remove_element

May 7, 2025

1 Remove Element

1.1 Problem Definition

Given an integer array nums and an integer val, remove all occurrences of val in nums in-place. The order of the elements may be changed. Then return the number of elements in nums which are not equal to val.

Consider the number of elements in nums which are not equal to val 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 elements which are not equal to val. The remaining elements of nums are not important as well as the size of nums.
- Return k.

Judge

```
[1]: def judge_remove_element(remove_element, nums, val, expected_nums):
         nums_copy = nums.copy()

         k = remove_element(nums_copy, val)

         assert k == len(expected_nums), f"Expected length {len(expected_nums)}, got␣
      ↪{k}"

         nums_copy[:k] = sorted(nums_copy[:k])

         for i in range(k):
             assert nums_copy[i] == expected_nums[i], f"Mismatch at index {i}:␣
      ↪expected {expected_nums[i]}, got {nums_copy[i]}"

         print("All tests passed!")
```

1.1.1 Example Cases

Example 1:

Input: nums = [3,2,2,3], val = 3
Output: 2, nums = [2,2,'‘,”]

Explanation: Your function should return k = 2, with the first two elements of nums being 2.
It does not matter what you leave beyond the returned k (hence they are underscores).

Example 2:

Input: nums = [0,1,2,2,3,0,4,2], val = 2
Output: 5, nums = [0,1,4,0,3,'‘,’‘,”]
Explanation: Your function should return k = 5, with the first five elements of nums containing 0, 0, 1, 3, and 4.
Note that the five elements can be returned in any order.
It does not matter what you leave beyond the returned k (hence they are underscores).

1.1.2 Constraints

Constraints:

- 0 <= nums.length <= 100
- 0 <= nums[i] <= 50
- 0 <= val <= 100

1.2 Example test cases

[4]:
```python
def test_cases():
    count, nums = removeElement([3,2,2,3], 3)
    assert (count, nums) == (2, [2,2,'',''])
    count, nums = removeElement([0,1,2,2,3,0,4,2], 2)
    assert (count, nums) == (5, [0,1,4,0,3,'','',''])
    print("All test cases passed!")
```

1.3 Solutions

$O(n * logn)$

The first solution I suggest to go with is to check if the number if equal to the target value or not. If yes, replace it with a number higher than the contraint largest value (101 for instance). If not, then add 1 to a counter. Sort the list and return the counter.

[30]:
```python
def removeElement(nums, val):
    occ = 0
    for index, num in enumerate(nums):
        if num == val:
            nums[index] = 101
        else:
            occ += 1

    return occ, sorted(nums)
```

[31]:
```python
removeElement([3,2,2,3], 3)
```

[31]:
```
(2, [2, 2, 101, 101])
```

[32]:
```
removeElement([0,1,2,2,3,0,4,2], 2)
```

[32]:
```
(5, [0, 0, 1, 3, 4, 101, 101, 101])
```

[33]:
```
def removeElement(nums, val):
    occ = 0
    for index, num in enumerate(nums):
        if num == val:
            nums[index] = 101
        else:
            occ += 1
    nums.sort()

    return occ
```

[34]:
```
judge_remove_element(removeElement, [0,1,2,2,3,0,4,2], 2, [0, 0, 1, 3, 4])
```

```
All tests passed!
```

[35]:
```
judge_remove_element(removeElement, [3,2,2,3], 3, [2, 2])
```

```
All tests passed!
```

1.3.1 Improvements

$O(n)$

We can keep a pointer on an equal value until we replace it with an unequal value. If we get all the unequal values from the right side and replace all equal values with those we will end up having unequal values on the left. The order may change as suggested in the problem.

[50]:
```
def removeElement(nums, val):
    pointer = 0
    occ = 0
    for num in nums:
        if num != val:
            nums[pointer] = num
            occ += 1
            pointer += 1

    return occ
```

[51]:
```
removeElement([3,2,2,3], 3)
```

[51]:
```
2
```

[52]:
```
removeElement([0,1,2,2,3,0,4,2], 2)
```

[52]:
```
5
```

[53]:
```
judge_remove_element(removeElement, [0,1,2,2,3,0,4,2], 2, [0, 0, 1, 3, 4])
```

```
All tests passed!
```

[54]:
```
judge_remove_element(removeElement, [3,2,2,3], 3, [2, 2])
```

```
All tests passed!
```