31. Next Permutation

Summary

We need to find the next lexicographic permutation of the given list of numbers than the number formed by the given array.

Solution

Approach 1: Brute Force

Algorithm

In this approach, we find out every possible permutation of list formed by the elements of the given array and find out the permutation which is just larger than the given one. But this one will be a very naive approach, since it requires us to find out every possible permutation which will take really long time and the implementation is complex. Thus, this approach is not acceptable at all. Hence, we move on directly to the correct approach.

Complexity Analysis

- Time complexity : $O(n!)$. Total possible permutations is $n!$.
- Space complexity : $O(n)$. Since an array will be used to store the permutations.

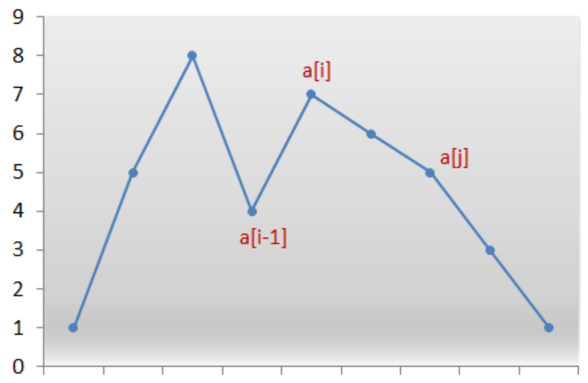
Approach 2: Single Pass Approach

Algorithm

First, we observe that for any given sequence that is in descending order, no next larger permutation is possible. For example, no next permutation is possible for the following array: `[9, 5, 4, 3, 1]`

We need to find the first pair of two successive numbers $a[i]$ and $a[i-1]$, from the right, which satisfy $a[i] > a[i-1]$. Now, no rearrangements to the right of $a[i-1]$ can create a larger permutation since that subarray consists of numbers in descending order. Thus, we need to rearrange the numbers to the right of $a[i-1]$ including itself.

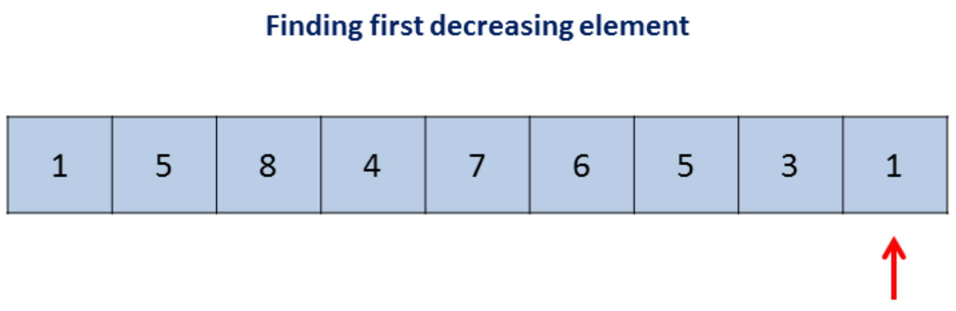
Now, what kind of rearrangement will produce the next larger number? We want to create the permutation just larger than the current one. Therefore, we need to replace the number $a[i-1]$ with the number which is just larger than itself among the numbers lying to its right section, say $a[j]$.



We swap the numbers $a[i-1]$ and $a[j]$. We now have the correct number at index $i-1$. But still the current permutation isn't the permutation that we are looking for. We need the smallest permutation that can be formed by using the numbers only to the right of $a[i-1]$. Therefore, we need to place those numbers in ascending order to get their smallest permutation.

But, recall that while scanning the numbers from the right, we simply kept decrementing the index until we found the pair $a[i]$ and $a[i-1]$ where, $a[i] > a[i-1]$. Thus, all numbers to the right of $a[i-1]$ were already sorted in descending order. Furthermore, swapping $a[i-1]$ and $a[j]$ didn't change that order. Therefore, we simply need to reverse the numbers following $a[i-1]$ to get the next smallest lexicographic permutation.

The following animation will make things clearer:



Java

```java
public class Solution {
    public void nextPermutation(int[] nums) {
        int i = nums.length - 2;
        while (i >= 0 && nums[i + 1] <= nums[i]) {
            i--;
        }
        if (i >= 0) {
            int j = nums.length - 1;
            while (j >= 0 && nums[j] <= nums[i]) {
                j--;
            }
            swap(nums, i, j);
        }
        reverse(nums, i + 1);
    }

    private void reverse(int[] nums, int start) {
        int i = start, j = nums.length - 1;
        while (i < j) {
            swap(nums, i, j);
            i++;
            j--;
        }
    }

    private void swap(int[] nums, int i, int j) {
        int temp = nums[i];
```

Complexity Analysis