



Full Name: Desmond Wong

Email: 29dweh@gmail.com

Test Name: Mock Test

Taken On: 9 Mar 2024 06:05:43 IST

Time Taken: 29 min 52 sec/ 30 min

Linkedin: <https://www.linkedin.com/in/desmond-wong-62579521b/>

Invited by: Ankush

Invited on: 5 Mar 2024 04:55:07 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 29 min 52 sec on 9 Mar 2024 06:05:43 IST

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Flipping the Matrix > Coding	24 min 49 sec	90/ 90	✓

QUESTION 1



Correct Answer

Score 90

Flipping the Matrix > Coding

- problem-solving
- Core CS
- Problem Solving

Algorithms

Medium

Greedy Algorithms

Constructive Algorithms

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

- $1 \leq q \leq 16$
- $1 \leq n \leq 128$
- $0 \leq \text{matrix}[i][j] \leq 4096$, 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**

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  string ltrim(const string &);
6  string rtrim(const string &);
7  vector<string> split(const string &);
8
9  /*
10 * Complete the 'flippingMatrix' function below.
11 *
12 * The function is expected to return an INTEGER.
13 * The function accepts 2D_INTEGER_ARRAY matrix as parameter.
14 */
15 int flippingMatrix(vector<vector<int>> matrix)
16 {
17     int sum = 0;
18     int n = matrix.size();
19
20     for (int i = 0; i < n / 2; i++)
21     {
22         for (int j = 0; j < n / 2; j++)
23         {
24             sum +=
25                 max(matrix[i][j],
26                     (max(matrix[i][n - 1 - j],
27                         (max(matrix[n - 1 - i][j], matrix[n - 1 - i][n - 1 -
28 j])))));
29         }
30     }
31     return sum;
32 }
33
34 int main()
35 {

```

```

36     ofstream fout(getenv("OUTPUT_PATH"));
37
38     string q_temp;
39     getline(cin, q_temp);
40
41     int q = stoi(ltrim(rtrim(q_temp)));
42
43     for (int q_itr = 0; q_itr < q; q_itr++)
44     {
45         string n_temp;
46         getline(cin, n_temp);
47
48         int n = stoi(ltrim(rtrim(n_temp)));
49
50         vector<vector<int>> matrix(2 * n);
51
52         for (int i = 0; i < 2 * n; i++)
53         {
54             matrix[i].resize(2 * n);
55
56             string matrix_row_temp_temp;
57             getline(cin, matrix_row_temp_temp);
58
59             vector<string> matrix_row_temp =
60 split(rtrim(matrix_row_temp_temp));
61
62             for (int j = 0; j < 2 * n; j++)
63             {
64                 int matrix_row_item = stoi(matrix_row_temp[j]);
65
66                 matrix[i][j] = matrix_row_item;
67             }
68         }
69
70         int result = flippingMatrix(matrix);
71
72         fout << result << "\n";
73     }
74
75     fout.close();
76
77     return 0;
78 }
79
80 string ltrim(const string &str)
81 {
82     string s(str);
83
84     s.erase(
85         s.begin(),
86         find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace))));
87
88     return s;
89 }
90
91 string rtrim(const string &str)
92 {
93     string s(str);
94
95     s.erase(
96         find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>
97 (isspace))).base(),
98         s.end());

```

```

99     return s;
100 }
101
102 vector<string> split(const string &str)
103 {
104     vector<string> tokens;
105
106     string::size_type start = 0;
107     string::size_type end = 0;
108
109     while ((end = str.find(" ", start)) != string::npos)
110     {
111         tokens.push_back(str.substr(start, end - start));
112
113         start = end + 1;
114     }
115
116     tokens.push_back(str.substr(start));
117
118     return tokens;
119 }

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✔ Success	0	0.0061 sec	8.75 KB
Testcase 2	Easy	Hidden case	✔ Success	15	0.0526 sec	9.21 KB
Testcase 3	Easy	Hidden case	✔ Success	15	0.0704 sec	9.36 KB
Testcase 4	Easy	Hidden case	✔ Success	15	0.0428 sec	9.39 KB
Testcase 5	Easy	Hidden case	✔ Success	15	0.0486 sec	9.14 KB
Testcase 6	Easy	Hidden case	✔ Success	15	0.088 sec	9.28 KB
Testcase 7	Easy	Hidden case	✔ Success	15	0.0695 sec	9.18 KB
Testcase 8	Easy	Sample case	✔ Success	0	0.0062 sec	8.99 KB

No Comments