[ARRAYS]

Algorithm Fundamentals - Hosted by WiCSE

Agenda

10 min - Review the **warm-up problem**

30 min – Independent work on **main problem**

- Optional: Work on the bonus problem if you finish early!

10 min – Review the **main problem** solution

10 min – Wrap-up

Goals

- Create a safe space to practice and build confidence around algorithm fundamentals and data structures
- Actively engage in the discussion of potential solutions to the problems
- Foster a community of study buddies. Join our Teams Channel!
- Have Fun!

What’s an array?

- A data structure that holds data items or elements
- Can be declared using a single identifier and processed efficiently using iteration techniques
- Each position is identified by an index (starting at 0 in most languages)
- The values stored are mutable, they can be changed while the program is running

- isaaccomputerscience.org

You are given an array `prices` where `prices[i]` is the price of a given stock on the ith day.

You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return `0`.

WARM UP: *BEST TIME TO BUY AND SELL STOCK*

Example 1:

```
Input: prices = [7,1,5,3,6,4]
Output: 5
Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.
Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.
```

Example 2:

```
Input: prices = [7,6,4,3,1]
Output: 0
Explanation: In this case, no transactions are done and the max profit = 0.
```

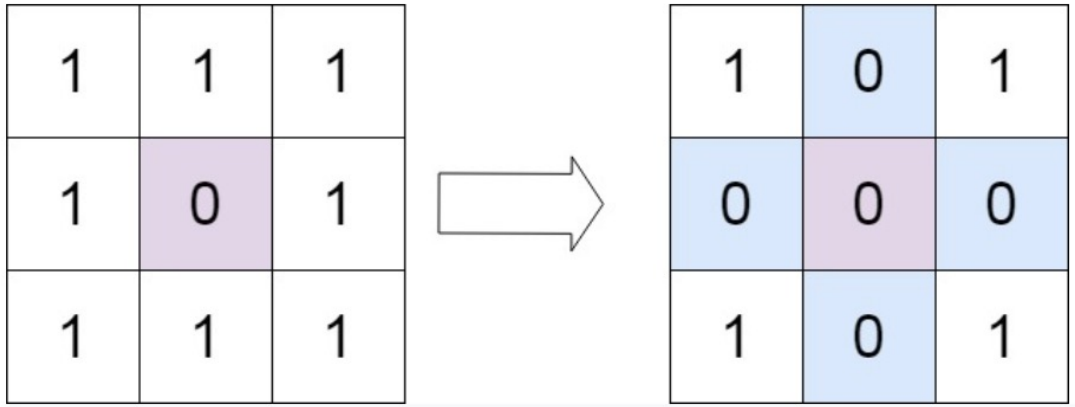
Constraints:

- `1 <= prices.length <=` 10^5
- `0 <= prices[i] <=` 10^4

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

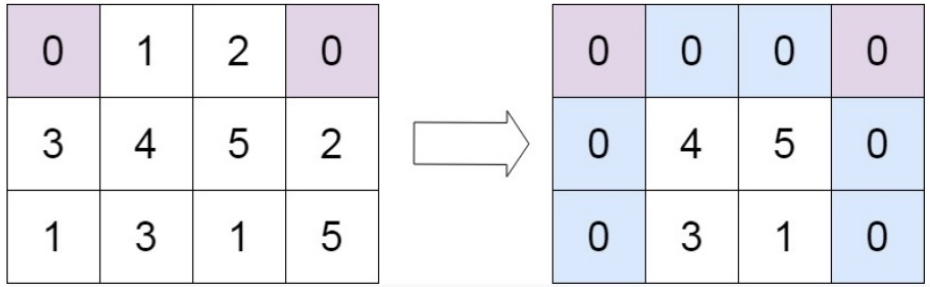
You must do it in place.

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:




```
Input: matrix = [[0,1,2,0],[3,4,5,2],[1,3,1,5]]
Output: [[0,0,0,0],[0,4,5,0],[0,3,1,0]]
```

Constraints:

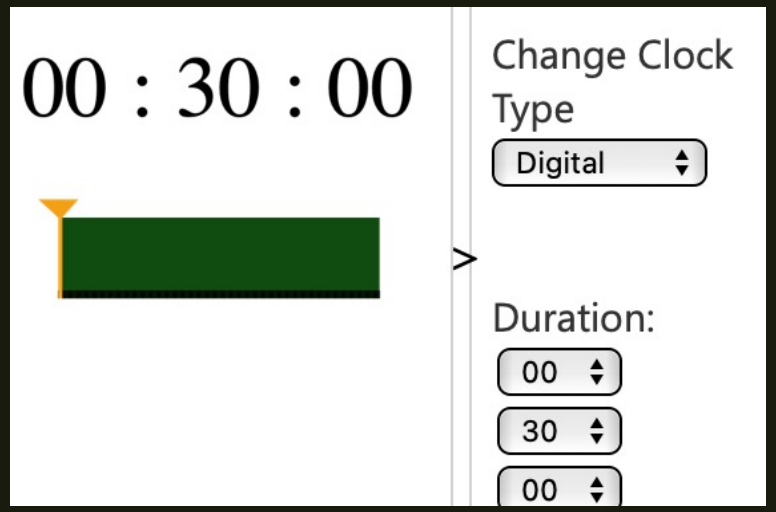
- `m == matrix.length`
- `n == matrix[0].length`
- `1 <= m, n <= 200`
- -2^{31} `<= matrix[i][j] <=` $2^{31} - 1$

Follow up:

- A straightforward solution using `O(mn)` space is probably a bad idea.
- A simple improvement uses `O(m + n)` space, but still not the best solution.
- Could you devise a constant space solution?

MAIN PROBLEM: SET MATRIX ZEROES

Independent Hack Time!

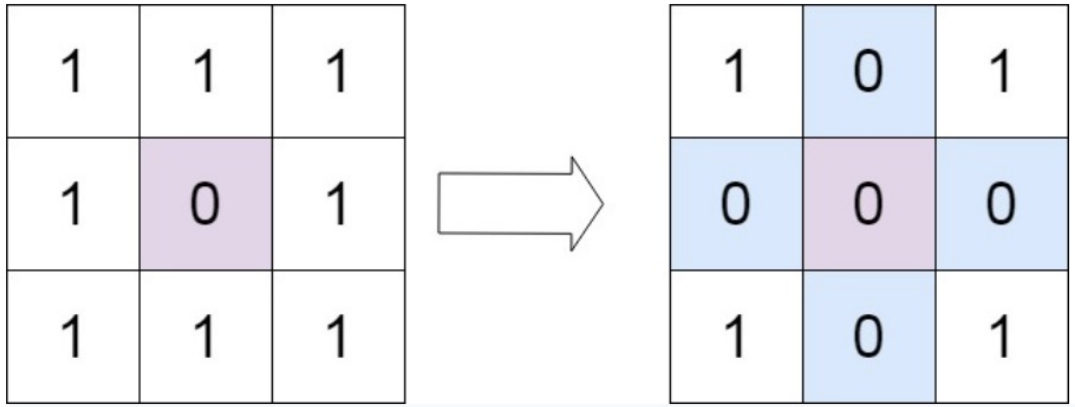


- **Warm Up:** Best Time to Buy And Sell Stock
  - https://leetcode.com/problems/best-time-to-buy-and-sell-stock/
- **Main:** Set Matrix Zeroes
  - https://leetcode.com/problems/set-matrix-zeroes/
- **Optional Bonus:** Search in Rotated Sorted Array
  - https://leetcode.com/problems/search-in-rotated-sorted-array/
- **Next Month's Warm Up (Linked Lists):** Palindrome Linked List
  - https://leetcode.com/problems/palindrome-linked-list/

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

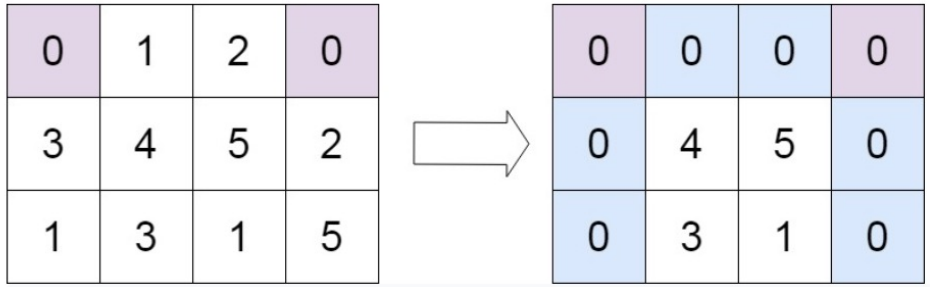
You must do it in place.

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:




```
Input: matrix = [[0,1,2,0],[3,4,5,2],[1,3,1,5]]
Output: [[0,0,0,0],[0,4,5,0],[0,3,1,0]]
```

Constraints:

- `m == matrix.length`
- `n == matrix[0].length`
- `1 <= m, n <= 200`
- -2^{31} `<= matrix[i][j] <=` $2^{31} - 1$

Follow up:

- A straightforward solution using `O(mn)` space is probably a bad idea.
- A simple improvement uses `O(m + n)` space, but still not the best solution.
- Could you devise a constant space solution?

MAIN PROBLEM: SET MATRIX ZEROES

There is an integer array `nums` sorted in ascending order (with **distinct** values).

Prior to being passed to your function, `nums` is **possibly rotated** at an unknown pivot index `k` (`1 <= k < nums.length`) such that the resulting array is `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]` (**0-indexed**). For example, `[0,1,2,4,5,6,7]` might be rotated at pivot index `3` and become `[4,5,6,7,0,1,2]`.

Given the array `nums` **after** the possible rotation and an integer `target`, return *the index of* `target` *if it is in* `nums`*, or* `-1` *if it is not in* `nums`.

You must write an algorithm with `O(log n)` runtime complexity.

Example 1:

```
Input: nums = [4,5,6,7,0,1,2], target = 0
Output: 4
```

Example 2:

```
Input: nums = [4,5,6,7,0,1,2], target = 3
Output: -1
```

Example 3:

```
Input: nums = [1], target = 0
Output: -1
```

Constraints:

- `1 <= nums.length <= 5000`
- $-10^4 <= nums[i] <= 10^4$
- All values of `nums` are **unique**.
- `nums` is an ascending array that is possibly rotated.
- $-10^4 <= target <= 10^4$

OPTIONAL BONUS PROBLEM:
SEARCH IN ROTATED SORTED ARRAY

THANK YOU!
SEE YOU
NEXT MONTH
FOR
LINKED LISTS!