Full Name:	Chetana Patil
Email:	chetanabmpatil@gmail.com
Test Name:	**Mock Test**
Taken On:	5 Apr 2024 23:55:28 IST
Time Taken:	23 min 38 sec/ 24 min
Resume:	https://hackerrank-resumes.s3.amazonaws.com/9410269/3lqHdh5uG5EoFRbiHHda9AxW0OoIT_PPY5jpC40pc_QQwn18ApZAttnF11gab6S2ug/Chetana_s_Resume.pdf
Linkedin:	https://www.linkedin.com/in/chetana-patil-497236193/
Invited by:	Ankush
Invited on:	5 Apr 2024 23:55:07 IST
Skills Score:	
Tags Score:	Algorithms 0/90 Constructive Algorithms 0/90 Core CS 0/90 Greedy Algorithms 0/90 Medium 0/90 Problem Solving 0/90 problem-solving 0/90

0%

0/90

scored in **Mock Test** in 23 min 38 sec on 5 Apr 2024 23:55:28 IST

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	**Flipping the Matrix** > **Coding**	21 min 7 sec	0/ 90	⊖

QUESTION 1

⊖

Not Submitted

Score 0

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

No answer was submitted for this question. Showing compiled/saved versions.

Language used: **Java 8**

```
1
2  class Result {
3
4      /*
5       * Complete the 'flippingMatrix' function below.
6       *
7       * The function is expected to return an INTEGER.
8       * The function accepts 2D_INTEGER_ARRAY matrix as parameter.
9       */
10
11     public static int flippingMatrix(List<List<Integer>> matrix) {
12     // Write your code here
13
14
15         int sum = 0;
16         int n = matrix.size();
17
18         for (int i = 0; i < n / 2; i++) {
19             for (int j = 0; j < n / 2; j++) {
20                 sum += Math.max(matrix.get(i).get(j),
21                         Math.max(matrix.get(i).get(n - 1 - j),
22                                 Math.max(matrix.get(n - 1 - i).get(j),
23 matrix.get(n - 1 - i).get(n - 1 - j))));
24             }
25         }
```

```
26          return sum;
27      }
28
29
30
31
32  }
```

No Comments