



Full Name:

Rahul Gangwar

Email:

rahulkurmi97194@gmail.com

Test Name:

Mock Test

Taken On:

19 Feb 2025 09:20:41 IST

Time Taken:

14 min 8 sec/ 30 min

Invited by:

Ankush

Invited on:

18 Feb 2025 09:32:40 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 8 sec on 19 Feb 2025 09:20:41 IST

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	13 min 55 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 2×2 and we want to maximize the top left quadrant, a 1×1 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: C#

```

1
2 class Result
3 {
4
5     /*
6      * Complete the 'flippingMatrix' function below.
7      *
8      * The function is expected to return an INTEGER.
9      * The function accepts 2D_INTEGER_ARRAY matrix as parameter.
10     */
11
12     public static int flippingMatrix(List<List<int>> matrix)
13     {
14         int n = matrix.Count / 2; // Half the matrix size (2n x 2n => n)
15         int totalSum = 0;
16
17         for (int i = 0; i < n; i++)
18         {
19             for (int j = 0; j < n; j++)
20             {
21                 // Get the four possible values for each element in the top-
22                 left n x n submatrix
23                 int topLeft = matrix[i][j];
24                 int topRight = matrix[i][2 * n - 1 - j];
25                 int bottomLeft = matrix[2 * n - 1 - i][j];
26                 int bottomRight = matrix[2 * n - 1 - i][2 * n - 1 - j];
27
28                 // Add the maximum of these four values to the result
29                 totalSum += Math.Max(Math.Max(topLeft, topRight),
30 Math.Max(bottomLeft, bottomRight));
31             }
32         }
33

```

```
34         return totalSum;
35     }
36 }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✔ Success	0	0.0455 sec	24.3 KB
Testcase 2	Easy	Hidden case	✔ Success	15	0.0869 sec	41 KB
Testcase 3	Easy	Hidden case	✔ Success	15	0.1259 sec	41.1 KB
Testcase 4	Easy	Hidden case	✔ Success	15	0.093 sec	38.5 KB
Testcase 5	Easy	Hidden case	✔ Success	15	0.1051 sec	41 KB
Testcase 6	Easy	Hidden case	✔ Success	15	0.1072 sec	41.1 KB
Testcase 7	Easy	Hidden case	✔ Success	15	0.1156 sec	40.9 KB
Testcase 8	Easy	Sample case	✔ Success	0	0.0531 sec	24.4 KB

No Comments