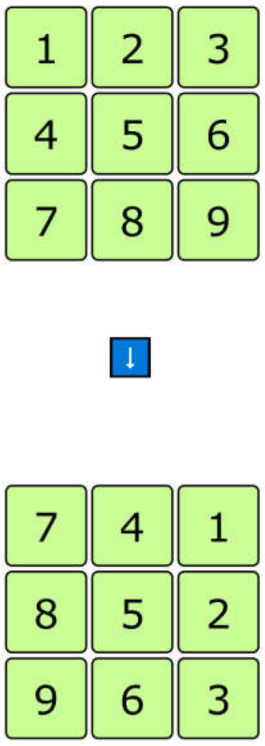
Rotate Matrix, Advanced

Learn how to rotate a matrix in place. We'll go over the complex algorithm required in heavy detail.

We'll cover the following

Rotate Square Matrix in Place#



Instructions#

Write a function that takes a square matrix as input. A square matrix has the same number of rows and columns, e.g. 3 x 3, 4 x 4, 5 x 5. It should return the same matrix rotated 90 degrees clockwise. The rotation should happen in place, meaning you may not create any extra matrixes or arrays in your function.

Input: Array of arrays of numbers

Output: Array of arrays of numbers

Example#

An input of:

```
[[1, 2, 3],
 [4, 5, 6],
 [7, 8, 9]]
```

yields an output of:

```
[[7, 4, 1],
 [8, 5, 2],
 [9, 6, 3]]
```

Hints#

1. Start at the border and work your way in.
2. We'll need some placeholder variables.

```
function rotateClockwise(mat) {
    // Your code here
}
```

Solution#

```
function rotateClockwise(mat) {
  const totalLayers = Math.floor(mat.length / 2);

  for(let layer = 0; layer < totalLayers; layer++) {
    const lastIndex = mat.length - 1 - layer;
    for(let forwardIterator = layer + 1; forwardIterator < mat.length - layer; forwa
      const reverseIterator = lastIndex - forwardIterator + layer;

      let temp1 = mat[forwardIterator][lastIndex];
      mat[forwardIterator][lastIndex] = mat[layer][forwardIterator];

      let temp2 = mat[lastIndex][reverseIterator];
      mat[lastIndex][reverseIterator] = temp1;

      temp1 = mat[reverseIterator][layer];
      mat[reverseIterator][layer] = temp2;

      mat[layer][forwardIterator] = temp1;
    }
  }

  return mat;
}
```

Show Results Show Console

1.19s

4 of 4 Public Tests Passed

Result	Input	Expected Output	Actual Output	Reason
✓	[[1]]	[[1]]	[[1]]	Succeeded
✓	[[1, 2], [3, 4]]	[[3, 1], [4, 2]]	[[3,1],[4,2]]	Succeeded
✓	[[1, 2, 3], [4, 5, 6], [7, 8, 9]]	[[7, 4, 1], [8, 5, 2], [9, 6, 3]]	[[7,4,1],[8,5,2],[9,6,3]]	Succeeded
✓	[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 1 …	[[13, 9, 5, 1], [14, 10, 6, 2], [15, 1 …	[[13,9,5,1],[14,10,6,2], [15,11,7,3],[16, …	Succeeded

The Strategy#

We'll treat the input matrix as a series of nested layers. We'll start from the outermost layer and work our way towards the center. Here's what that means, in terms of a 5x5 matrix.

If the matrix looks like this:

```
[[01, 02, 03, 04, 05],
 [06, 07, 08, 09, 10],
 [11, 12, 13, 14, 15],
 [16, 17, 18, 19, 20],
 [21, 22, 23, 24, 25]]
```

We'll focus on the outer layer first. We'll pretend the inside doesn't even exist:

```
[[01, 02, 03, 04, 05],
 [06,             10],
 [11,             15],
 [16,             20],
 [21, 22, 23, 24, 25]]
```

We'll go through and rotate the corners first. This means 01 goes to the right, 05 comes down. 25 goes left, and 21 goes up.

Then we'll send 02 to the right and down one, 10 down and to the left one, etc, etc. We'll do this until we've rotated every number in the layer.

Once the layer's done, we move inwards:

```
[[                  ],
 [    07, 08, 09    ],
 [    12,     14    ],
 [    17, 18, 19    ],
 [                  ]]
```

We do the same thing. 7 goes right, 9 goes down, etc.

After this layer is processed, we're finished. We don't need to touch the center.

Translating to Code#

Outside In#

To go from the outside in, we'll use a for-loop. Line 2 tells us how many layers we'll need to consider:

```
const totalLayers = Math.floor(mat.length / 2);
```

Here's the pattern:

- 2x2: 1 layer
- 3x3: 1 layer
- 4x4: 2 layers
- 5x5: 2 layers
- 6x6: 3 layers
- 7x7: 3 layers

The formula on line 2 determines the number of layers we'll need to consider using this pattern. It gives us the stop condition for the outer for-loop.

Once we're in the outer for-loop, we need to determine where to start and stop the inner for-loop. This logic is in line 7:

```
const reverseIterator = lastIndex - forwardIterator + layer;
```

You'll likely need to go through a matrix rotation by hand to observe how it works.

Swapping the Corners#

In the second for-loop, we start switching numbers. We need two temporary variables. To see why, look at the outermost layer of the 5x5 example above. We first need to move 01 to the right. When we do so, we need to store 05 in a temporary variable, since 01 is going to replace it.

We then need to move 05 down into 25’s spot, so we’ll need a second variable to store 25. Once we’ve replaced 25 with 05, we can overwrite the temporary variable holding 05. That’s exactly what we’ll do when we move 25 over to 21’s spot.

We’ll overwrite the variable holding 05 with 21. We’ll then replace 21 with 25. We’ll go back to the original start point and replace the top left corner with 21. We’ve successfully rotated the corners clockwise.

Swapping the Rest of the Layer#

We then need to swap 02, 10, 24, and 06, moving them all clockwise. We’ll use the same strategy as above and our inner loop will take care of it. The inner loop keeps going until we’ve rotated the entire layer.

Once the layer completes, our outer for-loop will increment and the inner loop will restart its work.

Time#

As we only process each number once, our time complexity is:

O(n),#

the same as if it weren’t an in-place rotation.

Space#

Think about the space complexity. We’re not creating any data structures at all inside our function. Our input matrix is not extended in any way. Only positions are swapped.

We *do* create some variables. However, the same exact number of variables is always created, regardless of the input, whether it’s a 2x2 or a 1000x1000. This means the space complexity is:

O(1).#

Conclusion#

This one was rough. If asked in an interview, you’ll most likely be asked to give the general strategy you would use and analyze the efficiency. Getting it correct could take a good programmer hours, much too long for an interview.

This problem allows us to think about spacially transforming data. The ability to understand this solution shows powerful logical reasoning skills and excellent spatial reasoning.

If this is too complex, try coming back to it another time. Try walking through the function using a simple matrix such as a 3 x 3. Keep track of each of the variables and what is happening to the matrix.

← Back
Rotate Matrix

Next →
Sorted Search

✓ Mark as Completed

Report an Issue