

flipping the matrix hackerrank solution

Flipping the matrix is a popular problem often encountered in coding challenges, particularly on platforms like HackerRank. This problem requires participants to manipulate a given matrix through specific operations, ultimately aiming to maximize the sum of elements in a particular quadrant of the matrix. In this article, we will explore the problem statement, break down the possible approaches to find a solution, and provide a detailed explanation of an optimal algorithm to solve the problem effectively.

Understanding the Problem

The flipping the matrix problem typically presents a square matrix of size $n \times n$, which contains integers. The goal is to perform a series of operations to maximize the sum of the elements in the top-left quadrant of the matrix after all possible flips.

Problem Statement

Given an $n \times n$ matrix, which can be divided into four quadrants:

- Quadrant 1 (Q1): top-left
- Quadrant 2 (Q2): top-right
- Quadrant 3 (Q3): bottom-left
- Quadrant 4 (Q4): bottom-right

You can flip elements from any quadrant into another quadrant to achieve the maximum possible sum of the elements in Q1. A flip can be described as taking the element from one quadrant and "flipping" it to the corresponding position in another quadrant.

Example

For example, consider the following 4×4 matrix:

```
...
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
...
```

The quadrants are:

- Q1: 1, 2

- Q2: 3, 4
- Q3: 9, 10
- Q4: 11, 12

You can choose to flip elements between these quadrants, and your goal is to maximize the sum of Q1.

Approach to the Solution

To tackle the flipping the matrix problem, we need a systematic approach. Here's a breakdown of how to achieve that:

Step 1: Matrix Representation

We will represent the matrix in a 2D list format in Python. The way we access the matrix will depend on the quadrant we're interested in.

Step 2: Identify Quadrant Elements

For each element in the top-left quadrant (Q1), we need to consider the corresponding elements in the other quadrants (Q2, Q3, Q4) that could potentially be flipped into Q1. Specifically, for each element at position (i, j) in Q1, we will check the elements at positions:

- Q2: $(i, n-j-1)$
- Q3: $(n-i-1, j)$
- Q4: $(n-i-1, n-j-1)$

Step 3: Calculate Maximum Values

For each of the four positions corresponding to the element in Q1, we will take the maximum value. This ensures that we are always maximizing the sum of elements in Q1 after performing the flips.

Step 4: Sum Up the Maximum Values

Once we have computed the maximum value for each position in Q1, we can sum these values to get the final result.

Implementation

Now that we have a clear approach, let's implement the solution in Python:

```
```python
def flippingMatrix(matrix):
 n = len(matrix) // 2 Since the matrix is n x n, n/2 gives us the size of the
```

```

quadrants
total_sum = 0

Iterate through each element in the top-left quadrant
for i in range(n):
 for j in range(n):
 Identify the four possible flips
 top_left = matrix[i][j]
 top_right = matrix[i][len(matrix) - j - 1]
 bottom_left = matrix[len(matrix) - i - 1][j]
 bottom_right = matrix[len(matrix) - i - 1][len(matrix) - j - 1]

 Take the maximum of the four
 total_sum += max(top_left, top_right, bottom_left, bottom_right)

return total_sum
'''

```

### Explanation of the Code

1. Input Matrix: The function `flippingMatrix` takes a 2D list called `matrix` as input.
2. Quadrant Size: We calculate `n`, which is half the size of the matrix.
3. Iterate Through Q1: We loop through each element of the top-left quadrant (Q1).
4. Calculate Flips: For each element, we determine the corresponding elements in Q2, Q3, and Q4.
5. Max Calculation: We use the `max()` function to find the highest value among the four possible options.
6. Sum the Values: Finally, we accumulate these maximum values into `total\_sum`.

## Complexity Analysis

The time complexity of the above solution can be analyzed as follows:

- Time Complexity:  $O(n^2)$ . Since we are iterating through half of the matrix ( $n \times n / 4$ ), the time complexity effectively remains  $O(n^2)$  due to the nested loops.
- Space Complexity:  $O(1)$ . We are not using any additional data structures that scale with the input size, as we are only using a few variables for computation.

## Conclusion

The flipping the matrix problem is an excellent exercise in matrix

manipulation and optimization. By understanding the quadrant structure and carefully considering the possible flips, we can devise an efficient solution to achieve the maximum sum in the desired quadrant. This problem not only helps in strengthening algorithmic skills but also enhances problem-solving capabilities by encouraging logical thinking about data structures.

## **Frequently Asked Questions**

### **What is the 'Flipping the Matrix' problem on HackerRank?**

The 'Flipping the Matrix' problem involves flipping specific quadrants of a given 2D matrix to maximize the sum of the top-left quadrant. The goal is to strategically flip the matrix's quadrants to achieve the highest possible sum.

### **How do you approach solving the 'Flipping the Matrix' challenge?**

To solve the 'Flipping the Matrix' challenge, you can iterate through the top-left quadrant and compare values from the opposite quadrants. For each cell, you take the maximum of the four possible values that can appear in that cell after any number of flips.

### **What is the time complexity of the optimal solution for 'Flipping the Matrix'?**

The optimal solution for 'Flipping the Matrix' has a time complexity of  $O(n^2)$ , where  $n$  is the dimension of the matrix. This is due to the need to iterate through each element of the matrix and perform constant-time operations for each element.

### **Can you explain the flipping mechanism in the 'Flipping the Matrix' problem?**

In the 'Flipping the Matrix' problem, flipping a quadrant means switching it with its opposite quadrant. For a given cell  $(i, j)$  in the top-left quadrant, you can consider its corresponding cells in the other three quadrants:  $(i, n-j-1)$ ,  $(n-i-1, j)$ , and  $(n-i-1, n-j-1)$  to determine the maximum value possible after flips.

### **What are common pitfalls when solving 'Flipping the Matrix'?**

Common pitfalls include not correctly identifying the pairs of indices that

correspond to the quadrants, failing to consider all four possible values for each position, and not properly accounting for the matrix's dimensions when iterating through the elements.

## **Is there a space complexity concern when solving 'Flipping the Matrix'?**

The space complexity for the optimal solution of 'Flipping the Matrix' is  $O(1)$  since you can compute the maximum values in-place without needing additional data structures, aside from a few variables for tracking sums.

## **[Flipping The Matrix Hackerrank Solution](#)**

Flipping The Matrix Hackerrank Solution

## **Related Articles**

- [ffiec cybersecurity assessment tool cat](#)
- [florida child support guideline worksheet](#)
- [fisher bluetooth speaker manual](#)

[Back to Home](#)