| | |
|---|---|
| Full Name: | Fabio Gaiera |
| Email: | easily.reachable@gmail.com |
| Test Name: | **Mock Test** |
| Taken On: | 1 Apr 2024 00:48:39 IST |
| Time Taken: | 0 min 41 sec/ 30 min |
| Invited by: | Ankush |
| Invited on: | 1 Apr 2024 00:48:20 IST |
| Skills Score: | |

**100%**

**90/90**

scored in **Mock Test** in 0 min 41 sec on 1 Apr 2024 00:48:39 IST

Tags Score:

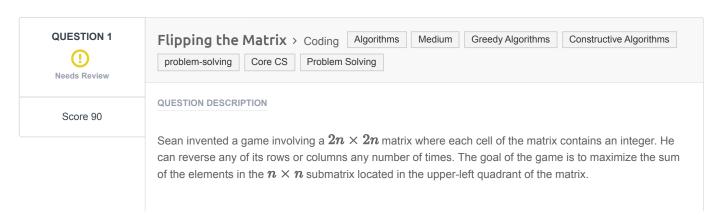| | |
|---|---|
| Algorithms | 90/90 |
| Constructive Algorithms | 90/90 |
| Core CS | 90/90 |
| Greedy Algorithms | 90/90 |
| Medium | 90/90 |
| Problem Solving | 90/90 |
| problem-solving | 90/90 |

**Recruiter/Team Comments:**

*No Comments.*

**Plagiarism flagged**

We have marked questions with suspected plagiarism below. Please review it in detail here -

| | Question Description | Time Taken | Score | Status |
|---|---|---|---|---|
| **Q1** | **Flipping the Matrix** > **Coding** | 24 sec | 90/ 90 | ⚠ |

**QUESTION 1**

⚠

**Needs Review**

Score 90

**Flipping the Matrix** > Coding    `Algorithms`  `Medium`  `Greedy Algorithms`  `Constructive Algorithms`  `problem-solving`  `Core CS`  `Problem Solving`

**QUESTION DESCRIPTION**

Sean invented a game involving a $2n \times 2n$ matrix where each cell of the matrix contains an integer. He can reverse any of its rows or columns any number of times. The goal of the game is to maximize the sum of the elements in the $n \times n$ submatrix located in the upper-left quadrant of the matrix.

Given the initial configurations for $q$ matrices, help Sean reverse the rows and columns of each matrix in the best possible way so that the sum of the elements in the matrix's upper-left quadrant is maximal.

**Example**
$matrix = [[1, 2], [3, 4]]$

```
1 2
3 4
```

It is $2 \times 2$ and we want to maximize the top left quadrant, a $1 \times 1$ matrix. Reverse row $1$:

```
1 2
4 3
```

And now reverse column $0$:

```
4 2
1 3
```

The maximal sum is $4$.

**Function Description**

Complete the *flippingMatrix* function in the editor below.

flippingMatrix has the following parameters:
- *int matrix[2n][2n]:* a 2-dimensional array of integers

**Returns**
- *int:* the maximum sum possible.

**Input Format**

The first line contains an integer $q$, the number of queries.

The next $q$ sets of lines are in the following format:
- The first line of each query contains an integer, $n$.
- Each of the next $2n$ lines contains $2n$ space-separated integers $matrix[i][j]$ in row $i$ of the matrix.

**Constraints**

- $1 \le q \le 16$
- $1 \le n \le 128$
- $0 \le matrix[i][j] \le 4096$, where $0 \le i, j < 2n$.

**Sample Input**

```
STDIN            Function
-----            --------
1                q = 1
2                n = 2
112 42 83 119    matrix = [[112, 42, 83, 119], [56, 125, 56, 49], \
56 125 56 49               [15, 78, 101, 43], [62, 98, 114, 108]]
15 78 101 43
62 98 114 108
```

**Sample Output**

```
414
```

**Explanation**

Start out with the following $2n \times 2n$ matrix:

$$matrix = \begin{bmatrix} 112 & 42 & 83 & 119 \\ 56 & 125 & 56 & 49 \\ 15 & 78 & 101 & 43 \\ 62 & 98 & 114 & 108 \end{bmatrix}$$

Perform the following operations to maximize the sum of the $n \times n$ submatrix in the upper-left quadrant:

2. Reverse column $2$ ($[83, 56, 101, 114] \rightarrow [114, 101, 56, 83]$), resulting in the matrix:

$$matrix = \begin{bmatrix} 112 & 42 & 114 & 119 \\ 56 & 125 & 101 & 49 \\ 15 & 78 & 56 & 43 \\ 62 & 98 & 83 & 108 \end{bmatrix}$$

3. Reverse row $0$ ($[112, 42, 114, 119] \rightarrow [119, 114, 42, 112]$), resulting in the matrix:

$$matrix = \begin{bmatrix} 119 & 114 & 42 & 112 \\ 56 & 125 & 101 & 49 \\ 15 & 78 & 56 & 43 \\ 62 & 98 & 83 & 108 \end{bmatrix}$$

The sum of values in the $n \times n$ submatrix in the upper-left quadrant is $119 + 114 + 56 + 125 = 414$

.

## CANDIDATE ANSWER

Language used: **Python 3**

```python
#
# Complete the 'flippingMatrix' function below.
#
# The function is expected to return an INTEGER.
# The function accepts 2D_INTEGER_ARRAY matrix as parameter.
#

def flippingMatrix(matrix):
    rows = len(matrix)
    cols = len(matrix[0])
    suma = 0
    for i in range(0, rows // 2):
        for j in range(0, cols // 2):
            suma += max(
                matrix[i][j],
                matrix[i][cols - j - 1],
                matrix[rows - 1 - i][j],
                matrix[rows - 1 - i][cols - 1 - j]
            )

    return suma
```

| TESTCASE | DIFFICULTY | TYPE | STATUS | SCORE | TIME TAKEN | MEMORY USED |
|---|---|---|---|---|---|---|
| Testcase 1 | Easy | Sample case | ✓ Success | 0 | 0.0189 sec | 10.4 KB |
| Testcase 2 | Easy | Hidden case | ✓ Success | 15 | 0.0831 sec | 12.7 KB |
| Testcase 3 | Easy | Hidden case | ✓ Success | 15 | 0.1075 sec | 13.1 KB |
| Testcase 4 | Easy | Hidden case | ✓ Success | 15 | 0.0651 sec | 12.6 KB |

| Testcase 5 | Easy | Hidden case | ⊘ Success | 15 | 0.0853 sec | 12.8 KB |
| Testcase 6 | Easy | Hidden case | ⊘ Success | 15 | 0.1179 sec | 12.8 KB |
| Testcase 7 | Easy | Hidden case | ⊘ Success | 15 | 0.1065 sec | 12.6 KB |
| Testcase 8 | Easy | Sample case | ⊘ Success | 0 | 0.0173 sec | 10.4 KB |

No Comments