Bài 1. Mật khẩu (7 điểm)

➢ **Phân bổ điểm**

- Có 20% số test ứng với 20% số điểm của bài có $m \leq 10; n_i \leq 3; k_i \leq 2 \times 10^5$;
- Có 20% số test ứng với 20% số điểm của bài có $m \leq 2 \times 10^5;\ n_i + k_i \leq 50$;
- Có 30% số test khác ứng với 30% số điểm của bài có $m \leq 5000;\ n_i, k_i \leq 5000$;
- Có 30% số test còn lại ứng với 30% số điểm của bài có giới hạn như dữ kiện bài ra.

➢ **Hướng dẫn thuật toán**

Subtask 1:

- Nếu $n = 1$, ta có: $\rho(1,k) = k.\rho(0,k) = k$;
- Nếu $n = 2$, ta có: $\rho(2,k) = \frac{k[\rho(0,k)+\rho(1,k)]}{2} = \frac{k(1+k)}{2}$;
- Nếu $n = 3$, ta có: $\rho(3,k) = \frac{k[\rho(0,k)+\rho(1,k)+\rho(2,k)]}{3} = \frac{k[1+k+\frac{k(k+1)}{2}]}{3} = \frac{k(1+k)(2+k)}{6}$.

Căn cứ vào các công thức trên, tính kết quả của biểu thức lấy theo modulo $10^9 + 7$.

Độ phức tạp thuật toán: $O(m)$.

Subtask 2:

Với $n \geq 1$, ta có:

$$\rho(n,k) = \frac{k[\rho(0,k)+\rho(1,k)+\cdots+\rho(n-1,k)]}{n} = \frac{n-1}{n} \cdot \frac{k[\rho(0,k)+\rho(1,k)+\cdots+\rho(n-2,k)]}{n-1} + \frac{k\rho(n-1,k)}{n}$$
$$= \frac{(n-1+k)\rho(n-1,k)}{n}.$$

Từ công thức truy hồi ta xây dựng hàm đệ quy để tính giá trị biểu thức và lấy kết quả theo modulo $10^9 + 7$. Nhưng với cách tính toán xây dựng hàm đệ quy để tính biểu thức, kết quả có thể bị tràn số khi $n + k > 50$.

Độ phức tạp thuật toán: $O(m.n)$.

Subtask 3:

Từ công thức truy hồi ở trên, ta có:

$$\rho(n,k) = \frac{(n-1+k)(n-2+k)\ldots(1+k)k}{n!}$$

Đến đây ta tính $\rho(n,k)\%(10^9+7)$ theo các bước sau:

Bước 1. Dùng vòng lặp tính:

$$a = (n-1+k)(n-2+k)\ldots(1+k)k\ \%\ (10^9+7), b = n!\,\%(10^9+7).$$

Bước 2. Dùng định lí Ferma nhỏ tính c là nghịch đảo mô đun của $n!$ (theo modulo $10^9 + 7$).

Định lí Ferma nhỏ: *Với số nguyên dương u và số nguyên tố p, ta có: $u^{p-1} = 1(mod\ p)$.*

Từ đó suy ra u^{p-2} là nghịch đảo mô đun của u theo mô đun p.

Do $p = 10^9 + 7$ là số nguyên tố nên $c = b^{p-2}\%p$ (tính bằng phương pháp chia để trị).

Bước 3. Kết quả của $\rho(n,k)\%p$ là: $(a.c)\%p$.

Độ phức tạp thuật toán: $O(m.n)$.

Subtask 4:

Ta có: $\rho(n,k) = \frac{(n-1+k)(n-2+k)\ldots(1+k)k}{n!} = \frac{(n-1+k)!}{n!(k-1)!}$.

Để giảm độ phức tạp tính toán ta tính trước 2 mảng sau:

$fc[i] = i!\%p, rfc[i] = (fc[i])^{p-2}\%p$ (với $p = 10^9 + 7, i = 0,1, \dots ,400000$).

Khi đó: $\rho(n,k)\%p = (fc[n-1+k].rfc[n]\%p).rfc[k-1]\%p$.

Tham khảo code lời giải mẫu để biết cách cài đặt thuật toán.

Độ phức tạp thuật toán: $O(m.(n+k))$.

Bài 2. Giá trị nhỏ nhất (7 điểm)

➢ **Phân bổ điểm**

- Có 20% số test ứng với 20% số điểm của bài có $m, n \leq 30$;
- Có 20% số test ứng với 20% số điểm của bài có $m, n \leq 100$;
- Có 30% số test khác ứng với 30% số điểm của bài có $m, n \leq 300$;
- Có 30% số test còn lại ứng với 30% số điểm của bài có giới hạn như dữ kiện bài ra.

➢ **Hướng dẫn thuật toán**

Subtask 1:

Dễ thấy giá trị X của mỗi lưới ô vuông con thỏa mãn yêu cầu là giá trị của một ô nằm trong lưới ô vuông con đó. Vì vậy ta dùng 4 vòng lặp lồng nhau để duyệt vét cạn các lưới ô vuông con theo mô tả của bài toán để tìm kết quả.

Độ phức tạp thuật toán: $O(m.n.(h.w)^2)$.

Subtask 2:

Căn cứ vào tính chất trung vị của dãy các số, ta đưa các giá trị của lưới ô vuông con vào mảng $d[1..hw]$ và sắp xếp theo thứ tự không giảm thì số X cần tìm là $d[\frac{hw+1}{2}]$ ($\frac{hw+1}{2}$ là phép chia nguyên).

Độ phức tạp thuật toán: $O(m.n.h.w.log2(hw))$.

Subtask 3, 4:

Đây chính là bài toán tìm trung vị của dãy số có các giá trị trong lưới ô vuông con kích thước $h \times w$, khi sắp xếp các số này theo thứ tự không giảm. Ta thấy rằng kết quả của bài toán nằm trong phạm vi từ $l = 0$ đến $r = 10^9$ nên ta tìm kiếm nhị phân trên kết quả trên đoạn này.

Gọi $d = \frac{h.w+1}{2}$, khi đó với mỗi giá trị $mid = \frac{l+r}{2}$ ta xây dựng hàm $check(mid)$ để kiểm tra xem mid có là trung vị một lưới ô vuông con kích thước $h \times w$ hay không. Việc làm này được thực hiện bằng cách đếm số lượng các số trong lưới nhỏ hơn hoặc bằng k nếu giá trị này nếu lớn hơn hoặc bằng d thì ta trả về cho hàm $check(mid)$ giá trị là $true$, ngược lại trả về kết quả $false$.

Để giảm độ phức tạp tính toán khi đếm số lượng các số trong lưới nhỏ hơn hoặc bằng mid, ta cần tính trược toán mảng cộng dồn 2 chiều $s[.][.]$ như sau:

$$s[i][j] = s[i-1][j] + s[i][j-1] - s[i-1][j-1] + (a[i][j] \leq mid), \forall i = 1..m, j = 1..n.$$

Trong đó: $s[0][j] = s[i][0], \forall i = 0..m, j = 0..n$.

Sau khi tính xong mảng $s[.][.]$, ta dễ dàng tính được số lượng các số trên lưới có giá trị nhỏ hơn hoặc bằng mid bằng hàm

$$getsum(i,j,u,v) = s[u][v] - s[i-1][v] - s[u][j-1] + s[i-1][j-1],$$

với $(i,j), (u,v)$ lần lượt là ô trên trái, dưới phải của lưới ô vuông kích thước $h \times w$ đang xét.

Việc cài đặt hàm $check(mid)$ và tìm kiếm nhị phân kết quả xin tham khảo code lời giải mẫu.

Độ phức tạp thuật toán: $O(m.n.log2(10^9))$.

Code tham khảo:

Bài 1

```
#include <bits/stdc++.h>
#define NAME "password."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
typedef long long ll;
const ll mod = 1e9+7;
ll q, n, k, fc[400001], rfc[400001];
ll pow1(ll a, ll st)
{
    ll ans = 1;
    while(st)
    {
        if(st&1) ans = (ans * a) % mod;
        a = (a * a) % mod;
        st>>=1;
    }
    return ans;
}

int main()
{
    fi >> q;
    fc[0] = 1;
    for(int i=0; i<400000; ++i)
    fc[i + 1] = (fc[i] * ll(i + 1)) % mod;
    rfc[400000] = pow1(fc[400000], mod - 2);
    for (int i = 400000 - 1; i >= 0; i--)
    rfc[i] = (rfc[i + 1] * ll(i + 1)) % mod;
    for(int i=0; i<q; ++i)
    {
        fi >> n >> k;
        ll ans = ((fc[n-1+k] * rfc[k-1]) % mod) * rfc[n] % mod;
        fo << ans << "\n";
    }
```

```
    return 0;
}
```

Bài 2

```
#include <bits/stdc++.h>
#define NAME "minimum."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int m,n,p,q,res;
int a[1001][1001],s[1001][1001];
int d;

void enter(){
    fi>>m>>n>>p>>q;
    for (int i=1;i<=m;++i)
        for (int j=1;j<=n;++j)
            fi>>a[i][j];
    d=(p*q+1)/2;
}

int getsum(int i,int j,int u,int v){
    return s[u][v]-s[i-1][v]-s[u][j-1]+s[i-1][j-1];
}

bool check(int k){
    for (int i=1;i<=m;++i)
        for (int j=1;j<=n;++j)
            s[i][j]=s[i-1][j]+s[i][j-1]-s[i-1][j-1]+(a[i][j]<=k);
    for (int i=1;i<=m-p+1;++i)
        for (int j=1;j<=n-q+1;j++){
            int u=i+p-1,v=j+q-1;
            if (getsum(i,j,u,v)>=d) return true;
        }
    return false;
}

void solve(){
```

```
    res=1e9;
    int left=0,right=1e9;
    while (left<right){
        int mid=(left+right)/2;
        if (check(mid)){
            right=mid;
            res=mid;
        }
        else
            left=mid+1;
    }
    fo<<res;
}

int main(){
    enter();
    solve();
    return 0;
}
```