



Full Name: Elton Rodrigues de Souza

Email: elton.rds@outlook.com

Test Name: Mock Test

Taken On: 14 Jan 2024 05:11:31 IST

Time Taken: 3 min 42 sec/ 40 min

Invited by: Ankush

Invited on: 14 Jan 2024 05:11:15 IST

Skills Score:

Tags Score:

- Algorithms195/195
- Constructive Algorithms90/90
- Core CS195/195
- Easy105/105
- Greedy Algorithms90/90
- Medium90/90
- Problem Solving195/195
- Search105/105
- Sorting105/105
- problem-solving195/195

100%

195/195

scored in Mock Test in 3 min 42 sec on 14 Jan 2024 05:11:31 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	Find the Median > Coding	1 min 3 sec	105/ 105	✔
Q2	Flipping the Matrix > Coding	2 min 21 sec	90/ 90	!

QUESTION 1

Correct Answer

Find the Median > Coding

Sorting

Search

Algorithms

Easy

problem-solving

Core CS

Problem Solving

QUESTION DESCRIPTION

Score 105

The median of a list of numbers is essentially its middle element after sorting. The same number of elements occur after it as before. Given a list of numbers with an odd number of elements, find the **median**?

**Example**

$arr = [5, 3, 1, 2, 4]$

The sorted array  $arr' = [1, 2, 3, 4, 5]$ . The middle element and the median is **3**.

**Function Description**

Complete the *findMedian* function in the editor below.

findMedian has the following parameter(s):

- *int arr[n]*: an unsorted array of integers

**Returns**

- *int*: the median of the array

**Input Format**

The first line contains the integer *n*, the size of *arr*.

The second line contains *n* space-separated integers  $arr[i]$

**Constraints**

- $1 \leq n \leq 1000001$
- *n* is odd
- $-10000 \leq arr[i] \leq 10000$

**Sample Input 0**

```
7
0 1 2 4 6 5 3
```

**Sample Output 0**

```
3
```

**Explanation 0**

The sorted  $arr = [0, 1, 2, 3, 4, 5, 6]$ . It's middle element is at  $arr[3] = 3$ .





**CANDIDATE ANSWER**

Language used: **JavaScript (Node.js)**

```
1
2  /*
3   * Complete the 'findMedian' function below.
4   *
5   * The function is expected to return an INTEGER.
6   * The function accepts INTEGER_ARRAY arr as parameter.
7   */
8
9  function findMedian(arr) {
10     // Write your code here
11
12     arr.sort((a, b) => a - b);
13
14     const middleIndex = Math.floor(arr.length / 2);
15     const median = arr[middleIndex];
16     return median;
17 }
```

18 }

19

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	 Success	0	0.0402 sec	41.8 KB
Testcase 2	Easy	Hidden case	 Success	35	0.0361 sec	43.4 KB
Testcase 3	Easy	Hidden case	 Success	35	0.0466 sec	44.7 KB
Testcase 4	Easy	Hidden case	 Success	35	0.0792 sec	53.2 KB

No Comments

QUESTION 2

!

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  $2 \times 2$  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

- $1 \leq q \leq 16$
- $1 \leq n \leq 128$
- $0 \leq matrix[i][j] \leq 4096$ , 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	

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: JavaScript (Node.js)

```
1
2 /*
3  * Complete the 'flippingMatrix' function below.
4  *
```

```

5  * The function is expected to return an INTEGER.
6  * The function accepts 2D_INTEGER_ARRAY matrix as parameter.
7  */
8
9  function flippingMatrix(matrix) {
10     // Write your code here
11
12     const n = matrix.length / 2;
13
14
15     let maxSum = 0;
16     for (let i = 0; i < n; i++) {
17         for (let j = 0; j < n; j++) {
18
19             const maxValue = Math.max(
20                 matrix[i][j],
21                 matrix[i][2 * n - 1 - j],
22                 matrix[2 * n - 1 - i][j],
23                 matrix[2 * n - 1 - i][2 * n - 1 - j]
24             );
25
26
27             maxSum += maxValue;
28         }
29     }
30
31     return maxSum
32 }
33

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✔ Success	0	0.0316 sec	41.9 KB
Testcase 2	Easy	Hidden case	✔ Success	15	0.1196 sec	53.5 KB
Testcase 3	Easy	Hidden case	✔ Success	15	0.1148 sec	56.9 KB
Testcase 4	Easy	Hidden case	✔ Success	15	0.0776 sec	53 KB
Testcase 5	Easy	Hidden case	✔ Success	15	0.1104 sec	56.4 KB
Testcase 6	Easy	Hidden case	✔ Success	15	0.1241 sec	54.8 KB
Testcase 7	Easy	Hidden case	✔ Success	15	0.1133 sec	55.7 KB
Testcase 8	Easy	Sample case	✔ Success	0	0.0354 sec	41.8 KB

No Comments