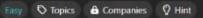


# 26. Remove Duplicates from Sorted Array



Given an integer array nums sorted in **non-decreasing order**, remove the duplicates **in-place** such that each unique element appears only **once**. The **relative order** of the elements should be kept the **same**. Then return *the number of unique elements in nums*.

Solved @

Consider the number of unique elements of nums to be k, to get accepted, you need to do the following things:

- Change the array nums such that the first k elements of nums contain the unique elements in the order they
  were present in nums initially. The remaining elements of nums are not important as well as the size of nums.
- Return k.

### **Custom Judge:**

The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int[] expectedNums = [...]; // The expected answer with correct length
```

# 80. Remove Duplicates from Sorted Array II





Given an integer array nums sorted in non-decreasing order, remove some duplicates in-place such that each unique element appears at most twice. The relative order of the elements should be kept the same.

Since it is impossible to change the length of the array in some languages, you must instead have the result be placed in the **first part** of the array nums. More formally, if there are k elements after removing the duplicates, then the first k elements of nums should hold the final result. It does not matter what you leave beyond the first k elements.

Return k after placing the final result in the first k slots of nums.

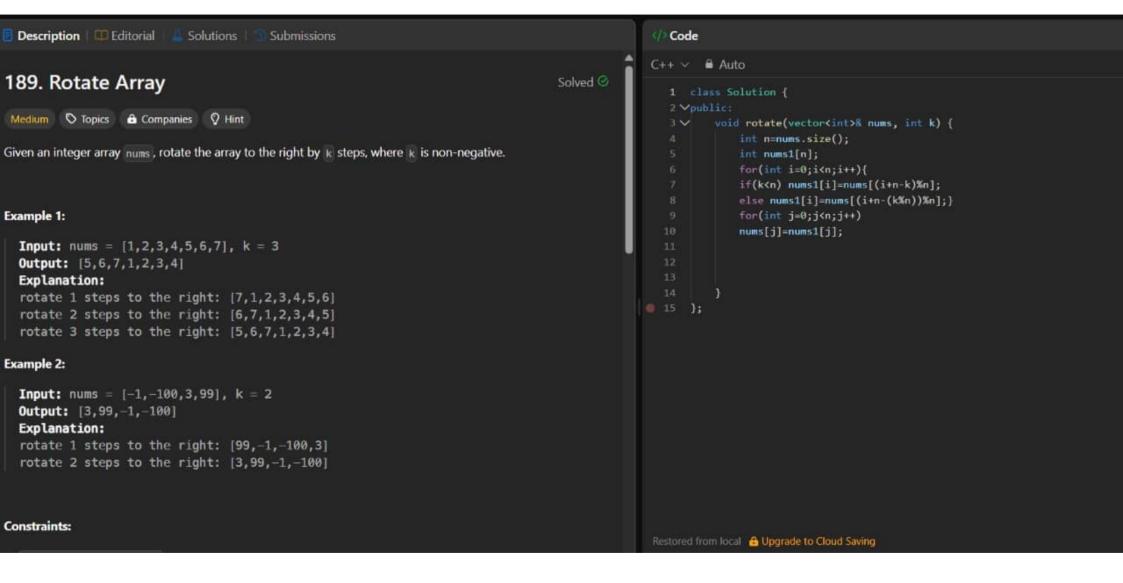
Do **not** allocate extra space for another array. You must do this by **modifying the input array in-place** with O(1) extra memory.

#### Custom Judge:

The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int[] expectedNums = [...]; // The expected answer with correct length
int k = removeDuplicates(nums); // Calls your implementation
assert k == expectedNums.length;
for (int i = 0; i < k; i++) {
    assert nums[i] == expectedNums[i];</pre>
```

```
int removeDuplicates(vector<int>% nums) {
    int n=nums.size():
   if(n==0) return 0;
   else if(n==1){return 1;}
   else if(n==2){return 2;}
       int k=0,i;
       for(i=0;i<n;i++){
            int j,c=0;
           for(j=i;j<n;j++){
               if(nums[j]!=nums[i]) break;
           if(c>=3){
                nums[k]=nums[i];
               nums[k+1]=nums[i];
               i=i+c-1;
               k=k+2;
            } else if(c==2){
               nums[k]=nums[i];
               nums[k+1]=nums[i];
               i=i+1:
               k=k+2;
            ] else[
                nums[k]=nums[i];
               k++;}
```



# 121. Best Time to Buy and Sell Stock

Easy 🛇 Topics 🔒 Companies

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

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

Solved @

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

### xample 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.
```

## xample 2:

**Input:** prices = [7,6,4,3,1]

```
Output: 0 Explanation: In this case, no transactions are done and the max profit = 0.
```

```
int maxProfit(vector(int>% prices) {
     int n=prices.size();
     if(n==1) return 0;
     else{
        int 1=0,r=1,max1=0;
        while(r<n){
            if(prices[r]>prices[l]){
                max1=max(prices[r]-prices[1],max1);
            } else{l=r;}
            r++;
        } return max1;
```

# 55. Jump Game

Medium ♥ Topics ♠ Companies

You are given an integer array nums. You are initially positioned at the array's **first index**, and each element in the array represents your maximum jump length at that position.

Return true if you can reach the last index, or false otherwise.

### Example 1:

**Input:** nums = [2,3,1,1,4]

Output: true

Explanation: Jump 1 step from index 0 to 1, then 3 steps to the last index.

## Example 2:

**Input:** nums = [3,2,1,0,4]

Output: false

Explanation: You will always arrive at index 3 no matter what. Its maximum jump

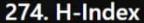
length is 0, which makes it impossible to reach the last index.

### Constraints:

Solved @

Solved @ 45. Jump Game II Medium O Topics Companies You are given a **0-indexed** array of integers nums of length n. You are initially positioned at nums [0]. Each element nums [i] represents the maximum length of a forward jump from index i. In other words, if you are at nums [i], you can jump to any nums [i + j] where: 0 <= j <= nums[i] and</li> • i + j < nReturn the minimum number of jumps to reach nums[n-1]. The test cases are generated such that you can reach nums [n - 1]. Example 1: **Input:** nums = [2,3,1,1,4]Output: 2 Explanation: The minimum number of jumps to reach the last index is 2. Jump 1 step from index 0 to 1, then 3 steps to the last index. Example 2: **Input:** nums = [2,3,0,1,4]Output: 2

```
class Solution {
      public:
          int jump(vector(int>& nums) [
              int n = nums.size();
              if (n <= 1) return 0;
              int jumps = 0, currentEnd = 0, farthest = 0;
              for (int i = 0; i < n - 1; ++i) {
                  farthest = max(farthest, i + nums[i]);
                  if (i == currentEnd) {
                      jumps++;
                      currentEnd = farthest;
                      if (currentEnd >= n - 1) {
                          break;
              return jumps;
Saved
```





Given an array of integers citations where citations [i] is the number of citations a researcher received for their ith paper, return the researcher's h-index.

According to the <u>definition of h-index on Wikipedia</u>: The h-index is defined as the maximum value of h such that the given researcher has published at least h papers that have each been cited at least h times.

## Example 1:

**Input:** citations = [3,0,6,1,5]

Output: 3

**Explanation:** [3,0,6,1,5] means the researcher has 5 papers in total and each of them had received 3, 0, 6, 1, 5 citations respectively.

Since the researcher has 3 papers with at least 3 citations each and the remaining two with no more than 3 citations each, their h-index is 3.

### Example 2:

Input: citations = [1,3,1]

Output: 1

```
Auto
Python V
  1 class Solution(object):
          def hIndex(self, citations):
              n=len(citations)
              c=0
              citations=sorted(citations,reverse=True)
              for i in range(1,n+1,1):
                  if(citations[i-1]<i):
                      c=1
                      break
              if(c==1):
                  return i-1
              else:
                  return n
```

Solved @