



Full Name:

Kaviya Ramesh

Email:

kaviya09052004@gmail.com

Test Name:

Mock Test

Taken On:

7 Aug 2025 19:04:34 IST

Time Taken:

21 min 10 sec/ 40 min

Invited by:

Ankush

Invited on:

7 Aug 2025 19:04:18 IST

Skills Score:

Tags Score:

Algorithms

160/195

Constructive Algorithms

90/90

Core CS

160/195

Easy

70/105

Greedy Algorithms

90/90

Medium

90/90

Problem Solving

160/195

Search

70/105

Sorting

70/105

problem-solving

160/195

82.1%

160/195

scored in **Mock Test** in 21 min 10 sec on 7 Aug 2025 19:04:34 IST

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Find the Median > Coding	5 min 20 sec	70/ 105	✓
Q2	Flipping the Matrix > Coding	15 min 33 sec	90/ 90	✓

QUESTION 1

Correct Answer

Score 70

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**

```
1 #include <stdio.h>
2 void findMedian(int arr[],int n)
3 {
4     for(int i=0;i<n;i++)
5     {
6         for(int j=0;j<n;j++)
7         {
8             if(arr[j]<arr[j+1])
9             {
10                 int temp=arr[j];
11                 arr[j]=arr[j+1];
12                 arr[j+1]=temp;
13             }
14         }
15     }
16     printf("%d",arr[n/2]);
17 }
18 }
19 int main()
20 {
21     int n;
22     scanf("%d",&n);
23     int arr[n];
24     for(int i=0;i<n;i++)
```

```

25     {
26         scanf("%d",&arr[i]);
27     }
28     findMedian(arr,n);
29 }

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✔ Success	0	0.0087 sec	7.25 KB
Testcase 2	Easy	Hidden case	✔ Success	35	0.0849 sec	7 KB
Testcase 3	Easy	Hidden case	✔ Success	35	0.1035 sec	7 KB
Testcase 4	Easy	Hidden case	✘ Terminated due to timeout	0	2.0026 sec	7 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	

Sample Output

414

Explanation

Start out with the following *2n* × *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* × *n* submatrix in the upper-left quadrant:

2. Reverse column 2 ([83, 56, 101, 114] → [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] → [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* × *n* submatrix in the upper-left quadrant is 119 + 114 + 56 + 125 = 414

CANDIDATE ANSWER

```

1  #include <stdio.h>
2  int main()
3  {
4      int q;
5      scanf("%d",&q);
6      while(q--)
7      {
8          int n;
9          scanf("%d",&n);
10         int size=2*n;
11         int matrix[256][256];
12         for(int i=0;i<size;i++)
13         {
14             for(int j=0;j<size;j++)
15             {
16                 scanf("%d",&matrix[i][j]);
17             }
18         }
19         int sum=0;
20         for(int i=0;i<n;i++)
21         {
22             for(int j=0;j<n;j++)
23             {
24                 int a=matrix[i][j];
25                 int b=matrix[i][size-1-j];
26                 int c=matrix[size-1-i][j];
27                 int d=matrix[size-1-i][size-1-j];
28                 int max=a;
29                 if (b>max)
30                 {
31                     max=b;
32                 }
33                 if (c>max)
34                 {
35                     max=c;
36                 }
37                 if (d>max)
38                 {
39                     max=d;
40                 }
41                 sum+=max;
42             }
43         }
44         printf("%d\n",sum);
45     }
46 }
47
48 return 0;
49 }

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✔ Success	0	0.0078 sec	6.88 KB
Testcase 2	Easy	Hidden case	✔ Success	15	0.0297 sec	7.13 KB
Testcase 3	Easy	Hidden case	✔ Success	15	0.0497 sec	7.5 KB
Testcase 4	Easy	Hidden case	✔ Success	15	0.0238 sec	7.5 KB
Testcase 5	Easy	Hidden case	✔ Success	15	0.0314 sec	7.38 KB
Testcase 6	Easy	Hidden case	✔ Success	15	0.0379 sec	7.5 KB

Testcase 7	Easy	Hidden case	✔ Success	15	0.04 sec	7.38 KB
Testcase 8	Easy	Sample case	✔ Success	0	0.0074 sec	7.38 KB

No Comments

PDF generated at: 7 Aug 2025 13:58:34 UTC