Description | Editorial | Solutions (5.7K) | Submissions

73. Set Matrix Zeroes

Hint

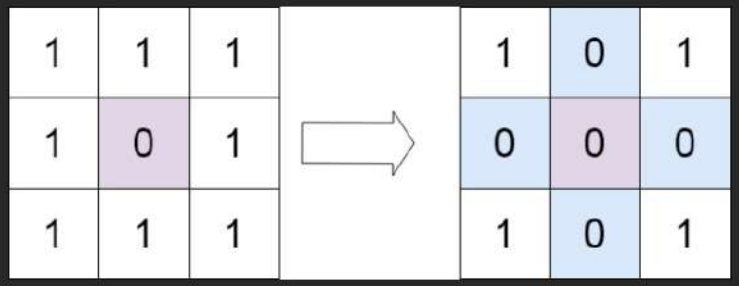
Medium 12.5K 639

Companies

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

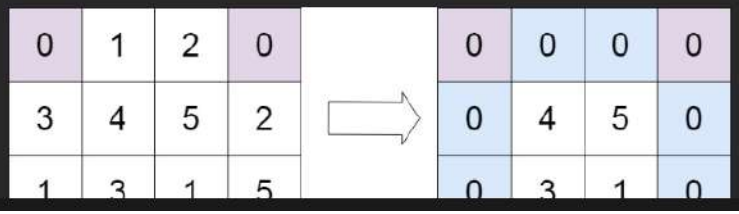
You must do it in place.

Example 1:




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

Example 2:



Java | Auto

```java
public class Solution {
public void setZeroes(int[][] matrix) {
    boolean fr = false,fc = false;
    for(int i = 0; i < matrix.length; i++) {
        for(int j = 0; j < matrix[0].length; j++) {
            if(matrix[i][j] == 0) {
                if(i == 0) fr = true;
                if(j == 0) fc = true;
                matrix[0][j] = 0;
                matrix[i][0] = 0;
            }
        }
    }
    for(int i = 1; i < matrix.length; i++) {
        for(int j = 1; j < matrix[0].length; j++) {
            if(matrix[i][0] == 0 || matrix[0][j] == 0) {
                matrix[i][j] = 0;
        }}
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

matrix =

```
[[1,1,1],[1,0,1],[1,1,1]]
```

Output

```
[[1,0,1],[0,0,0],[1,0,1]]
```

Console | Run | Submit

Description | Editorial | Solutions (3.8K) | Submissions

77. Combinations

Medium 7.1K 205

Companies

Given two integers `n` and `k`, return *all possible combinations of* `k` *numbers chosen from the range* `[1, n]`.

You may return the answer in **any order**.

Example 1:

```
Input: n = 4, k = 2
Output: [[1,2],[1,3],[1,4],[2,3],[2,4],[3,4]]
Explanation: There are 4 choose 2 = 6 total combinations.
Note that combinations are unordered, i.e., [1,2] and [2,1] are considered to be the same combination.
```

Example 2:

```
Input: n = 1, k = 1
Output: [[1]]
Explanation: There is 1 choose 1 = 1 total combination.
```

Constraints:

- `1 <= n <= 20`
- `1 <= k <= n`

Java | Auto

```java
class Solution {
    public List<List<Integer>> combine(int n, int k) {
        List<List<Integer>> result = new ArrayList<>();
        generateCombinations(1, n, k, new ArrayList<Integer>(), result);
        return result;
    }

    private void generateCombinations(int start, int n, int k, List<Integer> combination, List<List<Integer>> result) {
        if (k == 0) {
            result.add(new ArrayList<>(combination));
            return;
        }
        for (int i = start; i <= n - k + 1; i++) {
            combination.add(i);
            generateCombinations(i + 1, n, k - 1, combination, result);
            combination.remove(combination.size() - 1);
        }
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

n =
4

k =
2

Console | Run | Submit