

Full Name: Bugrahan Kara

Email: mbkara0106@gmail.com

Test Name: Mock Test

Taken On: 27 Mar 2022 12:25:46 IST

Time 2 min 15 sec/ 30 min

Taken:

Resume: https://hackerrank-

 $resumes.s3.amazonaws.com/14313602/2d5wcNSIVi9yCF6eHTOIaQiPWrOmbN2f\_3-qP995O3kQsQIIk7TcmvhCrNQrZkSyBA/Muhammet\_Bugrahan\_KARA\_CV6.pdf$ 

Invited by: Ankush

Invited on: 27 Mar 2022 12:25:40 IST

Skills Score:

Tags Score:



problem-solving 90/90

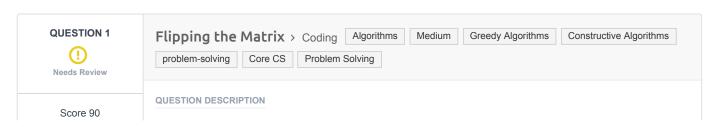
## **Recruiter/Team Comments:**

No Comments.

## Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review.





100% 90/90

scored in **Mock Test** in 2 min 15 sec on 27 Mar 2022 12:25:46 IST 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 = \left[ \left[ 1, 2 \right], \left[ 3, 4 \right] \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**

111

### Explanation

Start out with the following 2n imes 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] ightarrow [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] ightarrow [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 imes n submatrix in the upper-left quadrant is 119+114+56+125=414

.

#### **CANDIDATE ANSWER**

# Language used: C++14

```
1 /*
   * Complete the 'flippingMatrix' function below.
* The function is expected to return an INTEGER.
  * The function accepts 2D INTEGER ARRAY matrix as parameter.
9 int flippingMatrix(vector<vector<int>> matrix) {
     int q1sum=0;
     int n=matrix.size()/2;
    for(int i=0;i<n;i++){
         //vector<int> maxFinder(4,0);
         for(int j=0;j<n;j++){
             /* //it gives wrong result ?
            maxFinder.push back(matrix[i][j]);
           maxFinder.push back(matrix[i][2*n-j-1]);
             maxFinder.push back(matrix[2*n-i-1][j]);
           maxFinder.push_back(matrix[2*n-i-1][2*n-j-1]);
           q1sum+= *max_element(maxFinder.begin(),maxFinder.end());
            //cout<<q1sum<<endl;
            qlsum+= max({matrix[i][j],matrix[i][2*n-j-1],matrix[2*n-i-1][j],
             matrix[2*n-i-1][2*n-j-1]});
      }
```

}						
TEOTOAGE	DIFFIGURTY	TVDE	OTATLIO	00005	TIME TAKEN	MEMORYLIGER
TESTCASE Testcase 1	DIFFICULTY	TYPE	STATUS  Success	SCORE 0	0.0284 sec	MEMORY USED 8.78 KB
	Easy	Sample case				
Testcase 2	Easy	Hidden case	Success     ■	15	0.0792 sec	9.29 KB
Testcase 3	Easy	Hidden case	Success	15	0.1001 sec	9.33 KB
Testcase 4	Easy	Hidden case	Success	15	0.0759 sec	9.5 KB
Testcase 5	Easy	Hidden case	Success	15	0.0776 sec	9.28 KB
Testcase 6	Easy	Hidden case	Success	15	0.0936 sec	9.33 KB
Testcase 7	Easy	Hidden case	Success	15	0.0991 sec	9.38 KB
Testcase 8	Easy	Sample case	Success	0	0.0256 sec	8.96 KB

PDF generated at: 27 Mar 2022 06:59:41 UTC