LeetCode Solutions

Program Creek

Version 0.0

Contents

1 Rotate Array in Java

You may have been using Java for a while. Do you think a simple Java array question can be a challenge? Let's use the following problem to test.

Problem: Rotate an array of n elements to the right by k steps. For example, with n = 7 and k = 3, the array [1,2,3,4,5,6,7] is rotated to [5,6,7,1,2,3,4].

How many different ways do you know to solve this problem?

1.1 Solution 1 - Intermediate Array

In a straightforward way, we can create a new array and then copy elements to the new array. Then change the original array by using System.arraycopy().

```
public void rotate(int[] nums, int k) {
   if(k > nums.length)
      k=k%nums.length;

   int[] result = new int[nums.length];

   for(int i=0; i < k; i++){
      result[i] = nums[nums.length-k+i];
   }

   int j=0;
   for(int i=k; i<nums.length; i++){
      result[i] = nums[j];
      j++;
   }

   System.arraycopy( result, 0, nums, 0, nums.length );
}
```

Space is O(n) and time is O(n).

1.2 Solution 2 - Bubble Rotate

Can we do this in O(1) space?

This solution is like a bubble sort.

```
public static void rotate(int[] arr, int order) {
  if (arr == null || order < 0) {
     throw new IllegalArgumentException("Illegal argument!");
```

```
  }

  for (int i = 0; i < order; i++) {
    for (int j = arr.length - 1; j > 0; j--) {
      int temp = arr[j];
      arr[j] = arr[j - 1];
      arr[j - 1] = temp;
    }
  }
}
```

However, the time is O(n*k).

1.3 Solution 3 - Reversal

Can we do this in O(1) space and in O(n) time? The following solution does.

Assuming we are given 1,2,3,4,5,6 and order 2. The basic idea is:

```
1. Divide the array two parts: 1,2,3,4 and 5, 6
2. Rotate first part: 4,3,2,1,5,6
3. Rotate second part: 4,3,2,1,6,5
4. Rotate the whole array: 5,6,1,2,3,4
```

```
public static void rotate(int[] arr, int order) {
  order = order % arr.length;

  if (arr == null || order < 0) {
    throw new IllegalArgumentException("Illegal argument!");
  }

  //length of first part
  int a = arr.length - order;

  reverse(arr, 0, a-1);
  reverse(arr, a, arr.length-1);
  reverse(arr, 0, arr.length-1);

}

public static void reverse(int[] arr, int left, int right){
  if(arr == null || arr.length == 1)
    return;

  while(left < right){
    int temp = arr[left];
    arr[left] = arr[right];
    arr[right] = temp;
    left++;
    right--;
```