Class Notes 6

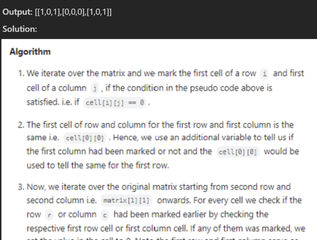
- Python
- JavaScript
- Java

Question 1

Given an m x n integer matrix matrix, if an element is 0, set its entire row and column to 0's.

You must do it in place.

Example 1:



Input: matrix = [[1,1,1],[1,0,1],[1,1,1]]

Output: [[1,0,1],[0,0,0],[1,0,1]]

Solution:

Algorithm

1. We iterate over the matrix and we mark the first cell of a row `i` and first cell of a column `j` , if the condition in the pseudo code above is satisfied. i.e. if `cell[i][j] == 0` .
2. The first cell of row and column for the first row and first column is the same i.e. `cell[0][0]` . Hence, we use an additional variable to tell us if the first column had been marked or not and the `cell[0][0]` would be used to tell the same for the first row.
3. Now, we iterate over the original matrix starting from second row and second column i.e. `matrix[1][1]` onwards. For every cell we check if the row `r` or column `c` had been marked earlier by checking the respective first row cell or first column cell. If any of them was marked, we set the value in the cell to 0. Note the first row and first column serve as the `row_set` and `column_set` that we used in the first approach.
4. We then check if `cell[0][0] == 0` , if this is the case, we mark the first row as zero.
5. And finally, we check if the first column was marked, we make all entries in it as zeros.

Complexity Analysis

- **Time Complexity:** *O(M×N)*
- **Space Complexity:** *O*(1)

```
class Solution {
  public void setZeroes(int[][] matrix) {
    Boolean isCol = false;
    int R = matrix.length;
    int C = matrix[0].length;

    for (int i = 0; i < R; i++) {

      // Since first cell for both first row and first column is the same i.e.
      // We can use an additional variable for either the first row/column.
      // For this solution we are using an additional variable for the first co
      // and using matrix[0][0] for the first row.
      if (matrix[i][0] == 0) {
        isCol = true;
      }

      for (int j = 1; j < C; j++) {
        // If an element is zero, we set the first element of the corresponding
        if (matrix[i][j] == 0) {
          matrix[0][j] = 0;
          matrix[i][0] = 0;
        }
      }
    }

    // Iterate over the array once again and using the first row and first colu
    for (int i = 1; i < R; i++) {
      for (int j = 1; j < C; j++) {
        if (matrix[i][0] == 0 || matrix[0][j] == 0) {
          matrix[i][j] = 0;
        }
      }
    }

    // See if the first row needs to be set to zero as well
    if (matrix[0][0] == 0) {
      for (int j = 0; j < C; j++) {
        matrix[0][j] = 0;
      }
    }

    // See if the first column needs to be set to zero as well
    if (isCol) {
      for (int i = 0; i < R; i++) {
        matrix[i][0] = 0;
      }
    }
  }
}
```

Question 2

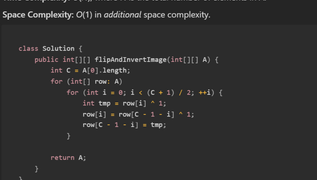
In MATLAB, there is a handy function called reshape which can reshape an m x n matrix into a new one with a different size r x c keeping its original data.

You are given an m x n matrix mat and two integers r and c representing the number of rows and the number of columns of the wanted reshaped matrix.

The reshaped matrix should be filled with all the elements of the original matrix in the same row-traversing order as they were.

If the reshape operation with given parameters is possible and legal, output the new reshaped matrix; Otherwise, output the original matrix.

Example 1:



Input: mat = [[1,2],[3,4]], r = 1, c = 4

Output:

[[1,2,3,4]]

Solution:

The simplest method is to extract all the elements of the given matrix by reading the elements in a row-wise fashion. In this implementation, we use a queue to put the extracted elements. Then, we can take out the elements of the queue formed in a serial order and arrange the elements in the resultant required matrix in a row-by-row order again.

The formation of the resultant matrix won't be possible if the number of elements in the original matrix isn't equal to the number of elements in the resultant matrix.

Complexity Analysis

- **Time complexity:** *O(m·n)*. We traverse over *m·n* elements twice. Here, *m* and *n* refer to the number of rows and columns of the given matrix respectively.
- **Space complexity:** *O(m·n)*. The queue formed will be of size *m·n*.

```
class Solution {
    public int[][] matrixReshape(int[][] nums, int r, int c) {
        int[][] res = new int[r][c];
        if (nums.length == 0 || r * c != nums.length * nums[0].length)
            return nums;
        Queue<Integer> queue = new LinkedList();
        for (int i = 0; i < nums.length; i++) {
            for (int j = 0; j < nums[0].length; j++) {
                queue.add(nums[i][j]);
            }
        }
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                res[i][j] = queue.remove();
            }
        }
        return res;
    }
}
```

Question 3

Given an n x n binary matrix image, flip the image **horizontally**, then invert it, and return *the resulting image*.

To flip an image horizontally means that each row of the image is reversed.

- For example, flipping [1,1,0] horizontally results in [0,1,1].

To invert an image means that each 0 is replaced by 1, and each 1 is replaced by 0.

- For example, inverting [0,1,1] results in [1,0,0].

Example 1:

Input: image = [[1,1,0],[1,0,1],[0,0,0]]

Output: [[1,0,0],[0,1,0],[1,1,1]]

Explanation: First reverse each row: [[0,1,1],[1,0,1],[0,0,0]].

Then, invert the image: [[1,0,0],[0,1,0],[1,1,1]]

Solution:

Intuition and Algorithm

We can do this in place. In each row, the ith value from the left is equal to the inverse of the ith value from the right.

We use (C+1) / 2 (with floor division) to iterate over all indexes i in the first half of the row, including the center.

Complexity Analysis

Time Complexity: *O(N)*, where N is the total number of elements in A.

Space Complexity: *O*(1) in *additional* space complexity.

```
class Solution {
    public int[][] flipAndInvertImage(int[][] A) {
        int C = A[0].length;
        for (int[] row: A)
            for (int i = 0; i < (C + 1) / 2; ++i) {
                int tmp = row[i] ^ 1;
                row[i] = row[C - 1 - i] ^ 1;
                row[C - 1 - i] = tmp;
            }

        return A;
    }
}
```