Easy Path

Limits: 1s, 1.0 GB

There is a rectangular lake with N rows and M columns and you are standing on a stone in the top left corner (1, 1) of the lake and you want to go to the stone at the bottom right corner (N, M). You can consider the lake as a 2D grid. But there is a problem. You can only jump to adjacent right and bottom cell from the current one. You are wearing your new year dress and don't want to get wet. So you only will jump from one stone to another stone.

But you are a stubborn lad. And you have some lucky cells. And you say, if you want to cross the lake then you at least have to go through at least one of your lucky cells otherwise you won't go through that path.

You are given the lake's dimension and the grid. **"#"** means a stone and **"."** means water.

Now you are given Q queries. In every query, you are given the lucky cell's positions. For each query you need to output the number of paths that you can follow to get to the bottom right corner.

Input

The first line will contain two integer N and M, number of rows and columns.

Then there will be N lines each having M characters representing the lake. Then Q which is the number of queries.

For each query, the first line will be L the number of lucky cells. Next L lines. The i^{th} line will contain (x_i, y_i).

$1 \le N * M \le 10^6$
$1 \le Q \le 2000$
$1 \le L \le 1000$
$1 \le x_{i+1} < x_i \le N$
$1 \le y_{i-1} < y_i \le M$

Output

For each query, print the number of paths explained above.

As the number can be very big print the answer modulo **1000000007 ($10^9 + 7$)**.

Samples

Input	Output
2 2 ## .# 2 1 1 2 1 2 1	

```cpp
#include<bits/stdc++.h>
using namespace std;
#define ll long long
const ll mod = 1000000007;
int x[1005],y[1005];
int main(){
    ios::sync_with_stdio(false); cin.tie(0);
    int n,m; cin>>n>>m;
    string s[n+5];
    for(int i=0; i<n; i++){
        cin>>s[i];
    }

    int a[n+5][m+5];
    ll dp[n+5][m+5], mp[n+5][m+5];
    for(int i=0; i<=n+2; i++){
        for(int j=0; j<=m+2; j++){
            dp[i][j]=mp[i][j]=a[i][j]=0;
        }
    }

    for(int i=0; i<n; i++){
        for(int j=0; j<m; j++){
            if(s[i][j]=='#')a[i+1][j+1]=1;
            else a[i+1][j+1]=0;
        }
    }

    dp[1][1] = 1;
    for(int i=1; i<=n; i++){
        for(int j=1; j<=m; j++){
            if(i==1&&j==1)continue;
            if(a[i][j]==0)dp[i][j]=0;
            else{
                dp[i][j] = dp[i-1][j]+dp[i][j-1];
                dp[i][j] %= mod;
            }
        }
    }

    mp[n][m]=1;
    for(int i=n; i>=1; i--){
        for(int j=m; j>=1; j--){
            if(i==n&&j==m)continue;
            if(a[i][j]==0)mp[i][j]=0;
            else{
                mp[i][j] = mp[i][j+1]+mp[i+1][j];
                mp[i][j] %= mod;
            }
        }
    }

    int q; cin>>q;
    while(q--)
    {
        int p; cin>>p;
        for(int k=1; k<=p; k++){
            cin>>x[k]>>y[k];
        }

        ll ans = 0;
        for(int k=1; k<=p; k++){
            ans += (dp[x[k]][y[k]]*mp[x[k]][y[k]])%mod;
            ans %= mod;
        }
        cout << ans << endl;
    }

    return 0;
}
```