```java
/*Author: Bochen (mddboc@foxmail.com)
Last Modified: Tue Apr 10 22:28:44 CST 2018*/

/*Given a sorted array, remove the duplicates in-place such that each element appear only once and return the new length.

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

        Example:

        Given nums = [1,1,2],

        Your function should return length = 2, with the first two elements of nums being 1 and 2 respectively.
        It doesn't matter what you leave beyond the new length.*/


import java.util.*;
import java.lang.Math;
import java.lang.System;
import java.lang.Integer;


public class Main {

    public static void main(String[] args) throws ArithmeticException {

        int[] input = {7, 1, 5, 3, 6, 4};

        Solution solution = new Solution();

        int result = solution.maxProfit(input);

        System.out.println("haha");
    }

}


class ListNode {
    int val;
    ListNode next;

    ListNode(int x) {
        val = x;
    }
}


class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int x) {
        val = x;
    }
}


class Solution {
    public int removeDuplicates(int[] nums) {

        if (nums == null || nums.length < 2) {
            return nums.length;
        }

        int numsLength = nums.length;
        int slowPointer = 1, fastPointer = 1;

        while (fastPointer < numsLength) {
            if (nums[fastPointer] != nums[fastPointer - 1]) {
```

```
                nums[slowPointer] = nums[fastPointer];
                slowPointer++;
            }

            fastPointer++;
        }

        return slowPointer;
    }
}
```