

Full Name: LAKSHMIVENKAT UPPALA Email: lakshmivenkatuppala@gmail.com Test Name: **Mock Test** Taken On: 12 Aug 2025 12:50:28 IST 24 min 9 sec/ 40 min Time Taken: Invited by: Ankush Invited on: 12 Aug 2025 12:49:52 IST Skills Score: Tags Score: Algorithms 195/195 Constructive Algorithms 90/90 Core CS 195/195 Easy 105/105 Greedy Algorithms 90/90 90/90 Medium Problem Solving 195/195 105/105 Search Sorting 105/105

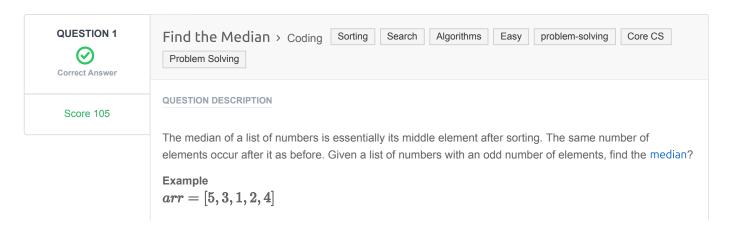


Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Find the Median > Coding	10 min 54 sec	105/ 105	Ø
Q2	Flipping the Matrix > Coding	10 min 3 sec	90/ 90	⊘

problem-solving 195/195



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

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

Sample Input 0

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

Sample Output 0

3

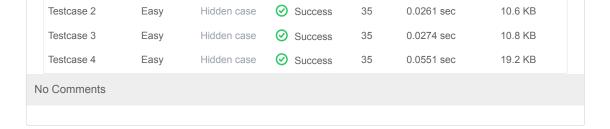
Explanation 0

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

CANDIDATE ANSWER

Language used: Python 3

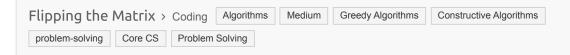
```
1
2 #
3 # Complete the 'findMedian' function below.
4 #
5 # The function is expected to return an INTEGER.
6 # The function accepts INTEGER_ARRAY arr as parameter.
7 #
8
9 def findMedian(arr):
10  # Write your code here
11  sorted_arr = sorted(arr)
12  m=0
13  if len(arr)>0:
14  m=len(arr)//2
15  median=sorted_arr[m]
16  return median
17
18
```







Score 90



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 ${f 4}$.

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, $oldsymbol{n}$.
- Each of the next 2n lines contains 2n space-separated integers matrix[i][j] in row i of the matrix.

Constraints

• $1 \le q \le 16$

- $1 \le n \le 128$
- $0 \leq matrix[i][j] \leq 4096$, 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 imes 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:

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

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: Python 3

```
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 def flippingMatrix(matrix):
10 # Write your code here
11 maximal_sum=0
```

```
n=len(matrix)
       q=n//2
14
       for i in range(q):
            for j in range(q):
                max_val=max(matrix[i][j], matrix[n-1-i][j],matrix[i][n-1-j],
17 matrix[n-1-i][n-1-j])
                 maximal_sum += max_val
        return maximal sum
   TESTCASE
                                          STATUS
              DIFFICULTY
                              TYPE
                                                     SCORE
                                                              TIME TAKEN
                                                                           MEMORY USED
                                                                              10.1 KB
  Testcase 1
                 Easy
                            Sample case
                                        Success
                                                               0.0251 sec
  Testcase 2
                 Easy
                            Hidden case
                                        Success
                                                              0.0906 sec
                                                                              13.1 KB
                                                       15
                                        Success
                                                              0.1199 sec
  Testcase 3
                 Easy
                            Hidden case
                                                       15
                                                                              13.3 KB
  Testcase 4
                 Easy
                            Hidden case
                                        Success
                                                       15
                                                               0.109 sec
                                                                              12.5 KB
                                        Success
  Testcase 5
                 Easy
                            Hidden case
                                                      15
                                                               0.103 sec
                                                                              13.3 KB
  Testcase 6
                 Easy
                            Hidden case
                                        Success
                                                       15
                                                              0.1057 sec
                                                                              13.4 KB
                            Hidden case
  Testcase 7
                                        Success
                                                              0.1355 sec
                                                                              13.3 KB
                 Easy
                                                       15
                                                                               10 KB
  Testcase 8
                 Easy
                           Sample case
                                        Success
                                                       0
                                                              0.0249 sec
No Comments
```

PDF generated at: 12 Aug 2025 07:48:04 UTC