Full Name:	Dody Apriyanto S
Email:	dody.apriyanto@gmail.com
Test Name:	**Mock Test**
Taken On:	19 Jun 2022 18:52:06 IST
Time Taken:	22 min/ 24 min
Invited by:	Ankush
Invited on:	19 Jun 2022 18:51:53 IST
Skills Score:	
Tags Score:	Algorithms 0/90
	Constructive Algorithms 0/90
	Core CS 0/90
	Greedy Algorithms 0/90
	Medium 0/90
	Problem Solving 0/90
	problem-solving 0/90

0%

0/90

scored in **Mock Test** in 22 min on 19 Jun 2022 18:52:06 IST

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Flipping the Matrix > **Coding**	21 min 52 sec	0/ 90	⊗

QUESTION 1

⊗ Wrong Answer

Score 0

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: **Python 3**

```python
#
# Complete the 'flippingMatrix' function below.
#
# The function is expected to return an INTEGER.
# The function accepts 2D_INTEGER_ARRAY matrix as parameter.
#
maximumValue_recorded_before = 0
maximumValue_recorded = 0
matrixIN = []

def flippingMatrix(matrix):
    # Write your code here
    global matrixIN
    matrixIN = matrix

    #filter
    if ((int(len(matrixIN)/2) < 1) or (int(len(matrixIN)/2) > 128)):
        return

    currentCalculated = 0
    for counterxx in range(0, int(len(matrixIN))):
        for counterzz in range(0, int(len(matrixIN))):
            if ((matrixIN[counterxx][counterzz] > 4096) or
(matrixIN[counterxx][counterzz] < 0)):
                return

    # flip it
    MaximumSum = False
    counterMax = 0
    while (MaximumSum == False):
        CalculateANDprintMaximum()
        JustFlip_row()
        JustFlip_coloumn()
        if (maximumValue_recorded == maximumValue_recorded_before):
            counterMax+=1
            if (counterMax > 500):
                MaximumSum = True
        else:
            counterMax=0
    return (maximumValue_recorded)


def CalculateANDprintMaximum():
    global maximumValue_recorded
    global maximumValue_recorded_before
    maximumValue_recorded_before = maximumValue_recorded
    currentCalculated = countMaximum()
    if (currentCalculated > maximumValue_recorded):
        maximumValue_recorded = currentCalculated

def countMaximum():
    currentCalculated = matrixIN[0][0]
    if (int(len(matrixIN)/2) > 1):
        currentCalculated = 0
        for counterxx in range(0, int(len(matrixIN)/2)):
            for counterzz in range(0, int(len(matrixIN)/2)):
                currentCalculated = currentCalculated + matrixIN[counterxx]
[counterzz]
    return(currentCalculated)

def JustFlip_row():
    global matrixIN
    matrixIN[random.randint(0, len(matrixIN)-1)].reverse()

def JustFlip_coloumn():
    global matrixIN
    i= random.randint(0, len(matrixIN)-1)
    # reverse it
    matrixtemp = []
    for counterii in range(0,len(matrixIN)):
        matrixtemp.append(matrixIN[counterii][i])
    matrixtemp.reverse()
    #copyback
    for counterii in range(0,len(matrixIN)):
        matrixIN[counterii][i] = matrixtemp[counterii]

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✓ Success	0	0.0794 sec	9.47 KB
Testcase 2	Easy	Hidden case	✗ Wrong Answer	0	3.992 sec	12.5 KB
Testcase 3	Easy	Hidden case	✗ Wrong Answer	0	7.4022 sec	14 KB
Testcase 4	Easy	Hidden case	✗ Wrong Answer	0	2.8459 sec	12.7 KB
Testcase 5	Easy	Hidden case	✗ Wrong Answer	0	3.2451 sec	13.5 KB
Testcase 6	Easy	Hidden case	✗ Wrong Answer	0	5.457 sec	12.5 KB
Testcase 7	Easy	Hidden case	✗ Wrong Answer	0	8.1315 sec	13.3 KB
Testcase 8	Easy	Sample case	✗ Wrong Answer	0	0.0726 sec	9.58 KB

No Comments