

### Mock Test > 29dweh@gmail.com

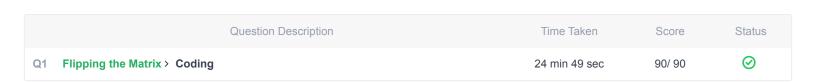
**Full Name:** Desmond Wong Email: 29dweh@gmail.com Test Name: **Mock Test** Taken On: 9 Mar 2024 06:05:43 IST Time Taken: 29 min 52 sec/ 30 min Linkedin: https://www.linkedin.com/in/desmond-wong-62579521b/ Invited by: Ankush Invited on: 5 Mar 2024 04:55:07 IST Skills Score: Tags Score: Algorithms 90/90 Constructive Algorithms 90/90 Core CS 90/90 Greedy Algorithms 90/90 90/90 Medium Problem Solving 90/90 problem-solving 90/90

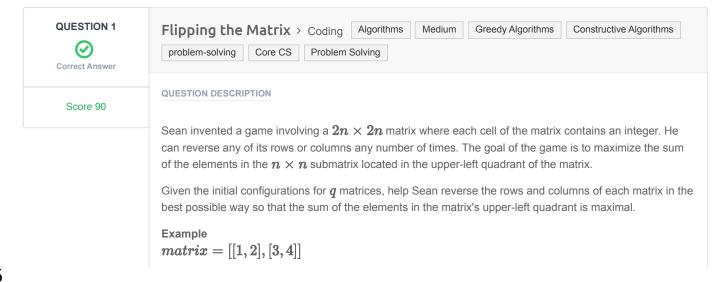
**100%** 90/90

scored in **Mock Test** in 29 min 52 sec on 9 Mar 2024 06:05:43 IST

# Recruiter/Team Comments:

No Comments.





```
1 2
3 4
```

It is  $2 \times 2$  and we want to maximize the top left quadrant, a  $1 \times 1$  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 imes 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] ightarrow [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] ightarrow [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 imes n submatrix in the upper-left quadrant is 119+114+56+125=414

### **CANDIDATE ANSWER**

### Language used: C++14

```
1 #include <bits/stdc++.h>
 3 using namespace std;
 5 string ltrim(const string &);
 6 string rtrim(const string &);
7 vector<string> split(const string &);
9 /*
* Complete the 'flippingMatrix' function below.
11 *
12 * The function is expected to return an INTEGER.
   * The function accepts 2D INTEGER ARRAY matrix as parameter.
15 int flippingMatrix(vector<vector<int>> matrix)
16 {
      int sum = 0;
     int n = matrix.size();
     for (int i = 0; i < n / 2; i++)
          for (int j = 0; j < n / 2; j++)
         {
              sum +=
                 max(matrix[i][j],
                   (\max(\max[i][n-1-j],
                          (\max(\max[n-1-i][j], \max[n-1-i][n-1-i]]
28 j])))));
      return sum;
32 }
34 int main()
35 {
```

```
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++)
54
               matrix[i].resize(2 * n);
               string matrix row temp temp;
               getline(cin, matrix_row_temp_temp);
               vector<string> matrix row temp =
60 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";</pre>
       fout.close();
       return 0;
78 }
80 string ltrim(const string &str)
81 {
      string s(str);
      s.erase(
           s.begin(),
           find if(s.begin(), s.end(), not1(ptr fun<int, int>(isspace))));
       return s;
89 }
91 string rtrim(const string &str)
92 {
       string s(str);
       s.erase(
           find if(s.rbegin(), s.rend(), not1(ptr fun<int, int>
97 (isspace))).base(),
           s.end());
```

```
return s;
10 }
10 vector<string> split(const string &str)
10 {
18
     vector<string> tokens;
10
16
     string::size_type start = 0;
16
      string::size_type end = 0;
10
      while ((end = str.find(" ", start)) != string::npos)
19
10
          tokens.push_back(str.substr(start, end - start));
12
          start = end + 1;
13
      }
14
15
      tokens.push back(str.substr(start));
16
17
      return tokens;
8 }
```

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

PDF generated at: 9 Mar 2024 01:07:46 UTC