Lego Blocks

by HackerRank

You have 4 types of lego blocks, of sizes given as (1 x 1 x 1), (1 x 1 x 2), (1 x 1 x 3), and (1 x 1 x 4). Assume that you have an infinite number of blocks of each type.

Using these blocks, you want to make a wall of height N and width M. The wall should not have any holes in it. The wall you build should be one solid structure. A solid structure can be interpreted in one of the following ways:
(1)It should not be possible to separate the wall along any vertical line without cutting any lego block used to build the wall.
(2)You cannot make a vertical cut from top to bottom without cutting one or more lego blocks.

The blocks can only be placed horizontally. In how many ways can the wall be built?

Input Format

The first line contains the number of test cases T. T test cases follow. Each case contains two integers N and M.

Constraints

1 <= T <= 100
1 <= N,M <= 1000

Output Format

Output T lines, one for each test case containing the number of ways to build the wall. As the numbers can be very large, output the result modulo 1000000007.

Sample Input

```
4
2 2
3 2
2 3
4 4
```

Sample Output

```
3
7
9
3375
```

Explanation

For the first case, we can have

- two (1 * 1 * 2) lego blocks stacked one on top of another.
- one (1 * 1 * 2) block stacked on top of two (1 * 1 * 1) blocks.
- two (1 * 1 * 1) blocks stacked on top of one (1 * 1 * 2) block.

For the second case, each row of the wall can contain either two blocks of width 1, or one block of width 2. However, the wall where all rows contain two blocks of width 1 is not a solid one as it can be divided vertically. Thus, the number of ways is 2 * 2 * 2 - 1 = 7.

Submissions: 2579
Max Score: 50
Difficulty: Medium

Rate This Challenge:

More

Current Buffer (saved locally, editable) Java 8

```
import java.io.*;
import java.util.*;

public class Solution {

    static Map<Integer,Long> OneHWalls = new HashMap<Integer,Long>();
    static Map<Integer,Long> solidWalls = new HashMap<Integer,Long>();
    static Map<Integer,Long> allWalls = new HashMap<Integer,Long>();

    static long mod = 1000000007;

    public static void main(String[] args) throws IOException{

        int[] arr = {1,2,3,4};

        findWidth(arr,1000);

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int tst = Integer.parseInt(br.readLine());

        for(int i = 0 ; i < tst ; i++){

            String[] num = br.readLine().split("\\s");
            int H = Integer.parseInt(num[0]);
            int W = Integer.parseInt(num[1]);


            allWalls = new HashMap<Integer,Long>();

            for(int j = 0 ; j < W; j++){
                long out = 1;
                for(int k = 0 ; k < H ; k++){
                    out = (out * OneHWalls.get(j+1)) % mod;
                }
                allWalls.put(j+1,out);
            }

            solidWalls = new HashMap<Integer,Long>();
            solidWalls.put(1,(long)1);

            long o = dp(W,H);

            System.out.println(o < 0 ?  mod + o : o);

        }
    }


    public static long dp(int W,int H){

        if(solidWalls.containsKey(W)){
            return solidWalls.get(W);
        }

        long output = allWalls.get(W);
```

```
56
57          for(int i = 1 ; i < W ; i++){
58              output = (output - ((dp(W - i,H) * allWalls.get(i)) % mod )) % mod;
59          }
60
61
62          solidWalls.put(W,output);
63          return output;
64      }
65
66      public static long findWidth(int[] arr, int sum){
67
68          long output = 0;
69
70          if(OneHWalls.containsKey(sum)){
71              return OneHWalls.get(sum);
72          }
73
74          if(sum < 0){
75              return 0;
76          }
77
78          if(sum == 1 || sum == 0){
79              OneHWalls.put(1,(long)1);
80              return 1;
81          }
82
83          for(int i = 0 ; i < arr.length ; i++){
84              output= (output +  findWidth(arr,sum-arr[i])) % mod;
85          }
86
87
88          OneHWalls.put(sum,output);
89          return output % mod;
90
91      }
92
93  }
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code