

Mock Test > shindshw1995@gmail.com

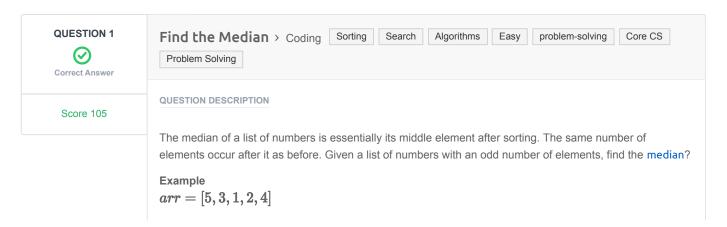
Full Name: Shweta Shinde Email: shindshw1995@gmail.com Test Name: **Mock Test** Taken On: 9 Jul 2024 04:09:04 IST 38 min 1 sec/ 40 min Time Taken: Invited by: Ankush Invited on: 9 Jul 2024 04:08:44 IST Skills Score: Tags Score: Algorithms 195/195 Constructive Algorithms 90/90 Core CS 195/195 Easy 105/105 Greedy Algorithms 90/90 90/90 Medium Problem Solving 195/195 105/105 Search Sorting 105/105 problem-solving 195/195



Recruiter/Team Comments:

No Comments.





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

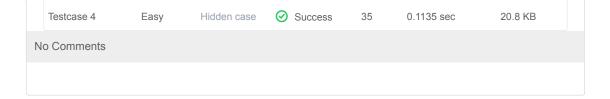
Explanation 0

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

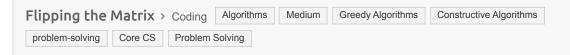
TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	Success	0	0.0335 sec	10.3 KB
Testcase 2	Easy	Hidden case	Success	35	0.0359 sec	10.9 KB
Testcase 3	Easy	Hidden case	Success	35	0.0352 sec	11.2 KB







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

$$matrix = \left[[1,2], [3,4] \right]$$

- 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, $oldsymbol{n}$.
- 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 $n \times n$ 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 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):
    # Write your code here
    result=0
12    n=len(matrix)//2
13    for i in range(n):
```

```
14
            for j in range(n):
                max value = max(
                     matrix[i][j], matrix[i][(2*n-1)-j], matrix[(2*n-1)-i][j],
17 matrix[(2*n-1)-i][(2*n-1)-j]
                result+=max_value
       return result
  TESTCASE
              DIFFICULTY
                             TYPE
                                         STATUS
                                                    SCORE
                                                             TIME TAKEN
                                                                           MEMORY USED
  Testcase 1
                 Easy
                           Sample case
                                        Success
                                                              0.0396 sec
                                                                              10.4 KB
 Testcase 2
                                                              0.1039 sec
                                                                              12.8 KB
                 Easy
                           Hidden case
                                        Success
                                                      15
 Testcase 3
                 Easy
                           Hidden case
                                        Success
                                                      15
                                                              0.1406 sec
                                                                              12.8 KB
 Testcase 4
                                                              0.0817 sec
                 Easy
                           Hidden case
                                        Success
                                                      15
                                                                              12.4 KB
 Testcase 5
                 Easy
                           Hidden case
                                        Success
                                                      15
                                                              0.1191 sec
                                                                              12.8 KB
```

Success

Success

Success

15

15

0

0.1325 sec

0.205 sec

0.0351 sec

12.8 KB 12.5 KB

10.3 KB

No Comments

Testcase 6

Testcase 7

Testcase 8

Easy

Easy

Easy

PDF generated at: 8 Jul 2024 23:18:44 UTC

Hidden case

Hidden case

Sample case