2018级软件学院算法习题整理

by aikx 如有疑问请联系1016194674@qq.com

C1

A C1-冒泡排序

时间限制：300ms 内存限制：65536kb

题面

使用冒泡排序将一个序列由小到大排序，请问这个过程中，序列中的元素一共交换了多少次?

输入

第一行一个数n，表示序列的长度。(1≤n≤1e5)

接下来一行，n个整数，保证在int范围内

输出

一行一个数，表示冒泡排序的交换次数

输入样例

```
5
1 2 3 4 5
```

输出样例

```
0
```

```cpp
#include <cstdio>
#include <iostream>
#include <algorithm>
using namespace std;
int a[1000001], b[1000001];
void Merge(int a[], int l, int m, int r, int tmp[])
{
    int p = 0, p1 = l, p2 = m + 1;
    while (p1 <= m && p2 <= r)
    {
        if (a[p1] > a[p2])
            tmp[p++] = a[p1++];
        else
            tmp[p++] = a[p2++];
    }

    while (p1 <= m)
        tmp[p++] = a[p1++];
```

```
    while (p2 <= m)
        tmp[p++] = a[p2++];

    for (int i = 0; i < p; i++)
        a[i + l] = tmp[i];
}

long long cal(int a[], int l, int m, int r, int tmp[])
{
    int i = l, j = m + 1;
    long long result = 0;
    while (i <= m && j <= r)
    {
        if (a[i] > a[j])
        {
            result += r - j + 1;
            i++;
        }
        else
            j++;
    }
    return result;
}

long long Merge_sort(int a[], int l, int r, int tmp[])
{
    long long cnt = 0;
    if (r > l)
    {
        int m = l + (r - l) / 2;
        cnt += Merge_sort(a, l, m, tmp);
        cnt += Merge_sort(a, m + 1, r, tmp);
        cnt += cal(a, l, m, r, tmp);
        Merge(a, l, m, r, tmp);
    }
    return cnt;
}

int main()
{
    long long ans = 0;
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
        scanf("%d", &a[i]);
    ans = Merge_sort(a, 0, n - 1, b);
    printf("%lld", ans);
    return 0;
}
```

B C1-Half

题面

有n个数。每次操作可以使一个数的值减半（除以2向下舍入）。请问k次操作后，这些数的和最小为多少。

输入

多组输入数据

每组数据第一行两个数n,k（1≤n≤1e5,1≤k≤1e5）

接下来一行n个数aiai(1≤ai≤1e9)

输出

每组数据输出一行，一个整数

输入样例

```
1 1
3
```

输出样例

```
1
```

```cpp
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int n;
    int k;
    while(~scanf("%d%d", &n, &k))
    {
        long long ans = 0;
        priority_queue<long long>q;
        for(int i = 0; i < n; i++)
        {
            long long x;
            scanf("%lld", &x);
            q.push(x);
        }
        for(int i = 0; i < k; i++)
        {
            long long tmp = q.top() / 2;
            q.pop();
            q.push(tmp);
        }
        while(!q.empty())
        {
            ans += q.top();
            q.pop();
        }
        printf("%lld\n", ans);
    }
    return 0;
}
```

C C1-pair

题面

c++中有一个非常方便的结构体pair<type,type>

现在我们定义一个合法的，只有pair和int构成的pair如下：

- 当且仅当type为int或者pair<type,type>时，pair<type,type> 合法

例如pair<int,int>和pair<pair<int,int>,int>是一个合法的pair

现在给你多个pair和int字符串，按照给出pair和int的顺序,添加'<' , '>' , ','这三个符号,使得给出的串成为一个合法的pair<type,type>。

如果不行，输出Error occurred

输入

第一行一个数字t表示数据组数(1≤t≤100)

每组数据两行，第一行一个数字n表示字符串个数，(0≤n≤1000),第二行n个字符串，只会是pair或者int

输出

一行一个字符串

输入样例1

```
3
5
pair pair int int int
2
pair int
1
int
```

输出样例1

```
pair<pair<int,int>,int>
Error occurred
Error occurred
```

```cpp
#include<iostream>
#include<cstdio>
using namespace std;
int num;//输入字符串移动下标
int endnum;//输出字符串移动下标
string endString[3000];//输出字符串
//只能前面定义后面防止代码，因为涉及互相调用
bool checkT(string *a);//type判断函数
bool checkP(string *a);//pair判断函数
int main()
{
    int t;
    cin >> t;
    while(t--)
    {
        int n;
        cin >> n;
```

```
        string temp;
        string *a = new string[4000];
        int i = 0;
        for(int i =0;i<n;i++)
        {
            cin >> temp;
            a[i] = temp;
            //这里是一次传入一个字符串，如利用char数组可以传入一个P或者I这种首字母
        }
        num = 0;
        endnum = 0;
        if(checkP(a))
        {
            if(num == n-1)//判断一个pair后是否还有内容
            {
                for(int i = 0;i<endnum;i++)
                {
                    cout << endString[i];
                }
            }
            else
            {
                cout << "Error occurred";
            }
        }
        else
        {
            cout << "Error occurred";
        }
        cout << endl;
        delete [] a;
    }
    return 0;
}
bool checkT(string *a)//type判断函数
{
    if(a[num] == "int")
    {
        endString[endnum++] = "int";
        return true;
    }
    if(a[num] == "pair")
    {
        if(checkP(a))
        {
            return true;
        }
    }
    return false;
}
bool checkP(string *a)//pair判断函数
{
    if(a[num] == "pair")
    {
        num++;//采取的是调用函数前移动下标num的形式
        endString[endnum++] = "pair";
        endString[endnum++] = "<";
        //输出字符串的增加直接endnum++个人感觉是最简便的形式
```

```
        if(checkT(a))
        {
            num++;
            endString[endnum++] = ",";
            if(checkT(a))
            {
                endString[endnum++] = ">";
                return true;
            }
        }
    }
    return false;
}
```

D C1-希尔伯特曲线

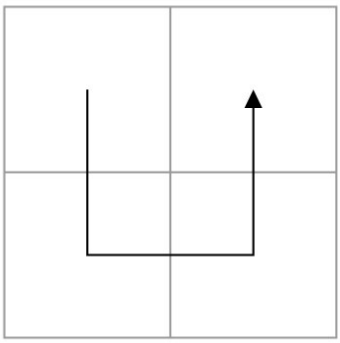
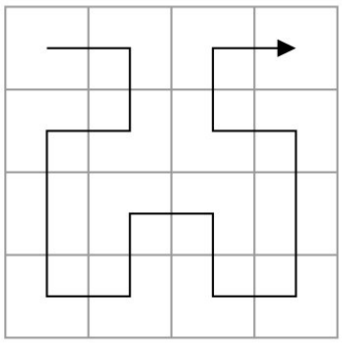
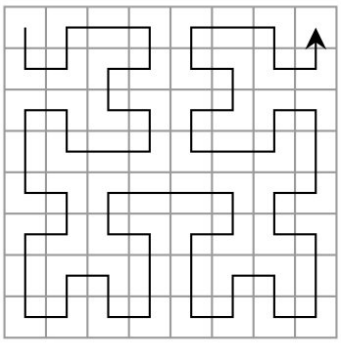
题面

由德国数学家戴维·希尔伯特（David Hilbert）发明的希尔伯特曲线是最著名的分形曲线之一。

希尔伯特曲线的形式定义是递归的。第i个希尔伯特曲线可以通过以下方式连接四个（i-1）个希尔伯特曲线来形成：

1. 将网格划分为四个象限；
2. 在主对角线上（从左上方到右下方）反转第（i-1）个希尔伯特曲线，并将其放置在左上象限中；
3. 将第（i-1）个希尔伯特曲线的两个副本分别放在左下象限和右下象限中；
4. 跨次对角线（从右上到左下）反转第（i-1）个希尔伯特曲线，并将其放置在右上象限中。

前三个希尔伯特曲线如下所示。

现在沿希尔伯特曲线对点进行编号，第i个希尔伯特曲线上的点依次编号为1~22i1~22i

那么，请问第n个希尔伯特曲线上的两点a,b之间的距离的平方。（所有点都在方格中心，方格边长为1）

输入

多组输入数据。

每组一行，三个数字n,a,b，分别表示第n个希尔伯特曲线上的两点a,b的编号（$1 \le n \le 31, 1 \le a, b \le 2^{2n}$）

输出

每行一个整数，表示ab两点之间的距离。

输入样例

```
1 1 4
```

输出样例

```
1
```

```
#include<algorithm>
#include<iostream>
using namespace std;
typedef long long ll;
typedef pair<ll, ll> point;
ll dis(const point&x, const point&y)
{
    return(x.first - y.first)*(x.first - y.first) + (x.second - y.second)*(x.second - y.second);
}
point cal(ll x, int k)
{
    if (k == 1)
    {
     if (x == 1)
        return { 1,1 };
     if (x == 2)
        return { 1,2 };
     if (x == 3)
        return { 2,2 };
     if (x == 4)
        return { 2,1 };
    }
    ll p = 1ll << (k - 1);
        point res;
    if (x <= p * p)
    {
        res = cal(x, k - 1);
        swap(res.first, res.second);
        return res;
    }
    if (x <= 2 * p * p)
    {
        res = cal(x - p * p, k - 1);
        res.second += p;
        return res;
    }
    if (x <= 3 * p * p)
    {
        res = cal(x - 2 * p * p, k - 1);
        res.first += p; res.second += p;
        return res;
    }
    res = cal(x - 3 * p * p, k - 1);
    swap(res.first, res.second);
    res.second = p - res.second + 1;
    res.first = p - res.first + 1;
    res.first += p; return res;
}
int main()
```

```
{
    int n;
    ll a, b;
    point A, B;
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    while (cin >> n >> a >> b)
    {
        A = cal(a, n);
        B = cal(b, n);
        cout << dis(A, B) << "\n";
    }
    return 0;
}
```

E C1-点灯

时间限制：300ms 内存限制：65536kb

题面

有n个灯，编号0~n-10~n-1，一开始都是关闭状态。

每次操作会拨动一个区间[L,R]灯的开关，也就是说，对于灯i，L≤i≤R，如果i是关闭状态，则操作会使灯亮，反之会使灯灭。

请问k次操作后有多少灯亮着。

输入

多组输入数据

每组数据第一行两个数n,k（1≤n≤1e9,1≤k≤1e5）

接下来k行，每行两个数l,r（0≤l≤r≤n-1）

输出

每组数据一行一个数，表示最后灯亮的个数

输入样例

```
10 1
2 6
```

输出样例

```
5
```

```
#include <cstdio>
#include <iostream>
#include <algorithm>
using namespace std;
typedef long long ll;
int a[201000];
int main()
{
```

```
    int t;
    int n, k;
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    while (cin >> n >> k) {
        int cnt = 0;
        for (int i = 0; i < k; ++i) {
            int l, r;
            cin >> l >> r;
            a[cnt++] = l;
            a[cnt++] = r + 1;
        }
        sort(a, a + cnt);
        int res = 0;
        for (int i = 1; i < cnt; i += 2) {
            res += a[i] - a[i - 1];
        }
        cout << res << "\n";
    }
    return 0;
}
```

F C1-Zexal的过河

时间限制：500ms 内存限制：65536kb

题目描述

Zexal打算借助河中间的石砖**过到河对岸去**。Zexal从**第一块石砖出发**，接下来他可以走到第二块石砖或第三块石砖，有时候走的很不爽，甚至可以直接跨过两个石砖，到达第四块石砖，但是**不能连续两次这种操作**，因为这样消耗体能比较大。现在假设河中含nn块石砖，且这些石砖呈直线分布，请你计算出Zexal**从第一块石砖出发**有多少种**安全**的过河方法。

输入

输入将由多组测试数据组成，以**EOF**结尾。

每组数据只有一行，为河中的总石砖数n（$0<n\leq 50$）。

输出

对于每组数据，输出一行，为过河的方法数。

输入样例

```
1
2
3
```

输出样例

```
1
2
4
```

样例解释

1：一步走完；
2：先走到2再走完，或者直接走完；
3：111或12或21或3。

```cpp
#include<bits/stdc++.h>
using namespace std;
long long f[52] = {1, 1, 2, 4};
void F(int n)
{
    if(n <= 3) return;
    else
    {
        f[n] = f[n-1] + f[n-2] + f[n-4] + f[n-5];
        //printf("%lld\n", f[n]);
    }
}
int main()
{
    int n;
    for(int i = 4; i < 51; i++)
        F(i);
    while(~scanf("%d", &n))
    {
        printf("%lld\n", f[n]);
    }

    return 0;
}
```

E1

A E1-位运算

时间限制：1000ms 内存限制：65536kb

题面

给出一个小于232232的正整数a，将其低16位和高16位交换后输出（用十进制表示的正整数）。

输入

第一行一个正整数t,表示数据组数(0<t<100) 接下来t行，每行一个正整数a(0<a<232)

输出

t行，每行一个正整数

输入样例

```
2
1
1314520
```

输出样例

```
65536
249036820
```

样例解释

1用二进制表示为0000 0000 0000 0000(高16位) 0000 0000 0000 0001(低16位) 交换后，0000 0000 0000 0001 0000 0000 0000 0000 值为65536

```cpp
#include<cstdio>
int main()
{
    unsigned int a;
    int k;
    scanf("%d", &k);
    while(k--)
    {
        scanf("%u", &a);
        printf("%u\n", (a<<16) | (a>>16));
    }

}
```

B E1-前缀和

时间限制：1000ms 内存限制：65536kb

知识点

前缀和是一种较为常见的预处理方式，能大大降低查询的时间复杂度。我们可以简单理解为“数列的前 n 项的和”。

对于一个给定的序列A，其前缀和S的定义如下：

$S[i] = \sum_{j=1}^{i} A[j]$

预处理出来前缀和序列S后我们就可以O(1)的查询区间和

$sum(l, r) = S[r] - S[l-1]$

题面

给定一个长度为n的整数序列A，问连续k个整数的和的最大值。

输入

第一行两个正整数n,k,(0<n<1e6,0<k<n)

第二行n个整数，表示序列A，(−1e6<Ai<1e6)

输出

一行一个整数

输入样例

```
5 3
2 5 -4 10 3
```

输出样例

```
11
```

```cpp
#include<cstdio>
long long a[1000001];
int main()
{
    int n, k, tmp;

    long long max = -9223372036854775808;
    scanf("%d%d", &n, &k);
    for(int i = 0; i < n; i++)
    {
        scanf("%d", &tmp);
        if(i > 0) a[i] = a[i - 1] + tmp;
        else a[i] = tmp;
    }
    if(n == 1)
    {
        printf("%d", tmp);
        return 0;
    }
    for(int i = k, j = 0; i < n; i++, j++)
    {
        max = max > a[i] - a[j] ? max : a[i] - a[j];
    }
    printf("%lld", max);
}
```

C E1-差分

时间限制：1000ms 内存限制：65536kb

知识点

差分，是一种和前缀和相对的策略。

对于一个给定的序列A，其差分序列S的定义如下：

$$B[i] = A[i] - A[i-1]$$

易得对这个B序列做一遍前缀和就得到了原来的A序列。

差分可以帮助我们将原序列上的区间操作转换为单点操作，降低复杂度。

譬如使$A[l,r]$每个数加上一个d，可以转换操作为$B[l]+d, B[r+1]-d$

题面

有n个长度为1的木板，编号依次为1-n，在木板上执行k次操作，每次将编号为x到编号为x+y-1的木板染上一种颜色，每次操作颜色不同。问k次操作后，一个木板最多被染几次色。

输入

第一行两个正整数n, k$(0 < n < 10^6, 0 < k < 10^5)$

接下来k行，每行两个正整数x, y$(1 \le x \le n, 1 \le y \le n)$

输出

一行一个整数

输入样例

```
5 3
1 3
1 4
1 2
```

输出样例

```
3
```

```cpp
#include<cstdio>
int a[1000001];
int main()
{
    int n, k, max = -9999999;
    scanf("%d%d", &n, &k);
    for(int i = 0; i < k; i++)
    {
        int x, y;
        scanf("%d%d", &x, &y);
        a[x]++;
        a[x+y]--;
    }
    for(int i = 1; i <= n; i++)
    {
        a[i] = a[i] + a[i-1];
        max = max > a[i] ? max : a[i];
    }
    printf("%d", max);
}
```

D E1-递归

时间限制：1000ms 内存限制：65536kb

知识点

递归大家应该都会吧

题面

从1~n中选取任意多（大于0）个数字，输出所有可能的选择方案

输入

一行一个整数n(1=<n<=10)

输出

多行，每行一种方案

同一行内的数必须升序排列，相邻两个数用恰好1个空格隔开。

方案按照字典序由小到大输出。

输入样例

```
3
```

输出样例

```
1
1 2
1 2 3
1 3
2
2 3
3
```

```
#include<iostream>
#include<cstdio>
#include<vector>
using namespace std;
vector<int> num;
int n;

void f(int top, int now)
{
    if(now > n) return;


    num.push_back(now);
    if(num.size() >= top)
    {
        for(int i = 0; i < num.size(); i++)
            printf("%d ", num[i]);
        printf("\n");
    }
    f(top + 1, now + 1);
    num.pop_back();
    f(top, now + 1);

}

int main()
{
    scanf("%d", &n);
    f(1, 1);
}
```

E E1-分治

求逆序对 代码同C1-冒泡排序

F E1-斐波那契数列

输出斐波那契数列的第n项，不再赘述

G E1-等比数列求和

时间限制：200ms 内存限制：65536kb

题面

已知$a_i = a_1 \times q^{i-1}$，求$\sum_{i=1}^{n} a_i$

结果可能很大，请对987654323取模

快速幂 快速模

输入

第一行一个正整数t,表示数据组数

接下来t行，每行三个整数n,a1,qn,a1,q 。

(0<n,a1,q<1e9,0<t<10000)

输出

t行，每行输出一个整数，表示等比数列的和 mod 987654323的值。

输入样例

```
2
3 2 7
3 2 1
```

输出样例

```
114
6
```

```cpp
#include<cstdio>
#include<iostream>
#include<cmath>
#define ll long long
using namespace std;

ll quick(ll a, ll b, ll c)
{
    ll ans = 1;
    a %= c;
    while(b)
    {
        if(b & 1) ans = ans * a % c;
        a = a * a % c;
        b >>= 1;
    }
    return ans % c;
}

ll sum(ll q, ll n, ll p)
```

```
{
    if(n == 0 || q == 0) return 0;
    if(n & 1) // 奇数
        return ((1 + quick(q, n / 2 + 1, p)) * sum(q, n / 2, p) % p +
quick(q, n / 2 + 1, p)) % p;
    else
        return (1 + quick(q, n / 2, p)) * sum(q, n / 2, p) % p;
}

int main()
{
    ll n, a1, p = 987654323, q;
    int t;
    scanf("%d", &t);
    while(t--)
    {
        scanf("%lld%lld%lld", &n, &a1, &q);
        long long t = n - 1;
        cout << (a1 % p * ((sum(q, t, p) + 1) % p)) % p << endl;
    }
    return 0;
}
```

H E1-妙妙趣排序

时间限制：1000ms 内存限制：65536kb

题面

本题中：

一个即将排好序的序列定义为：将这个序列去除至多一个值后，新序列是严格递增的。

一个过滤器[u,v]定义为：一个序列a经过过滤器[u,v]后，$a_u = min(a_u, a_v), a_v = max(a_u, a_v)$，其他值不变。

妙妙趣排序器由k个过滤器组成，一个序列的妙妙趣排序需要依次经过排序器的k个过滤器。

那么请问，1~n的全排列中，有几个序列经过给定的妙妙趣排序后可以变成即将排好序的序列。

输入

第一行一个正整数t表示数据组数(0<t<100)

接下来t组数据

每组数据第一行两个整数n，k (2≤n≤50,0≤k≤10)

每组数据接下来k行，每行两个整数u,v，表示一个过滤器(1≤u<v≤n)

输出

每组数据输出一行，一个整数

输入样例

```
4
4 0
4 1
```

```
1 2
4 3
1 2
2 3
1 2
4 6
1 2
2 3
1 2
3 4
2 3
1 2
```

输出样例

```
10
14
24
24
```

```cpp
#include<cstdio>
#include<iostream>
#include<vector>
#include<cstdlib>
#include<cstring>
using namespace std;
long long ans;
int a[2600][51];
long long cnt = 1;
void create(int n)
{
    for(int i = 0; i < n; i++)
    {
        int tmp = a[0][i]; // 枚举第i个数进行插入
        for(int j = 0; j < n; j++) // 枚举插入位置
        {
            if(j == i || j == i - 1) // 除了第一个数以外，剩下的绕过原位和前一个数的左侧
                continue;
            a[cnt][j] = tmp;
            for(int k = 0, m = 0; k < n && m < n; k++, m++)
            {
                if(m == j) m++;//已填上
                if(k == i) k++;//已使用
                a[cnt][m] = a[0][k];
            }
            cnt++;
        }
    }
}
void Find(int now, int u[], int v[], int i, int is_change)
{
    if(i < 0)
    {
        ans++;
        return;
```

```
    }
    if(a[now][u[i] - 1] > a[now][v[i] - 1]) return;
    int tmpu = a[now][u[i] - 1], tmpv = a[now][v[i] - 1];
    Find(now, u, v, i - 1, 0);

    a[now][u[i] - 1] = tmpv, a[now][v[i] - 1] = tmpu;
    Find(now, u, v, i - 1, 1);
    a[now][u[i] - 1] = tmpu, a[now][v[i] - 1] = tmpv;
}
int main()
{
    int n, k, t;
    scanf("%d", &t);
    while(t--)
    {
        scanf("%d%d", &n, &k);
        int u[51] = {0}, v[51] = {0};
        for(int i = 0; i < n; i++)
            a[0][i] = i + 1; // base
        create(n);

        for(int i = 0; i < k; i++)
        {
            scanf("%d%d", &u[i], &v[i]);
        }
        //printf("cnt = %d\n", cnt);
        for(int now = 0; now < cnt; now++)
            Find(now, u, v, k - 1, 0);
        if(k == 0) printf("%lld\n", cnt);
        else printf("%lld\n", ans);
        ans = 0;
        cnt = 1;
        memset(a, 0, sizeof(a));
    }
    return 0;
}
```

C2

A C2-妙妙趣和值

时间限制：1000ms 内存限制：65536kb

题面

定义一个长度为k的数组$a_1, a_2, \cdots, a_k$的妙妙趣值是$\min\limits_{1\le i<j\le k}|a_i - a_j|$

那么请问对于一个长度为n的`序列`，所有长度为k的`子序列`的妙妙趣值和为多少。

某个序列的`子序列`是从最初序列通过去除某些元素但不破坏余下元素的相对位置（在前或在后）而形成的新序列。

答案为`100000007`取模

输入

第一行两个数n, k，表示序列的长度，子序列的长度。$(2 \le k \le n \le 1000)$

接下来一行，n个整数$a_1, a_2, \cdots, a_n, (0 \leq a_i \leq 10^5)$

输出

一行一个数，表示妙妙趣值和

输入样例

```
4 3
1 7 3 5
```

输出样例

```
8
```

样例解释

长度为3的子序列有 [1,7,3], [1,3,5], [7,3,5], [1,7,5]，每一个妙妙趣值都是2

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
typedef long double ld;
typedef long long ll;
const int mod = 100000007;
const int ms = 1005;
int n, k;
int a[ms], dp[ms][ms];
int solve(int x)
{
    dp[0][0] = 1;
    int li = 0;
    for (int i = 1; i <= n; ++i) {
        while (a[i] - a[li + 1] >= x) li++;
        dp[i][0] = dp[i - 1][0];
        for (int j = 1; j <= k && j <= i; ++j) {
            dp[i][j] = dp[i - 1][j] + dp[li][j - 1];
            dp[i][j] %= mod;
        }
    }
    return dp[n][k];
}
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n >> k;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
    sort(a + 1, a + n + 1);
    int ans = 0;
    for (int i = 1; i * (k - 1) <= a[n]; i++) {
        ans = ans + solve(i);
        ans %= mod;
    }
```

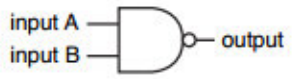
```
    cout << ans;
    return 0;
}
```

B C2-与非门

时间限制：1000ms 内存限制：65536kb

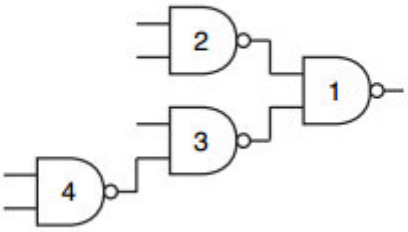
题面

数字电路中有一种基本逻辑电路叫做`与非门`，它有两个输入和一个输出。如下图：



input A	input B	output
0	0	1
0	1	1
1	0	1
1	1	0

现在将n个与非门拼接到一起，形成了一个形如二叉树的电路，如下图：



两个与非门相连表示一个与非门的输出作为另一个与非门的输入。不与与非门相连的部分表示外部输入，可能是0或者1。也就是说所有子节点数不为2的节点都会连一个外部输入，自底向上处理，最后从根节点输出，保证根节点编号为1.

由于外界的干扰，有一些与非门损坏，只能输出1或者只能输出0。

现在给出一个电路，请问有多少输入对应的输出会出错。

输入

第一行一个整数n（1≤n≤1e5）

接下来n行，每行三个整数a,b,c(0≤a,b≤n,−1≤c≤1)。对于第i行，如果a,b不为0，则a表示第i个与非门与第a个与非门相连，b表示第i个与非门与第b个与非门相连；如果a或者b为0，表示第i个与非门对应的输入接外部输入。c如果等于-1，表示第i个与非门正常工作，1表示第i个与非门只输出1，0表示第i个与非门只输出0

从1开始编号，保证根节点为1。

输出

输出一行，一个整数，答案可能很大，请对1e9+7取模

输入样例

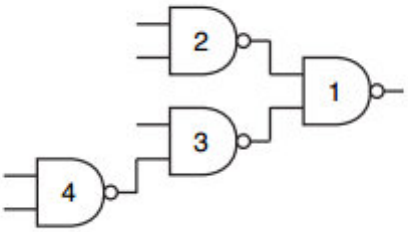
```
4
2 3 1
0 0 -1
4 0 0
0 0 -1
```

输出样例

```
15
```

样例说明

这个样例对应的电路如下:



其中1号损坏只能输出1,3号损坏只能输出0

一共有5个外部输入，每个都可能是0,1,所以输入一共有2^5种情况

对于输入`0 0 0 0 0`（上图中，由上到下），理论输出为0，实际输出为1，出错

对于输入`1 1 1 1 1`（上图中，由上到下），理论输出为1，实际输出为1，正确

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mod = 1e9 + 7;
const int ms = 1e5 + 10;
ll dp[ms][2], err[ms][2];
vector<int> g[ms];
int flag[ms], n, tot;
void dfs(int cur) {
    if (g[cur].empty())
        dp[cur][0] = 1, dp[cur][1] = 3, tot += 2;
    else if (g[cur].size() == 1) {
        tot++;
        dfs(g[cur][0]);
        dp[cur][1] = dp[g[cur][0]][0] * 2 + dp[g[cur][0]][1] + err[g[cur][0]][0] + err[g[cur][0]][1];
        dp[cur][0] = dp[g[cur][0]][1];
        err[cur][1] = err[g[cur][0]][0];
        err[cur][0] = err[g[cur][0]][1];
    }
    else {
        dfs(g[cur][0]), dfs(g[cur][1]);
```

```
        dp[cur][1] = dp[g[cur][0]][0] * dp[g[cur][1]][0] + dp[g[cur][0]][1] * dp[g[cur][1]][0] + dp[g[cur][0]][0] * dp[g[cur][1]][1] + err[g[cur][0]][1] * err[g[cur][1]][0] + err[g[cur][0]][0] * err[g[cur][1]][1] + dp[g[cur][0]][0] * err[g[cur][1]][0] + dp[g[cur][0]][0] * err[g[cur][1]][1] + err[g[cur][0]][1] * dp[g[cur][1]][0] + err[g[cur][0]][0] * dp[g[cur][1]][0];
        dp[cur][0] = dp[g[cur][0]][1] * dp[g[cur][1]][1];
        err[cur][1] = err[g[cur][0]][0] * err[g[cur][1]][0] + dp[g[cur][0]][1] * err[g[cur][1]][0] + err[g[cur][0]][0] * dp[g[cur][1]][1]; err[cur][0] = err[g[cur][0]][1] * err[g[cur][1]][1] + dp[g[cur][0]][1] * err[g[cur][1]][1] + err[g[cur][0]][1] * dp[g[cur][1]][1];
    }
    if (flag[cur] == 1)
    {
        dp[cur][1] += err[cur][0];
        err[cur][1] += dp[cur][0];
        dp[cur][0] = 0;
        err[cur][0] = 0;
    }
    else if (flag[cur] == 0) {
        dp[cur][0] += err[cur][1];
        err[cur][0] += dp[cur][1];
        dp[cur][1] = 0; err[cur][1] = 0;
    }
    dp[cur][0] %= mod;
    dp[cur][1] %= mod;
    err[cur][0] %= mod;
    err[cur][1] %= mod;
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n;
    int a, b, c;
    for (int i = 1; i <= n; ++i) {
        cin >> a >> b >> c;
        flag[i] = c;
        if (a != 0) g[i].push_back(a);
        if (b != 0) g[i].push_back(b);
    }
    dfs(1);
    cout << (err[1][0] + err[1][1]) % mod;
    return 0;
}
```

C C2-Zexal的钢管切割

时间限制：1000ms 内存限制：65536kb

题目描述

在钢管切割的背景下，已经知道长度为1-n的钢管的价值，给定长度为n的钢管在切割若干次（也可以不切割）所带来的**最小价值**是?

输入

多组数据输入

第一行一个整数n，为起始钢管长度（0<n≤1000）

第二行n个整数，分别为长度为i的钢管的价值ti（0<ti≤1e6）

输出

对于每组数据，输出一行，为这根钢管所带来的最小价值T

输入样例

```
3
2 3 7
```

输出样例

```
5
```

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
using namespace std;
#define N 100000
int a[N], dp[N];
int MAX (int a, int b)
{
    return a>b?a:b;
}
int main(){
    int n;
    while (~scanf("%d", &n)) {

        memset(dp, -9999999, sizeof(int) * (n+1));
        dp[0] = 0;
        for (int i = 1; i <= n; i++)
        {
            scanf("%d", &a[i]);
            a[i] = -a[i];
        }
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                dp[i] = MAX(dp[i], dp[i - j] + a[j]);
            }
        }
        printf("%d\n", -dp[n]);
    }
    return 0;
}
```

D C2-Zexal的流水线问题

时间限制：1000ms 内存限制：65536kb

题目描述

Zexal的偶像SkyLee趁着假期来逛漫展，漫展只有一条通道，只能从入口进入，出口离开，且左边有n个店铺，右边也有n个店铺，编号都是1~n。为了照顾大家没法兼顾逛两边的痛苦，左边和右边编号相同的店铺卖同样的周边。

现在SkyLee站在入口处的路中间，立志要从头到尾按顺序逛完所有的店铺。

因为每家店铺排队的人数不同，SkyLee在左边第i家要停留 p1[i] 的时间，在右边第i家要停留 p2[i] 的时间。而且这条路还挺宽的，所以**从左边第i家移动到右边第i+1家需要$t[1][i]$的时间；从右边第i家移动到左边第i+1家需要$t[2][i]$的时间。**

由于时间安排的紧，SkyLee想知道逛完漫展至少需要多长时间。

开始时从路中间到任意一边的时间以及同一边相邻店铺移动时间忽略不计。

输入

多组数据输入

对于每一组测试数据，第一行1个数 n （0<=n<=500）表示店铺数 。

接下来一行 n个数，表示在左边店铺停留时间 p1。再接下来一行n个数，表示在右边店铺停留时间 p2（0<=p1,p2<=1000）。

接下来2行，每行n−1个数，分别表示$t[1][i]$和$t[2][i]$（0<=t<=300）。

输出

对于每组数据，输出一行，为最少时间TT。

输入样例

```
3
10 1 10
8 5 10
3 1
1 3
```

输出样例

```
20
```

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cstdio>
using namespace std;
const int maxn = 501;
int dp[2][maxn];
int p[2][maxn];
int t[2][maxn];
int main()
{
    int n;
    while (~scanf("%d", &n))
    {
        memset(dp, 0, sizeof(dp));
        int i, j;
        for (i = 0; i <= 1; i++)
```

```
            for (j = 0; j<n; j++)
                scanf("%d", &p[i][j]);
        for (i = 0; i <= 1; i++)
            for (j = 0; j < n - 1; j++)
                scanf("%d", &t[i][j]);

        dp[0][0] = p[0][0];
        dp[1][0] = p[1][0];
        for (j = 1; j<n; j++)
        {
            dp[0][j] = min(dp[0][j - 1], dp[1][j - 1] + t[1][j - 1]) + p[0][j];
            dp[1][j] = min(dp[1][j - 1], dp[0][j - 1] + t[0][j - 1]) + p[1][j];
        }
        long long  ans = dp[0][n - 1] < dp[1][n - 1] ? dp[0][n - 1] : dp[1][n - 1];
        printf("%lld\n", ans);
    }
}
```

E C2-Zexal的排座位

时间限制：1000ms 内存限制：65536kb

题目描述

在一个班级中挑选NN个学生排成一列座位（保证有足够多的男生与足够多的女生），要求座位序列中**男生**互不相邻，求解有多少种排列方式?（挑选男生与女生的数量与排列方式均为任意）

例如挑选三个学生，那么所有排列为: 女女女、女男女，男女女，女女男，男女男

输入

第一个数为学生总数N(0<N<30)

输出

只有一行，保证男生与男生不相邻，座位排列的所有情况数目的结果

输入样例

```
3
```

输出样例

```
5
```

```
#include<cstdio>
long long f[32];
int main()
{
    int n;
    scanf("%d", &n);
    f[1] = 2;
    f[0] = 1;
```

```
    for(int i = 2; i <= n; i++)
    {
        f[i] = f[i-1] + f[i-2];
    }
    printf("%lld", f[n]);
}
```

F C2-Zexal的竞赛

时间限制：1000ms 内存限制：65536kb

题目描述

在一场竞赛中有n道题目，每道题都会对应一个分值，你可以选择任意一道题作答，比如你选择了x分的题目，那么在作答完毕（假设一定可以得分）后，你将获得x分，但是这场比赛中分值**等于**x−1和x+1的其他题目就会消失，那么这场比赛中你最多可以得到多少分？

输入

第一个数为题目总数n（0<n<1e5)

接下来为n个n个整数a1，a2，a3.....(0<ai<1e5)

输出

输出你可以得到的最高分

输入样例

```
9
1 2 1 3 2 2 2 2 3
```

输出样例

```
10
```

```
#include<cstdio>
#include<iostream>
#include<algorithm>
using namespace std;
long long a[100003];//原数组
long long aa[100003]; //新数组
long long num[100003];//记录个数
long long f[100003];//答案
int main()
{
    int n;
    scanf("%d", &n);
    for(int i = 0; i < n; i++)
    {
        scanf("%lld", &a[i]);
    }
    sort(a, a + n);
    int base = -1, k = 0;
    for(int i = 0; i < n; i++)
    {
```

```
        if(a[i] != base) num[a[i]]++, aa[k++] = a[i], base = a[i];
        else
        {
            num[a[i]]++;
        }
    }
    f[aa[0]] = aa[0]*num[aa[0]];
    f[0] = 0;
    for(int i = 1; i < k; i++)
    {
        if(aa[i-1] < aa[i] - 1) f[aa[i]] = f[aa[i-1]] + aa[i] * num[aa[i]];
        else
            f[aa[i]] = max(aa[i] * num[aa[i]] + f[aa[i-2]], f[aa[i-1]]);
    }
    printf("%lld\n",f[aa[k-1]]);
}
```

C3

A C3-Zexal的多路流水线调度

时间限制：1000ms 内存限制：65536kb

题目描述

Zexal的偶像SkyLee想要组装一台电脑，而电脑需要按照固定的顺序进行安装，不能把配件都买好一起安装（因为SkyLee只会按照顺序安装，他分不清内存条和显卡）。

城市里有n个电脑城，并且每个电脑城都有所有的配件卖，除了价格不同外完全一样。一台电脑一共有m个配件，按照安装顺序编号为1-m。

假设第i个电脑城的编号为jj的配件售价为$p[i][j]p[i][j]$，从第i个电脑城到第j个电脑城的交通费用为$f[i][j]$。

那么SkyLee组装好整台电脑最少需要多少钱呢？（配件费用+交通费用）

输入

多组数据输入

第一行两个整数n和m，分别为电脑城数量和配件数量（2<n,m≤500）

接下来n行，每行m个整数，表示配件的价格$p[i][j](0 \leq p[i][j] \leq 500)$

接下来n行，每行n个整数，表示交通费用$f[i][j](0 \leq f[i][j] \leq 500)$

输出

对于每组数据，输出一行，为最小费用

输入样例

```
3 3
10 1 10
8 5 10
10 10 2
0 5 2
1 0 5
1 1 0
```

输出样例

```
14
```

```
#include<cstdio>
#include<iostream>
#include<algorithm>
#include<cstring>
#include<cstdlib>
using namespace std;
const int maxx= 510;
long long p[maxx][maxx];
long long t[maxx][maxx];
long long cost[maxx][maxx];
const int MAX = (1<<30);
void Resolve(int n,int m)
{
    int i, j;
    for(i = 0; i < n; i++)
        for(j = 0; j < m; j++)
            cost[i][j] = MAX;
    for(int i = 0; i < n; i++)
        cost[i][0] = p[i][0];
    for(int k = 1; k < m; k++)
        for(int i = 0; i < n; i++)
            for(int j = 0; j < n; j++)
            {
                cost[i][k] = min(cost[i][k], cost[j][k-1] + t[j][i] + p[i]
[k]);
            }
    long long ans = 0x7fffffff;
    for(i = 0; i < n; i++)
    {
        if(cost[i][m-1] < ans)
            ans = cost[i][m-1];
    }
    printf("%lld\n",ans);
}
int main()
{
    int n, m;
    while(scanf("%d%d", &n, &m) != EOF)
    {
        int i, j;
        for(i = 0; i < n; i++)
            for(j = 0; j < m; j++)
                scanf("%lld", &p[i][j]);
        for(i = 0; i < n; i++)
```

```
            for(j = 0; j < n; j++)
                scanf("%lld", &t[i][j]);
        Resolve(n, m);
    }
}
```

B C3-炮弹杀伤力

最长上升子序列

```
#include<cstdio>
#include<iostream>
#include<algorithm>
#include<cstring>
#include<cstdlib>
using namespace std;
#define N 25001
int a[N], dp1[N], dp2[N];

int main() {
    int n;
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        scanf("%d", &a[i]);
    }
    for (int i = 1; i <= n; i++) {
        int j, maxx = 0;
        for (j = 1; j < i; j++)
            if (a[i] > a[j] && dp2[j] > dp2[maxx])
                maxx = j;
        dp2[i] = dp2[maxx] + 1;
        dp1[i] = max(dp1[i - 1], dp2[i]);
    }
    printf("%d\n", dp1[n]);
}
```

C C3-Zexal的矩阵链乘

时间限制：1000ms 内存限制：65536kb

题目描述

用加括号的方式给出最优的矩阵相乘方案

输入

多组数据输入

第一行一个整数 n，表示矩阵链的长度（1<=n<=300）

接下来一行n+1数表示这些矩阵的行数和列数

别问我为什么只有n+1个数，每相邻的两个数表示一个矩阵的大小

输出

对于每组数据，输出两行，第一行为计算次数，第二行为计算方案，用加括号的方式给出最优的矩阵相乘方案

如果不幸最优方案不唯一，选择优先计算左边的矩阵

输入样例

```
3
10 30 5 60
3
10 20 5 4
```

输出样例

```
4500
((A1A2)A3)
1200
((A1A2)A3)
```

Hint

在第二组样例中，选择((A1A2)A3)时，结果为10×20×5+10×5×4=1200

选择A1(A2A3)时，结果为20×5×4 + 10×20×4 = 1200

这时选择第一种，优先计算左边的!

```
#include <stdio.h>
int m[502][502],s[502][502];
void matrix_chain(int a[], int n)
{
    int l, i, j, k, tmp;
    for(l = 2; l <= n; l++)
    {
        for(i = 1; i <= n - l + 1; i++)
        {
            j = i + l - 1;
            m[i][j] = 0x7fffffff;
            for(k = i; k < j; k++)
            {
                tmp = m[i][k] + m[k+1][j] + a[i-1] * a[k] * a[j];
                if(tmp <= m[i][j])
                {
                    m[i][j] = tmp;
                    s[i][j] = k;
                }
            }
        }
    }

}
void print(int i, int j)
{
    if(i == j)
        printf("A%d",i);
    else{
        printf("(");
```

```
        print(i, s[i][j]);
        print(s[i][j] + 1, j);
        printf(")");
    }
}
int main()
{
    int n, a[303];
    int i, j, l;
    while(~scanf("%d", &n))
    {
        for(i = 0; i < n + 1; i++)
            scanf("%d", &a[i]);
        //memset(m, 0x7fffffff,sizeof(m));
        for(i = 0; i < n + 1; i++)
            m[i][i] = 0;
        matrix_chain(a, n);
        printf("%d\n", m[1][n]);
        print(1, n);
        printf("\n");
    }
    return 0;
}
```

D C3-Zexal的OBST

时间限制：1000ms 内存限制：65536kb

题目描述

假设给定一个n个不同关键字的严格升序序列K=<k[1], k[2], ..., k[n]>，用这些关键字构造二叉搜索树。对关键字k[i]，有p[i]次被检索到。有些搜索的值可能不在K中，假设n+1个伪关键字D=<d[0], d[1], ..., d[n]>，对i=1, 2, ..., n-1，d[i]表示在k[i]和k[i+1]之间的值，d[0]表示小于k[1]的值，d[n]表示大于k[n]的值。对每个伪关键字d[i]，有q[i]次被检索到。请注意这里规定了每个关键字和伪关键字的检索次数。

用上述D序列作叶节点，K序列做内部节点，（可以参考算法导论第三版中文版226-230页，但注意题目定义的不同之处）构造一棵最优二叉搜索树。假设根节点深度为1，给定p, q，求出这二叉搜索棵树的最小总代价。

总代价定义为下面两式之和：

$$\sum_{i=1}^{n} depth(k[i]) \times p[i]$$

$$\sum_{i=0}^{n} depth(d[i]) \times q[i]$$

输入

第一行两个整数n。1≤n≤500

第二行n个整数 p[i]，表示关键字的出现次数。

第三行n+1个整数q[i]，表示i-1关键字与i关键字之间的出现次数。0≤p[i],q[i]≤1000

输出

一个整数，表示最小总代价。

输入样例

```
5
15 10 5 10 20
5 10 5 5 5 10
```

输出样例

```
275
```

```
#include <iostream>
#include <iomanip>
#include <cstring>
#include<cstdio>
#define N 505
using namespace std;
typedef int T;
int a[N],b[N];
T w[N][N],q[N],p[N],e[N][N];
int r[N][N];
int n;
T dep;
void PRINT_OPTIMAL_BST(int i,int j,T depth)
{
    if(i == 1 && j == n)
        {
           // cout <<"k"<<r[i][j]<<" is the root"<<endl;
            dep += p[r[i][j]];
        }
    if(i == j )
    {
        //cout<<"d"<<i-1<<" is the left child of k"<<i<<endl;
        dep += depth*q[i-1];
        //cout<<"d"<<i<<" is the right child of k"<<i<<endl;
        dep += depth*q[i];
    }
    else if(r[i][j] == i)
    {
        //cout<<"d"<<i-1<<" is the left child of k"<<i<<endl;
        dep += depth*q[i-1];
        //cout<<"k"<<r[i+1][j]<<" is the right child of k"<<r[i][j]<<endl;
        dep += depth*p[r[i+1][j]];
        PRINT_OPTIMAL_BST(i+1,j,depth+1);
    }
    else if(r[i][j] == j)
    {
        //cout<<"k"<<r[i][j-1]<<" is the left child of k"<<r[i][j]<<endl;
        dep += depth*p[r[i][j-1]];
        PRINT_OPTIMAL_BST(i,j-1,depth+1);
        //cout<<"d"<<j<<" is the right child of k"<<j<<endl;
        dep += depth*q[j];
    }
    else
    {
        //cout<<"k"<<r[i][r[i][j]-1]<<" is the left child of k"<<r[i][j]
<<endl;
```

```
        dep += depth*p[r[i][r[i][j]-1]];
        PRINT_OPTIMAL_BST(i,r[i][j]-1,depth+1);
        //cout<<"k"<<r[r[i][j]+1][j]<<" is the right child of k"<<r[i][j]
<<endl;
        dep += depth*p[r[r[i][j]+1][j]];
        PRINT_OPTIMAL_BST(r[i][j]+1,j,depth+1);
    }
}
void OPTIMAL_BST()
{
    for(int i = 1; i <= n+1; i++)
    {
        e[i][i-1] = w[i][i-1] = q[i-1];
    }
    for(int l = 1; l <= n; l++)
    {
        for(int i = 1; i <= n-l+1; i++)
        {
            int j = i+l-1;
            e[i][j] = 0x7fffffff;
            w[i][j] = w[i][j-1] + p[j]+q[j];
            for(int k = i ; k <= j ; k++)
            {
                T t = e[i][k-1]+e[k+1][j]+w[i][j];
                if(t < e[i][j])
                {
                    e[i][j] = t;
                    r[i][j] = k;
                }
            }
        }
    }
    PRINT_OPTIMAL_BST(1,n,2);
}

int main()
{
    while(cin>>n)
    {
        memset(w,0,sizeof(w));
        memset(e,0,sizeof(e));
        memset(r,0,sizeof(r));
        dep = 0;
        int sum = 0;
        for(int i = 1; i <= n; i++) cin>>p[i],sum+=p[i];
        for(int i = 0; i <= n; i++) cin >>q[i],sum+=q[i];
        //for(int i = 1; i <= n; i++) p[i]=(float)a[i]/(float)sum;
        //for(int i = 0; i <= n; i++) q[i]=(float)b[i]/(float)sum;
        OPTIMAL_BST();
        printf("%d\n",(int)(dep));
    }

    return 0;
}
```

E C3-Zexal的浩瀚星辰

时间限制：1000ms 内存限制：65536kb

题目描述

ZexalZexal想要发射火箭，但是由于能源供应不足了，所以一些火箭需要延迟发射。 每个火箭每延迟一小时发射都会相应的损失。ZexalZexal了解到，一共有n个火箭，其中第i个火箭原计划在第i小时发射，即1~n时间段发射，现预计kk小时后电力可以恢复正常，即所有火箭将在k+1~k+n时间段内发射，

新的火箭发射计划**不要求**按照最初的发射计划顺序，唯一的要求是每个火箭都不能早于原定时间发射。请你帮忙计算一下最小的损失吧。

注意：时间均以小时为最小单位。由于条件有限，一次只能发射一枚火箭。

输入

输入包含多组数据。

每组数据第一行为正整数n和k（1≤n≤500000,1≤k≤500000），为火箭总数和延迟时间。

接下来是n个正整数 pi，代表第i个火箭每延迟一小时的损失费（1≤pi≤1e4）。

输出

对于每组数据，输出一行，为最小的损失费用。

输入样例

```
1 2
10
2 1
10 100
```

输出样例

```
20
20
```

```cpp
#include<cstdio>
#include<iostream>
#include<algorithm>
#include<cstring>
#include<cstdlib>
#include<queue>
using namespace std;
long long p[500005];
long long cost[500005];
int main()
{
    int n, k;
    while(~scanf("%d%d", &n, &k))
    {
        long long ans = 0;
        for(int i = 1; i <= n; i++)
            scanf("%lld", &p[i]);
        priority_queue<long long> q;
        for(int j = 1; j <= k + 1; j++)
            q.push(p[j]);
```

```
        for(int i = 1; i <= n; i++)
            ans += (k - i) * p[i];
        long long tmp = 0;
        int x = k + 2;
        for(int i = 1; i <= n; i++)
        {
            tmp = q.top();
            q.pop();
            ans += i * tmp;
            if(x <= n)
                q.push(p[x++]);
        }
        printf("%lld\n", ans);
    }
    return 0;
}
```

F C3-排座位

时间限制：1000ms 内存限制：65536kb

题目描述

将n个女生和m个男生排成**一列**，要求是连续的女生不能超过x，连续的男生不能超过y

请问：一共有多少种排座位的方式（结果对1000007取模）。

注：所有的女生看作相同，所有男生看做相同。

注意内存限制。

输入

多组数据输入，数据组数小于20

每组数据包括四个正整数n、m、x、y（1≤n,m≤2000,1≤x,y≤2000），含义见描述

输出

对于每组数据，输出一行，为排座位的方法数（结果对1000007取模）

输入样例

```
1 2 1 1
2 3 1 2
```

输出样例

```
1
5
```

Hint

1表示女生，0表示男生

第一组数据，方法只有一种，即：010

第二组数据，方法有五种，即：01011，01101，10101,10110，11010

```cpp
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int mod = 1000007;
const int N = 2010;
const int M = 2010;
int a[N][M];//a[i][j]表示以女生结尾并且有i个女生前提下，1个男  到  j个男  j种方案方案总和
int b[N][M];//b[i][j]表示以男生结尾并且有j个男生前提下，1个女  到  i个女  i种方案方案总和
int n, m, x, y;
int main()
{
    while (scanf("%d%d%d%d", &n, &m, &x, &y) != EOF)
    {
        memset(a, 0, sizeof(a));
        memset(b, 0, sizeof(b));
        for (int i = 1; i <= x; i++)
            a[i][0] = 1;//初始化只有女生的情况
        for (int j = 1; j <= y; j++)
            b[0][j] = 1;//初始化只有男生的情况
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= m; j++)
            {
                if (i <= x)
                {
                    a[i][j] = (a[i][j - 1] + b[i - 1][j]) % mod;
                    //当i<=x时，b[i-1][j]就等于以女生结尾，有i个男生j个女生
                }
                else
                {
                    a[i][j] = (a[i][j - 1] + b[i - 1][j] - b[i - x - 1][j] + mod) % mod;
                    /*[i - 1][j] - b[i - x - 1][j] 代表在有j个男生的并且以男生结尾的前提下，有i-x个女生到i-1个女生的方案总数。
                    这个方案总数恰好等于以女生结尾,有i个女生j个男生的方案书。
                    */
                }
                if (j <= y)
                    b[i][j] = (b[i - 1][j] + a[i][j - 1]) % mod;
                else
                    b[i][j] = (b[i - 1][j] + a[i][j - 1] - a[i][j - y - 1] + mod) % mod;
            }
        int out = (a[n][m] - a[n][m - 1] + b[n][m] - b[n - 1][m] + mod + mod) % mod;
        printf("%d\n", out);
    }
    return 0;
}
```

G C3-等差数列

时间限制：2000ms 内存限制：204800kb

题目描述

现有一数字序列，从中取出一些数字元素，就可以组成一个等差数列，我们想知道这个等差数列最多能有多少个元素，原序列每个元素最多只能取一次。

输入

输入包括多组数据。

每组数据第一行为整数n，表示输入序列的元素个数（3≤ n ≤10^4）.

接下来一行是n个**不同的**正整数Ai（1≤ Ai ≤10^9）。

输出

对于每组数据，输出一行，为最长的等差数列的长度（元素个数）。

输入样例

```
3
1 2 3
```

输出样例

```
3
```

```cpp
#include <cstdio>
#include <iostream>
#include <algorithm>
#define MaxSize 10005
using namespace std;
int n, ans;
int A[MaxSize];
short int dp[MaxSize][MaxSize];

int main()
{
    while(~scanf("%d", &n))
    {
        for (int i = 0; i < n; ++i)
            scanf("%d", &A[i]);
        sort(A, A+n);

        for (int i = 0; i < n; ++i)
            for (int j = i+1; j < n; ++j)
                dp[i][j] = 2;

        ans = 2;
        for (int j = n-2; j > 0; --j) {
            int i = j-1, k = j+1;
            while(i>=0 && k<n)
            {
                if(A[i]+A[k] < 2*A[j])
                    k++;
                else if(A[i]+A[k] > 2*A[j])
                    i--;
                else
                {
                    dp[i][j] = dp[j][k] + 1;
```

```
                    if(dp[i][j] > ans)
                        ans = dp[i][j];
                    i--, k++;
                }
            }
        }
        printf("%d\n", ans);
    }
}
```

C4

A C4-非负权单源最短路

时间限制：1000ms 内存限制：65536kb

考点

Shortest path

题目描述

给一个n(1<=n<=2500)个点 m(1<=m<=6200)条边的无向图，求s到ts到t的最短路。

输入

第一行四个由空格隔开的整数n,m,s,t之后的m行，每行三个正整数a、b、c，表示一条从a到b长度为c的边。

输出

一个整数表示从s到t的最短路长度。数据保证至少存在一条道路。

输入样例

```
7 11 5 4
2 4 2
1 4 3
7 2 2
3 4 3
5 7 5
7 3 3
6 1 1
6 3 4
2 4 3
5 6 3
7 2 1
```

输出样例

```
7
```

```
#include <iostream>
#include <algorithm>
#include <queue>
```

```cpp
#include <vector>
#include <cstdio>
#include <cstring>
#include <climits>
using namespace std;
#define INF 0x3f3f3f3f

struct edge {int to,cost;};
typedef pair<int,int> P;
int V;
vector<edge> G[2520];
int d[2520];

void dijkstra(int s)
{
    priority_queue<P,vector<P>,greater<P> > que;
    memset(d,INF,sizeof(d));
    d[s] = 0;
    que.push(P(0,s));
    while(!que.empty())
    {
        P p = que.top(); que.pop();
        int v = p.second;
        if (d[v] < p.first) continue;
        for(int i = 0; i < G[v].size(); i++)
        {
            edge e = G[v][i];
            if (d[e.to] > d[v] + e.cost)
            {
                d[e.to] = d[v] + e.cost;
                que.push(P(d[e.to],e.to));
            }
        }
    }
}

int main()
{
    int n, m, s, t;
    scanf("%d %d %d %d", &n, &m, &s, &t);
    for (int i = 0; i < m; i++)
    {
        int a, b, c;
        scanf("%d %d %d", &a, &b, &c);
        G[a].push_back({b, c});
        G[b].push_back({a, c});
    }
    dijkstra(s);
    printf("%d\n", d[t]);
    return 0;
}
```

B C4-排列

时间限制：1000ms 内存限制：65536kb

考点

DFS

题目描述

给定n,mn,m，输出从 1~n中选择 m个数的所有排列。 要求按照字典序输出。

输入

单组数据。

一行，两个空格分隔的整数，分别表示 n,m(1≤m≤n≤8)。

输出

输出若干行，表示答案。

输入样例

```
3 2
```

输出样例

```
1 2
1 3
2 1
2 3
3 1
3 2
```

```
#include<bits/stdc++.h>
using namespace std;
int n, m, top = 0;
int num[11];
bool used[11];
void dfs()
{
    if (top == m)
    {
        for (int i = 0; i<top; i++)
            printf("%d ", num[i]);
        printf("\n");
        return;
    }

    for (int i = 1; i <= n; i++)
    {
        if (!used[i])
        {
            used[i] = 1;
            num[top++] = i;
            dfs();
            --top;
            used[i] = 0;
        }
    }
}
int main()
```

```
{
    scanf("%d%d", &n, &m);
    dfs();
}
```

C C4-食物链

时间限制：1000ms 内存限制：65536kb

考点

DFS

题目描述

现在给你n个物种和m条能量流动关系，求其中的食物链条数。

物种的名称为从1到n的编号。 m条能量流动关系形如a b表示能量从物种a 流向物种b。注意单独的一种孤立生物不算一条食物链。

此处的食物链指的是从生产者到当前链上**最高级消费者**的一条链。（入度为0出度为0的一条独立路径）

输入

第一行两个整数n和m,接下来 m行每行两个整数a b描述 m条能量流关系。

（保证输入数据符合生物学特点，且不会有重复的能量流动关系出现）

(1<=n<=100000 1<=m<=200000)

输出

一个整数，即食物网中的食物链条数。

输入样例

```
10 16
1 2
1 4
1 10
2 3
2 5
4 3
4 5
4 8
6 5
7 6
7 9
8 5
9 8
10 6
10 7
10 9
```

输出样例

```
9
```

```cpp
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<algorithm>
using namespace std;
int i, j, l, m, n, temp;
long long ans;
int head[100001], y[200001], net[200001], rd[100001], t[100001],
cd[100001];
void add(int a, int b)
{
    net[++temp] = head[a];
    y[temp] = b;
    head[a] = temp;
    rd[b]++;
    cd[a]++;
    return;
}

long long dfs(int x)
{
    long long ans = 0;
    if(t[x]) return t[x];
    if(!cd[x] && rd[x]) ans++;
    for(int p = head[x]; p != 0; p = net[p])
    {
        ans += dfs(y[p]);
    }
    t[x] = ans;
    return ans;
}

int main()
{
    int n, m, a, b;
    scanf("%d%d", &n, &m);
    for(i = 1; i <= m; i++)
    {
        int a, b;
        scanf("%d%d", &a, &b);
        add(a, b);
    }
    for(i = 1; i <= n; i++)
    {
        if(!rd[i])
        {
            ans += dfs(i);
        }
    }
    printf("%lld\n", ans);
    return 0;
}
```

D C4-最小乘法

时间限制：1000ms 内存限制：65536kb

考点

Dynamic programming

题目描述

如果在一个正整数中间插入一些乘号，会得到一个更小的数字。 一个数字有不同的插入方法，会得到不同的数字，我们想知道最大的那个数字是多少?

输入

输入包括多组数据。

每组数据第一行只含一个正整数n（0≤n≤10），代表添加乘号的个数。

第二行是一个数字串（0 < 长度≤18，保证长度大于n）。

输出

对于每组数据，输出添加乘号后计算得到的数字（在long long范围内）。

输入样例

```
1
233
1
1111
```

输出样例

```
69
121
```

```
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<iostream>
using namespace std;
char str[20];
long long change(int l, int r)
{
    long long ans = 0;
    for(int i = l; i <= r; i++)
    {
        ans = str[i] - '0' + ans * 10;
    }
    return ans;
}
long long dp[18][18];
int main()
{
    int n;
    while(~scanf("%d", &n))
    {
        scanf("%s", str);
        memset(dp, 0, 400*sizeof(int));
        int len = strlen(str);
```

```
        dp[0][0] = str[0] - '0';
        for(int i = 1; i < len; i++)
            dp[i][0] = dp[i - 1][0] * 10 + str[i] - '0';
        //dp[i][j]表示在str[0~i]中添加j个乘号的最大值
        //dp[i][j] = max(dp[i][j], dp[i-k][j-1] * change(i - k + 1, i)) k from 1 to i - j + 1
        for(int i = 1; i < len; i++)
        {
            for(int j = 1; j <= min(n, i); j++)
            {
                for(int k = 1; k <= i - j + 1; k++)
                {
                    dp[i][j] = max(dp[i][j], dp[i-k][j-1] * change(i - k + 1, i));
                }
            }
        }
        printf("%lld\n", dp[len-1][n]);
    }
    return 0;
}
```

E C4-商人卖鱼

时间限制：1000ms 内存限制：65536kb

考点

Greedy Algorithm

题目描述

商人养了好多好多的鱼，他决定把各种鱼卖掉，很多人排队来买鱼，而且他们都会在推销下买给推荐的鱼。

现在知道卖掉第i条鱼需要Ti的时间（在推销时不需要给鱼喂食），而养第i条鱼一分钟需要Di价值的饲料。

现在要求你告诉他应该如何卖鱼才能在用最少饲料的情况下卖掉所有的鱼。

输入

第一行一个数n，表示有多少条鱼。

第二行到第n+1行，每行两个数Ti和Di如题目描述意义

$2 <= N <= 100000,\ 1 <= T <= 2000000,\ 1 <= D <= 100$

输出

一个数表示最少总共需要多少饲料。

输入样例

```
6
3 1
2 5
2 3
3 2
4 1
1 6
```

输出样例

```
43
```

```cpp
#include <bits/stdc++.h>
using namespace std;
struct data
{
    int t, d;
    double cost;
}a[100002];
bool cmp(data a, data b)
{
    if (a.cost > b.cost )
    {
        return true;
    }
    return false;
}
int main()
{
    int n;
    scanf("%d", &n);
    long long ans = 0;
    for(int i = 0; i < n; i++)
    {
        scanf("%d%d", &a[i].d, &a[i].t);
        a[i].cost = (double)a[i].d / a[i].t;
    }
    sort(a, a + n, cmp);
    for(int i = 0; i < n; i++)
    {
        for(int j = i + 1; j < n; j++)
            ans += a[j].d * a[i].t;
    }
    printf("%lld", ans);
}
```

F C4-白雪皑皑

时间限制：1000ms 内存限制：65536kb

考点

BFS

题目描述

在一片矩形的雪地上，有2种动物——小白兔和小白狐活动。小白兔走过草地会留下 R，小白狐走过雪地会留下 F。每只动物从左上角进入雪地，从右下角走出雪地。其间，它可以上下左右乱跳（可以重复），经过的格子会被覆盖上它的脚印。每次雪地上最多只有一只动物。

```
........            RRR.....       FFR.....

........            ..RRR...       .FRRR...

........            ..R.....       .FFFFF..

........            ..RRRR.R       ..RRRFFR

........            .....RRR       .....FFF
```

给你地图，问最少有多少只动物走过了雪地。

输入

第一行：宽度和高度H和W（1<=H,W<=4000）

下面一个 H*W 的矩阵

输出

至少有多少只动物走过了草地。

输入样例

```
5 8
FFR.....
.FRRR...
.FFFFF..
..RRRFFR
.....FFF
```

输出样例

```
2
```

```cpp
#include <iostream>
#include <algorithm>
#include <string>
#include <queue>
#include <cstring>
using namespace std;
typedef long long ll;
typedef pair<int, int> pt;
const int ms = 4000 + 10;
string s[ms];
int h, w;
int p[4] = { -1,1,0,0 };
int bfs()
{
 int res = 0;
 queue<pt> nq, q;
 char t;
```

```
 nq.push(make_pair(0, 0));
 while (!nq.empty())
 {
    t = s[nq.front().first][nq.front().second];
    res++;
    while (!nq.empty())
        q.push(nq.front()),s[nq.front().first][nq.front().second] = '.',
nq.pop();
    while (!q.empty())
    {
        int x = q.front().first;
        int y = q.front().second;
        q.pop();
        for (int i = 0; i < 4; i++)
        {
            int fx = x + p[i];
            int fy = y + p[3 - i];
            if (fx < 0 || fx >= h) continue;
            if (fy < 0 || fy >= w) continue;
            if (s[fx][fy] == '.') continue;
            if (s[fx][fy] != t)
        {
        nq.push(make_pair(fx, fy));
        continue;
        }
        q.push(make_pair(fx, fy)); s[fx][fy] = '.';
    }
  }
 }
return res;
}
int main()
{
 ios::sync_with_stdio(false); cin.tie(nullptr);
 cin >> h >> w;
 for (int i = 0; i < h; ++i)
 {
 cin >> s[i];
 }
 cout << bfs();
 return 0; }
```

G C4-完全图的最小生成树

时间限制：1000ms 内存限制：65536kb

考点

Greedy Algorithm

题目描述

给定一个图，图中包括 n 个点，每个点 i 有一个权值A[i]。

这个图是一个完全图，任意两点 i 和 j 之间有一条长度为 A[i]+A[j]的无向边，问该图最小生成树所有边的长度和。

输入

第一个数为数据组数T， T≤50。

对于每组数据，首先输入一个数nn，n≤100000。

接下来一行nn个数，分别表示A[1]，...，A[n]，1≤A[i]≤10,000,000。

输出

对于每组数据，输出一行，最小生成树的边权和。

输入样例

```
1
4
5 6 5 9
```

输出样例

```
35
```

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int T;
    scanf("%d", &T);
    while(T--)
    {
        int MIN = 0x3f3f3f3f;
        long long ans = 0;
        int n;
        scanf("%d", &n);
        for(int i = 0; i < n; i++)
        {
            int a;
            scanf("%d", &a);
            ans += a;
            MIN = MIN < a ? MIN : a;
        }
        ans += (n - 2) * MIN;
        printf("%lld\n", ans);
    }
}
```

E2-背包股票大礼包，请自行参阅相关文献

E3

A E3-网络流问题来了

时间限制：1000ms 内存限制：65536kb

题目描述

想哥带着叶姐去了游乐园，有个项目可以让他们在一个m*n (m,n<=30)方格中，取走一些礼物，同时要求任意2个取走的礼物所在方格没有公共边，且取出的礼物让叶姐的满意度最大。

想哥忙于学(lian)习(ai)，难以完成，所以求助于你。

输入

第1 行有2 个正整数m和n，分别表示棋盘的行数和列数。

接下来的m行，每行有n个正整数，表示方格中的礼物的满意度。

输出

输出一行，为最大满意度

输入样例

```
3 3
1 2 3
3 2 3
2 3 1
```

输出样例

```
11
```

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int oo=1e9;
const int mm=111111111;
const int mn=1010;
int node,src,dest,edge;
int ver[mm],flow[mm],nex[mm];
int head[mn],work[mn],dis[mn],q[mn];
void prepare(int _node, int _src,int _dest)
{
    node=_node,src=_src,dest=_dest;
    for(int i=0; i<=node; ++i)head[i]=-1;
    edge=0;
}
void addedge( int u,  int v,  int c)
{
    ver[edge]=v,flow[edge]=c,nex[edge]=head[u],head[u]=edge++;
    ver[edge]=u,flow[edge]=0,nex[edge]=head[v],head[v]=edge++;
}
bool Dinic_bfs()
{
    int i,u,v,l,r=0;
    for(i=0; i<node; ++i)dis[i]=-1;
    dis[q[r++]=src]=0;
    for(l=0; l<r; ++l)
        for(i=head[u=q[l]]; i>=0; i=nex[i])
            if(flow[i]&&dis[v=ver[i]]<0)
            {
                dis[q[r++]=v]=dis[u]+1;
```

```
                if(v==dest)  return 1;
            }
    return 0;
}
int Dinic_dfs(  int u, int exp)
{
    if(u==dest)  return exp;
    for(  int &i=work[u],v,tmp; i>=0; i=nex[i])
        if(flow[i]&&dis[v=ver[i]]==dis[u]+1&&
(tmp=Dinic_dfs(v,min(exp,flow[i])))>0)
        {
            flow[i]-=tmp;
            flow[i^1]+=tmp;
            return tmp;
        }
    return 0;
}
int Dinic_flow()
{
    int i,ret=0,delta;
    while(Dinic_bfs())
    {
        for(i=0; i<node; ++i)work[i]=head[i];
        while((delta=Dinic_dfs(src,oo)))ret+=delta;
    }
    return ret;
}

int a[111][111];
int fx[4][2]= {{1,0},{0,1},{-1,0},{0,-1}};
int main()
{
    int m,n;
    while(~scanf("%d%d",&m,&n))
    {
        int sum=0;
        for(int i=1; i<=m; i++)
            for(int j=1; j<=n; j++)
                scanf("%d",&a[i][j]),sum+=a[i][j];
        prepare(m*n+2,0,m*n+1);
        for(int i=1; i<=m; i++)
            for(int j=1; j<=n; j++)
            {
                int tem=(i-1)*n+j;
                if((i+j)&1)
                    addedge(0,tem,a[i][j]);
                else
                    addedge(tem,m*n+1,a[i][j]);
                for(int k=0; k<4; k++)
                {
                    int x=i+fx[k][0],y=j+fx[k][1];
                    if(x>=1&&x<=m&&y>=1&&y<=n)
                    {
                        int tt=(x-1)*n+y;
                        if((i+j)&1)
                            addedge(tem,tt,oo);
                        else
                            addedge(tt,tem,oo);
```

```
                    }
                }
            }
        printf("%d\n",sum-Dinic_flow());
    }
    return 0;
}
```

B E3-婚车

最大流模板题

C E3-要成为魔法少女吗！！

二分图最大匹配模板题

D E3-SkyLee的脱单大计

二分图最大匹配模板

E E3-计网的烦恼

时间限制：1000ms 内存限制：65536kb

题目描述

计网课上有一道题：一条街道安装无线网络，需要放置M个路由器。整条街道上一共有N户居民，分布在一条直线上，每一户居民必须被至少一台路由器覆盖到。现在的问题是所有路由器的覆盖半径是一样的，我们希望用覆盖半径尽可能小的路由器来完成任务，因为这样可以节省成本。

输入

输入第一行包含两个整数M和N，以下N行每行一个整数Hi表示该户居民在街道上相对于某个点的坐标。

输出

输出仅包含一个数，表示最小的覆盖半径，保留一位小数。

输入样例

```
2 3
1
3
10
```

输出样例

```
1.0
```

HINT

【样例输出】（在2，10位置上各放一个）

【数据规模】

对于100%的数据，有1 ≤N, M ≤100000，-10000000 ≤Hi ≤10000000。

```
//二分答案
#include<bits/stdc++.h>
using namespace std;
int N, M;
int a[100001];
bool cover(int d){ //这里把半径换成了直径会好写一点
    int c = 0, cnt = 1;
    while((c = (upper_bound(a, a + N, a[c] + d) - a)) < N)
    {
        //cout << "c = " << c << endl;
        cnt++;
        if(cnt > M)
            return 0;
    }
    return 1;
}
int main()
{

    cin >> M >> N;
    for(int i = 0; i < N; ++i)
    {
        cin >> a[i];
    }
    sort(a, a + N);
    int l = 0, r = a[N-1] - a[0];
    //cout << "r = " << r << endl;
    int mid;
    while(l <= r)
    {
        mid = (l + r) >> 1;
        if(cover(mid))
            r = mid - 1;
        else l = mid + 1;
    }
    printf("%.1lf", (double)mid / 2.0);
}
```

C5

A C5-毛毛虫

简单程设题

B C5-图1

时间限制：1000ms 内存限制：65536kb

题面

一个无向图，`N`个点编号`1~N`。`M`条边，每条边有一个权值`c`。

对于一个点集`A`，这个点集的权值`s`定义为$SA = maxc_{ij}$，其中$i \in A \wedge j \in A \wedge i \neq j$。

现在将N个点分割为两个点集A、B，请问$max(SA, SB)$的最小值

输入

第一行两个正整数N、M。(2≤N≤20000,1≤M≤100000)

接下来M行，每行三个整数a,b,c，表示ab之间有一条权值为c的边（1≤a,b≤N,1≤c≤1e9）

输出

一行一个数

输入样例

```
4 6
1 4 2534
2 3 3512
1 2 28351
1 3 6618
2 4 1805
3 4 12884
```

输出样例

```
3512
```

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int ms = 20000 + 10;
int c[ms];
vector<pair<int, int>>g[ms];
bool dfs(int cur, int color, int mid) {
    c[cur] = color;
    for (const auto& i : g[cur]) {
        if (i.second > mid)
        {
            if (c[i.first])
            {
                if (c[i.first] == color)
                    return false;
            }
            else
            {
                if (!dfs(i.first, 3 - color, mid))
                    return false;
            }
        }
    }
    return true;
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t = 1;
```

```cpp
    //cin >> t;
    while (t--)
    {
        int n, m, l = 0, r = 0, x, y, z;
        cin >> n >> m;
        for (int i = 0; i < m; ++i)
        {
            cin >> x >> y >> z;
            g[x].emplace_back(y, z);
            g[y].emplace_back(x, z);
            r = max(r, z);
        }
        int mid, res = 0;
        while (l < r)
        {
            mid = (l + r) / 2;
            bool flag = true;
            memset(c, 0, sizeof(c));
            for (int i = 1; i <= n; ++i)
            {
                if (!c[i]) flag &= dfs(i, 1, mid);
                if (!flag)
                    break;
            }
            if (flag)
            {
                res = mid;
                r = mid;
            }
            else
            {
                l = mid + 1;
            }
        }
        cout << res << "\n";
    }
    return 0;
}
```

C C5-图2

时间限制：1000ms 内存限制：65536kb

题面

一个有向图，`N`个点编号`1~N`。`M`条边，每条边有一个权值`c`。

点`i`、`j`之间的最短路长度定义为 Sij。如果ij不连通，则Sij=-1

输出所有使得Sij最大的ij

输入

第一行一个整数t表示数据组数(t≤50)

对于每组数据：

第一行两个正整数N、M。(2≤N≤200,1≤M≤1000)

接下来M行，每行三个整数a,b,c，表示ab之间有一条权值为c的边（1≤a,b≤N,1≤c≤1e3）

输出

输出所有使得Sij最大的ij，每一对ij输出一行，用空格隔开，按i的大小由小到大输出，i相同时按j的大小由小到大输出。

输入样例

```
2
2 1
1 2 3
3 3
1 2 2
2 3 3
3 1 5
```

输出样例

```
1 2
2 1
```

```
#include<cstdio>
#include<cstring>
#include<iostream>
#define MAX 201
using namespace std;
int n,m;
int e[MAX][MAX], mmax;
void Init()
{
  for(int i=1; i<=n; ++i)
    for(int j=1; j<=n; ++j)
    {
      if(i==j)
        e[i][j]=0;
      else
        e[i][j]=1e9;
    }
}
void Input()
{
  int a,b,c;
  for(int i=1; i<=m; ++i)
  {
    cin>>a>>b>>c;
    e[a][b]=c;
      mmax = max(e[a][b], mmax);
  }
}
void Floyd()
{
  for(int k=1; k<=n; k++)
    for(int i=1; i<=n; i++)
      for(int j=1; j<=n; j++)

```

```
            if(e[i][j] > e[i][k] + e[k][j])
            {
                e[i][j]=e[i][k]+e[k][j];

            }
}
void Output()
{
    int maxpath = 0;
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
        {
            if (e[i][j] != 1e9)
                maxpath = max(maxpath, e[i][j]);
        }
    }
  for(int i=1; i<=n; ++i)
      for(int j=1; j<=n; ++j) {
          if (e[i][j] == maxpath ) {
              printf("%d %d\n", i, j);
          }
      }

}
int main()
{
      int t;
      cin >> t;
      while (t--) {
          mmax = 0;
          cin>>n;
          cin>>m;
          Init();
          Input();
          Floyd();
          Output();
          //cout << e[1][4] << endl;
      }
}
```

D C5-图3

时间限制：1000ms 内存限制：65536kb

通过率：1/4 `(25.00%)` 正确率：1/12 `(8.33%)`

题面

一个无向图，`N`个点编号`1~N`。`M`条边，每条边有一个权值`c`。

问对于每条边，最少删除多少条边后，可以使得存在一个最小生成树包含这条边。

输入

第一行两个正整数N、M(2≤N,M≤100)

接下来M行，每行三个整数abc，表示ab之间存在一条权值为c的边。（1≤a,b≤N，1≤c≤500）

输出

输出一行M个数，数之间用空格隔开

输入样例

```
3 3
1 2 1
3 1 2
3 2 3
```

输出样例

```
0 0 1
```

```cpp
//对每条边，用小于这条边权值的边建图，跑最小割
#include <bits/stdc++.h>
using namespace std;
const int maxv = 100 + 5;
const int maxm = 550;
const int inf = 0x3f3f3f3f;
struct Edge { int v, e, f; };
Edge e[maxm * 2];
int head[maxv * 2], depth[maxv * 2];
int s, t, cnt = 2;
queue<int>q;
inline void addEdge(int from, int to, int val) {
    e[cnt] = { to,head[from],val };
    head[from] = cnt++;
    e[cnt] = { from,head[to],val };
    head[to] = cnt++;
}
bool bfs() {
    memset(depth, 0, sizeof(depth));
    while (!q.empty())
        q.pop();
    q.push(s);
    depth[s] = 1;
    while (!q.empty())
    {
        int cur = q.front();
        q.pop();
        for (int i = head[cur]; i; i = e[i].e)
        {
            if (!depth[e[i].v] && e[i].f)
            {
                depth[e[i].v] = depth[cur] + 1;
                q.push(e[i].v);
                if (e[i].v == t) return true;
            }
        }
    }
    return false;
}
int dfs(int x, int flow) {
```

```
    if (x == t) return flow;
    int rest = flow;
    for (int i = head[x]; i && rest; i = e[i].e)
    {
        if (depth[x] + 1 == depth[e[i].v] && e[i].f)
        {
            int f = dfs(e[i].v, min(rest, e[i].f));
            if (!f) depth[e[i].v] = 0;
            e[i].f -= f; e[i ^ 1].f += f; rest -= f;
        }
    }
    return flow - rest;
}
int dinic() {
    int flow = 0;
    while (bfs())
    {
        flow += dfs(s, inf);
    }
    return flow;
}
Edge ee[maxm];
int main()
{
    int n, m, a, b, c;
    ios::sync_with_stdio(false); cin.tie(nullptr);
    cin >> n >> m;
    for (int i = 0; i < m; ++i)
    {
        cin >> a >> b >> c;
        ee[i] = { a,b,c };
    }
    for (int i = 0; i < m; ++i)
    {
        cnt = 2;
        memset(head, 0, sizeof(head));
        for (int j = 0; j < m; ++j)
        {
            if (ee[j].f < ee[i].f)
                addEdge(ee[j].v, ee[j].e, 1);
        }
        s = ee[i].v;
        t = ee[i].e;
        cout << dinic() << " ";
    }
    return 0;
}
```

E C5-棋盘

时间限制：1000ms 内存限制：65536kb

题面

一个`N`行，`M`列的棋盘。棋盘每个格子都是边长为1的正方形。

现在要在棋盘上放一些1×2大小的骨牌。骨牌的边线与格子重合（必须占2个格子），任意两个骨牌不能重叠。

但是棋盘上的一些格子已经被占用，请问最多可以放多少个骨牌。

输入

第一行三个正整数N、M、q。(2≤N,M≤100,0≤q≤1000)

接下来q行，每行两个整数a,b,表示第a行第b列的格子被占用（1≤a,b≤N）

输出

输出一行

输入样例

```
8 8 0
```

输出样例

```
32
```

```cpp
//二分图
#include<bits/stdc++.h>
using namespace std;
const int ms = 105 * 105;
int n, m;
vector<int> G[ms];
int match[ms];
bool visit[ms];
bool dfs(int x) {
    int len = G[x].size();
    for (int i = 0; i < len; ++i)
    {
        int to = G[x][i];
        if (!visit[to])
        {
            visit[to] = true;
            if (!match[to] || dfs(match[to])) {
                match[to] = x; return true;
            }
        }
    }
    return false;
}
bool p[ms];
inline int to1(int a, int b){ return (a - 1)*m + b; }
int MaxMatch()
{
    int ans = 0;
    memset(match, 0, sizeof(match));
    for (int i = 1; i <= n; ++i)
    {
        for (int j = 1; j <= m; ++j)
        {
            if ((i + j) & 1 || p[to1(i, j)])
                continue;
            memset(visit, false, sizeof(visit));
```

```
            if (dfs(to1(i, j))) ans++;
        }
    }
    return ans;
}
int main()
{
    int k;
    scanf("%d%d%d", &n, &m, &k);
    int a, b;
    for (int i = 0; i < k; ++i)
    {
        scanf("%d%d", &a, &b);
        p[to1(a, b)] = true;
    }
    for (int i = 1; i <= n; ++i)
    {
        for (int j = 1; j <= m; ++j)
        {
            if (!((i + j) & 1) && !p[to1(i, j)])
            {
                if (i > 1 && !p[to1(i - 1, j)])
                    G[to1(i, j)].push_back(to1(i - 1, j));
                if (j > 1 && !p[to1(i, j - 1)])
                    G[to1(i, j)].push_back(to1(i, j - 1));
                if (i < n && !p[to1(i + 1, j)])
                    G[to1(i, j)].push_back(to1(i + 1, j));
                if (j < m && !p[to1(i, j + 1)])
                    G[to1(i, j)].push_back(to1(i, j + 1));
            }
        }
    }
    printf("%d\n", MaxMatch());
}
```

F C5-垃圾粉碎机

时间限制：1000ms 内存限制：65536kb

题目描述

垃圾分类快来了，垃圾场主某楠希望赶在垃圾分类之前将厂里的垃圾全部粉碎填埋。为此场长专门去租了n台垃圾粉碎机，每种垃圾粉碎机都有一个最长使用时间ti，在这段时间里总共可以处理mi吨垃圾,可以在任意时间使用任意时长，但是用完就不能再用。由于场里太穷，同一时间只能运行一台垃圾粉碎机，现在想问在垃圾分类来临之前，最多能粉碎多少垃圾。为了简化计算，所有时间单位以小时计算。

输入

前两个数为垃圾粉碎机的个数N和距离垃圾分类来临时间T小时

接下来N行每行2个整数，对应的ti和mi

所有数字均不大于1e5

输出

输出一行，能处理的垃圾最大重量，保留2位小数，单位为吨

输入样例

```
1 2
2 3
```

输出样例

```
3.00
```

```cpp
//简单贪心
#include<cstdio>
#include<cstring>
#include<iostream>
#include<algorithm>
#include<queue>
using namespace std;
#define MAXN 1000000
typedef pair<double, int> pd;
pd a[MAXN];

int main()
{
    int n, T, t, m;
    while (~scanf("%d %d",&n, &T)) {
        for (int i = 0; i < n; i++) {
            scanf("%d %d", &t, &m);
            a[i].first = 1.0 * m / t;
            a[i].second = t;
        }
        sort(a, a + n);
        double ans = 0;
        for (int i = n - 1; i >= 0 && T > 0; i--) {

            ans += 1.0 * min(T, a[i].second) * a[i].first;
            T -= a[i].second;

        }
        printf("%.2lf\n", ans);
    }
    return 0;
}
```

G C5-小面包

时间限制：1000ms 内存限制：65536kb

题目描述

又要发小面包了。这次我们有许多 `3*6` 的小面包和 `6*6` 的方糕，以及一个 `6*N` 的长方形盒子，强迫症的某楠一定要把它们整齐的装到盒子里，并且要尽量装满。请问有多少总不同装法?

输入

多组数据输入。 每组一个3的倍数N（0<=N<=750）

输出

对于每组数据，输出一行，为最终计算对1000007取模得到的结果。

输入样例

```
3
6
```

输出样例

```
1
3
```

样例解释

输入为3时，只能放入一块小面包。

输入为6时，有三种情况：

（1）竖着放两块小面包

（2）横着放两块小面包

（3）放一块方糕

```cpp
#include <iostream>
using namespace std;
const int MOD = 1000007;
const int N = 255;
long long fib[N];
void setfib(int n)
{
    fib[0] = 0;
    fib[1] = 1;
    fib[2] = 3;
    for(int i = 3; i <= n; i++)
       fib[i] = (2 * fib[i - 2] % MOD + fib[i - 1] % MOD) % MOD;
}

int main()
{
    int n, i;
    setfib(N);
    while(cin >> i)
    {
        cout << fib[i/3] << endl;
    }
    return 0;
}
```

H C5-点线面

时间限制：1000ms 内存限制：65536kb

题面

二维平面上有n个点。现在用一根（毛）线将这些点围起来，问线的最小长度和围起来的面积。

输入

第一行一个正整数N。(2≤N≤100000)

接下来N行，每行两个整数a，b，表示一个点的坐标。 (-1e6≤a,b≤1e6)

输出

输出一行一个数，保留两位小数。

输入样例

```
4
0 0
0 4
3 0
1 1
```

输出样例

```
12.00 6.00
```

```cpp
//凸包面积周长
#include <iostream>
#include <cstdio>
#include <cmath>
#include <algorithm>
using namespace std;
const double epsi = 1e-8;
const double pi = acos(-1.0);
const int maxn = 100001;
struct Point{
    double x;
    double y;

    Point(double _x = 0,double _y = 0):x(_x),y(_y){

    }

    Point operator - (const Point& op2) const{
        return Point(x - op2.x,y - op2.y);
    }

    double operator^(const Point &op2)const{
        return x*op2.y - y*op2.x;
    }
};


inline int sign(const double &x){
    if(x > epsi){
        return 1;
    }

```

```
    if(x < -epsi){
        return -1;
    }

    return 0;
}

inline double sqr(const double &x){
    return  x*x;
}


inline double mul(const Point& p0,const Point& p1,const Point& p2){
    return (p1 - p0)^(p2 - p0);
}


inline double dis2(const Point &p0,const Point &p1){
    return sqr(p0.x - p1.x) + sqr(p0.y - p1.y);
}

inline double dis(const Point& p0,const Point& p1){
    return sqrt(dis2(p0,p1));
}

int n;
double l;
Point p[maxn];
Point convex_hull_p0;

inline bool convex_hull_cmp(const Point& a,const Point& b){
    return sign(mul(convex_hull_p0,a,b)>0)|| sign(mul(convex_hull_p0,a,b))
== 0 && dis2(convex_hull_p0,a) < dis2(convex_hull_p0,b);
}

int convex_hull(Point* a,int n,Point* b){
    if(n < 3){
        return -1;
    }

    int i;
    for(i = 1 ; i < n ; ++i){
        if(sign(a[i].x - a[0].x) < 0 || (sign(a[i].x - a[0].x) == 0 &&
sign(a[i].y < a[0].y) < 0 )){
            swap(a[i],a[0]);
        }
    }

    convex_hull_p0 = a[0];
    sort(a,a+n,convex_hull_cmp);//排序

    int newn = 2;
    b[0] = a[0];
    b[1] = a[1];

    for(i = 2 ; i < n ; ++i){
        while(newn > 1 && sign(mul(b[newn-1],b[newn-2],a[i])) >= 0){
            newn--;
```

```
        }

        b[newn++] = a[i];
    }

    return newn;
}
double getarea()
{
    double sum = 0;
    for(int i = 0;i < n;i++){
    sum += (p[i]^p[(i+1)%n]);
    }
    return fabs(sum)/2;
}

int main(){
    scanf("%d", &n);
    int i;
    for(i = 0 ; i < n ; ++i){
        scanf("%lf %lf", &p[i].x, &p[i].y);
    }
        n = convex_hull(p,n,p);
        p[n] = p[0];
        double ans = 0;
        for(i = 0 ; i < n ; ++i){
            ans += dis(p[i],p[i+1]);
        }
        printf("%.2lf %.2lf\n",ans, getarea());
    return 0;
}
```

C6

A C6-A*B!

简单FFT大数乘积模板，带负数

B C6-定向越野

时间限制：1000ms 内存限制：65536kb

题目描述

为了锻炼身体，某楠参加了一个定向越野比赛，定向越野是利用地图和指北针导航的一项竞技运动，通常由起点出发，在多个点标处打卡，再返回终点。但是非酋某楠的指北针居然是坏的，所以只能靠记住来时的方向和各个点的坐标来判断下一步。现在希望你能够帮忙判断下一步是左转还是右转。对于每次转弯输出一个字符，左转输出'L'，右转输出'R'，直走不输出。

输入

多组数据输入

每组数据第一行一个数n，n表示按顺序经历的点的数量，包括起点、各个点标以及终点。1<n<10000

接下来n行每行两个整数为点的坐标，均在INT范围内。

输出

每组数据一行，每次转弯的方向'L'或'R'，中间用空格分隔

输入样例

```
 5
 0 0
-1 1
 0 1
-1 2
 0 3
```

输出样例

```
R L R
```

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
int n, x[20000], y[20000];
int ndx, ndy;
using namespace std;
    int main()
    {
        memset(x, 0, sizeof(x));
        memset(y, 0, sizeof(y));
        while(~scanf("%d", &n))
        {
            for (int i = 0; i < n; i++) {
            scanf("%d %d", &x[i], &y[i]);
        }
        int ldx = x[1] - x[0], ldy = y[1] - y[0];

        for (int i = 2; i < n; ++i)
        {
            ndx = x[i] - x[i - 1]; ndy = y[i] - y[i - 1];
            long long flag = 1LL * ldx * ndy - 1LL * ldy * ndx;
            if (flag < 0) printf("R ");
            else if(flag > 0) printf("L ");
            ldx = ndx;
            ldy = ndy;
        }
        printf("\n");
    }
    return 0;
 }
```

C C6-危机合约

时间限制：1000ms 内存限制：65536kb

题目描述

一天起床，你突然发现自己成为了整合运动的一员，作为火刀哥的手下前去探路。由于危机合约的特殊性，博士只能布置没有阻挡数的干员，每路过一个干员就会受到一次他的攻击，你的目的就是在不被干掉的情况下，从位于最左第一列某点的红色出生点走到位于最右某点的蓝色目的地。作为一个普通宿主士兵，你只能走向右上，右，右下三个格子。

输入

第一行n和m，表示地图有n行m列 (n,m<100)

第二行h，a和b，h表示你现有的血量，红色出生点在第0列a行，蓝色目的地在第m+1列b行 (1<=a,b<=n)

接下来n行，每行m列，其中'*'表示这个点不能走，数字表示这个点上干员对你的伤害，范围0到9

输出

如果能够活着走到目的地，则输出剩余血量

如果已经死亡，则输出"doctor win"

输入样例

```
3 5
20 1 3
0 1 2 * 4
2 3 * 3 5
6 1 2 * 4
```

输出样例

```
10
```

样例解释


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

```cpp
const int INF = -1e9;
int d[107][107], dp[107][107], flag[107][107];

int main(int argc, const char * argv[]) {

    ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    int n, m, h, a, b;
    cin >> n >> m >> h >> a >> b;
    for (int i = 0; i <= n + 1 ; i++) {
        for (int j = 0; j <= m + 1; j++) {
            dp[i][j] = INF;
            d[i][j] = INF;
        }
    }
    char c;
    for (int i = 1; i <= n ; i++) {
        for (int j = 1; j <= m; j++) {
            cin >> c;
            if (c != '*') {
                d[i][j] = c - '0';
            }
        }
    }
    dp[a][0] = h;
    flag[a][0] = 1;
    d[b][m + 1] = 0;
    for (int j = 1; j <= m + 1; j++) {
        for (int i = 1; i <= n; i++) {
            if (d[i][j] == INF) {
                continue;
            }
            if (flag[i - 1][j - 1]) {
                dp[i][j] = max(dp[i][j], dp[i - 1][j - 1]);
                flag[i][j] = 1;
            }
            if (flag[i][j - 1]) {
                dp[i][j] = max(dp[i][j], dp[i][j - 1]);
                flag[i][j] = 1;
            }
            if (flag[i + 1][j - 1]) {
                dp[i][j] = max(dp[i][j], dp[i + 1][j - 1]);
                flag[i][j] = 1;
            }
            dp[i][j] -= d[i][j];
        }
    }

    if (dp[b][m + 1] <= 0) {
        printf("doctor win\n");
    } else printf("%d\n", dp[b][m + 1]);

    return 0;
}
```

D C6-不Nan的过河

时间限制：1000ms 内存限制：65536kb

题目描述

某楠也要过Zexal过的那条河，通过借助河中间的石砖过到河对岸去，这些石砖以直线排列。河的长度为L，当某楠走到或跨过坐标为L的点时，就算到达了河对岸。但是强迫症的某楠最多只能跨m次，请你计算某楠过河最长的一步最少是多少。注意从岸边迈向石头和从石头迈向岸边也算1步。

输入

多组数据输入

每组数据第一行有3个正整数L，n，m，L表示河的宽度，n表示有n个石砖，m表示某楠最多只能跨m步。(1≤L≤1e9,1≤n≤1e5,1≤m≤1e5)

第二行有n个不同的正整数分别表示这n个石砖在数轴上的位置（所有相邻的整数之间用一个空格隔开。

输出

每组数据输出一个整数，表示某楠迈的最长一步的最小距离。

输入样例

```
5 2 3
1 3
```

输出样例

```
2
```

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
using namespace std;
#define N 100007
int a[N], L, n, m;

int check(int t) {

    int cnt = 0, i = 1, cur = a[0];
    while (i < n) {
        if (cur + t < a[i]) {
            return 0;
        }
        while (a[i] <= cur + t && i < n) {
            i++;
        }
        cur = a[i - 1];
        cnt++;
    }
    if (cnt > m) {
        return 0;
    }
    return 1;
}

int main(int argc, const char * argv[]) {

```

```
    while (~scanf("%d %d %d", &L, &n, &m)) {
        a[0] = 0;
        for (int i = 1; i <= n; i++) {
            scanf("%d", a + i);
        }
        a[n + 1] = L;
        n = n + 2;
        sort(a, a + n);
        int l = 1, r = L + 2;
        while (l + 1 < r) {
            if (check((l + r) / 2)) {
                r = (l + r) / 2;
            } else l = (l + r) / 2;
        }
        printf("%d\n", r);
    }

    return 0;
}
```

E C6-线段交点

时间限制：1000ms 内存限制：65536kb

题面

有两条线段，求线段的交点。

输入

多组输入数据

每组数据两行

每行两个整数x，y，分别表示一条线段的x坐标，y坐标(−100≤x,y≤100)

输出

每组数据输出一行，两个数表示交点的坐标，中间用空格隔开。如果没有交点，或者线段重合，输出 `none`

输入样例

```
0 0 1 1
1 0 0 1
0 0 2 2
1 1 3 3
```

输出样例

```
0.50 0.50
none
```

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

```cpp
#include<cmath>
const int maxn = 1e3+7;
using namespace std;
const double p1 = acos(-1.0);
const double eps = 1e-8;
struct Point{
    double x, y;
    Point(double x=0, double y=0):x(x),y(y){}
    bool operator < (const Point &b) const
    {
        return x<b.x-eps || (fabs(x-b.x)<eps&&y<b.y-eps);
     }
};
typedef Point Vector;
Point operator + (Point A, Point B ) {return Point(A.x+B.x, A.y+B.y);}
Point operator - (Point A, Point B ) {return Point(A.x-B.x, A.y-B.y);}
Point operator * (Point A, double B ) {return Point(A.x*B, A.y*B);}
Point operator / (Point A, double B ) {return Point(A.x/B, A.y/B);}
int dcmp(double x)
{
    if(fabs(x) <eps) return 0;
    return x<0 ? -1 : 1;
}
int operator == (const Point &a, const Point &b)
{
    return dcmp(a.x-b.x)==0&&dcmp(a.y-b.y)==0;
}
double dot(Vector a, Vector b)
{
    return a.x*b.x+a.y+b.y;
}
double length(Vector a)
{
    return sqrt(dot(a, a));
}
double cross(Vector a, Vector b)
{
    return a.x*b.y-a.y*b.x;
}
double DistanceToLine(Point P, Point A, Point B)
{
    Vector v1 = B-A, v2 = P-A;
    return fabs(cross(v1, v2))/length(v1);
}
Point GetLineIntersection(Point P, Vector V, Point Q, Point W)
{
    Vector u = P-Q;
    double t = cross(W, u)/cross(V ,W);
    return P+V*t;
}
bool betw(double l, double r, double x)
{
    return min(l , r) <= x + eps && x <= max(l, r) +eps;
}
bool intersect_1d(double a, double b, double c, double d)
{
    if(a>b) swap(a, b);
    if(c>d) swap(c, d);
```

```
    return max(a, c) <= min(b, d) + eps;
}
bool intersect(Point a, Point b, Point c, Point d, Point& left, Point& right)
{
    if(!intersect_1d(a.x, b.x, c.x, d.x) || !intersect_1d(a.y, b.y, c.y, d.y))
        return 0;
    double zn = cross(b-a, d-c);
    if(abs(zn) < eps)
    {
        if(fabs(DistanceToLine(c, a, b)) > eps || fabs(DistanceToLine(a, c, d)) > eps)
            return 0;
        if(b<a)
            swap(a, b);
        if(d < c)
            swap(c, d);
        left = max(a, c);
        right = min(b, d);
        return 1;
    }else{
        left = right = GetLineIntersection(a, b-a, c, d-c);
        return betw(a.x, b.x, left.x) && betw(a.y, b.y, left.y) && betw(c.x, d.x, left.x) && betw(c.y, d.y, left.y);
    }
}
int main()
{
    Point a, b, c, d, l, r;
    ios::sync_with_stdio(false);
    cin.tie(0);cout.tie(0);
    while(cin >> a.x >> a.y >> b.x >> b.y >> c.x >> c.y >> d.x >> d.y)
    {
        if(intersect(a, b, c, d, l, r))
        {
            if(fabs(l.x - r.x) < eps && fabs(l.y - r.y) < eps)
                cout << l.x << ' ' << l.y << '\n';
            else cout << "none" << '\n';
        }else
        cout << "none" << '\n';
    }
}
```

F C6-直线

时间限制：1000ms 内存限制：65536kb

通过率：20/33 `(60.61%)` 正确率：20/82 `(24.39%)`

题面

二维平面上有n个黑点m个白点，现在请问是否存在一条直线使得所有的黑点白点分别在直线两侧（黑点都在一侧，白点都在另一侧）。

输入

对于每组数据：

第一行两个正整数n、m。(1≤n,m≤1001≤n,m≤100)

接下来n行，每行两个正整数x,y，表示一个黑点的xy坐标(1≤x,y≤10001≤x,y≤1000)

接下来m行，每行两个正整数x,y，表示一个白点的xy坐标(1≤x,y≤10001≤x,y≤1000)

输出

每组数据输出一行，存在输出`YES`，否则输出`NO`

输入样例

```
3 3
100 700
200 200
600 600
500 100
500 300
800 500
3 3
100 300
400 600
400 100
600 400
500 900
300 300
```

输出样例

```
YES
NO
```

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

struct Point
{
    double x,y;
    Point (double x=0, double y=0) :x(x),y(y) {}
    void show() { cout<<"("<<x<<","<<y<<")"<<endl; }
};
typedef Point Vector;

bool operator < (const Point& A, const Point& B )
{
    return A.x < B.x || (A.x == B.x && A.y < B.y);
}
Vector operator - ( Point A, Point B) { return Vector(A.x-B.x, A.y-B.y); }
double Cross(Vector A, Vector B) { return A.x*B.y - B.x*A.y; }

const double eps = 1e-10;
int dcmp(double x)
{
```

```
    if(fabs(x) < eps) return 0;
    return x>0? 1: -1;
}

int CH ( Point* p, int n, Point* ch)
{
    sort(p, p+n);
    int m = 0;
    for(int i = 0; i<n; i++)
    {
        while(m > 1 && Cross( ch[m-1] - ch[m-2], p[i] - ch[m-2]) <= 0) m--;
        ch[m++] = p[i];
    }
    int k = m;
    for(int i = n-2; i>=0; i--)
    {
        while(m > k && Cross( ch[m-1] - ch[m-2], p[i] - ch[m-2]) <= 0) m--;
        ch[m++] = p[i];
    }
    ch[m++] = ch[0];
    if(n > 1) m--;
    return m;
}

Point black[120],white[120],cw[120],cb[120];
int nb,nw,n,m;

void reset()
{
    memset(black,0,sizeof(black));
    memset(white,0,sizeof(white));
    memset(cw,0,sizeof(cw));
    memset(cb,0,sizeof(cb));
}

bool SideCross(Point a1, Point a2, Point b1, Point b2)
{
    double c1 = Cross(a2 - a1, b1 - a1), c2 = Cross(a2 - a1, b2 - a1),
           c3 = Cross(b2 - b1, a1 - b1), c4 = Cross(b2 - b1, a2 - b1);
    return dcmp(c1)*dcmp(c2)<0 && dcmp(c3)*dcmp(c4)<0;
}

bool pInConvex(Point a, Point* ch, int n)
{
    int i;
    if(n==3)
    {
        if(a.x < min(ch[0].x, ch[1].x) || a.x > max(ch[0].x, ch[1].x)) return 0;
        if(a.y < min(ch[0].y, ch[1].y) || a.y > max(ch[0].y, ch[1].y)) return 0;
    }
    for(i=1;i<n;i++)
    {
        if(Cross( ch[i]-ch[i-1], a-ch[i-1] ) < 0 ) return 0;
    }
```

```
    return 1;
}

int solve()
{
    int i,j;
    if(n==1&&m==1) return 1;
    else if(n==1)
    {
        nw=CH(white, m, cw);
        if(pInConvex(black[0], cw, nw)) return 0;
    }
    else if(m==1)
    {
        nb=CH(black, n, cb);
        if(pInConvex(white[0], cb, nb)) return 0;
    }
    else
    {
        nb=CH(black, n, cb);
        nw=CH(white, m, cw);
        for(i=1;i<nb;i++)
        {
            for(j=1;j<nw;j++)
                if(SideCross( cb[i], cb[i-1], cw[j] ,cw[j-1] )) return 0;
        }
        for(i=0;i<n;i++)
            if(pInConvex(black[i], cw, nw)) return 0;
        for(i=0;i<m;i++)
            if(pInConvex(white[i], cb, nb)) return 0;
    }
    return 1;
}

int main()
{
    int i,tx,ty;
    while(cin>>n>>m)
    {
        reset();
        for(i=0;i<n;i++)
        {
            cin>>tx>>ty;
            black[i]=Point(tx,ty);
        }
        for(i=0;i<m;i++)
        {
            cin>>tx>>ty;
            white[i]=Point(tx,ty);
        }
        int ans=solve();
        if(ans)
            cout<<"YES"<<endl;
        else
            cout<<"NO"<<endl;
    }
    return 0;
}
```

G C6-逆序对

时间限制：1000ms 内存限制：65536kb

题面

逆序对的定义：在一个序列aa中，如果i<j且 ai>aj 那么ai aj就是一个逆序对。

问相距最远的逆序对的距离（j-i）。如果没有逆序对，输出0。

输入

第一行一个数n，表示序列的长度。(1≤n≤1e5)

接下来一行，n个整数，保证在int范围内

输出

一行一个数，表示最远逆序对的距离

输入样例

```
4
4 3 5 1
```

输出样例

```
3
```

```
#include<stdio.h>
#include<algorithm>
#include<iostream>
using namespace std;
struct x
{
    int n;
    int p;
}num[100001];
bool cmp(x a, x b)
{
    return a.n < b.n;
}
int main()
{
    int n;
    scanf("%d", &n);
    for(int i = 0; i < n; i++)
    {
        scanf("%d", &num[i].n);
        num[i].p = i + 1;
    }
    sort(num, num + n, cmp);
    int MAX = -1, ans = 0;
    for(int i = 0; i < n; i++)
    {
        //printf("%d %d\n", num[i].n, num[i].p);
```

```
        MAX = max(MAX, num[i].p);
        ans = max(ans, MAX - num[i].p);
    }
    printf("%d", ans);
}
```