educative

Solution Review: Right Rotate List

This review provides a detailed analysis of the different ways to right rotate the list.

We'll cover the following

Solution #1: Manual Rotation

```
def right_rotate(lst, k):
    k = k % len(lst)
    rotatedList = []
    # get the elements from the end
    for item in range(len(lst) - k, len(lst)):
        rotatedList.append(lst[item])
    # get the remaining elements
    for item in range(0, len(lst) - k):
        rotatedList.append(lst[item])
    return rotatedList

```

```
12
13  print(right_rotate([10, 20, 30, 40, 50], abs(3)))
```

In this solution, we first create an empty list. We then iterate through the last k elements of the list and place them at the start of the new list. Lastly, we append the first `length(lst)-k` elements to the new list and return.

Time Complexity

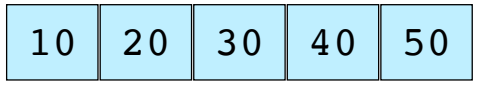
Since the entire list is iterated over, the time complexity of this solution is $O(n)$

Solution #2: Pythonic Rotation

```
def right_rotate(lst, k):
    # get rotation index
    k = k % len(lst)

    return lst[-k:] + lst[:-k]


print(right_rotate([10, 20, 30, 40, 50], abs(3)))
```



Pythonic rotation where k = 3

1 of 3

This solution simply uses list slicing to join together the last `k` and the first `len(lst) - k` elements and returns.

Time Complexity

List slicing is in $O(k)$ where k represents the number of elements that are sliced, and since the entire list is sliced, hence the total time complexity is in $O(n)$.

Back

Next

Challenge 8: Right Rotate List

Challenge 9: Rearrange Positive & Ne...

Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/solution-review-right-rotate-list__introduction-to-lists__data-structures-for-coding-interviews-in-python)