Daily Coding Problem #95

Problem

This problem was asked by Palantir.

Given a number represented by a list of digits, find the next greater permutation of a number, in terms of lexicographic ordering. If there is not greater permutation possible, return the permutation with the lowest value/ordering.

For example, the list `[1,2,3]` should return `[1,3,2]`. The list `[1,3,2]` should return `[2,1,3]`. The list `[3,2,1]` should return `[1,2,3]`.

Can you perform the operation without allocating extra memory (disregarding the input memory)?

Solution

The brute force approach to this problem would be to generate all permutations of the number/list. One we have all of

the permutations, we choose the one that comes right after our input list, by taking the minimum of their distances. This approach would take `O(N!)` time to generate the permutations and find the one that is the closest to our given number.

We can instead observe a pattern in how the next permutation is obtained. First, consider the case where the entire sequence is decreasing (e.g. `[3,2,1]`). Regardless of the values, we cannot generate a permutation where the value is greater. We can see this since, if there were two possible values a and b that could be swapped to obtain a higher value of a, b must be greater than a. Since b comes after a in the sequence and the sequence is decreasing, we have a contradiction. Next, we have to generate the lowest possible value from the sequence. By similar logic, the smallest number we can generate is from an increasing sequence -- or the reverse of the decreasing sequence.

Now, we can break down other problems in terms of this subproblem. We will start from the right side of the list, and try to find the smallest digit that we can swap and obtain a higher number. While the sublist is a decreasing sequence, we cannot get a higher permutation. Instead, we try adding to the sublist by appending the digit to the left. Once we get a digit that is less than the one to the right, we swap it with the smallest digit that is higher than it. Then, we must make sure the remaining digits to the right are in their lowest-value permutation. We can do this by ordering the remaining digits on the right from least-to-greatest. Since our swap preserves the fact that the right sublist is decreasing, we simply reverse it. For example:

```
[1, 3, 5, 4, 2]
             ^ decreasing sublist
[1, 3, 5, 4, 2]
          ^  ^ decreasing sublist
[1, 3, 5, 4, 2]
       ^     ^ decreasing sublist
[1, 3, 5, 4, 2]
    ^        ^ non-decreasing, so swap with next-highest digit
[1, 4, 5, 3, 2]
    ^        ^ reverse digits to the right, sorting least-to-greatest
[1, 4, 2, 3, 5]
```

```
def nextPermutation(self, nums):
```

```
def swap(nums, a, b):
    # Perform an in-place swap
    nums[a], nums[b] = nums[b], nums[a]

def reverse(nums, a, b):
    # Reverses elements at index a to b (inclusive) in-place
    nums[a:b+1] = reversed(nums[a:b+1])

# Find first index where nums[idx] < nums[idx + 1]
pivot = len(nums) - 2
while pivot >= 0 and nums[pivot] >= nums[pivot + 1]:
    pivot -= 1

if pivot >= 0:
    # Find the next-largest number to swap with
    successor = len(nums) - 1
    while (successor > 0 and nums[successor] <= nums[pivot]):
        successor -= 1
    swap(nums, pivot, successor)

reverse(nums, pivot + 1, len(nums) - 1)
```