



Full Name: Jayasakthi Subramanian

Email: jayasakthisubramanian4@gmail.com

Test Name: Mock Test

Taken On: 13 Aug 2025 11:24:05 IST

Time Taken: 20 min 18 sec/ 40 min

Invited by: Ankush

Invited on: 9 Aug 2025 22:38:01 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 20 min
18 sec on 13 Aug 2025 11:24:05
IST

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Find the Median > Coding	11 min 41 sec	105/ 105	✓
Q2	Flipping the Matrix > Coding	8 min 18 sec	90/ 90	✓

QUESTION 1

✓

Correct Answer

Score 105

Find the Median > Coding

SortingSearchAlgorithmsEasyproblem-solvingCore 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

```
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 static int cmp_int(const void *a, const void *b)
9 {
10     int ia = *(const int*)a;
11     int ib = *(const int*)b;
12     if (ia < ib) return -1;
13     if (ia > ib) return 1;
14     return 0;
15 }
16 int findMedian(int arr_count, int* arr) {
17     fprintf(stderr, "DEBUG: arr_count = %d\n", arr_count);
18     for (int i=0; i<arr_count; i++)
19     {
20         fprintf(stderr, "DEBUG before[%d] = %d\n", i, arr[i]);
21     }
22     qsort(arr, arr_count, sizeof(int), cmp_int);
23
24     for (int i =0; i < arr_count; ++i)
```

```

25 {
26     fprintf(stderr, "DEBUG after[%d] = %d\n", i, arr[i]);
27 }
28
29 int mid_index = arr_count / 2;
30 fprintf(stderr, "DEBUG median_index = %d, median_value = %d\n", mid_index,
31 arr[mid_index]);
32
33 return arr[mid_index];
34 }

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✔ Success	0	0.0071 sec	7.25 KB
Testcase 2	Easy	Hidden case	✔ Success	35	0.0313 sec	7.5 KB
Testcase 3	Easy	Hidden case	✔ Success	35	0.0314 sec	7.25 KB
Testcase 4	Easy	Hidden case	✔ Success	35	0.3204 sec	8.88 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

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

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

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:

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

3. Reverse row 0 ([112, 42, 114, 119] \rightarrow [119, 114, 42, 112]), resulting in the matrix:

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

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

```
1  /*
2  * Complete the 'flippingMatrix' function below.
3  *
4  * The function is expected to return an INTEGER.
5  * The function accepts 2D_INTEGER_ARRAY matrix as parameter.
6  */
7
8  int flippingMatrix(int matrix_rows, int matrix_columns, int** matrix) {
9      int n = matrix_rows / 2;
10     long long sum = 0;
11
12     for(int i =0; i<n; ++i)
13     {
14         for (int j=0;j<n;++j)
15         {
16             int a = matrix[i][j];
17             int b = matrix[i][matrix_columns - 1 - j];
18             int c = matrix[matrix_rows - 1 - i][j];
19             int d = matrix[matrix_rows - 1 - i][matrix_columns - 1 - j];
20
21             int mx = a;
22             if (b > mx) mx = b;
23             if (c > mx) mx = c;
24             if (d > mx) mx = d;
25
26             sum += mx;
27         }
28     }
29     return (int)sum;
30 }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	 Success	0	0.0081 sec	6.88 KB
Testcase 2	Easy	Hidden case	 Success	15	0.0383 sec	12 KB
Testcase 3	Easy	Hidden case	 Success	15	0.042 sec	15.1 KB
Testcase 4	Easy	Hidden case	 Success	15	0.0193 sec	11.3 KB
Testcase 5	Easy	Hidden case	 Success	15	0.0436 sec	13.3 KB
Testcase 6	Easy	Hidden case	 Success	15	0.0571 sec	14.1 KB
Testcase 7	Easy	Hidden case	 Success	15	0.0494 sec	14.8 KB
Testcase 8	Easy	Sample case	 Success	0	0.0093 sec	7.25 KB

No Comments