Mock Test › mhbagheri3@gmail.com

Full Name:	Mohammad Hossein Bagheri
Email:	mhbagheri3@gmail.com
Test Name:	**Mock Test**
Taken On:	9 Sep 2023 20:17:37 IST
Time Taken:	34 min 24 sec/ 40 min
Linkedin:	http://www.linkedin.com/in/mhbagheri
Invited by:	Ankush
Invited on:	9 Sep 2023 02:25:30 IST
Skills Score:	
Tags Score:	Algorithms 195/195 Constructive Algorithms 90/90 Core CS 195/195 Easy 105/105 Greedy Algorithms 90/90 Medium 90/90 Problem Solving 195/195 Search 105/105 Sorting 105/105 problem-solving 195/195

100%

195/195

scored in **Mock Test** in 34 min 24 sec on 9 Sep 2023 20:17:37 IST

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	**Find the Median › Coding**	5 min 2 sec	105/ 105	✓
Q2	**Flipping the Matrix › Coding**	29 min 13 sec	90/ 90	✓

QUESTION 1

✓

Correct Answer

Score 105

Find the Median › Coding

Sorting · Search · Algorithms · Easy · problem-solving · Core CS · Problem Solving

QUESTION DESCRIPTION

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: **C++14**

```
/*
 * Complete the 'findMedian' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts INTEGER_ARRAY arr as parameter.
 */

int findMedian(vector<int> arr) {
    int median;
    sort(arr.begin(), arr.end());
    int median_index = (arr.size() - 1)/2;
    median = arr[median_index];
    return median;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✓ Success	0	0.0329 sec	8.82 KB
Testcase 2	Easy	Hidden case	✓ Success	35	0.0881 sec	9 KB
Testcase 3	Easy	Hidden case	✓ Success	35	0.0302 sec	9.02 KB

Testcase 4	Easy	Hidden case	✓ Success	35	0.0468 sec	13.2 KB

No Comments

QUESTION 2

✓

Correct Answer

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 2×2 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, 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]]
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: **C++14**

```cpp

/*
 * Complete the 'flippingMatrix' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts 2D_INTEGER_ARRAY matrix as parameter.
 */

int flippingMatrix(vector<vector<int>> matrix) {
    int maxSum = 0;
    int size_n = matrix.size() / 2, size_2n = matrix.size() - 1;
    for (int y = 0; y < size_n; y++){
        for (int x = 0; x < size_n; x++){
            int m1 = max(matrix[y][x], matrix[size_2n - y][x]);

```

```
                int m2 = max(matrix[y][size_2n - x], matrix[size_2n - y][size_2n
- x]);
                maxSum += max(m1, m2);
            }
        }
        return maxSum;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✓ Success	0	0.1416 sec	8.97 KB
Testcase 2	Easy	Hidden case	✓ Success	15	0.1357 sec	9.31 KB
Testcase 3	Easy	Hidden case	✓ Success	15	0.1047 sec	9.14 KB
Testcase 4	Easy	Hidden case	✓ Success	15	0.1132 sec	9.18 KB
Testcase 5	Easy	Hidden case	✓ Success	15	0.1039 sec	9 KB
Testcase 6	Easy	Hidden case	✓ Success	15	0.1286 sec	9.23 KB
Testcase 7	Easy	Hidden case	✓ Success	15	0.1393 sec	9.24 KB
Testcase 8	Easy	Sample case	✓ Success	0	0.0862 sec	8.82 KB

No Comments

PDF generated at: 9 Sep 2023 15:23:32 UTC