

Mock Test > lorenzo@deluca.pro

 Full Name:
 Lorenzo De Luca

 Email:
 lorenzo@deluca.pro

 Test Name:
 Mock Test

 Taken On:
 7 Jun 2025 00:50:26 IST

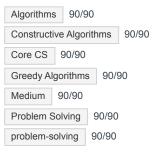
Time Taken: 7 Jun 2025 00:50:26 15 1

Invited by: Ankush

Invited on: 7 Jun 2025 00:50:18 IST

Skills Score:

Tags Score:



100% 90/90

scored in **Mock Test** in 5 min 47 sec on 7 Jun 2025 00:50:26 IST

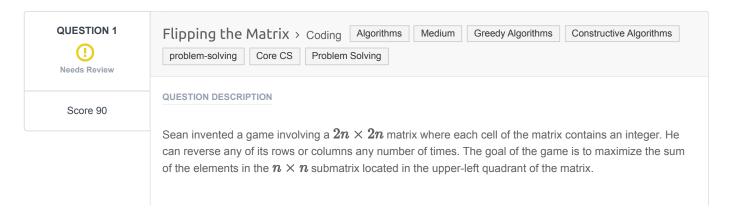
Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review it in detail here -

Question Description	Time Taken	Score	Status
Q1 Flipping the Matrix > Coding	5 min 38 sec	90/ 90	(!)



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

Constraints

- $1 \le q \le 16$
- $1 \le n \le 128$
- $ullet 0 \leq matrix[i][j] \leq 4096$, where $0 \leq i,j < 2n$.

Sample Input

```
Function

q = 1

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:

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

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:

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

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: Java 8

```
2 class Result {
4
       * Complete the 'flippingMatrix' function below.
       * The function is expected to return an INTEGER.
       * The function accepts 2D INTEGER ARRAY matrix as parameter.
       */
       public static int flippingMatrix(List<List<Integer>> matrix) {
         int n2=matrix.size(); //2*n
           int n=matrix.size()/2;
          List<Integer> nToSum = new ArrayList<Integer>();
           //for each position of the submatrix i find the highest number that
18 can go there
           for(int r=0,c=0;r<n&&c<n;){//row column of the submatrix</pre>
               //i find all the numbers that could go in that cell
               List<Integer> poss=new ArrayList<Integer>();
              //top-left
               poss.add(matrix.get(r).get(c));
               //top-right
               poss.add(matrix.get(r).get(n2-1-c));
               //bottom-right
               poss.add(matrix.get(n2-1-r).get(n2-1-c));
               //bottom-left
               poss.add(matrix.get(n2-1-r).get(c));
               nToSum.add(poss.stream().max(Comparator.naturalOrder()).get());
               c++;
               if(c==n){
```

```
34
                       c=0;
                       r++;
                  }
             return nToSum.stream().mapToInt(Integer::intValue).sum();
40
42 }
                                            STATUS
                                                        SCORE
                                                                               MEMORY USED
   TESTCASE
               DIFFICULTY
                                TYPE
                                                                 TIME TAKEN
                                                                                   32 KB
  Testcase 1
                  Easy
                             Sample case
                                           Success
                                                          0
                                                                  0.1185 sec
                                                                                   90.1 KB
  Testcase 2
                  Easy
                             Hidden case
                                           Success
                                                          15
                                                                  0.4238 sec
                                                                  0.4747 sec
                                                                                   126 KB
  Testcase 3
                  Easy
                             Hidden case
                                           Success
                                                          15
  Testcase 4
                  Easy
                             Hidden case
                                           Success
                                                          15
                                                                  0.4611 sec
                                                                                   76.9 KB
  Testcase 5
                  Easy
                             Hidden case
                                           Success
                                                          15
                                                                  0.4164 sec
                                                                                   102 KB
  Testcase 6
                  Easy
                                                          15
                                                                  0.4541 sec
                                                                                   112 KB
                             Hidden case
                                           Success
  Testcase 7
                  Easy
                             Hidden case
                                           Success
                                                          15
                                                                  0.5115 sec
                                                                                   118 KB
  Testcase 8
                  Easy
                             Sample case

    Success

                                                                  0.1213 sec
                                                                                   31.8 KB
No Comments
```

PDF generated at: 6 Jun 2025 19:27:38 UTC