

**UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA**

**FACULTAD DE PRODUCCION Y SERVICIOS**

**ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS**



**Curso: Laboratorio Análisis y Diseño de Algoritmos**

## **Evidencias Ejercicios 8**

**Docente:** Florez Farfan, Alex Josue

**Estudiante:**

Neira Carrasco, Darwin Jesus

**Grupo - “B”**

Arequipa - Perú

## Ejercicio 1

Success Details

Runtime: 22 ms, faster than 5.22% of Java online submissions for Unique Paths II.

Memory Usage: 38.8 MB, less than 34.95% of Java online submissions for Unique Paths II.

Next challenges:

Unique Paths Unique Paths III

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language
11/30/2021 20:01	Accepted	22 ms	38.8 MB	java
11/30/2021 20:00	Wrong Answer	N/A	N/A	java
11/30/2021 20:00	Compile Error	N/A	N/A	java
11/30/2021 16:59	Wrong Answer	N/A	N/A	java
11/30/2021 16:58	Wrong Answer	N/A	N/A	java
11/30/2021 16:55	Wrong Answer	N/A	N/A	java
11/30/2021 16:53	Wrong Answer	N/A	N/A	java
11/30/2021 16:53	Wrong Answer	N/A	N/A	java

```
class Solution {
    public int uniquePathsWithObstacles(int[][] obstacleGrid) {
        // inicio de la seccion de convert
        for (int i = 0; i < obstacleGrid.length; i++) {
            for (int j = 0; j < obstacleGrid[i].length; j++) {
                if (obstacleGrid[i][j] == 1) {
                    obstacleGrid[i][j] = -2;
                }
            }
        }
        // de la seccion de convert
        int x = obstacleGrid.length;
        int y = obstacleGrid[0].length;
        int [][] all = new int [x][y];

        // en caso la posicion de destino este con una piedra
        if (obstacleGrid[x-1][y-1] == -2) {
            return 0;
        }

        // llenado fila
        for (int i = 0; i < y; i++) {
            if (obstacleGrid[0][i] == -2) {
                break; // mediante esto se evita seguir ,cuando se tiene una piedra en el camino
            }
            else {
                all[0][i] = 1;
            }
        }
        // llenado columna
        for (int i = 0; i < x; i++) {
            if (obstacleGrid[i][0] == -2) {
                break; // mediante esto se evita seguir ,cuando se tiene una piedra en el camino
            }
            else {
                all[i][0] = 1;
            }
        }

        // esto sera el llenado par las posiciones de i >= 1 & j >= 1
        for (int i = 1; i < x; i++) {
            for (int j = 1; j < y; j++) {
                /*if (obstacleGrid[i][j] == -2) {
                    if (obstacleGrid[i-1][j] == -2) {
                        continue;
                    }
                    continue;
                }
                if (obstacleGrid[i-1][j] == -2) {
                    continue;
                }
                if (obstacleGrid[i-1][j-1] == -2) {
                    continue;
                }
                all[i][j] = all[i-1][j] + all[i-1][j-1];
            }
        }

        return all[x-1][y-1];
    }
}
```

## Ejercicio 2

Submission details

Task: [Book Shop](#)

Sender: DarwinJNeira

Submission time: 2021-12-02 06:30:19

Language: Java

Status: READY

Result: ACCEPTED

Test results

test	verdict	time
#1	ACCEPTED	0.13 s
#2	ACCEPTED	0.19 s
#3	ACCEPTED	0.22 s
#4	ACCEPTED	0.22 s
#5	ACCEPTED	0.13 s
#6	ACCEPTED	0.96 s
#7	ACCEPTED	1.00 s
#8	ACCEPTED	1.00 s
#9	ACCEPTED	0.95 s
#10	ACCEPTED	0.96 s
#11	ACCEPTED	1.00 s
#12	ACCEPTED	0.13 s
#13	ACCEPTED	0.96 s
#14	ACCEPTED	0.13 s

Code

Dynamic Programming

- Coin Combinations II
- Removing Digits
- Grid Paths
- Book Shop
- Array Description
- Counting Towers
- Edit Distance
- Rectangle Cutting

Your submissions

Submission Time	Status
2021-12-02 06:30:19	✓
2021-12-02 06:29:21	✗
2021-12-02 06:19:03	✗
2021-12-02 06:02:48	✗
2021-12-02 05:57:36	✗
2021-12-02 05:54:17	✗

## Ejercicio 3

Success Details >

Runtime: 100 ms, faster than 13.23% of Java online submissions for Longest Increasing Subsequence.

Memory Usage: 39.2 MB, less than 37.20% of Java online submissions for Longest Increasing Subsequence.

Next challenges:

- Increasing Triplet Subsequence
- Russian Doll Envelopes
- Maximum Length of Pair Chain
- Number of Longest Increasing Subsequence
- Minimum ASCII Delete Sum for Two Strings
- Minimum Number of Removals to Make Mountain Array
- Find the Longest Valid Obstacle Course at Each Position

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Runtime	Memory	Language
12/02/2021 15:27	Accepted	100 ms	39.2 MB	java
12/02/2021 14:49	Wrong Answer	N/A	N/A	java
12/01/2021 19:42	Wrong Answer	N/A	N/A	java
12/01/2021 19:42	Compile Error	N/A	N/A	java
12/01/2021 19:42	Compile Error	N/A	N/A	java
12/01/2021 15:20	Wrong Answer	N/A	N/A	java

```
1 class Solution {
2     public int lengthOfLIS(int[] nums) {
3         int n = nums.length;
4         int[] cant = new int[n];
5         for (int i = 0; i < cant.length; i++) {
6             cant[i] = 1;
7         }
8
9         for (int i = 0; i < n-1; i++) {
10            for (int j = i+1; j < n; j++) {
11                if (nums[i] < nums[j]) {
12                    cant[j] = Math.max(cant[j], cant[i] + 1);
13                }
14            }
15        }
16
17        for (int i : cant) {
18            System.out.println(i);
19        }
20
21        int u = cant[0];
22        for (int i = 1; i < cant.length; i++) {
23            u = Math.max(u, cant[i]);
24        }
25
26        return u;
27    }
28 }
```

Your previous code was restored from your local storage. [Reset to default](#)

## Ejercicio 4

CSES Problem Set

### Rectangle Cutting

[TASK](#) | [SUBMIT](#) | [RESULTS](#) | [STATISTICS](#) | [HACKING](#)

**Submission details**

Task: [Rectangle Cutting](#)

Sender: DarwinJNeira

Submission time: 2021-12-03 01:31:36

Language: Java

Status: READY

Result: **ACCEPTED**

**Test results**

test	verdict	time	
#1	ACCEPTED	0.13 s	>>
#2	ACCEPTED	0.13 s	>>
#3	ACCEPTED	0.13 s	>>
#4	ACCEPTED	0.13 s	>>
#5	ACCEPTED	0.13 s	>>
#6	ACCEPTED	0.31 s	>>

**Dynamic Programming**

- ...
- [Array Description](#)
- [Counting Towers](#)
- [Edit Distance](#)
- [Rectangle Cutting](#) ✓
- [Money Sums](#)
- [Removal Game](#)
- [Two Sets II](#)
- [Increasing Subsequence](#)
- ...

**Your submissions**

2021-12-03 01:31:36	✓
2021-12-03 01:30:52	✗

## Ejercicio 5

Success

Details

Runtime: 200 ms, faster than 47.42% of Python online submissions for Maximal Square.

Memory Usage: 19.5 MB, less than 96.66% of Python online submissions for Maximal Square.

Next challenges:

Maximal Rectangle

Largest Plus Sign

Show off your acceptance:

Facebook

Twitter

LinkedIn

Time Submitted	Status	Runtime	Memory	Language
12/03/2021 15:26	Accepted	200 ms	19.5 MB	python
12/03/2021 14:18	Runtime Error	N/A	N/A	python
12/03/2021 14:17	Wrong Answer	N/A	N/A	python
12/03/2021 14:06	Wrong Answer	N/A	N/A	python
12/03/2021 12:39	Wrong Answer	N/A	N/A	python
12/03/2021 12:30	Wrong Answer	N/A	N/A	python
12/03/2021 12:30	Runtime Error	N/A	N/A	python

Python

Autocomplete

```
1 class Solution(object):
2     def maximalSquare(self, matrix):
3         # recibimos un arreglo de tipo str lo convertiremos a int
4         for x in range(len(matrix)):
5             for y in range(len(matrix[x])):
6                 matrix[x][y] = int(matrix[x][y])
7
8         w = len(matrix)
9         h = len(matrix[0])
10        arr = [[0 for x in range(h+1)] for y in range(w+1)]
11
12
13        if len(matrix) == 1 and len(matrix[0]) == 1:
14            if matrix[0][0] == 1:
15                return 1
16            else:
17                return 0
18        else:
19            if len(matrix) == 1 or len(matrix[0]) == 1:
20                result1 = matrix[0][0]
21                for x in range(len(matrix)):
22                    for y in range(len(matrix[x])):
23                        if matrix[x][y] > result1:
24                            result1 = matrix[x][y]
25
26            return result1
27        else:
28            for x in range(1,w+1):
29                for y in range(1,h+1):
30                    if matrix[x-1][y-1] == 1:
31                        matrix[x][y] = min(matrix[x-1][y-1],matrix[x][y-1],matrix[x-1][y]) + 1
32                        arr[x][y] = min(arr[x-1][y-1],arr[x][y-1],arr[x-1][y]) + 1
33                        #if matrix[x-1][y-1] == matrix[x][y-1] and matrix[x-1][y-1] == arr[x][y] == min(matrix[x-1][y-1],matrix[x][y-1],matrix[x-1][y]) + 1
34                        # if matrix[x-1][y-1] != 0:
35                        #     matrix[x][y] += 1
36                        # else:
37                        #     continue
38                        # continue
39
40            result = arr[0][0]
41            for x in range(len(arr)):
42                for y in range(len(arr[x])):
43                    if arr[x][y] > result:
44                        result = arr[x][y]
45
46            return result + result
47
48
49
```

Your previous code was restored from your local storage. [Reset to default](#)

Run Code

Submit