Full Name:	Shahar Asher
Email:	shaharas30@gmail.com
Test Name:	**Mock Test**
Taken On:	24 Nov 2024 16:14:16 IST
Time Taken:	7 min 18 sec/ 24 min
Invited by:	Ankush
Invited on:	24 Nov 2024 16:14:06 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 7 min 18 sec on 24 Nov 2024 16:14:16 IST

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Flipping the Matrix > **Coding**	6 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 <algorithm>
#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);



/*
 * Complete the 'flippingMatrix' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts 2D_INTEGER_ARRAY matrix as parameter.
 */

int flippingMatrix(vector<vector<int>> matrix) {
    int res = 0,
    c_r_len = matrix.size();
    for(size_t r = 0; r<c_r_len/2; ++r)
    {
        for(size_t c = 0; c<c_r_len/2; ++c)
        {
            int temp_sum = matrix[r][c];
            temp_sum = std::max(temp_sum, matrix[r][c_r_len-1-c]);
            temp_sum = std::max(temp_sum, matrix[c_r_len-1-r][c_r_len-1-c]);
            temp_sum = std::max(temp_sum, matrix[c_r_len-1-r][c]);
            res+=temp_sum;
        }
    }
    return res;
}

```

```cpp
int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    string q_temp;
    getline(cin, q_temp);

    int q = stoi(ltrim(rtrim(q_temp)));

    for (int q_itr = 0; q_itr < q; q_itr++) {
        string n_temp;
        getline(cin, n_temp);

        int n = stoi(ltrim(rtrim(n_temp)));

        vector<vector<int>> matrix(2 * n);

        for (int i = 0; i < 2 * n; i++) {
            matrix[i].resize(2 * n);

            string matrix_row_temp_temp;
            getline(cin, matrix_row_temp_temp);

            vector<string> matrix_row_temp =
split(rtrim(matrix_row_temp_temp));

            for (int j = 0; j < 2 * n; j++) {
                int matrix_row_item = stoi(matrix_row_temp[j]);

                matrix[i][j] = matrix_row_item;
            }
        }

        int result = flippingMatrix(matrix);

        fout << result << "\n";
    }

    fout.close();

    return 0;
}

string ltrim(const string &str) {
    string s(str);

    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
    );

    return s;
}

string rtrim(const string &str) {
    string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>
(isspace))).base(),
        s.end()
    );

```

```
    return s;
}

vector<string> split(const string &str) {
    vector<string> tokens;

    string::size_type start = 0;
    string::size_type end = 0;

    while ((end = str.find(" ", start)) != string::npos) {
        tokens.push_back(str.substr(start, end - start));

        start = end + 1;
    }

    tokens.push_back(str.substr(start));

    return tokens;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	Success	0	0.0105 sec	8.9 KB
Testcase 2	Easy	Hidden case	Success	15	0.0554 sec	9.37 KB
Testcase 3	Easy	Hidden case	Success	15	0.0778 sec	9.27 KB
Testcase 4	Easy	Hidden case	Success	15	0.0377 sec	9.2 KB
Testcase 5	Easy	Hidden case	Success	15	0.0653 sec	9.34 KB
Testcase 6	Easy	Hidden case	Success	15	0.0725 sec	9.3 KB
Testcase 7	Easy	Hidden case	Success	15	0.087 sec	9.35 KB
Testcase 8	Easy	Sample case	Success	0	0.0085 sec	8.9 KB

No Comments