Lab9 Solution

YAO ZHAO

Lab9.A: Rectangle

- Seaflowery noticed a cockroach on a sheet. In order to kill it, she had to draw rectangles inside the sheet.
- To start with, she draws a $n \times m$ rectangle. Then, each time she draws, she can only draw rectangles **strictly inside** the last rectangle. She can draw rectangles any times, as long as the rectangle she draws has positive area.
- Now she wonders the number of possible patterns she can draw modulo $10^9 + 7$.
- Note that all the rectangles she draws must have their edges parallel to the coordinate axis and must have integer length and width. Also, strictly inside means that there will be no intersections(edges/points) between rectangles.
- Note that two patterns are considered different iff the final pictures are different.

Sample 1 Input

4 4

X	X	X	X
X	1	2	X
X	3	4	X
X	X	X	X

$\longrightarrow$

no draw $\longrightarrow$ 1

All possible 1*1 rectangles: (1,1) (2,2) (3,3) (4,4) $\longrightarrow$ 4

All possible 1*2 rectangles: (1,2) (1,3) (2,4) (3,4) $\longrightarrow$ 4

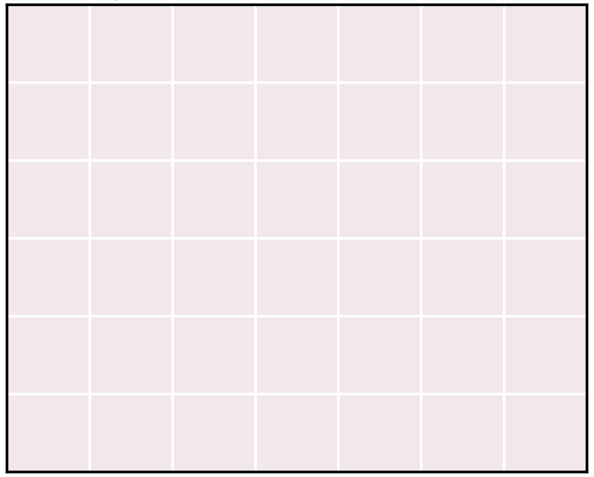
All possible 2*2 rectangles:(1,2,3,4) $\longrightarrow$ 1

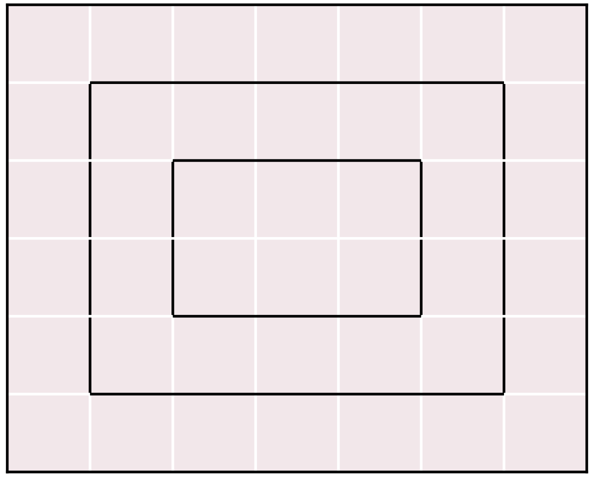
Sample 1 output

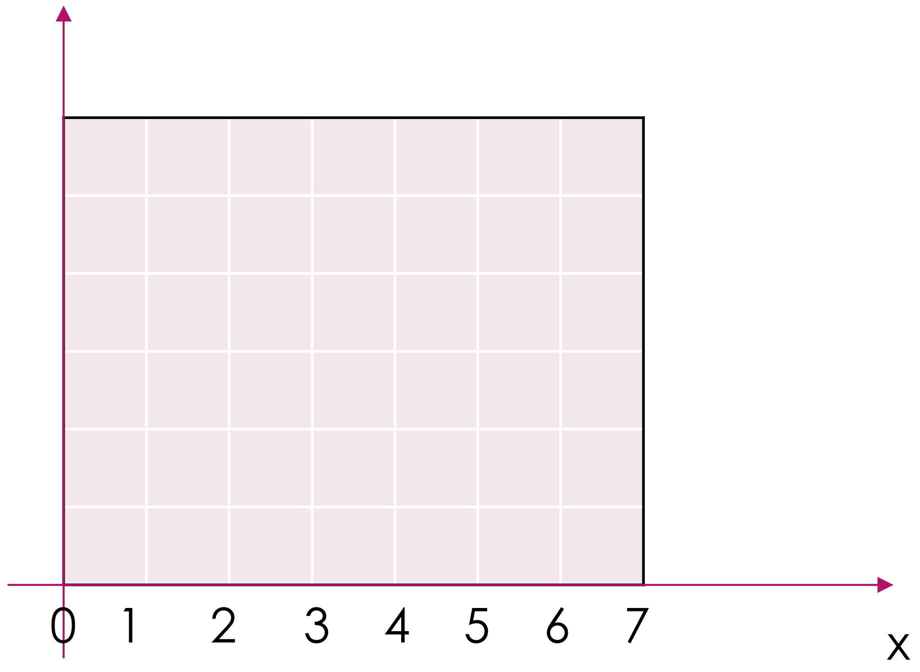
10

Maximum nesting depth: (min(n,m)+1)/2 = 3

n=6, m=7







Let f[i][j] represent the number of edges when the nest number is i, length is j in x direction.

Maximum nesting depth: (min(n,m)+1)/2 = 3

depth = 1: f[1][j] = 1
f[1][1] = 1,
f[1][2] = 1,
…
f[1][7] = 1

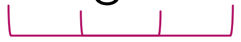
You should precomputation and record $\sum_{k=1}^{j-2} f[i-1][k] \; and \sum_{k=1}^{j-2} f[i-1][k] * k$

depth = 2: f[2][j]

f[2][1] = 0, if length is 1, no 2 nest

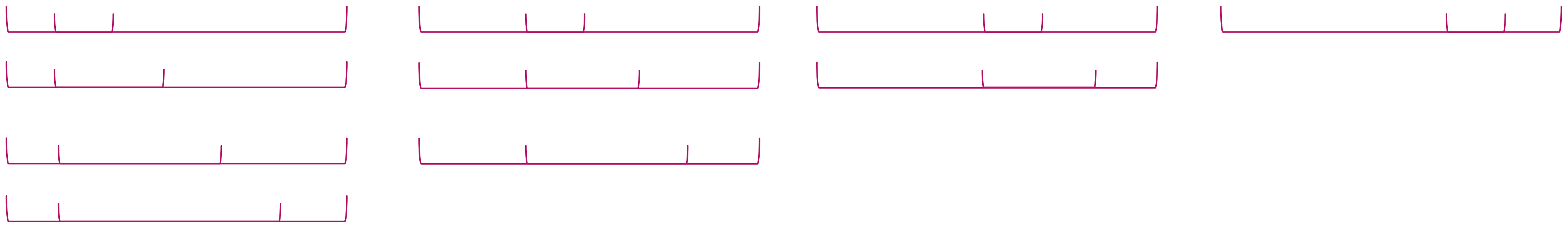
f[2][2] = 0, if length is 2, no 2 nest

f[2][3] = 1,



…

f[2][6] = 4* f[1][1] + 3*f[1][2] +…+1*f[1][4]



f[2][7] = 5* f[1][1] + 4*f[1][2] +…+1*f[1][5]

f[i][j] = $(j-1-k)\sum_{k=1}^{j-2} f[i-1][k] = (j-1)\sum_{k=1}^{j-2} f[i-1][k] - \sum_{k=1}^{j-2} f[i-1][k]*k$

You can precomputation and record $\sum_{k=1}^{j-2} f[i-1][k]\ and\ \sum_{k=1}^{j-2} f[i-1][k]*k$

Time complexity: $O(n^2)$

Total number of possible pattern: number of x * number of y
depth = 1: f[1][n]*f[1][m]
depth = 2: f[2][n]*f[2][m]
…
depth = i: f[i][n]*[i][m]

Total number: $\sum_{i=1}^{Maximum\ nesting\ depth} f[i][n] * f[i][m]$

Lab9.B: K-Valid

- A 01-sequence A is called $K-valid$ for a positive integer K, if there are no K consecutive 1s in the sequence.
- You are given two integers N, M. Satori wonders the number of sequences of length N that are $M-valid$, modulo 10^9+7.

Sample 1 Input

5 2

1	0	1	0	1
1	0	1	0	0
1	0	0	1	0
1	0	0	0	1
0	1	0	1	0
0	1	0	0	1
0	0	1	0	1
1	0	0	0	0
		…		
0	0	0	0	1
0	0	0	0	0

7 sequence contains more than 2 K,
but no K consecutive 1s

5 sequence only contains 1 K

1 sequence only contains 0 K

Sample 1 output

13

Let f[i][j] represent the number that the first i bit of the sequence has been filled in, and that the sequence has exactly j consecutive 1s at the end. There cannot be K 1s in a sequence, so we need to record how many 1s there are at the end of the sequence.

Sample 1 Input

5 3

f[0][0] = 1 f[0][1] = 0….f[0][2]=0

initial : ø

append a bit:
if the bit == 0: f[1][0] = f[0][0] + f[0][1] + f[0][2] = 1
if the bit ==1:   f[1][1] = f[0][0] = 1
                  f[1][2] = f[0][1] = 0

i=1

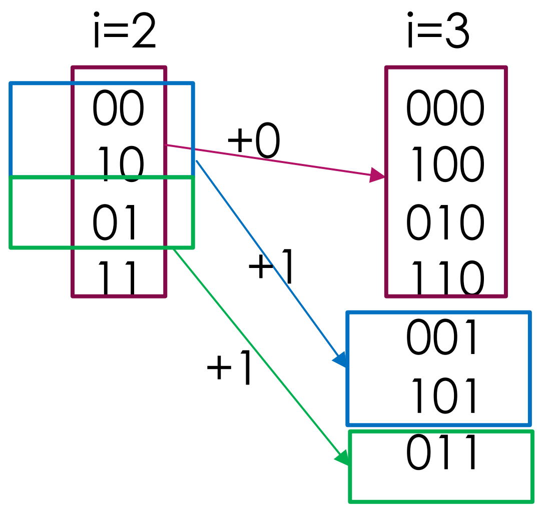
0
1

i=2

00
10
01
11

continue append a bit:
if the bit == 0: f[2][0] = f[1][0] + f[1][1] + f[1][2] = 2
if the bit ==1:   f[2][1] = f[1][0] = 1
                  f[2][2] = f[1][1] = 1

continue append a bit:
if the bit == 0: f[3][0] = f[2][0] + f[2][1] + f[2][2] = 4
if the bit ==1: f[3][1] = f[2][0] = 2
f[3][2] = f[2][1] = 1



f[i][0] = f[i-1][0]+f[i-1][1]+…f[i-1][k-1]
f[i][1] = f[i-1][0]
…
f[i][k-1] = f[i-1][k-2]

$$[f[i][0] \quad f[i][1] \quad \ldots f[i][k-1]] = [f[i-1][0] \quad f[i-1][1] \quad \ldots f[i-1][k-1]] \times \begin{bmatrix} 1 & 1 & & 0 & 0 \\ 1 & 0 & & 1 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & 0 & & 0 & 0 \end{bmatrix}$$

$$\text{answer} = [f[0][0] \quad f[0][1] \quad \ldots f[0][k-1]] \times \left.\underbrace{\begin{bmatrix} 1 & 1 & & 0 & 0 \\ 1 & 0 & & 1 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & 0 & & 0 & 0 \end{bmatrix}}_{K}^{n}\right\} k$$

In mathematics and computer programming, **exponentiating by squaring** is a general method for fast computation of large positive integer powers of a number, or more generally of an element of a semigroup, like a polynomial or a square matrix. Some variants are commonly referred to as **square-and-multiply algorithms** or **binary exponentiation**.

https://en.wikipedia.org/wiki/Exponentiation_by_squaring

https://oi-wiki.org/math/quick-pow/