Full Name:	Shaurya kajiwala
Email:	kajiwalashaurya29@gmail.com
Test Name:	**Mock Test**
Taken On:	15 Jun 2023 18:18:28 IST
Time Taken:	0 min 38 sec/ 30 min
Invited by:	Ankush
Invited on:	15 Jun 2023 18:18:18 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 0 min 38 sec on 15 Jun 2023 18:18:28 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	25 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: **Java 8**

```java
import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.function.*;
import java.util.regex.*;
import java.util.stream.*;
import static java.util.stream.Collectors.joining;
import static java.util.stream.Collectors.toList;

class Result {

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

    public static int flippingMatrix(List<List<Integer>> matrix) {
    int leftQudrantSize=(int) matrix.size()/2;
    int matLength=matrix.size()-1;
    int sum=0;
    for(int i=0; i<leftQudrantSize; i++){

        for(int j=0;j<leftQudrantSize; j++){
            int largestElement=matrix.get(i).get(j);
            if(matrix.get(i).get(matLength-j)>largestElement){
                largestElement=matrix.get(i).get(matLength-j);
            }
            if(matrix.get(matLength-i).get(j)>largestElement){
```

```
             largestElement=   matrix.get(matLength-i).get(j);
            }
            if(matrix.get(matLength-i).get(matLength-j)>largestElement){
            largestElement=matrix.get(matLength-i).get(matLength-j);
            }
            sum+=largestElement;
        }

    }
    return sum;


    }

}

public class Solution {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new
FileWriter(System.getenv("OUTPUT_PATH")));

        int q = Integer.parseInt(bufferedReader.readLine().trim());

        IntStream.range(0, q).forEach(qItr -> {
            try {
                int n = Integer.parseInt(bufferedReader.readLine().trim());

                List<List<Integer>> matrix = new ArrayList<>();

                IntStream.range(0, 2 * n).forEach(i -> {
                    try {
                        matrix.add(

Stream.of(bufferedReader.readLine().replaceAll("\\s+$", "").split(" "))
                                .map(Integer::parseInt)
                                .collect(toList())
                        );
                    } catch (IOException ex) {
                        throw new RuntimeException(ex);
                    }
                });

                int result = Result.flippingMatrix(matrix);

                bufferedWriter.write(String.valueOf(result));
                bufferedWriter.newLine();
            } catch (IOException ex) {
                throw new RuntimeException(ex);
            }
        });

        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	Success	0	0.1108 sec	30 KB
Testcase 2	Easy	Hidden case	Success	15	0.3536 sec	44.3 KB

Testcase 3	Easy	Hidden case	Success	15	0.3839 sec	56.2 KB
Testcase 4	Easy	Hidden case	Success	15	0.3603 sec	45.1 KB
Testcase 5	Easy	Hidden case	Success	15	0.3399 sec	54.5 KB
Testcase 6	Easy	Hidden case	Success	15	0.4958 sec	54.6 KB
Testcase 7	Easy	Hidden case	Success	15	0.3665 sec	52.3 KB
Testcase 8	Easy	Sample case	Success	0	0.1388 sec	29.8 KB

No Comments

PDF generated at: 15 Jun 2023 12:51:00 UTC