



Full Name: Joel Varghese

Email: joelvarghese356@gmail.com

Test Name: Mock Test

Taken On: 28 Mar 2024 09:52:28 IST

Time Taken: 14 min 43 sec/ 24 min

Contact Number: +916376422354

Linkedin: http://www.linkedin.com/in/joel-varghese-54b51923a

Invited by: Ankush

Invited on: 28 Mar 2024 09:51:48 IST

Skills Score:

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

100%

90/90

scored in Mock Test in 14 min 43 sec on 28 Mar 2024 09:52:28 IST

Recruiter/Team Comments:

No Comments.

|    | Question Description         | Time Taken   | Score  | Status |
|----|------------------------------|--------------|--------|--------|
| Q1 | Flipping the Matrix > Coding | 14 min 9 sec | 90/ 90 | ✔      |

QUESTION 1



Correct Answer

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

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

### 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: **PyPy3**

```

1
2 #
3 # Complete the 'flippingMatrix' function below.
4 #
5 # The function is expected to return an INTEGER.
6 # The function accepts 2D_INTEGER_ARRAY matrix as parameter.
7 #
8
9 def flippingMatrix(matrix):
10     # Write your code here
11     rows = len(matrix)
12     col = len(matrix[0])
13     sums = 0
14     for i in range(0, rows//2):
15         for j in range(0, col//2):
16             sums += max(matrix[i][j], matrix[i][col-j-1], matrix[rows-1-i]
17 [j], matrix[rows-1-i][col-j-1])
18     return sums

```

| TESTCASE   | DIFFICULTY | TYPE        | STATUS    | SCORE | TIME TAKEN | MEMORY USED |
|------------|------------|-------------|-----------|-------|------------|-------------|
| Testcase 1 | Easy       | Sample case | ✔ Success | 0     | 0.2624 sec | 71.8 KB     |
| Testcase 2 | Easy       | Hidden case | ✔ Success | 15    | 0.3957 sec | 75.7 KB     |
| Testcase 3 | Easy       | Hidden case | ✔ Success | 15    | 0.4358 sec | 76.7 KB     |
| Testcase 4 | Easy       | Hidden case | ✔ Success | 15    | 0.3883 sec | 75.8 KB     |
| Testcase 5 | Easy       | Hidden case | ✔ Success | 15    | 0.4039 sec | 75.9 KB     |
| Testcase 6 | Easy       | Hidden case | ✔ Success | 15    | 0.434 sec  | 76.2 KB     |
| Testcase 7 | Easy       | Hidden case | ✔ Success | 15    | 0.4194 sec | 76.5 KB     |
| Testcase 8 | Easy       | Sample case | ✔ Success | 0     | 0.2788 sec | 71.6 KB     |

