Question: https://leetcode.com/problems/remove-duplicates-from-sorted-array/

Constraints:do this by ****modifying the input array**[in-place](https://en.wikipedia.org/wiki/In-place_algorithm" \t "https://leetcode.com/problems/remove-duplicates-from-sorted-array/_blank)** with O(1) extra memory.

So we need to modify the array in such a way that we calculate the number of unique(let the number be u) elements in nums array.

And the elements in nums array from 0 to u-1 will be the unique elements.

If we are not faced with the challenge of not using extra space then the problem is very simple.

Let’s look into the simple solution first.

So in that case we maintain a res array which will contain the unique elements.

Set res pointer as j=0

Initially res[0]=nums[0]

Now we will iterate over the nums from 1.

We will check if res[j]!=nums[i]

Increase j by 1

Then set res[j]=nums[i]

So now we can see the whole operation is taking place based on 2 pointers. So we can use two pointers approach to solve the problem with O(1) space.

We set 2 pointers i and j.

So we do the quite same as the previous approach the only difference is we update the nums array itself instead of creating a new array, cause once we have iterated ith position we will not comeback to that position.

Code:  
class Solution {

public int removeDuplicates(int[] nums) {

if(nums.length==0){

return 0;

}

int j=0;

for(int i=1;i<nums.length;i++){

if(nums[j]!=nums[i]){

nums[++j]=nums[i];

}

}

return j+1;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>