - 1];
    }
    int t = sum[n][R] - v;
    if (t >= k) {
        return Q(l + v, l + sum[n][R] - 1, k, l, m, n + 1)
        ;
    }
    return Q(m + 1 + L - l - v, m + 1 + R - l - sum[n][R],
     k - t, m + 1, r,
             n + 1);
}
int main() {
    int n, m;
    while (~scanf("%d%d", &n, &m)) {
        for (int i = 1; i <= n; ++i) {
            scanf("%d", p + i);
        }
        for (int i = 0; i < 20; ++i) {
            sum[i][0] = 0;
        }
        for (int i = 1; i <= n; ++i) {
            s[0][i] = p[i];
        }
        sort(p + 1, p + n + 1);
        B(1, n);
        while (m--) {
            int l, r, k;
            scanf("%d%d%d", &l, &r, &k);
            printf("%d\n", Q(l, r, k, 1, n));
        }
    }
    return 0;
}
```

2.9 RMQ

```
// st 算法更快，树状数组空间小。
// RMQ    初始化     查询     空间
// st算法  O(nlogn)  O(1)    O(nlogn)
// 树状数组 O(nlogn)  O(logn) O(n)
// st 算法
int p[MX], d[MX][20];
void init(int n) {
    for (int i = 0; i < n; ++i)
        d[i][0] = p[i];
    for (int j = 1; (1 << j) <= n; ++j)
        for (int i = 0; i + (1 << j) - 1 < n; ++i)
            d[i][j] = max(d[i][j - 1], d[i + (1 << (j - 1)
            )][j - 1]);
}
int rmq(int x, int y) {
    if (x > y)
        return 0;
    int k = 0;
    while ((1 << (k + 1)) <= y - x + 1)
        ++k;
    return max(d[x][k], d[y - (1 << k) + 1][k]);
}

// 树状数组
int p[MX], d[MX];
void init(int n) {
    for (int i = 1; i < n; ++i) {
        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;
            }
        } else {
            puts("Lexical Error!");
            return;
        }
    }
    while (!s.empty()) {
```

```
        if (s.top() == '(') {
            puts("Syntax Error!");
            return;
        }
        q.push(s.top());
        s.pop();
    }
    while (!q.empty()) {
        putchar(q.front());
        q.pop();
    }
    puts("");
}
int main() {
    char p[1024];
    while (~scanf("%s", p)) {
        shunting_yard(p);
    }
    return 0;
}
```

2.11 Union Find

```
int p[MX], q[MX], n;
void init() {
    for (int i = 0; i < MX; ++i) {
        p[i] = i;
        q[i] = 1;
    }
}
int F(int x) {
    return x == p[x] ? x : (p[x] = F(p[x]));
}
// 同组
bool equal(int x, int y) {
    return F(x) == F(y);
}
void uni(int x, int y) {
    x = F(x), y = F(y);
    if (x == y)
        return;
    --n;
    p[x] = y;
    q[y] += q[x];
    q[x] = 0;
}
```

3 String

3.1 AC

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

struct Trie {
    int next[500010][26], fail[500010], end[500010];
    int root, L;
    int newnode() {
        for (int i = 0; i < 26; i++) {
            next[L][i] = -1;
        }
        end[L++] = 0;
        return L - 1;
    }
    void init() {
        L = 0;
        root = newnode();
    }
    void insert(char buf[]) {
        int len = strlen(buf);
        int now = root;
        for (int i = 0; i < len; i++) {
            if (next[now][buf[i] - 'a'] == -1) {
                next[now][buf[i] - 'a'] = newnode();
            }
            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 = 100000007ULL;  // 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];
```

```
    int n = Max(len, s_num);  // 调用#define的Max()
    Forr(i, 1, n + 1) bp[i] = bp[i - 1] * B;
}

ull get_hash(char *s) {  // 直接返回整个字符串的hash
    ull ha = OULL;
    for (int i = 0; s[i]; ++i)
        ha = ha * B + s[i];
    return ha;
}

ull get_hash(int *a, int n) {  // 返回整个int数组的hash值
    int i;
    ull ha = OULL;
    For(i, n) ha = ha * B + (ull)a[i];
    return ha;
}

// 注意pos一定不能是0!!!!
inline ull get_hash(ull *Ha, int pos,
                    int l) {  // 返回Ha[pos...pos+l-1]的值，pos与l必须是正数
    return Ha[pos + l - 1] - Ha[pos - 1] * bp[l];
}

inline ull merge_hash(ull ha1, ull ha2, int len2) {  // 返回s1+s2拼接后的hash值
    return ha1 * bp[len2] + ha2;
}

bool contain(int ida, int idb) {  // b是否为a的子串
                                  // ，ida和idb为字符串下标，若只有两个字符串，使用时传入参数(0,
                                  // 1)、(1, 0)就行
    if (len[ida] < len[idb])
        return false;
    ull hab = ha[idb][len[idb]];
    for (int i = 1; i + len[idb] <= len[ida]; ++i)
        if (get_hash(ha[ida], i, len[idb]) == hab)
            return true;
    return false;
}

int overlap(
    int ida,
    int idb) {  // 求a后缀与b前缀的最长公共子串，ida和idb为字符串下标，若只有两个字符串，使用时传入参数(0,1)、(1,
                // 0)就行
    int ans = 0, i;
    Forr(i, 1, min(len[ida], len[idb]) +
                   1) if (get_hash(ha[ida], len[ida] - i +
                    1, i) ==
                          get_hash(ha[idb], 1, i)) ans = i
                          ;
    // 可在if中加上 && strncmp(s[ida] + len[ida] - i + 1, s[idb] + 1, i) ==
    // 0(不过这就失去意义了，还不如双hash)
    return ans;
}

#include <cstdio>
#include <cstring>
#include <algorithm>
#define MAXL 222
typedef unsigned long long ull;
using namespace std;
const ull B = 100000007ULL;
char s[MAXL];
int len;
ull ha[MAXL], bp[MAXL] = {1ULL};
void hash() {
    ha[0] = s[0];
    for (int i = 1; i < len; ++i)
        ha[i] = ha[i - 1] * B + s[i], bp[i] = bp[i - 1] * B;
}
ull get(int pos, int l) {
    return ha[pos + l - 1] - ha[pos - 1] * bp[l];
}
bool check(int p, int l) {
    return get(p, min(l, len - p - l)) == get(p + l, min(l, len - p - l));
}
int main() {
    int k;
    scanf("%s%d", s, &k);
    len = strlen(s);
    if (k >= len) {
        printf("%d\n", 2 * ((k + len) >> 1));
        return 0;
    }
    hash();
    for (int i = (len + k) >> 1; i >= k; --i) {
        for (int j = 0; j <= len + k - 2 * i; ++j) {
            if (check(j, i)) {
                printf("%d\n", 2 * i);
                return 0;
            }
        }
    }
}
```

3.3 KMP

```
int m, n, p[10010];
char a[1000100], b[10010];
void init() {
    p[0] = -1;
    int i = 0, j = -1;
    while (i < n) {
        if (j == -1 || b[i] == b[j]) {
            ++i, ++j, p[i] = j;
        } else {
            j = p[j];
        }
    }
}
int kmp() {
    int i = 0, j = 0, ans = 0;
    while (i < m) {
        if (j == -1 || b[j] == a[i]) {
            ++i, ++j;
        } else {
            j = p[j];
        }
        if (j == n) {
            ++ans;
        }
    }
    return ans;
}
```

3.4 Manacher

```
char s[MX];
int l[MX];

// len[i] 是 i/2 为中心的最大回文长度
void palindrome(char cs[], int len[], int n) {
    for (int i = 0; i < n * 2; ++i) {
        len[i] = 0;
    }
    for (int i = 0, j = 0, k; i < n * 2; i += k, j = max(j - k, 0)) {
        while (i - j >= 0 && i + j + 1 < n * 2 &&
               cs[(i - j) / 2] == cs[(i + j + 1) / 2]) {
            j++;
        }
        len[i] = j;
        for (k = 1; i - k >= 0 && j - k >= 0 && len[i - k] != j - k; k++) {
            len[i + k] = min(len[i - k], j - k);
        }
    }
}

int main() {
    while (~scanf("%s", s)) {
        int ans = 0, sl = strlen(s);
        palindrome(s, l, sl);
        for (int i = 0; i < sl * 2; ++i) {
            ans = max(ans, l[i]);
        }
```

```
        printf("%d\n", ans);
    }
    return 0;
}
```

3.5 Trie

```
#define MAXW 100010
#define MAXL 12
const int MAXN = MAXW * MAXL;

struct node {
    int next[26];
    int cnt;  // 附加信息
} t[MAXN];

int ts;

void clear() {
    ts = 0;
    memset(t, 0, sizeof(t));
}

int insert(char s[]) {
    int len = strlen(s), p = 0;
    for (int i = 0; i < len; ++i) {
        if (!t[p].next[s[i] - 'a']) {
            t[p].next[s[i] - 'a'] = ++ts;
        }
        ++t[p].cnt;
        p = t[p].next[s[i] - 'a'];
    }
    return ++t[p].cnt;
}

int query(char s[]) {
    int len = strlen(s), p = 0;
    for (int i = 0; i < len; ++i) {
        if (!t[p].next[s[i] - 'a']) {
            return 0;
        }
        p = t[p].next[s[i] - 'a'];
    }
    return t[p].cnt;
}

int main() {
    char s[MAXL];
    int n;
    scanf("%d", &n);
    while (n--) {
        scanf("%s", s);
        insert(s);
    }
    scanf("%d", &n);
    while (n--) {
        scanf("%s", s);
        printf("%d\n", query(s));
    }
    return 0;
}
```

4 Geometry

4.1 include

```
#include <vector>
#include <list>
#include <map>
#include <set>
#include <deque>
#include <queue>
#include <stack>
#include <bitset>
#include <algorithm>
#include <functional>
#include <numeric>
#include <utility>
#include <iostream>
#include <sstream>
#include <iomanip>
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <cctype>
#include <string>
#include <cstring>
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <climits>
#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))));
```

```
    }
    point trunc(double r) {
        double l = len();
        if (!dblcmp(l))
            return *this;
        r /= l;
        return point(x * r, y * r);
    }
    point rotleft() {
        return point(-y, x);
    }
    point rotright() {
        return point(y, -x);
    }
    point rotate(point p, double angle)  //绕点p逆时针旋转angle角度
    {
        point v = this->sub(p);
        double c = cos(angle), s = sin(angle);
        return point(p.x + v.x * c - v.y * s, p.y + v.x * s + v.y * c);
    }
};
```

4.3 line

```
struct line {
    point a, b;
    line() {
    }
    line(point _a, point _b) {
        a = _a;
        b = _b;
    }
    bool operator==(line v) {
        return (a == v.a) && (b == v.b);
    }
    //倾斜角angle
    line(point p, double angle) {
        a = p;
        if (dblcmp(angle - pi / 2) == 0) {
            b = a.add(point(0, 1));
        } else {
            b = a.add(point(1, tan(angle)));
        }
    }
    // ax+by+c=0
    line(double _a, double _b, double _c) {
        if (dblcmp(_a) == 0) {
            a = point(0, -_c / _b);
            b = point(1, -_c / _b);
        } else if (dblcmp(_b) == 0) {
            a = point(-_c / _a, 0);
            b = point(-_c / _a, 1);
        } else {
            a = point(0, -_c / _b);
            b = point(1, (-_c - _a) / _b);
        }
    }
    void input() {
        a.input();
        b.input();
    }
    void adjust() {
        if (b < a)
            swap(a, b);
    }
    double length() {
        return a.distance(b);
    }
    double angle()  //直线倾斜角 0<=angle<180
    {
        double k = atan2(b.y - a.y, b.x - a.x);
        if (dblcmp(k) < 0)
            k += pi;
        if (dblcmp(k - pi) == 0)
            k -= pi;
        return k;
    }
    //点和线段关系
    // 1 在逆时针
    // 2 在顺时针
    // 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

```
struct circle {
    point p;
    double r;
    circle() {
```

```
    }
    circle(point _p, double _r) : p(_p), r(_r){};
    circle(double x, double y, double _r) : p(point(x, y)), r(_r){};
    circle(point a, point b, point c)  //三角形的外接圆
    {
        p = line(a.add(b).div(2), a.add(b).div(2).add(b.sub(a).rotleft()))
                .crosspoint(line(c.add(b).div(2),
                                 c.add(b).div(2).add(b.sub(c).rotleft())));
        r = p.distance(a);
    }
    circle(point a, point b, point c, bool t)  //三角形的内切圆
    {
        line u, v;
        double m = atan2(b.y - a.y, b.x - a.x), n = atan2(c.y - a.y, c.x - a.x);
        u.a = a;
        u.b = u.a.add(point(cos((n + m) / 2), sin((n + m) / 2)));
        v.a = b;
        m = atan2(a.y - b.y, a.x - b.x), n = atan2(c.y - b.y, c.x - b.x);
        v.b = v.a.add(point(cos((n + m) / 2), sin((n + m) / 2)));
        p = u.crosspoint(v);
        r = line(a, b).dispointtoseg(p);
    }
    void input() {
        p.input();
        scanf("%lf", &r);
    }
    void output() {
        printf("%.2lf %.2lf %.2lf\n", p.x, p.y, r);
    }
    bool operator==(circle v) {
        return ((p == v.p) && dblcmp(r - v.r) == 0);
    }
    bool operator<(circle v) const {
        return ((p < v.p) || (p == v.p) && dblcmp(r - v.r) < 0);
    }
    double area() {
        return pi * sqr(r);
    }
    double circumference() {
        return 2 * pi * r;
    }
    // 0 圆外
    // 1 圆上
    // 2 圆内
    int relation(point b) {
        double dst = b.distance(p);
        if (dblcmp(dst - r) < 0)
            return 2;
        if (dblcmp(dst - r) == 0)
            return 1;
        return 0;
    }
    int relationseg(line v) {
        double dst = v.dispointtoseg(p);
        if (dblcmp(dst - r) < 0)
            return 2;
        if (dblcmp(dst - r) == 0)
            return 1;
        return 0;
    }
    int relationline(line v) {
        double dst = v.dispointtoline(p);
        if (dblcmp(dst - r) < 0)
            return 2;
        if (dblcmp(dst - r) == 0)
            return 1;
        return 0;
    }
    //过a b两点 半径r的两个圆
    int getcircle(point a, point b, double r, circle &c1, circle &c2) {
        circle x(a, r), y(b, r);
        int t = x.pointcrosscircle(y, c1.p, c2.p);
        if (!t)
            return 0;
        c1.r = c2.r = r;
        return t;
    }
    //与直线u相切 过点q 半径r1的圆
    int getcircle(line u, point q, double r1, circle &c1, circle &c2) {
        double dis = u.dispointtoline(q);
        if (dblcmp(dis - r1 * 2) > 0)
            return 0;
        if (dblcmp(dis) == 0) {
            c1.p = q.add(u.b.sub(u.a).rotleft().trunc(r1));
            c2.p = q.add(u.b.sub(u.a).rotright().trunc(r1));
            c1.r = c2.r = r1;
            return 2;
        }
        line u1 = line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),
                       u.b.add(u.b.sub(u.a).rotleft().trunc(r1)));
        line u2 = line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),
                       u.b.add(u.b.sub(u.a).rotright().trunc(r1)));
        circle cc = circle(q, r1);
        point p1, p2;
        if (!cc.pointcrossline(u1, p1, p2))
            cc.pointcrossline(u2, p1, p2);
        c1 = circle(p1, r1);
        if (p1 == p2) {
            c2 = c1;
            return 1;
        }
        c2 = circle(p2, r1);
        return 2;
    }
    //同时与直线u,v相切 半径r1的圆
    int getcircle(line u, line v, double r1, circle &c1, circle &c2, circle &c3,
                  circle &c4) {
        if (u.parallel(v))
            return 0;
        line u1 = line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),
                       u.b.add(u.b.sub(u.a).rotleft().trunc(r1)));
        line u2 = line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),
                       u.b.add(u.b.sub(u.a).rotright().trunc(r1)));
        line v1 = line(v.a.add(v.b.sub(v.a).rotleft().trunc(r1)),
                       v.b.add(v.b.sub(v.a).rotleft().trunc(r1)));
        line v2 = line(v.a.add(v.b.sub(v.a).rotright().trunc(r1)),
                       v.b.add(v.b.sub(v.a).rotright().trunc(r1)));
        c1.r = c2.r = c3.r = c4.r = r1;
        c1.p = u1.crosspoint(v1);
        c2.p = u1.crosspoint(v2);
        c3.p = u2.crosspoint(v1);
        c4.p = u2.crosspoint(v2);
        return 4;
    }
    //同时与不相交圆cx,cy相切 半径为r1的圆
    int getcircle(circle cx, circle cy, double r1, circle &c1, circle &c2) {
        circle x(cx.p, r1 + cx.r), y(cy.p, r1 + cy.r);
        int t = x.pointcrosscircle(y, c1.p, c2.p);
        if (!t)
            return 0;
        c1.r = c2.r = r1;
        return t;
    }
    int pointcrossline(line v, point &p1,
                       point &p2)  //求与线段交要先判断 relationseg
    {
        if (!(*this).relationline(v))
            return 0;
```

```
        point a = v.lineprog(p);
        double d = v.dispointtoline(p);
        d = sqrt(r * r - d * d);
        if (dblcmp(d) == 0) {
            p1 = a;
            p2 = a;
            return 1;
        }
        p1 = a.sub(v.b.sub(v.a).trunc(d));
        p2 = a.add(v.b.sub(v.a).trunc(d));
        return 2;
    }
    // 5 相离
    // 4 外切
    // 3 相交
    // 2 内切
    // 1 内含
    int relationcircle(circle v) {
        double d = p.distance(v.p);
        if (dblcmp(d - r - v.r) > 0)
            return 5;
        if (dblcmp(d - r - v.r) == 0)
            return 4;
        double l = fabs(r - v.r);
        if (dblcmp(d - r - v.r) < 0 && dblcmp(d - l) > 0)
            return 3;
        if (dblcmp(d - l) == 0)
            return 2;
        if (dblcmp(d - l) < 0)
            return 1;
    }
    int pointcrosscircle(circle v, point &p1, point &p2) {
        int rel = relationcircle(v);
        if (rel == 1 || rel == 5)
            return 0;
        double d = p.distance(v.p);
        double l = (d + (sqr(r) - sqr(v.r)) / d) / 2;
        double h = sqrt(sqr(r) - sqr(l));
        p1 = p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).
        rotleft().trunc(h)));
        p2 = p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).
        rotright().trunc(h)));
        if (rel == 2 || rel == 4) {
            return 1;
        }
        return 2;
    }
    //过一点做圆的切线 (先判断点和圆关系)
    int tangentline(point q, line &u, line &v) {
        int x = relation(q);
        if (x == 2)
            return 0;
        if (x == 1) {
            u = line(q, q.add(q.sub(p).rotleft()));
            v = u;
            return 1;
        }
        double d = p.distance(q);
        double l = sqr(r) / d;
        double h = sqrt(sqr(r) - sqr(l));
        u = line(q, p.add(q.sub(p).trunc(l).add(q.sub(p).
        rotleft().trunc(h))));
        v = line(q, p.add(q.sub(p).trunc(l).add(q.sub(p).
        rotright().trunc(h))));
        return 2;
    }
    double areacircle(circle v) {
        int rel = relationcircle(v);
        if (rel >= 4)
            return 0.0;
        if (rel <= 2)
            return min(area(), v.area());
        double d = p.distance(v.p);
        double hf = (r + v.r + d) / 2.0;
        double ss = 2 * sqrt(hf * (hf - r) * (hf - v.r) *
        (hf - d));
        double a1 = acos((r * r + d * d - v.r * v.r) /
        (2.0 * r * d));
        a1 = a1 * r * r;
        double a2 = acos((v.r * v.r + d * d - r * r) /
        (2.0 * v.r * d));
        a2 = a2 * v.r * v.r;
        return a1 + a2 - ss;
    }
    double areatriangle(point a, point b) {
        if (dblcmp(p.sub(a).det(p.sub(b)) == 0))
            return 0.0;
        point q[5];
        int len = 0;
        q[len++] = a;
        line l(a, b);
        point p1, p2;
        if (pointcrossline(l, q[1], q[2]) == 2) {
            if (dblcmp(a.sub(q[1]).dot(b.sub(q[1]))) < 0)
                q[len++] = q[1];
            if (dblcmp(a.sub(q[2]).dot(b.sub(q[2]))) < 0)
                q[len++] = q[2];
        }
        q[len++] = b;
        if (len == 4 && (dblcmp(q[0].sub(q[1]).dot(q[2].
        sub(q[1]))) > 0))
            swap(q[1], q[2]);
        double res = 0;
        int i;
        for (i = 0; i < len - 1; i++) {
            if (relation(q[i]) == 0 || relation(q[i + 1])
            == 0) {
                double arg = p.rad(q[i], q[i + 1]);
                res += r * r * arg / 2.0;
            } else {
                res += fabs(q[i].sub(p).det(q[i + 1].sub(p
                )) / 2.0);
            }
        }
        return res;
    }
};
```

4.5 polygon

```
struct polygon {
    int n;
    point p[maxp];
    line l[maxp];
    void input() {
        n = 4;
        for (int i = 0; i < n; i++) {
            p[i].input();
        }
    }
    void add(point q) {
        p[n++] = q;
    }
    void getline() {
        for (int i = 0; i < n; i++) {
            l[i] = line(p[i], p[(i + 1) % n]);
        }
    }
    struct cmp {
        point p;
        cmp(const point &p0) {
            p = p0;
        }
        bool operator()(const point &aa, const point &bb)
        {
            point a = aa, b = bb;
            int d = dblcmp(a.sub(p).det(b.sub(p)));
            if (d == 0) {
                return dblcmp(a.distance(p) - b.distance(p
                )) < 0;
            }
            return d > 0;
        }
    };
    void norm() {
        point mi = p[0];
        for (int i = 1; i < n; i++)
            mi = min(mi, p[i]);
        sort(p, p + n, cmp(mi));
    }
    void getconvex(polygon &convex) {
        int i, j, k;
        sort(p, p + n);
        convex.n = n;
        for (i = 0; i < min(n, 2); i++) {
            convex.p[i] = p[i];
```

```
        }
        if (n <= 2)
            return;
        int &top = convex.n;
        top = 1;
        for (i = 2; i < n; i++) {
            while (top &&
                   convex.p[top].sub(p[i]).det(convex.p[
                   top - 1].sub(p[i])) <=
                       0)
                top--;
            convex.p[++top] = p[i];
        }
        int temp = top;
        convex.p[++top] = p[n - 2];
        for (i = n - 3; i >= 0; i--) {
            while (top != temp &&
                   convex.p[top].sub(p[i]).det(convex.p[
                   top - 1].sub(p[i])) <=
                       0)
                top--;
            convex.p[++top] = p[i];
        }
    }
    bool isconvex() {
        bool s[3];
        memset(s, 0, sizeof(s));
        int i, j, k;
        for (i = 0; i < n; i++) {
            j = (i + 1) % n;
            k = (j + 1) % n;
            s[dblcmp(p[j].sub(p[i]).det(p[k].sub(p[i]))) +
             1] = 1;
            if (s[0] && s[2])
                return 0;
        }
        return 1;
    }
    // 3 点上
    // 2 边上
    // 1 内部
    // 0 外部
    int relationpoint(point q) {
        int i, j;
        for (i = 0; i < n; i++) {
            if (p[i] == q)
                return 3;
        }
        getline();
        for (i = 0; i < n; i++) {
            if (l[i].pointonseg(q))
                return 2;
        }
        int cnt = 0;
        for (i = 0; i < n; i++) {
            j = (i + 1) % n;
            int k = dblcmp(q.sub(p[j]).det(p[i].sub(p[j]))
            );
            int u = dblcmp(p[i].y - q.y);
            int v = dblcmp(p[j].y - q.y);
            if (k > 0 && u < 0 && v >= 0)
                cnt++;
            if (k < 0 && v < 0 && u >= 0)
                cnt--;
        }
        return cnt != 0;
    }
    // 1 在多边形内长度为正
    // 2 相交或与边平行
    // 0 无任何交点
    int relationline(line u) {
        int i, j, k = 0;
        getline();
        for (i = 0; i < n; i++) {
            if (l[i].segcrossseg(u) == 2)
                return 1;
            if (l[i].segcrossseg(u) == 1)
                k = 1;
        }
        if (!k)
            return 0;
        vector<point> vp;
        for (i = 0; i < n; i++) {
            if (l[i].segcrossseg(u)) {
                if (l[i].parallel(u)) {
                    vp.pb(u.a);
                    vp.pb(u.b);
                    vp.pb(l[i].a);
                    vp.pb(l[i].b);
                    continue;
                }
                vp.pb(l[i].crosspoint(u));
            }
        }
        sort(vp.begin(), vp.end());
        int sz = vp.size();
        for (i = 0; i < sz - 1; i++) {
            point mid = vp[i].add(vp[i + 1]).div(2);
            if (relationpoint(mid) == 1)
                return 1;
        }
        return 2;
    }
    //直线u切割凸多边形左侧
    //注意直线方向
    void convexcut(line u, polygon &po) {
        int i, j, k;
        int &top = po.n;
        top = 0;
        for (i = 0; i < n; i++) {
            int d1 = dblcmp(p[i].sub(u.a).det(u.b.sub(u.a)
            ));
            int d2 = dblcmp(p[(i + 1) % n].sub(u.a).det(u.
            b.sub(u.a)));
            if (d1 >= 0)
                po.p[top++] = p[i];
            if (d1 * d2 < 0)
                po.p[top++] = u.crosspoint(line(p[i], p[(i
                 + 1) % n]));
        }
    }
    double getcircumference() {
        double sum = 0;
        int i;
        for (i = 0; i < n; i++) {
            sum += p[i].distance(p[(i + 1) % n]);
        }
        return sum;
    }
    double getarea() {
        double sum = 0;
        int i;
        for (i = 0; i < n; i++) {
            sum += p[i].det(p[(i + 1) % n]);
        }
        return fabs(sum) / 2;
    }
    bool getdir()  // 1代表逆时针 0代表顺时针
    {
        double sum = 0;
        int i;
        for (i = 0; i < n; i++) {
            sum += p[i].det(p[(i + 1) % n]);
        }
        if (dblcmp(sum) > 0)
            return 1;
        return 0;
    }
    point getbarycentre() {
        point ret(0, 0);
        double area = 0;
        int i;
        for (i = 1; i < n - 1; i++) {
            double tmp = p[i].sub(p[0]).det(p[i + 1].sub(p
            [0]));
            if (dblcmp(tmp) == 0)
                continue;
            area += tmp;
            ret.x += (p[0].x + p[i].x + p[i + 1].x) / 3 *
            tmp;
            ret.y += (p[0].y + p[i].y + p[i + 1].y) / 3 *
            tmp;
        }
        if (dblcmp(area))
            ret = ret.div(area);
        return ret;
```

```
    }
    double areaintersection(polygon po) {
    }
    double areaunion(polygon po) {
        return getarea() + po.getarea() - areaintersection
        (po);
    }
    double areacircle(circle c) {
        int i, j, k, l, m;
        double ans = 0;
        for (i = 0; i < n; i++) {
            int j = (i + 1) % n;
            if (dblcmp(p[j].sub(c.p).det(p[i].sub(c.p)))
            >= 0) {
                ans += c.areatriangle(p[i], p[j]);
            } else {
                ans -= c.areatriangle(p[i], p[j]);
            }
        }
        return fabs(ans);
    }
    //多边形和圆关系
    // 0 一部分在圆外
    // 1 与圆某条边相切
    // 2 完全在圆内
    int relationcircle(circle c) {
        getline();
        int i, x = 2;
        if (relationpoint(c.p) != 1)
            return 0;
        for (i = 0; i < n; i++) {
            if (c.relationseg(l[i]) == 2)
                return 0;
            if (c.relationseg(l[i]) == 1)
                x = 1;
        }
        return x;
    }
    void find(int st, point tri[], circle &c) {
        if (!st) {
            c = circle(point(0, 0), -2);
        }
        if (st == 1) {
            c = circle(tri[0], 0);
        }
        if (st == 2) {
            c = circle(tri[0].add(tri[1]).div(2),
                       tri[0].distance(tri[1]) / 2.0);
        }
        if (st == 3) {
            c = circle(tri[0], tri[1], tri[2]);
        }
    }
    void solve(int cur, int st, point tri[], circle &c) {
        find(st, tri, c);
        if (st == 3)
            return;
        int i;
        for (i = 0; i < cur; i++) {
            if (dblcmp(p[i].distance(c.p) - c.r) > 0) {
                tri[st] = p[i];
                solve(i, st + 1, tri, c);
            }
        }
    }
    circle mincircle()  //点集最小圆覆盖
    {
        random_shuffle(p, p + n);
        point tri[4];
        circle c;
        solve(n, 0, tri, c);
        return c;
    }
    int circlecover(double r)  //单位圆覆盖
    {
        int ans = 0, i, j;
        vector<pair<double, int> > v;
        for (i = 0; i < n; i++) {
            v.clear();
            for (j = 0; j < n; j++)
                if (i != j) {
                    point q = p[i].sub(p[j]);
                    double d = q.len();
                    if (dblcmp(d - 2 * r) <= 0) {
                        double arg = atan2(q.y, q.x);
                        if (dblcmp(arg) < 0)
                            arg += 2 * pi;
                        double t = acos(d / (2 * r));
                        v.push_back(make_pair(arg - t + 2
                        * pi, -1));
                        v.push_back(make_pair(arg + t + 2
                        * pi, 1));
                    }
                }
            sort(v.begin(), v.end());
            int cur = 0;
            for (j = 0; j < v.size(); j++) {
                if (v[j].second == -1)
                    ++cur;
                else
                    --cur;
                ans = max(ans, cur);
            }
        }
        return ans + 1;
    }
    int pointinpolygon(point q)  //点在凸多边形内部的判定
    {
        if (getdir())
            reverse(p, p + n);
        if (dblcmp(q.sub(p[0]).det(p[n - 1].sub(p[0]))) ==
         0) {
            if (line(p[n - 1], p[0]).pointonseg(q))
                return n - 1;
            return -1;
        }
        int low = 1, high = n - 2, mid;
        while (low <= high) {
            mid = (low + high) >> 1;
            if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0])))
            >= 0 &&
                dblcmp(q.sub(p[0]).det(p[mid + 1].sub(p
                [0]))) < 0) {
                polygon c;
                c.p[0] = p[mid];
                c.p[1] = p[mid + 1];
                c.p[2] = p[0];
                c.n = 3;
                if (c.relationpoint(q))
                    return mid;
                return -1;
            }
            if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0])))
            > 0) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        return -1;
    }
};
```

4.6 polygons

```
struct polygons {
    vector<polygon> p;
    polygons() {
        p.clear();
    }
    void clear() {
        p.clear();
    }
    void push(polygon q) {
        if (dblcmp(q.getarea()))
            p.pb(q);
    }
    vector<pair<double, int> > e;
    void ins(point s, point t, point X, int i) {
        double r = fabs(t.x - s.x) > eps ? (X.x - s.x) / (
        t.x - s.x)
                                         : (X.y - s.y) / (
                                         t.y - s.y);
        r = min(r, 1.0);
        r = max(r, 0.0);
        e.pb(mp(r, i));
```

```
    }
    double polyareaunion() {
        double ans = 0.0;
        int c0, c1, c2, i, j, k, w;
        for (i = 0; i < p.size(); i++) {
            if (p[i].getdir() == 0)
                reverse(p[i].p, p[i].p + p[i].n);
        }
        for (i = 0; i < p.size(); i++) {
            for (k = 0; k < p[i].n; k++) {
                point &s = p[i].p[k], &t = p[i].p[(k + 1)
                % p[i].n];
                if (!dblcmp(s.det(t)))
                    continue;
                e.clear();
                e.pb(mp(0.0, 1));
                e.pb(mp(1.0, -1));
                for (j = 0; j < p.size(); j++)
                    if (i != j) {
                        for (w = 0; w < p[j].n; w++) {
                            point a = p[j].p[w], b = p[j].
                            p[(w + 1) % p[j].n],
                                  c = p[j].p[(w - 1 + p[j
                                  ].n) % p[j].n];
                            c0 = dblcmp(t.sub(s).det(c.sub
                            (s)));
                            c1 = dblcmp(t.sub(s).det(a.sub
                            (s)));
                            c2 = dblcmp(t.sub(s).det(b.sub
                            (s)));
                            if (c1 * c2 < 0)
                                ins(s, t, line(s, t).
                                crosspoint(line(a, b)),
                                    -c2);
                            else if (!c1 && c0 * c2 < 0)
                                ins(s, t, a, -c2);
                            else if (!c1 && !c2) {
                                int c3 = dblcmp(t.sub(s).
                                det(
                                    p[j].p[(w + 2) % p[j].
                                    n].sub(s)));
                                int dp = dblcmp(t.sub(s).
                                dot(b.sub(a)));
                                if (dp && c0)
                                    ins(s, t, a, dp > 0
                                                     ? c0
                                                     * ((j
                                                      > i)
                                                      ^ (
                                                     c0 <
                                                     0))
                                                     : -(
                                                     c0 <
                                                     0));
                                if (dp && c3)
                                    ins(s, t, b,
                                        dp > 0 ? -c3 * ((j
                                         > i) ^ (c3 < 0))
                                               : c3 < 0);
                            }
                        }
                    }
                sort(e.begin(), e.end());
                int ct = 0;
                double tot = 0.0, last;
                for (j = 0; j < e.size(); j++) {
                    if (ct == 2)
                        tot += e[j].first - last;
                    ct += e[j].second;
                    last = e[j].first;
                }
                ans += s.det(t) * tot;
            }
        }
        return fabs(ans) * 0.5;
    }
};
```

4.7 circles

```
const int maxn = 500;
struct circles {
    circle c[maxn];
    double ans[maxn];  // ans[i]表示被覆盖了i次的面积
    double pre[maxn];
    int n;
    circles() {
    }
    void add(circle cc) {
        c[n++] = cc;
    }
    bool inner(circle x, circle y) {
        if (x.relationcircle(y) != 1)
            return 0;
        return dblcmp(x.r - y.r) <= 0 ? 1 : 0;
    }
    void init_or()  //圆的面积并去掉内含的圆
    {
        int i, j, k = 0;
        bool mark[maxn] = {0};
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++)
                if (i != j && !mark[j]) {
                    if ((c[i] == c[j]) || inner(c[i], c[j
                    ]))
                        break;
                }
            if (j < n)
                mark[i] = 1;
        }
        for (i = 0; i < n; i++)
            if (!mark[i])
                c[k++] = c[i];
        n = k;
    }
    void init_and()  //圆的面积交去掉内含的圆
    {
        int i, j, k = 0;
        bool mark[maxn] = {0};
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++)
                if (i != j && !mark[j]) {
                    if ((c[i] == c[j]) || inner(c[j], c[i
                    ]))
                        break;
                }
            if (j < n)
                mark[i] = 1;
        }
        for (i = 0; i < n; i++)
            if (!mark[i])
                c[k++] = c[i];
        n = k;
    }
    double areaarc(double th, double r) {
        return 0.5 * sqr(r) * (th - sin(th));
    }
    void getarea() {
        int i, j, k;
        memset(ans, 0, sizeof(ans));
        vector<pair<double, int> > v;
        for (i = 0; i < n; i++) {
            v.clear();
            v.push_back(make_pair(-pi, 1));
            v.push_back(make_pair(pi, -1));
            for (j = 0; j < n; j++)
                if (i != j) {
                    point q = c[j].p.sub(c[i].p);
                    double ab = q.len(), ac = c[i].r, bc =
                     c[j].r;
                    if (dblcmp(ab + ac - bc) <= 0) {
                        v.push_back(make_pair(-pi, 1));
                        v.push_back(make_pair(pi, -1));
                        continue;
                    }
                    if (dblcmp(ab + bc - ac) <= 0)
                        continue;
                    if (dblcmp(ab - ac - bc) > 0)
                        continue;
                    double th = atan2(q.y, q.x),
                           fai = acos((ac * ac + ab * ab -
                            bc * bc) /
                                      (2.0 * ac * ab));
                    double a0 = th - fai;
                    if (dblcmp(a0 + pi) < 0)
                        a0 += 2 * pi;
```

```
                    double a1 = th + fai;
                    if (dblcmp(a1 - pi) > 0)
                        a1 -= 2 * pi;
                    if (dblcmp(a0 - a1) > 0) {
                        v.push_back(make_pair(a0, 1));
                        v.push_back(make_pair(pi, -1));
                        v.push_back(make_pair(-pi, 1));
                        v.push_back(make_pair(a1, -1));
                    } else {
                        v.push_back(make_pair(a0, 1));
                        v.push_back(make_pair(a1, -1));
                    }
                }
            sort(v.begin(), v.end());
            int cur = 0;
            for (j = 0; j < v.size(); j++) {
                if (cur && dblcmp(v[j].first - pre[cur]))
                {
                    ans[cur] += areaarc(v[j].first - pre[
                    cur], c[i].r);
                    ans[cur] +=
                        0.5 *
                        point(c[i].p.x + c[i].r * cos(pre[
                        cur]),
                              c[i].p.y + c[i].r * sin(pre[
                              cur]))
                            .det(point(c[i].p.x + c[i].r *
                             cos(v[j].first),
                                       c[i].p.y + c[i].r *
                                        sin(v[j].first)));
                }
                cur += v[j].second;
                pre[cur] = v[j].first;
            }
        }
        for (i = 1; i <= n; i++) {
            ans[i] -= ans[i + 1];
        }
    }
};
```

4.8 halfplane

```
struct halfplane : public line {
    double angle;
    halfplane() {
    }
    //表示向量 a->b逆时针(左侧)的半平面
    halfplane(point _a, point _b) {
        a = _a;
        b = _b;
    }
    halfplane(line v) {
        a = v.a;
        b = v.b;
    }
    void calcangle() {
        angle = atan2(b.y - a.y, b.x - a.x);
    }
    bool operator<(const halfplane &b) const {
        return angle < b.angle;
    }
};
struct halfplanes {
    int n;
    halfplane hp[maxp];
    point p[maxp];
    int que[maxp];
    int st, ed;
    void push(halfplane tmp) {
        hp[n++] = tmp;
    }
    void unique() {
        int m = 1, i;
        for (i = 1; i < n; i++) {
            if (dblcmp(hp[i].angle - hp[i - 1].angle))
                hp[m++] = hp[i];
            else if (dblcmp(hp[m - 1]
                                .b.sub(hp[m - 1].a)
                                .det(hp[i].a.sub(hp[m -
                                1].a)) > 0))
                hp[m - 1] = hp[i];
        }
        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)
            ed--;
        while (st < ed &&
               dblcmp(hp[que[ed]]
                          .b.sub(hp[que[ed]].a)
                          .det(p[st + 1].sub(hp[que[ed]].a
                          ))) < 0)
            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

```
struct point3 {
    double x, y, z;
    point3() {
    }
    point3(double _x, double _y, double _z) : x(_x), y(_y)
    , z(_z){};
    void input() {
        scanf("%lf%lf%lf", &x, &y, &z);
    }
    void output() {
        printf("%.2lf %.2lf %.2lf\n", x, y, z);
    }
    bool operator==(point3 a) {
        return dblcmp(a.x - x) == 0 && dblcmp(a.y - y) ==
        0 &&
               dblcmp(a.z - z) == 0;
    }
    bool operator<(point3 a) const {
        return dblcmp(a.x - x) == 0
                   ? dblcmp(y - a.y) == 0 ? dblcmp(z - a.z
                   ) < 0 : y < a.y
                   : x < a.x;
    }
    double len() {
        return sqrt(len2());
    }
    double len2() {
        return x * x + y * y + z * z;
    }
```

```
    double distance(point3 p) {
        return sqrt((p.x − x) * (p.x − x) + (p.y − y) * (p.y − y) +
                    (p.z − z) * (p.z − z));
    }
    point3 add(point3 p) {
        return point3(x + p.x, y + p.y, z + p.z);
    }
    point3 sub(point3 p) {
        return point3(x − p.x, y − p.y, z − p.z);
    }
    point3 mul(double d) {
        return point3(x * d, y * d, z * d);
    }
    point3 div(double d) {
        return point3(x / d, y / d, z / d);
    }
    double dot(point3 p) {
        return x * p.x + y * p.y + z * p.z;
    }
    point3 det(point3 p) {
        return point3(y * p.z − p.y * z, p.x * z − x * p.z, x * p.y − p.x * y);
    }
    double rad(point3 a, point3 b) {
        point3 p = (*this);
        return acos(a.sub(p).dot(b.sub(p)) / (a.distance(p) * b.distance(p)));
    }
    point3 trunc(double r) {
        r /= len();
        return point3(x * r, y * r, z * r);
    }
    point3 rotate(point3 o, double r) {
    }
};
```

4.10 line3

```
struct line3 {
    point3 a, b;
    line3() {
    }
    line3(point3 _a, point3 _b) {
        a = _a;
        b = _b;
    }
    bool operator==(line3 v) {
        return (a == v.a) && (b == v.b);
    }
    void input() {
        a.input();
        b.input();
    }
    double length() {
        return a.distance(b);
    }
    bool pointonseg(point3 p) {
        return dblcmp(p.sub(a).det(p.sub(b)).len()) == 0 &&
               dblcmp(a.sub(p).dot(b.sub(p))) <= 0;
    }
    double dispointtoline(point3 p) {
        return b.sub(a).det(p.sub(a)).len() / a.distance(b);
    }
    double dispointtoseg(point3 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);
    }
    point3 lineprog(point3 p) {
        return a.add(b.sub(a).trunc(b.sub(a).dot(p.sub(a)) / b.distance(a)));
    }
    point3 rotate(point3 p, double ang)  // p绕此向量逆时针arg角度
    {
        if (dblcmp((p.sub(a).det(p.sub(b)).len())) == 0)
            return p;
        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

```
struct plane {
    point3 a, b, c, o;
    plane() {
    }
    plane(point3 _a, point3 _b, point3 _c) {
        a = _a;
        b = _b;
        c = _c;
        o = pvec();
    }
    plane(double _a, double _b, double _c, double _d) {
        // ax+by+cz+d=0
        o = point3(_a, _b, _c);
        if (dblcmp(_a) != 0) {
            a = point3((−_d − _c − _b) / _a, 1, 1);
        } else if (dblcmp(_b) != 0) {
            a = point3(1, (−_d − _c − _a) / _b, 1);
        } else if (dblcmp(_c) != 0) {
            a = point3(1, 1, (−_d − _a − _b) / _c);
        }
    }
    void input() {
        a.input();
        b.input();
        c.input();
        o = pvec();
    }
    point3 pvec() {
        return b.sub(a).det(c.sub(a));
    }
    bool pointonplane(point3 p)  //点是否在平面上
    {
        return dblcmp(p.sub(a).dot(o)) == 0;
    }
    // 0 不在
    // 1 在边界上
    // 2 在内部
    int pointontriangle(point3 p)  //点是否在空间三角形abc上
    {
        if (!pointonplane(p))
            return 0;
        double s = a.sub(b).det(c.sub(b)).len();
        double s1 = p.sub(a).det(p.sub(b)).len();
        double s2 = p.sub(a).det(p.sub(c)).len();
        double s3 = p.sub(b).det(p.sub(c)).len();
        if (dblcmp(s − s1 − s2 − s3))
            return 0;
        if (dblcmp(s1) && dblcmp(s2) && dblcmp(s3))
            return 2;
        return 1;
    }
    //判断两平面关系
    // 0 相交
    // 1 平行但不重合
    // 2 重合
    bool relationplane(plane f) {
        if (dblcmp(o.det(f.o).len()))
            return 0;
        if (pointonplane(f.a))
            return 2;
        return 1;
    }
    double angleplane(plane f)  //两平面夹角
    {
        return acos(o.dot(f.o) / (o.len() * f.o.len()));
    }
    double dispoint(point3 p)  //点到平面距离
    {
```

```
        return fabs(p.sub(a).dot(o) / o.len());
    }
    point3 pttoplane(point3 p)  //点到平面最近点
    {
        line3 u = line3(p, p.add(o));
        crossline(u, p);
        return p;
    }
    int crossline(line3 u, point3 &p)  //平面和直线的交点
    {
        double x = o.dot(u.b.sub(a));
        double y = o.dot(u.a.sub(a));
        double d = x - y;
        if (dblcmp(fabs(d)) == 0)
            return 0;
        p = u.a.mul(x).sub(u.b.mul(y)).div(d);
        return 1;
    }
    int crossplane(plane f,line3 &u)//平面和平面的交线
    {
        point3 oo=o.det(f.o);
        point3 v=o.det(oo);
        double d=fabs(f.o.dot(v));
        if (dblcmp(d)==0)return 0;
        point3 q=a.add(v.mul(f.o.dot(f.a.sub(a))/d));
        u=line3(q,q.add(oo));
        return 1;
    }
};
```

5 Number Theory

5.1 GCD

```
int gcd(int a, int b)
{
    return (b ? gcd(b, a % b) : a);
}

int gcd(int a, int b)
{
    int t = 1, c, d;
    while(a != b)
    {
        if(a < b) swap(a, b);
        if(!(a & 1))
        {
            a >>= 1;
            c = 1;
        }
        else c = 0;
        if(!(b & 1))
        {
            b >>= 1;
            d = 1;
        }
        else d = 0;
        if(c && d) t <<= 1;
        else if(!c && !d) a -= b;
    }
    return t * a;
}
```

5.2 Prime

```
// TODO
bool prime[MAXN];
void findprime() {
    for (int i = 0; i < MAXN; i++)
        prime[i] = true;

    prime[0] = false;
    prime[1] = false;

    for (int i = 2; i < MAXN; i++)
        if (prime[i])
            for (int j = i * i; j < MAXN; j += i)
                prime[j] = false;
}
int main() {
    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 <cstdio>
#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];
}
```

```
int find(int x) {
    return p[x] == x ? x : p[x] = find(p[x]);
}
void kruskal(int n, int m) {
    int ans = 0;
    for (int i = 0; i <= n; ++i)
        p[i] = i;
    for (int i = 0; i <= m; ++i)
        r[i] = i;
    sort(r, r + m, cmp);
    for (int i = 0; i < m; ++i) {
        int e = r[i], x = find(u[e]), y = find(v[e]);
        if (x != y) {
            ans += w[e];
            p[x] = y;
        }
    }
    cout << ans << endl;
}
int main() {
    int n;
    while (cin >> n && n) {
        int m = 0;
        for (int i = 1; i <= n; ++i) {
            for (int j = 1; j <= n; ++j) {
                cin >> w[m];
                u[m] = i;
                v[m] = j;
                ++m;
            }
        }
        kruskal(n, m);
    }
    return 0;
}
```

```
int lis[MAXN];
int lis() {
    int n, i, j, x, len = 0;
    for (i = 1; i <= n; ++i) {
        scanf("%d", &x);  // 或者 x = num[i]
        j = lower_bound(lis + 1, lis + len + 1, x) - lis;
        // LDS: j = lower_bound(lds + 1, lds + len + 1, x,
         greater<int>()) - lds;
        lis[j] = x;
        len = max(len, j);
    }
    return len;
}
// 打印路径: 使用 pos 数组, 从后往前扫, (len--)的第一次出现就是答案
// LIS[i] = max{1 , LIS[k] + 1}  (∀ k < i, arr[i] > arr[k])
/*
Longest Not-decrease Sequence:
bool cmp(int a, int b) {
    return a <= b;
}
j = lower_bound(lis + 1, lis + len + 1, x, cmp) - lis;
*/
```

7 DP

7.1 LCS

```
const int MX = 1000;

char a[MX], b[MX];
int dp[MX][MX], path[MX][MX];

int Lcs(char x[], char y[]) {
    int i, j, len1 = strlen(x + 1), len2 = strlen(y + 1);
    memset(dp, 0, sizeof(dp));
    for (i = 1; i <= len1; ++i)
        for (j = 1; j <= len2; ++j) {
            if (x[i] == y[j])
                dp[i][j] = dp[i - 1][j - 1] + 1, path[i][j
                ] = 1;
            else if (dp[i - 1][j] >= dp[i][j - 1])
                dp[i][j] = dp[i - 1][j], path[i][j] = 2;
            else
                dp[i][j] = dp[i][j - 1], path[i][j] = 3;
        }
    return dp[len1][len2];
}

void PrintLcs(int i, int j) {
    if (i == 0 || j == 0) return;
    if (path[i][j] == 1) {
        PrintLcs(i - 1, j - 1);
        putchar(a[i]);
    } else if (path[i][j] == 2) PrintLcs(i - 1, j);
    else PrintLcs(i, j - 1);
}

int main() {
    while (gets(a + 1)) {
        gets(b + 1);
        printf("%d\n", Lcs(a, b));
        PrintLcs(strlen(a + 1), strlen(b + 1));
        putchar(10);
    }
    return 0;
}
```

7.2 LIS