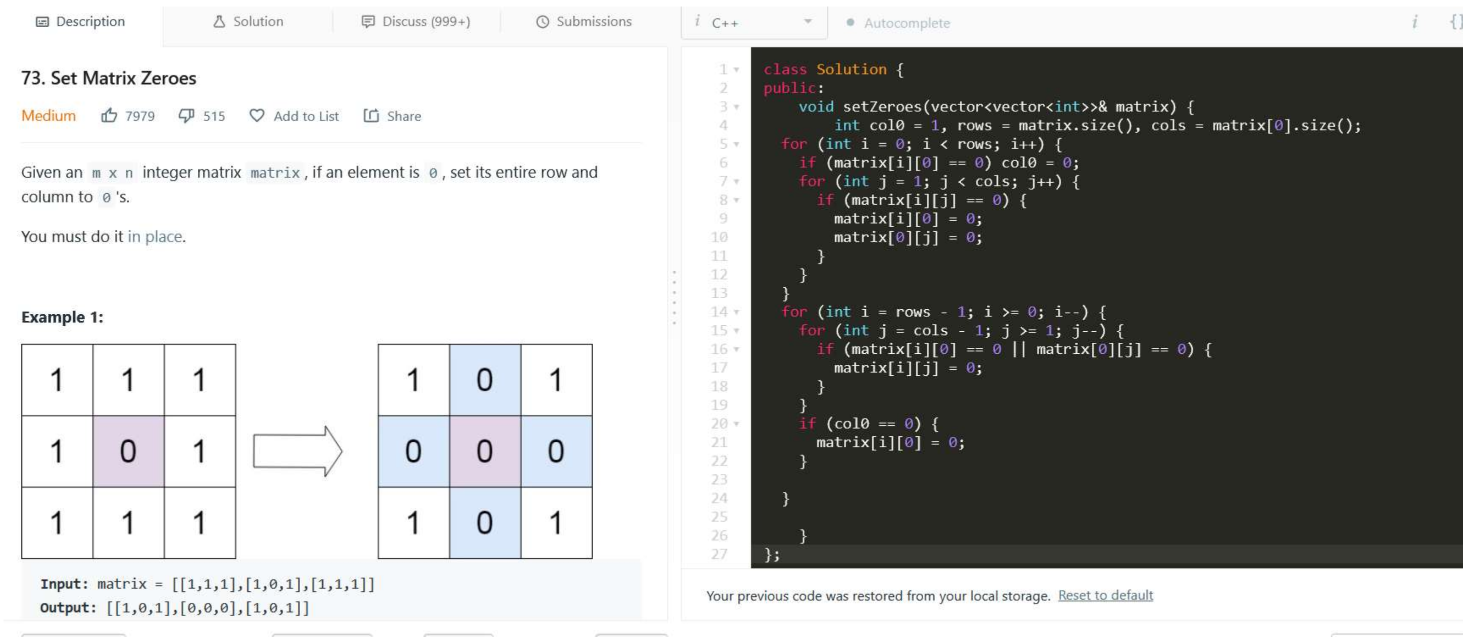
STRIVER SHEET QUESTIONS : WEEK-1

AMAN PRAJAPATI



73. Set Matrix Zeroes

Medium 7979 515 Add to List Share

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:

1	1	1
1	0	1
1	1	1

1	0	1
0	0	0
1	0	1

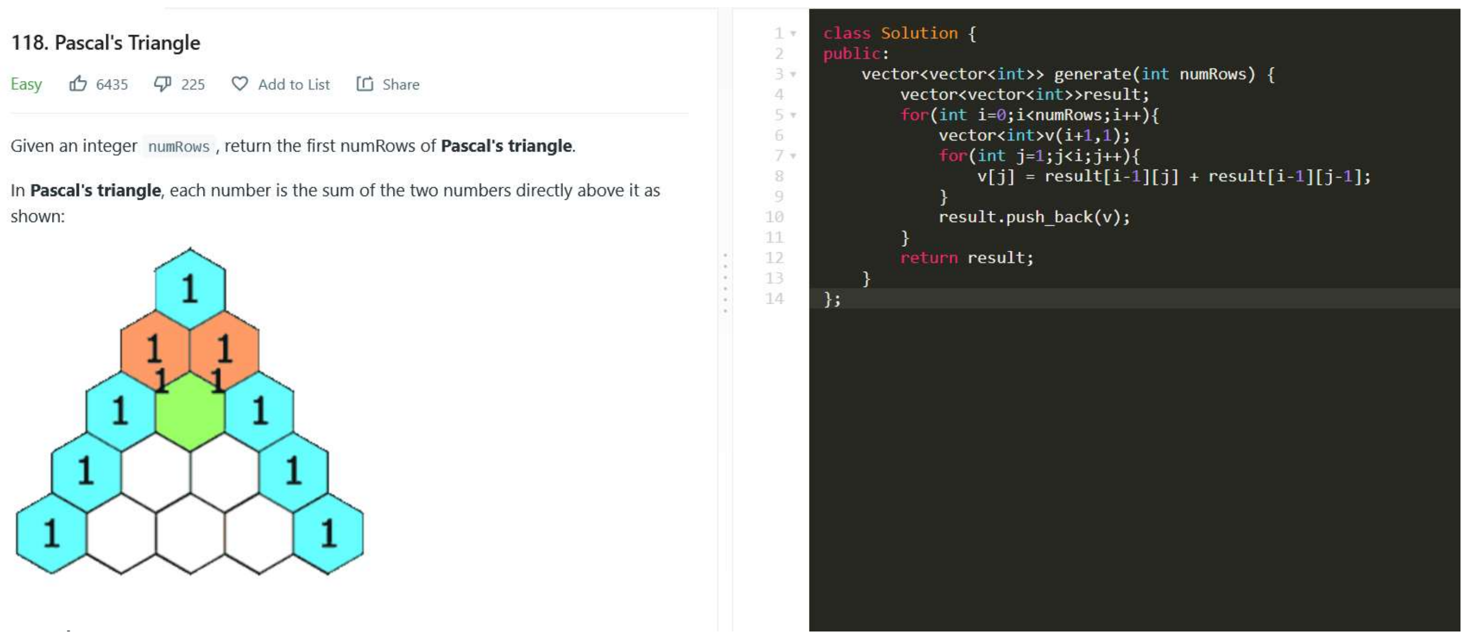
```
Input: matrix = [[1,1,1],[1,0,1],[1,1,1]]
Output: [[1,0,1],[0,0,0],[1,0,1]]
```

```
class Solution {
public:
    void setZeroes(vector<vector<int>>& matrix) {
        int col0 = 1, rows = matrix.size(), cols = matrix[0].size();
  for (int i = 0; i < rows; i++) {
    if (matrix[i][0] == 0) col0 = 0;
    for (int j = 1; j < cols; j++) {
      if (matrix[i][j] == 0) {
        matrix[i][0] = 0;
        matrix[0][j] = 0;
      }
    }
  }
  for (int i = rows - 1; i >= 0; i--) {
    for (int j = cols - 1; j >= 1; j--) {
      if (matrix[i][0] == 0 || matrix[0][j] == 0) {
        matrix[i][j] = 0;
      }
    }
    if (col0 == 0) {
      matrix[i][0] = 0;
    }

  }

    }
};
```

Your previous code was restored from your local storage. Reset to default



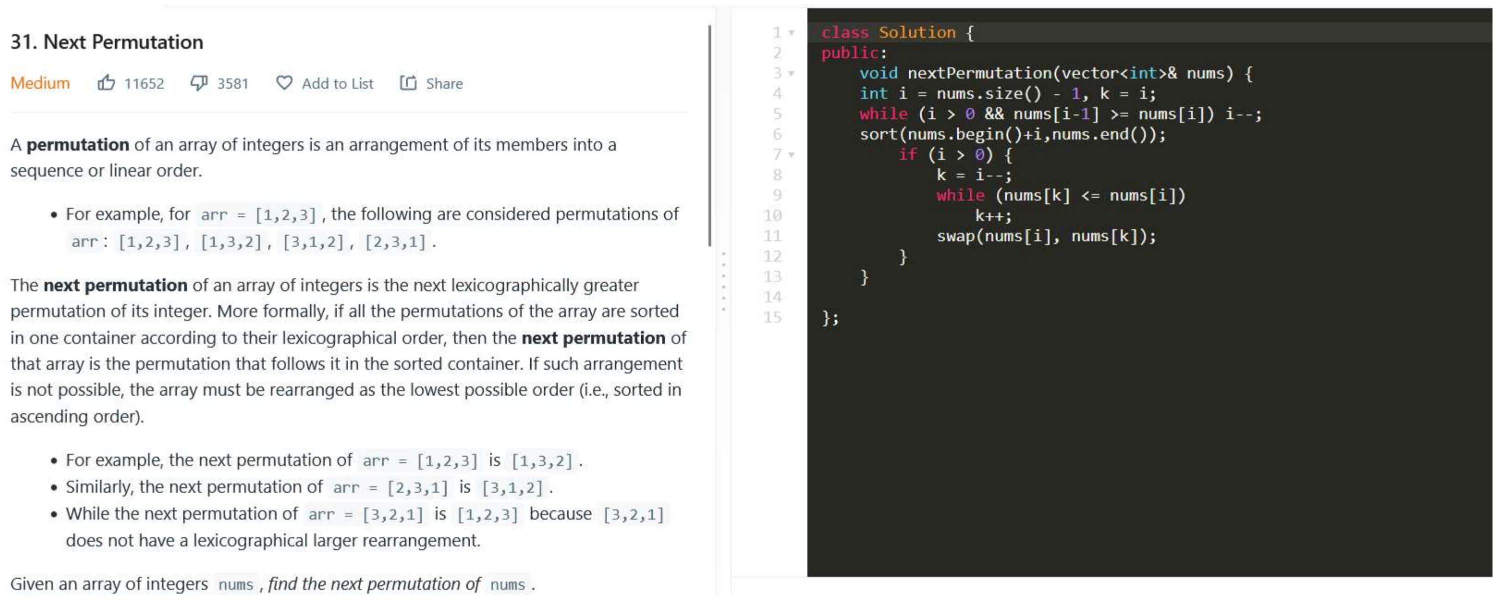
118. Pascal's Triangle

Easy 6435 225 Add to List Share

Given an integer `numRows`, return the first numRows of **Pascal's triangle**.

In **Pascal's triangle**, each number is the sum of the two numbers directly above it as shown:

```
class Solution {
public:
    vector<vector<int>> generate(int numRows) {
        vector<vector<int>>result;
        for(int i=0;i<numRows;i++){
            vector<int>v(i+1,1);
            for(int j=1;j<i;j++){
                v[j] = result[i-1][j] + result[i-1][j-1];
            }
            result.push_back(v);
        }
        return result;
    }
};
```



31. Next Permutation

Medium 11652 3581 Add to List Share

A **permutation** of an array of integers is an arrangement of its members into a sequence or linear order.

- For example, for `arr = [1,2,3]`, the following are considered permutations of `arr`: `[1,2,3]`, `[1,3,2]`, `[3,1,2]`, `[2,3,1]`.

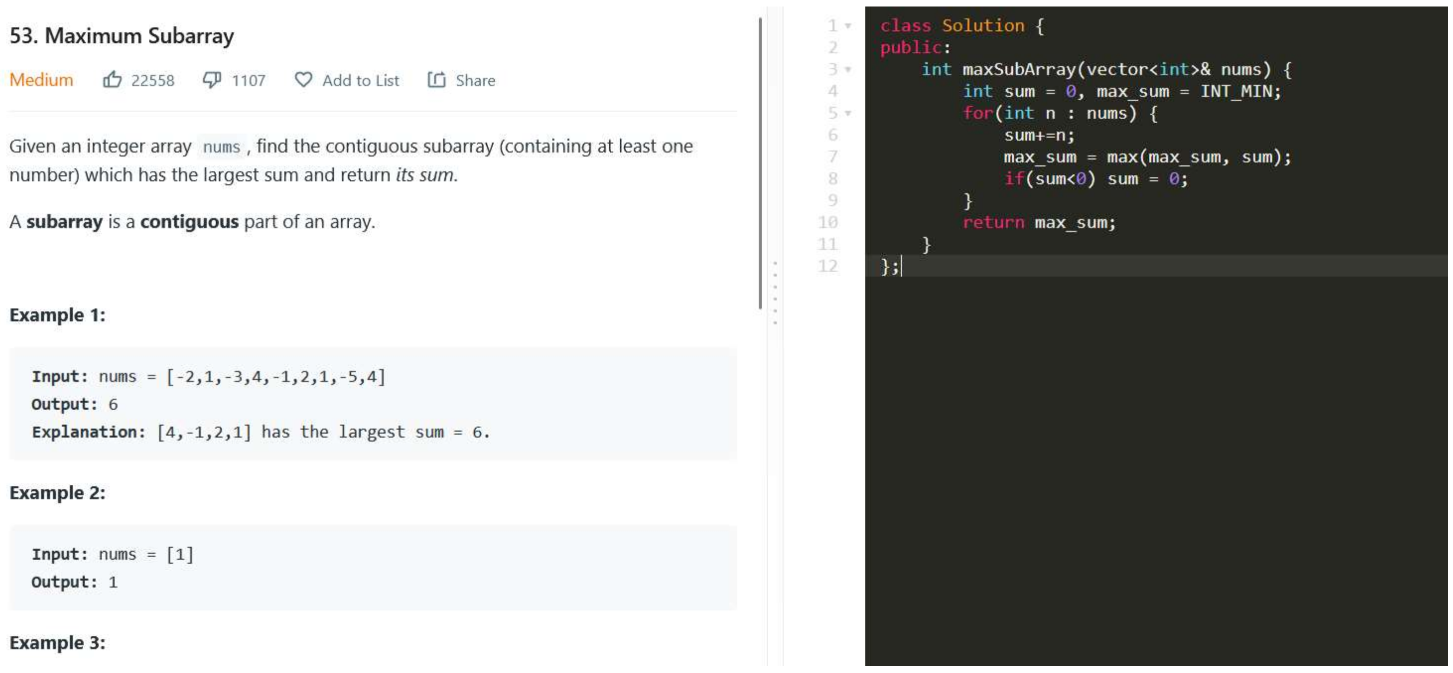
The **next permutation** of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the **next permutation** of that array is the permutation that follows it in the sorted container. If such arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

- For example, the next permutation of `arr = [1,2,3]` is `[1,3,2]`.
- Similarly, the next permutation of `arr = [2,3,1]` is `[3,1,2]`.
- While the next permutation of `arr = [3,2,1]` is `[1,2,3]` because `[3,2,1]` does not have a lexicographical larger rearrangement.

Given an array of integers `nums`, *find the next permutation of* `nums`.

```cpp
class Solution {
public:
    void nextPermutation(vector<int>& nums) {
    int i = nums.size() - 1, k = i;
    while (i > 0 && nums[i-1] >= nums[i]) i--;
    sort(nums.begin()+i,nums.end());
        if (i > 0) {
            k = i--;
            while (nums[k] <= nums[i])
                k++;
            swap(nums[i], nums[k]);
        }
    }

};
```



53. Maximum Subarray

Medium 22558 1107 Add to List Share

Given an integer array `nums`, find the contiguous subarray (containing at least one number) which has the largest sum and return *its sum*.

A **subarray** is a **contiguous** part of an array.

Example 1:

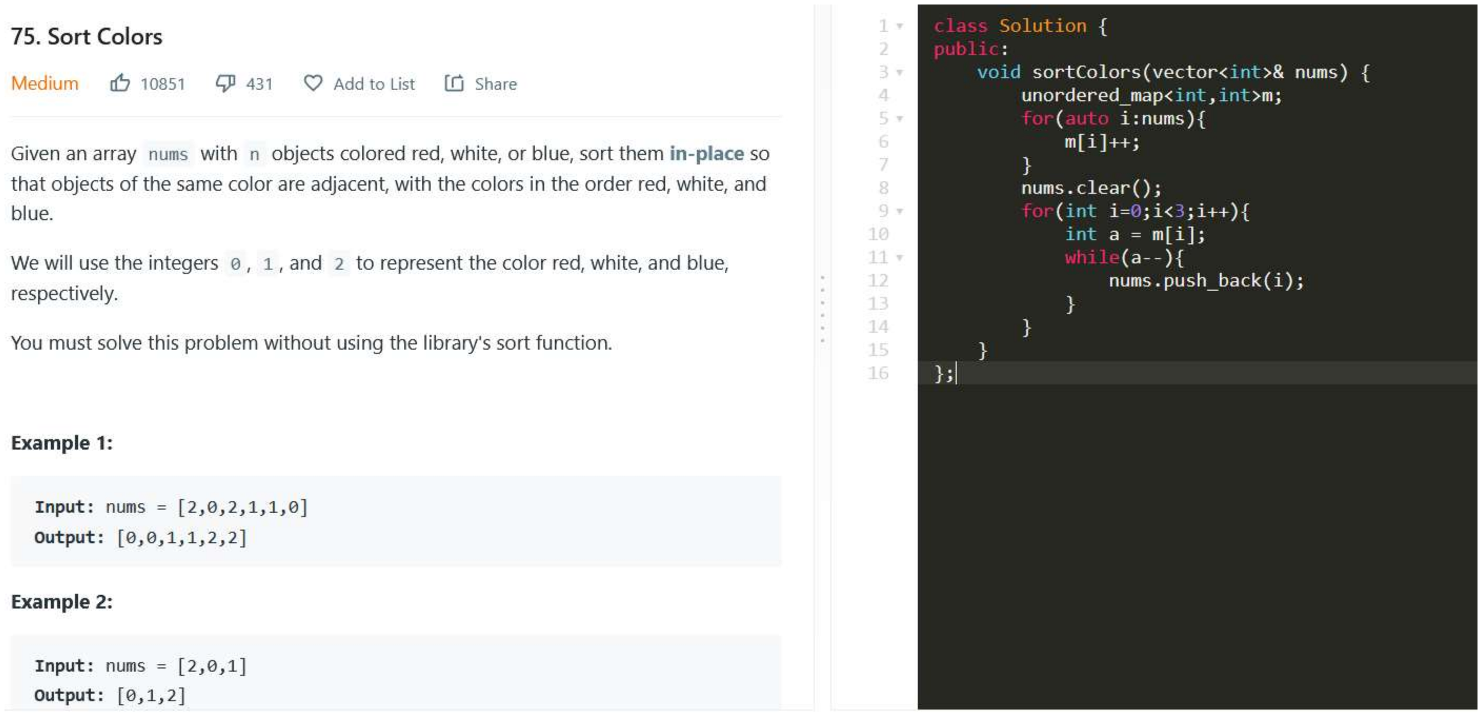
```
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6.
```

Example 2:

```
Input: nums = [1]
Output: 1
```

Example 3:

```cpp
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int sum = 0, max_sum = INT_MIN;
        for(int n : nums) {
            sum+=n;
            max_sum = max(max_sum, sum);
            if(sum<0) sum = 0;
        }
        return max_sum;
    }
};
```



75. Sort Colors

Medium 10851 431 Add to List Share

Given an array `nums` with `n` objects colored red, white, or blue, sort them **in-place** so that objects of the same color are adjacent, with the colors in the order red, white, and blue.

We will use the integers `0`, `1`, and `2` to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

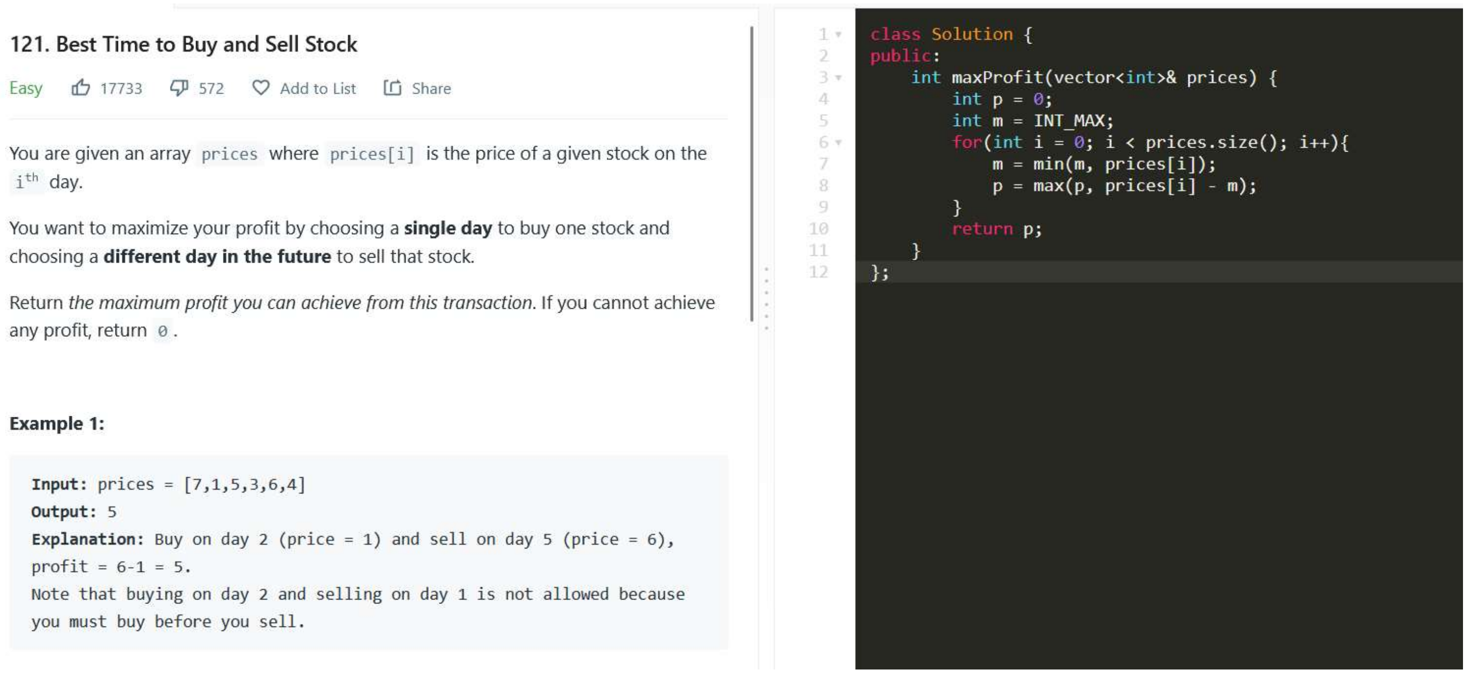
Example 1:

```
Input: nums = [2,0,2,1,1,0]
Output: [0,0,1,1,2,2]
```

Example 2:

```
Input: nums = [2,0,1]
Output: [0,1,2]
```

```cpp
class Solution {
public:
    void sortColors(vector<int>& nums) {
        unordered_map<int,int>m;
        for(auto i:nums){
            m[i]++;
        }
        nums.clear();
        for(int i=0;i<3;i++){
            int a = m[i];
            while(a--){
                nums.push_back(i);
            }
        }
    }
};
```



121. Best Time to Buy and Sell Stock

Easy 17733 572 Add to List Share

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

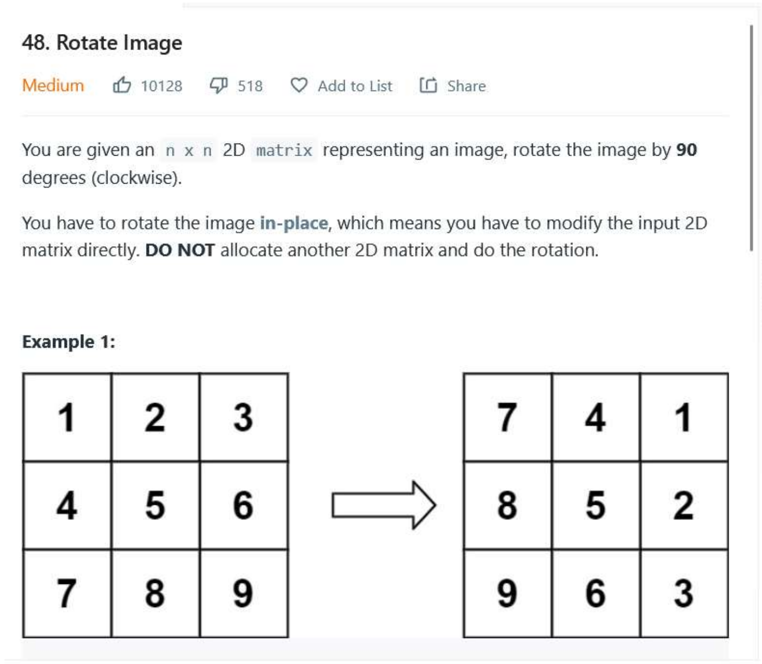
You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

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

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.
```

```cpp
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int p = 0;
        int m = INT_MAX;
        for(int i = 0; i < prices.size(); i++){
            m = min(m, prices[i]);
            p = max(p, prices[i] - m);
        }
        return p;
    }
};
```



48. Rotate Image

Medium 10128 518 Add to List Share

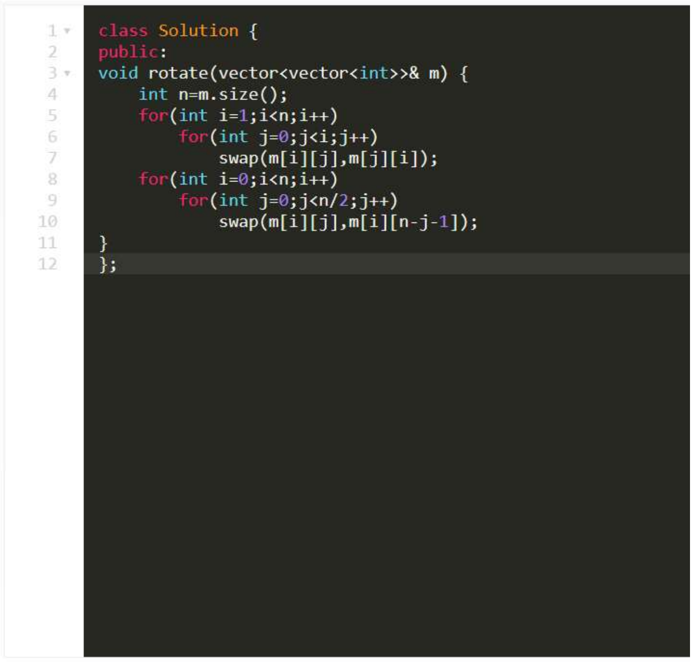
You are given an `n x n` 2D `matrix` representing an image, rotate the image by **90** degrees (clockwise).

You have to rotate the image **in-place**, which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

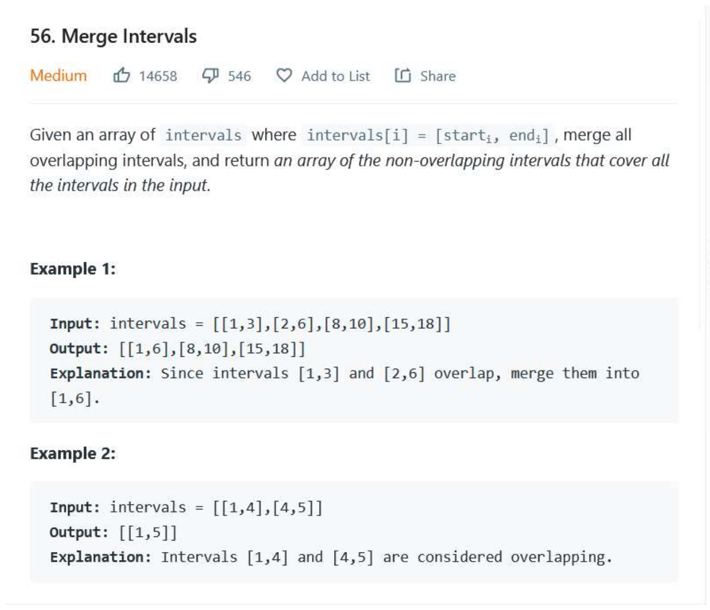
Example 1:

1	2	3
4	5	6
7	8	9

7	4	1
8	5	2
9	6	3



```cpp
class Solution {
public:
void rotate(vector<vector<int>>& m) {
    int n=m.size();
    for(int i=1;i<n;i++)
        for(int j=0;j<i;j++)
            swap(m[i][j],m[j][i]);
    for(int i=0;i<n;i++)
        for(int j=0;j<n/2;j++)
            swap(m[i][j],m[i][n-j-1]);
}
};
```



56. Merge Intervals

Medium 14658 546 Add to List Share

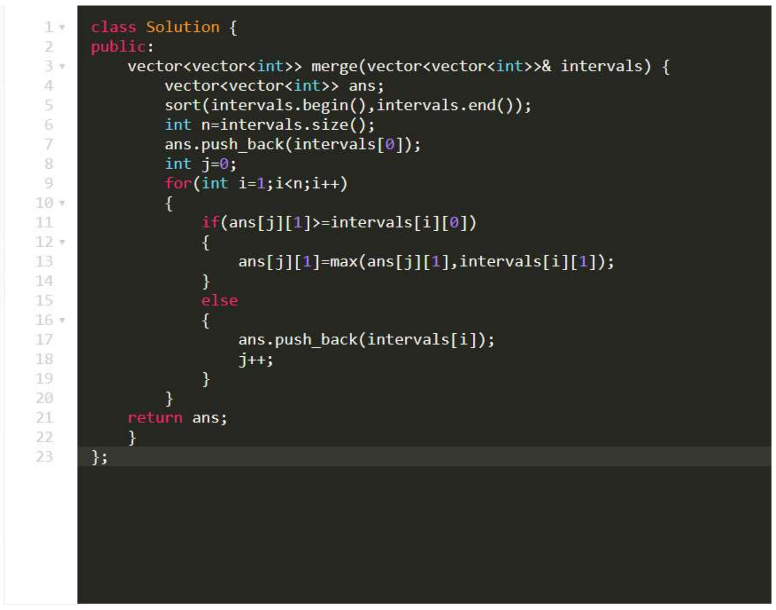
Given an array of `intervals` where `intervals[i] = [start`$_i$`, end`$_i$`]`, merge all overlapping intervals, and return *an array of the non-overlapping intervals that cover all the intervals in the input.*

Example 1:

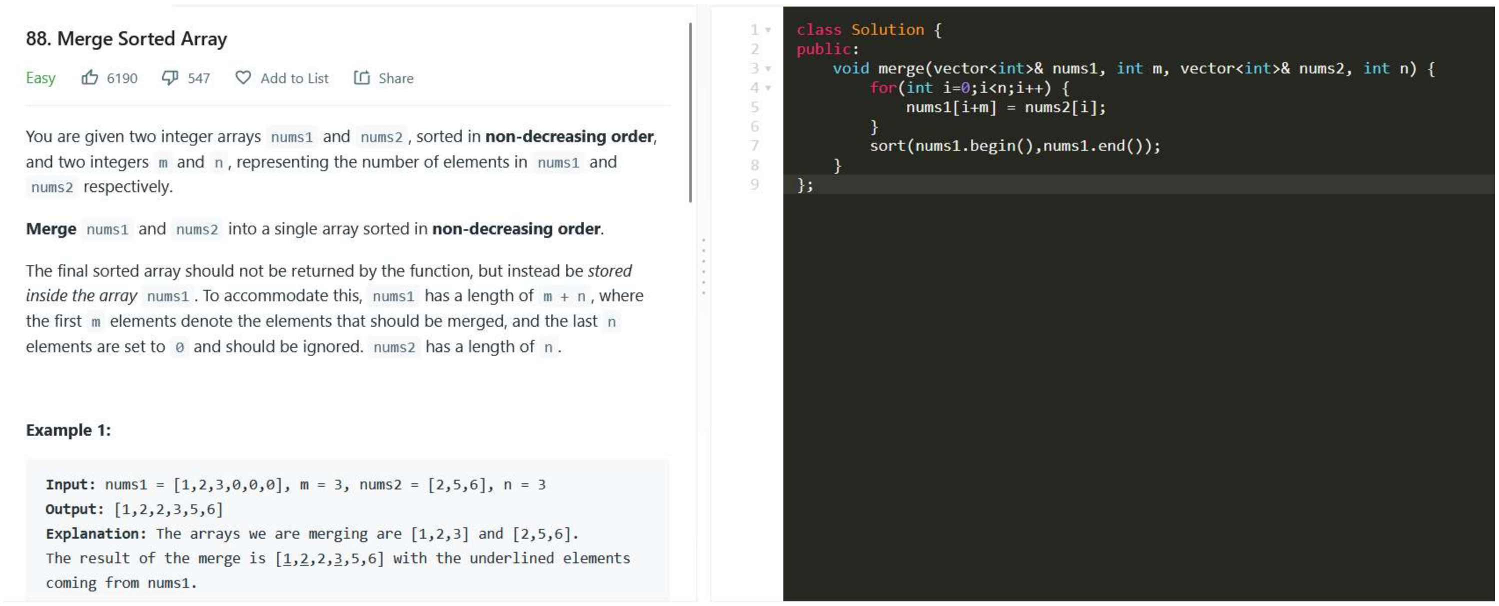
```
Input: intervals = [[1,3],[2,6],[8,10],[15,18]]
Output: [[1,6],[8,10],[15,18]]
Explanation: Since intervals [1,3] and [2,6] overlap, merge them into
[1,6].
```

Example 2:

```
Input: intervals = [[1,4],[4,5]]
Output: [[1,5]]
Explanation: Intervals [1,4] and [4,5] are considered overlapping.
```



```cpp
class Solution {
public:
    vector<vector<int>> merge(vector<vector<int>>& intervals) {
        vector<vector<int>> ans;
        sort(intervals.begin(),intervals.end());
        int n=intervals.size();
        ans.push_back(intervals[0]);
        int j=0;
        for(int i=1;i<n;i++)
        {
            if(ans[j][1]>=intervals[i][0])
            {
                ans[j][1]=max(ans[j][1],intervals[i][1]);
            }
            else
            {
                ans.push_back(intervals[i]);
                j++;
            }
        }
    return ans;
    }
};
```



88. Merge Sorted Array

You are given two integer arrays `nums1` and `nums2`, sorted in **non-decreasing order**, and two integers `m` and `n`, representing the number of elements in `nums1` and `nums2` respectively.

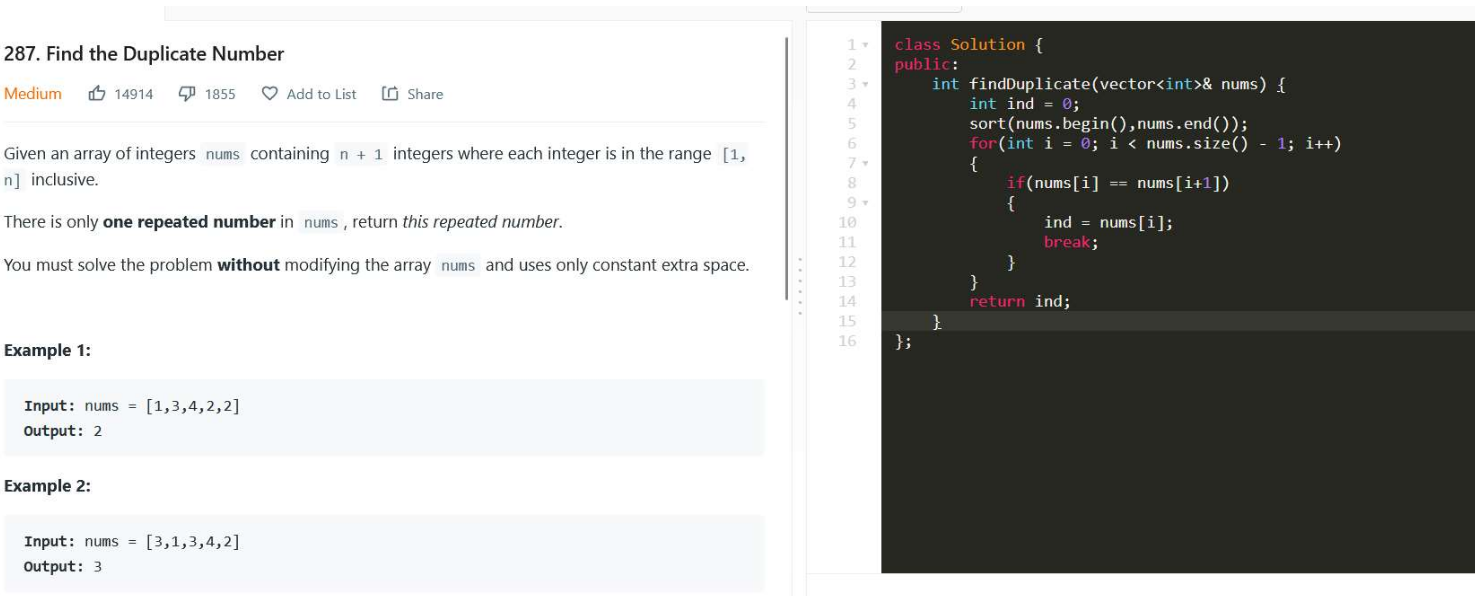
Merge `nums1` and `nums2` into a single array sorted in **non-decreasing order**.

The final sorted array should not be returned by the function, but instead be *stored inside the array* `nums1`. To accommodate this, `nums1` has a length of `m + n`, where the first `m` elements denote the elements that should be merged, and the last `n` elements are set to `0` and should be ignored. `nums2` has a length of `n`.

Example 1:

```
Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3
Output: [1,2,2,3,5,6]
Explanation: The arrays we are merging are [1,2,3] and [2,5,6].
The result of the merge is [1,2,2,3,5,6] with the underlined elements
coming from nums1.
```

```cpp
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        for(int i=0;i<n;i++) {
            nums1[i+m] = nums2[i];
        }
        sort(nums1.begin(),nums1.end());
    }
};
```



287. Find the Duplicate Number

Given an array of integers `nums` containing `n + 1` integers where each integer is in the range `[1, n]` inclusive.

There is only **one repeated number** in `nums`, return *this repeated number*.

You must solve the problem **without** modifying the array `nums` and uses only constant extra space.

Example 1:

```
Input: nums = [1,3,4,2,2]
Output: 2
```

Example 2:

```
Input: nums = [3,1,3,4,2]
Output: 3
```

```cpp
class Solution {
public:
    int findDuplicate(vector<int>& nums) {
        int ind = 0;
        sort(nums.begin(),nums.end());
        for(int i = 0; i < nums.size() - 1; i++)
        {
            if(nums[i] == nums[i+1])
            {
                ind = nums[i];
                break;
            }
        }
        return ind;
    }
};
```