H

Full Name: Wilson Javier Almario Rodriguez

Email: wilson.jar21@gmail.com

Test Name: Mock Test

Taken On: 28 May 2022 09:51:36 IST

Time Taken: 5 min 3 sec/ 24 min

Linkedin: http://www.linkedin.com/in/wjalmarior

Invited by: Ankush

Invited on: 28 May 2022 09:51:30 IST

Skills Score:

Tags Score: Algorithms 90/90

Constructive Algorithms 90/90

Core CS 90/90

Greedy Algorithms 90/90

Medium 90/90

Problem Solving 90/90 problem-solving 90/90

100% 90/90

scored in **Mock Test** in 5 min 3 sec on 28 May 2022 09:51:36 IST

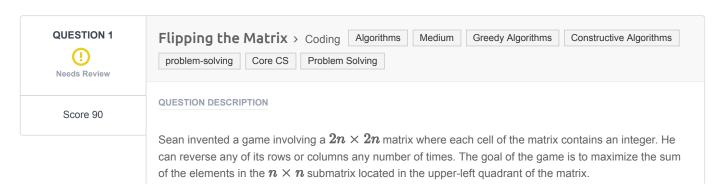
Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review.

Question Description	Time Taken	Score	Status
Q1 Flipping the Matrix > Coding	4 min 33 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

- $1 \le q \le 16$
- $1 \le n \le 128$
- $0 \leq matrix[i][j] \leq 4096$, 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 $n \times n$ submatrix in the upper-left quadrant is 119+114+56+125=414 .

CANDIDATE ANSWER

Language used: Python 3

```
1 #!/bin/python3
 3 import math
 4 import os
5 import random
6 import re
7 import sys
 8
11 #
12 # Complete the 'flippingMatrix' function below.
13 #
14 # The function is expected to return an INTEGER.
15 # The function accepts 2D INTEGER ARRAY matrix as parameter.
16 #
18 def flippingMatrix(matrix,n):
      # Write your code here
      #n = len(matrix)/2
     m = 0
     lst = []
      for i in range(n):
         for j in range(n):
             lst = []
             lst.append(matrix[i][j])
              lst.append(matrix[i][2*n-1-j])
             lst.append(matrix[2*n-1-i][j])
             lst.append(matrix[2*n-1-i][2*n-1-j])
              m += max(lst)
      return m:
```

```
34 if __name__ == '__main__':
     fptr = open(os.environ['OUTPUT PATH'], 'w')
      q = int(input().strip())
      for q_itr in range(q):
40
         n = int(input().strip())
41
42
          matrix = []
         for _{-} in range(2 * n):
44
45
              matrix.append(list(map(int, input().rstrip().split())))
46
47
         result = flippingMatrix(matrix,n)
48
          fptr.write(str(result) + '\n')
       fptr.close()
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	Success	0	0.0414 sec	9.32 KB
Testcase 2	Easy	Hidden case	Success	15	0.2264 sec	11.6 KB
Testcase 3	Easy	Hidden case	Success	15	0.2685 sec	11.7 KB
Testcase 4	Easy	Hidden case	Success	15	0.1476 sec	11.6 KB
Testcase 5	Easy	Hidden case	Success	15	0.2865 sec	11.7 KB
Testcase 6	Easy	Hidden case	Success	15	0.2562 sec	11.5 KB
Testcase 7	Easy	Hidden case	Success	15	0.2531 sec	11.6 KB
Testcase 8	Easy	Sample case	Success	0	0.109 sec	9.55 KB

No Comments

PDF generated at: 28 May 2022 04:28:01 UTC