

#### Mock Test > rahulkurmi97194@gmail.com

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:**

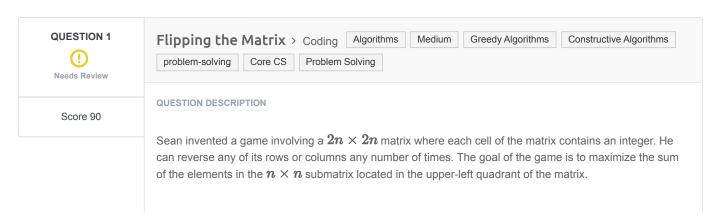
No Comments.

# 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 1



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 = \left[ [1,2], [3,4] \right]
```

```
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

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

#### Sample Input

#### **Sample Output**

414

### **Explanation**

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

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

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:

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

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

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

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#

```
2 class Result
3 {
4
        * 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<int>> matrix)
         int n = matrix.Count / 2; // Half the matrix size (2n \times 2n \Rightarrow n)
14
          int totalSum = 0;
           for (int i = 0; i < n; i++)
               for (int j = 0; j < n; j++)
                   // Get the four possible values for each element in the top-
  left n x n submatrix
                   int topLeft = matrix[i][j];
                   int topRight = matrix[i][2 * n - 1 - j];
                   int bottomLeft = matrix[2 * n - 1 - i][j];
                   int bottomRight = matrix[2 * n - 1 - i][2 * n - 1 - j];
                   // Add the maximum of these four values to the result
                   totalSum += Math.Max(Math.Max(topLeft, topRight),
30 Math.Max(bottomLeft, bottomRight));
              }
           }
```

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							

PDF generated at: 19 Feb 2025 04:06:44 UTC