**Question**

**1**

Max. Marks 100.00

**Fibonacci with GCD**

Monk is really fond of [Recurrence Relations](https://en.wikipedia.org/wiki/Recurrence_relation), and he likes to study their characteristics in any possible manner. As you might know, his favorite one among all such recurrences is the famous Fibonacci series. For those of you who haven't,

Fibonacci series is defined as:

F(N)F(N) = F(N−1)+F(N−2)F(N−1)+F(N−2) ∀ N≥3∀ N≥3

F(1)=1F(1)=1 , F(2)=1F(2)=1.

Now, in addition to such sequences, Number Theory is a really interesting concept, and Monk loves solving problems of those kinds too. GCD is a nice concept within the scope of this topic, and is defined to be :

The GCD of two numbers is the greatest common divisor of those numbers. Eg: GCD(2,4)=2. Here, he has challenged you to the following task as he feels that this one is amazingly challenging . So, this is how it goes :

You are given NN integers, A1,A2,...,ANA1,A2,...,AN and QQ queries. In each query, you are given two integers LL and R(1≤L≤R≤N)R(1≤L≤R≤N). For each query, you have to find the value of:

GCD(F(AL),F(AL+1),F(AL+2)......F(AR))%109+7GCD(F(AL),F(AL+1),F(AL+2)......F(AR))%109+7

Can you do it ?

**Constraints:**   
1≤N≤1051≤N≤105  
1≤Q≤1051≤Q≤105  
1≤Ai≤1091≤Ai≤109  
1≤L≤R≤N1≤L≤R≤N

**Format of the input file:**First line : Two integers N and Q.  
Second line : N space separated integers denoting array A.  
Next Q lines : Two space separated integers L and R.

**Format of the output file:**Output the result of each query in a separate line.

**Sample Input**

3 2

2 4 8

1 3

2 3

**Sample Output**

1

3

**Explanation**

For query 1: GCD(F(2) , F(4), F(8))= GCD(1,3,21)=1 For query 2: GCD(F(4), F(8))= GCD(3,21)=3

**Note:**Your code should be able to convert the sample input into the sample output. However, this is not enough to pass the challenge, because the code will be run on multiple test cases. Therefore, your code must solve this problem statement.

Time Limit: 2.0 sec(s) for each input file

Memory Limit: 256 MB

Source Limit: 1024 KB

Marking Scheme: Marks are awarded if any testcase passes

Allowed Languages: C, C++, Clojure, C#, D, Erlang, F#, Go, Groovy, Haskell, Java, Java 8, JavaScript(Rhino), JavaScript(Node.js), Lisp, Lisp (SBCL), Lua, Objective-C, OCaml, Octave, Pascal, Perl, PHP, Python, Python 3, R(RScript), Racket, Ruby, Rust, Scala 2.11.8, Swift, Visual Basic

-------NO ACEPTADO---------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static long gcd(long a, long b)

{

if (a == 0)

return b;

return gcd(b % a, a);

}

static List<long> fibonacci(long n)

{

List<long> fib = new List<long>();

fib.Add(1);

fib.Add(1);

for (int i = 2; i <= n; i++)

{

fib.Add((fib[i - 1] + fib[i - 2]) % 1000000007);

}

return fib;

}

static void Main(string[] args)

{

string[] input = Console.ReadLine().Split(' ');

int n = int.Parse(input[0]);

int q = int.Parse(input[1]);

long[] a = Array.ConvertAll(Console.ReadLine().Trim().Split(' '), e => long.Parse(e));

//int n = 3, q= 2;

//long[] a = { 2, 4, 8 };

long max = a.Max();

List<long> fibo = fibonacci(max);

while (q-- > 0)

{

//int l = 2, r = 3;

string[] lr = Console.ReadLine().Split(' ');

int l = int.Parse(lr[0]);

int r = int.Parse(lr[1]);

l--; r--;

long GCD = 0;

for (int i = l; i <= r; i++)

{

GCD = gcd(GCD, fibo[(int)(a[i] - 1)]) % 1000000007;

}

Console.WriteLine(GCD);

}

Console.ReadLine();

}

}

}

**Question**

**2**

Max. Marks 100.00

**Turtle's Path**

To be good at problem solving, Monk thinks that Graphs are a topic one can definitely not skip. They have a variety of applications in Networks, flows , Routing and so on.

He has prepared some really interesting problems based on the same for talented programmers like you. He really adores his friends, and has chosen one of this close friends Mona as the main character for this task. So, this is how it goes :

You've got a table of size **N\*M** containing positive integers. We'll consider the table rows numbered from top to bottom **1** through **N**, and the columns numbered from left to right **1** through **M**. Then we'll denote the cell in row **x** and column **y** as **(x, y)**.

Initially cell **(1, 1)** contains one turtle named Mona. Mona wants to get to cell **(N, M)**. Some cells of the table have obstacles. A cell is considered to be containing an **obstacle** if value of the cell is a **NON-PRIME NUMBER**. That means that Mona can only move through PRIME Numbers. It is guaranteed that upper left corner **(1,1)** contains a prime number.

Mona can go from cell **(x, y)** to one of three cells **(x + 1, y )**, **( x , y + 1 )** and **( x + 1, y + 1 )** only if the required cell doesn't contain an obstacle. Help him find the number of ways in which it can go from cell (1, 1) to cell (N, M).

In addition, you need to print the lexicographical largest path to reach cell (N,M).

**Note:** A cell (x1,y1x1,y1) is lexicographical larger than another cell (x2,y2x2,y2) if either x1>x2x1>x2 or if x1=x2x1=x2 and y1>y2y1>y2. A path XX is lexicographical larger than another path YY, if the first cell that does not match is lexicographical larger in XX than in YY. For example, cell (3,2)(3,2) is lexicographical larger than cell (3,1)(3,1).   
Let, Path Y: (1,1)(2,1)(3,1)(3,2)(3,3)(1,1)(2,1)(3,1)(3,2)(3,3)  
Path X: (1,1)(2,1)(3,2)(3,3)(1,1)(2,1)(3,2)(3,3)  
Path X is lexicographical larger than another path Y, because the first cell that does not match (i.e. (3,23,2) in X and (3,13,1) in Y) is lexicographical larger in X than in Y.

**Input Format**

The first line contains two space separated integers, **N** (number of rows) and **M** (number of columns).

Then N lines follow, each containing **M** space separated integers.

**Constraints**

1 <= N,M <= 103  
2 <= A[x][y] <= 105 (the elements of the matrix)

**Output Format**

In the first line, print the total number of possible ways to reach **(N,M)** from **(1,1)**. Since this number may be too large, print the answer modulo **109 +7**.

Print the cell indexes (space separated) of each step of his lexicographically largest path in a new line .  
If no path exists, only print **0** in first line.

(See sample test case for clarification)

**Sample Input**

3 3

2 3 7

5 4 2

3 7 11

**Sample Output**

4

1 1

2 1

3 2

3 3

**Explanation**

There are four ways to reach (3,3) from (1,1).

Path 1. (1,1) (1,2) (1,3) (2,3) (3,3)  
Path 2. (1,1) (1,2) (2,3) (3,3)  
Path 3. (1,1) (2,1) (3,1) (3,2) (3,3)  
Path 4. (1,1) (2,1) (3,2) (3,3)

Lexicographical Order -> 4 > 3 > 2 > 1

**Note:**Your code should be able to convert the sample input into the sample output. However, this is not enough to pass the challenge, because the code will be run on multiple test cases. Therefore, your code must solve this problem statement.

Time Limit: 1.0 sec(s) for each input file

Memory Limit: 256 MB

Source Limit: 1024 KB

Marking Scheme: Marks are awarded if any testcase passes

Allowed Languages: C, C++, Clojure, C#, D, Erlang, F#, Go, Groovy, Haskell, Java, Java 8, JavaScript(Rhino), JavaScript(Node.js), Lisp, Lisp (SBCL), Lua, Objective-C, OCaml, Octave, Pascal, Perl, PHP, Python, Python 3, R(RScript), Racket, Ruby, Rust, Scala 2.11.8, Swift, Visual Basic

<https://www.hackerrank.com/contests/iit-bhu-opc/challenges/turtle-path>

----ACEPTADO DE EDITORIAL DE HACKERRANK---------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication2

{

class Program

{

static int[] prime = new int[100007];

static void sieve()

{

for (int i = 2; i \* i <= 100005; i++)

{

if (prime[i] == 0)

{

for (int j = i \* i; j <= 100005; j += i)

prime[j] = 1;

}

}

}

static int[,] a = new int[1005, 1005];

static long[,] dp = new long[1005, 1005];

static int[,] mark = new int[1005, 1005];

static int q = 0, n, m;

static KeyValuePair<int, int>[] ans = new KeyValuePair<int, int>[100005];

static void dfs(int i, int j, int k)

{

if (a[i, j] == 0 || i > n || j > m || mark[i, j] != 0 || q != 0)

return;

mark[i, j] = 1;

ans[k] = new KeyValuePair<int, int>(i, j); // mp(i, j);

if (i == n && j == m)

{

q = k;

return;

}

dfs(i + 1, j + 1, k + 1);

dfs(i + 1, j, k + 1);

dfs(i, j + 1, k + 1);

}

static void Main(string[] args)

{

int MOD = 1000000007;

string[] input = Console.ReadLine().Split(' ');

n = int.Parse(input[0]);

m = int.Parse(input[1]);

sieve();

for (int i = 0; i < n; i++)

{

string[] fila = Console.ReadLine().Trim().Split(' ');

for (int j = 0; j < m; j++)

{

a[i+1, j+1] = int.Parse(fila[j]);

if (prime[a[i+1, j+1]] == 0)

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;

dp[i, j] = (dp[i - 1, j] + dp[i, j - 1] + dp[i - 1, j - 1]) % MOD;

if (a[i, j] == 0)

dp[i, j] = 0;

}

}

Console.WriteLine(dp[n, m]);

dfs(1, 1, 1);

for (int i = 1; i <= q; i++)

{

// cout << ans[i].F << " " << ans[i].S << "\n";

Console.WriteLine(ans[i].Key + " " + ans[i].Value);

}

}

}

}