

flipping matrix hackerrank solution

Flipping matrix hackerrank solution is a popular problem that challenges programmers to maximize the sum of a matrix after performing a series of flips. This problem not only tests the algorithmic skills of programmers but also provides insights into matrix manipulation and optimization techniques. In this article, we will delve into the details of the Flipping Matrix problem, discuss its requirements, explore different approaches to solving it, and finally provide a comprehensive solution.

Understanding the Problem

The Flipping Matrix problem is set in a grid or matrix format, where each cell in the matrix has a specific value. The goal is to maximize the sum of the values in the upper-left quadrant of the matrix after performing several flipping operations. A flip operation allows you to choose a cell and flip the values of that cell and its symmetric counterpart across the center of the matrix.

Problem Statement

Given a matrix of size $n \times n$, you can flip the values as follows:

- Choose a cell (i, j) .
- Flip the values of the cell (i, j) and its symmetric counterparts:
 - $(n-1-i, j)$
 - $(i, n-1-j)$
 - $(n-1-i, n-1-j)$

Your task is to perform these flips in such a way as to maximize the sum of the upper-left quadrant of the matrix (the first $n/2 \times n/2$ section).

Constraints

- The matrix size n will always be even.
- The values in the matrix can be positive or negative.
- You can perform as many flips as needed.

Approach to the Solution

To solve the Flipping Matrix problem, we will follow a systematic approach. Here are the key steps to consider:

1. Identify the Symmetric Positions

For any cell (i, j) in the upper-left quadrant, its symmetric counterparts in the other quadrants can be identified as:

- (i, j)

- (n-1-i, j)
- (i, n-1-j)
- (n-1-i, n-1-j)

This means that for each cell in the upper-left quadrant, you need to consider the four possible values from the symmetric positions.

2. Calculate Maximum Values

For each group of symmetric positions, determine the maximum value. Instead of simply summing the values in the upper-left quadrant, we will sum the maximum values from each group. This ensures that we are always selecting the best option available after flips.

3. Implement the Solution

The implementation can be done using a nested loop to traverse only the upper-left quadrant of the matrix while calculating the maximum values for each group of four symmetric cells.

Implementation of the Solution

Here is a Python implementation of the Flipping Matrix solution:

```
```python
def flippingMatrix(matrix):
 n = len(matrix) Assuming matrix is n x n
 total_sum = 0

 Traverse the upper-left quadrant
 for i in range(n // 2):
 for j in range(n // 2):
 Identify the four symmetric positions
 top_left = matrix[i][j]
 top_right = matrix[i][n - 1 - j]
 bottom_left = matrix[n - 1 - i][j]
 bottom_right = matrix[n - 1 - i][n - 1 - j]

 Find the maximum among the four positions
 total_sum += max(top_left, top_right, bottom_left, bottom_right)

 return total_sum

Example usage:
matrix = [
 [1, 2, 3, 4],
 [5, 6, 7, 8],
 [9, 10, 11, 12],
 [13, 14, 15, 16]
]
print(flippingMatrix(matrix)) Output should be maximum sum
```
```

Understanding the Code

- The function `flippingMatrix(matrix)` takes a 2D list (matrix) as input.
- It calculates the size of the matrix and initializes a variable `total_sum` to store the maximum sum.
- The nested loops iterate through each cell in the upper-left quadrant.
- For each cell, it retrieves the values from the four symmetric positions and computes the maximum value, adding it to the `total_sum`.
- Finally, the function returns the maximum sum.

Testing the Solution

To ensure the solution works correctly, you should test it with various matrices of different sizes and values. Here are a few test cases to consider:

- Test Case 1: Matrix with all positive values
- Test Case 2: Matrix with a mix of positive and negative values
- Test Case 3: Matrix with all negative values
- Test Case 4: Larger matrix sizes to test efficiency

Conclusion

The **flipping matrix hackerrank solution** is a great way to deepen your understanding of matrix manipulation and optimization techniques. By focusing on symmetric properties and maximizing values, you can solve the problem efficiently. We hope this article has provided a comprehensive overview of the problem and a clear solution approach. Happy coding!

Frequently Asked Questions

What is the 'Flipping Matrix' problem on HackerRank?

The 'Flipping Matrix' problem involves a 2D matrix where you can perform specific operations to maximize the sum of the top-left quadrant after flipping any of the four quadrants.

How can I approach solving the 'Flipping Matrix' problem?

A common approach is to iterate through the top-left quadrant and calculate the maximum possible value for each corresponding position in the matrix after considering all possible flips.

What are the constraints of the 'Flipping Matrix' problem?

The constraints typically include the size of the matrix, which is usually even, and limits on the values within the matrix elements, often ranging from -10^9 to 10^9 .

Can you describe a brute-force solution for the 'Flipping Matrix' problem?

A brute-force solution would involve generating all possible configurations of the matrix after flipping each quadrant and then calculating the sum of the top-left quadrant for each configuration to find the maximum.

What is a more efficient algorithm for the 'Flipping Matrix' problem?

A more efficient approach is to calculate the maximum possible value for each position in the top-left quadrant based on the corresponding positions in all four quadrants, allowing you to compute the result in $O(n^2)$ time.

How do you handle input and output for the 'Flipping Matrix' problem on HackerRank?

Input is usually read as a single matrix from standard input, and output is printed as a single integer representing the maximum sum of the top-left quadrant after optimal flips.

What programming languages can I use to solve the 'Flipping Matrix' problem?

HackerRank supports multiple programming languages for solving the 'Flipping Matrix' problem, including Python, Java, C++, and Ruby.

Are there any common pitfalls to avoid when solving the 'Flipping Matrix' problem?

Common pitfalls include not accounting for all possible flips or incorrectly indexing the matrix, which can lead to incorrect calculations of the sums.

Where can I find additional resources or discussions about the 'Flipping Matrix' problem?

You can find additional resources, sample solutions, and discussions on platforms like HackerRank's discussion forums, GitHub repositories, and programming communities such as Stack Overflow.

Flipping Matrix Hackerrank Solution

Flipping Matrix Hackerrank Solution

Related Articles

- [financial markets and institutions solution manual](#)
- [finite mathematics and calculus with applications 8th edition](#)
- [fire truck driver training](#)

[Back to Home](#)