Leetcode 179.
Largest Number

2024.02.04

원루빈

Problem Description - Largest Number

Given a list of non-negative integers nums, arrange them such that they form the largest number and return it.

Since the result may be very large, so you need to return a string instead of an integer.

Example 1:

Input: nums = [10,2]
Output: "210"

Example 2:

Input: nums = [3,30,34,5,9]
Output: "9534330"

1 Apply the Key Function (lambda x: x*3)

[Approach 1] The key function is applied to each element in the list to create a new list where each number is repeated 3 times

```
nums_str = [str(item) for item in nums]
nums_str.sort(key=lambda x: x*3, reverse=True)
```

- Python sort method modifies the list in-place
- ‘Key’ argument determines the basis on which the sorting is done. In this case it takes the element x from the list and returns x*3 (e.g., '9' becomes '999')
- reverse=True since sort method sorts the list in ascending order by default.

1 Apply the Key Function (lambda x: x*3)

E.g.

nums = [3,30,34,5,9]
nums_str = [‘3’,’30’,’34’,’5’,’9’]

nums_str = [str(item) for item in nums]
nums_str.sort(key=lambda x: x*3, reverse=True)

'3' → '333'
'30' → '303030'
'34' → '343434'
'5' → '555'
'9' → '999’

Before sorting: ['333', '303030', '343434', '555', '999']
After sorting: ['999', '555', '343434', '333', '303030']

Result: ['9', '5', '34', '3', '30']

```
def largestNumber_1(self, nums) -> str:
    # fails testcase: nums = [999999991,9]

    nums_str = [str(item) for item in nums]
    # sort_str_nums = sorted(nums_str, key=lambda x:

    nums_str.sort(key=lambda x: x*3, reverse=True)

    return ''.join(nums_str)
```

1 Apply the Key Function (lambda x: x*3)

```
nums_str = [str(item) for item in nums]
nums_str.sort(key=lambda x: x*3, reverse=True)
```

Constraints:

- `1 <= nums.length <= 100`
- `0 <= nums[i] <= 10^9`

Test Case Failure:

nums = [999999991, 9]

Time complexity: O(n log n)
Space complexity: O(n)

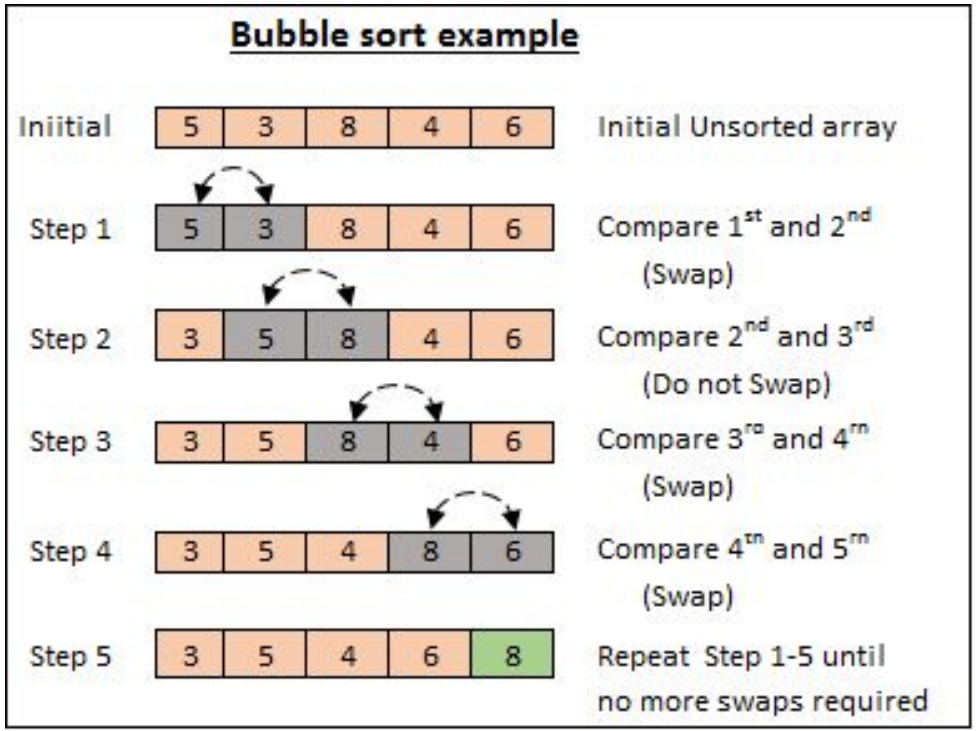
2 Bubble sort (concatenation of the pairs of numbers)

[Approach 2]
We can use Bubble sort based on the concatenation of the pairs of numbers in string format.

For example, when comparing 9 and 34, we are comparing 923 and 349.

Time complexity: O(n^2) because of the bubble sort
Space complexity: O(n)

2 Bubble sort (concatenation of the pairs of numbers)



Bubble Sort is based on the idea of **repeatedly comparing pairs** of **adjacent** elements
and **then swapping** their positions if they exist in the wrong order.

2 Bubble sort (concatenation of the pairs of numbers)

```
def largestNumber_2(self, nums) -> str:
    nums_str = [str(item) for item in nums]
    n = len(nums_str)

    for i in range(n):
        for j in range(0, n - i - 1):
            comp1 = nums_str[j] + nums_str[j + 1]
            comp2 = nums_str[j + 1] + nums_str[j]
            if comp1 < comp2:  # If the current pair is in the wrong order
                nums_str[j], nums_str[j + 1] = nums_str[j + 1], nums_str[j]  # Swap

    result = ''.join(nums_str)
    if result[0] == '0':  # Handle the edge case where the largest number is '0'
        return '0'
    return result
```

3 Using custom comparison function

[Approach 3]

Define a custom comparison function compare that dictates the sorting order based on the concatenation of pairs of numbers in string format.

```
def compare(n1, n2):
    if n1 + n2 > n2 + n1:
        return -1
    else:
        return 1
```

3 Using custom comparison function

```
from functools import cmp_to_key
```

cmp_to_key() is used to **convert a comparison function** into a **key function** that can be used by sorting functions like sort() or sorted()

*** comparison function:**
a function that compares two items and returns negative, zero, or positive

- Returns:
  - negative number if a < b,
  - zero if a == b,
  - positive number if a > b.

3 Using custom comparison function

```
def compare(n1, n2):
    if n1 + n2 > n2 + n1:
        return -1
    else:
        return 1
```

```
nums = sorted(nums, key=cmp_to_key(compare))
```

3 Handling Test case: [0, 0]

Note that we need to handle test cases with multiple zeros.

For example, if nums = [0, 0]
instead of "00", we need to return "0"

Instead of returning:

```
"".join(nums)
```

Return:

```
str(int("".join(nums)))
```

3 Using custom comparison function

```python
def largestNumber_3(self, nums) -> str:
    for i, n in enumerate(nums):
        nums[i] = str(n)

    def compare(n1, n2):
        if n1 + n2 > n2 + n1:
            return -1
        else:
            return 1

    nums = sorted(nums, key=cmp_to_key(compare))

    # handle [0, 0] instead of "00", we would like to return "0"

    return str(int("".join(nums)))
```

3 Other Solution (leetcode editorial)

```python
class LargerNumKey(str):
    def __lt__(x, y):
        return x+y > y+x

class Solution:
    def largestNumber(self, nums):
        largest_num = ''.join(sorted(map(str, nums), key=LargerNumKey))
        return '0' if largest_num[0] == '0' else largest_num
```

감사합니다!

THANK YOU