Full Name:	Mohamed Samir Salem Hassan
Email:	mohamedkhalilalmasri@gmail.com
Test Name:	**Mock Test**
Taken On:	21 Oct 2023 01:48:42 IST
Time Taken:	12 min 23 sec/ 24 min
Resume:	https://hackerrank-resumes.s3.amazonaws.com/12216067/NeyaRbrS_h60_OczMKX3xAyHjQwG1mkEovOLeLB9fntAeIVcNTf7D8FqE5dM72JFyA/Mohamed_Samir_Salem___Flutter_Developer.pdf
Linkedin:	https://www.linkedin.com/in/mohamed-samir-9b0b2a203
Invited by:	Ankush
Invited on:	21 Oct 2023 01:48:26 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 12 min 23 sec on 21 Oct 2023 01:48:42 IST

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	**Flipping the Matrix** > **Coding**	12 min 16 sec	90/ 90	✓

QUESTION 1

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]]
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: **C++14**

```cpp
#include "bits/stdc++.h"

using namespace std;

typedef long long int ll;
#define all(a)  a.begin(),a.end()

ll dx[] = {+0, +0, -1, +1, +1, +1, -1, -1};
ll dy[] = {-1, +1, +0, +0, +1, -1, +1, -1};

void _Depressed() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
// #ifndef ONLINE_JUDGE
//     freopen("input.in", "r", stdin);
//     freopen("output.txt", "w", stdout);
// #endif
}

const ll OO = 0X3F3F3F3F3F3F3F3F;
const ll N = 1e5 + 5, INF = INT_MAX, MOD = 1e9 + 7, LOG = 20;



void solve(ll test_case) {
    int q;
    cin >> q;

    while(q--) {
        int n;
        cin >> n;


        int mtrx[2*n+1][2*n+1];
        for(int i=1;i<=n * 2 ; i++) {
            for(int j= 1; j <= n*2 ; j++ ) {
                cin >> mtrx[i][j];
            }
        }

        int ANS = 0;
        for(int i=1;i<=n;i++) {
            for(int j = 1 ; j <= n; j ++ ){
                int i2 = n + n + 1 - i;
                int j2 = n + n + 1 - j;

```

```
                ANS += max(max(mtrx[i][j], mtrx[i2][j]), max(mtrx[i][j2],
mtrx[i2][j2]));
            }
        }
        cout << ANS <<endl;
    }

}

int main() {
    _Depressed();
    ll tc;
    tc = 1;
//    cin >> tc;
    for (ll test_case = 1; test_case <= tc; test_case++) {
        solve(test_case);
    }
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	Success	0	0.0591 sec	8.68 KB
Testcase 2	Easy	Hidden case	Success	15	0.0599 sec	9.25 KB
Testcase 3	Easy	Hidden case	Success	15	0.082 sec	9.25 KB
Testcase 4	Easy	Hidden case	Success	15	0.0844 sec	9.14 KB
Testcase 5	Easy	Hidden case	Success	15	0.0792 sec	9.2 KB
Testcase 6	Easy	Hidden case	Success	15	0.0914 sec	8.94 KB
Testcase 7	Easy	Hidden case	Success	15	0.0869 sec	8.78 KB
Testcase 8	Easy	Sample case	Success	0	0.0433 sec	8.89 KB

No Comments