179. Largest Number

Dec. 12, 2017 | 73.9K views

Previous Next

Average Rating: 3.97 (57 votes)

Given a list of non negative integers, arrange them such that they form the largest number.

Example 1:

```
Input: [10,2]
Output: "210"
```

Example 2:

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

Note: The result may be very large, so you need to return a string instead of an integer.

Approach #1 Sorting via Custom Comparator [Accepted]

Intuition

To construct the largest number, we want to ensure that the most significant digits are occupied by the largest digits.

Algorithm

First, we convert each integer to a string. Then, we sort the array of strings.

While it might be tempting to simply sort the numbers in descending order, this causes problems for sets of numbers with the same leading digit. For example, sorting the problem example in descending order would produce the number 9534303, while the correct answer can be achieved by transposing the 3 and the 30. Therefore, for each pairwise comparison during the sort, we compare the numbers achieved by concatenating the pair in both orders. We can prove that this sorts into the proper order as follows:

Assume that (without loss of generality), for some pair of integers a and b, our comparator dictates that a should precede b in sorted order. This means that $a \frown b > b \frown a$ (where $\frown$ represents concatenation). For the sort to produce an incorrect ordering, there must be some c for which b precedes c and c precedes a. This is a contradiction because $a \frown b > b \frown a$ and $b \frown c > c \frown b$ implies $a \frown c > c \frown a$. In other words, our custom comparator preserves transitivity, so the sort is correct.

Once the array is sorted, the most "signficant" number will be at the front. There is a minor edge case that comes up when the array consists of only zeroes, so if the most significant number is 0, we can simply return 0. Otherwise, we build a string out of the sorted array and return it.

Java | Python3 — Copy

```
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
```

Complexity Analysis

- Time complexity : $\mathcal{O}(nlgn)$

  Although we are doing extra work in our comparator, it is only by a constant factor. Therefore, the overall runtime is dominated by the complexity of `sort`, which is $\mathcal{O}(nlgn)$ in Python and Java.

- Space complexity : $\mathcal{O}(n)$

  Here, we allocate $\mathcal{O}(n)$ additional space to store the copy of `nums`. Although we could do that work in place (if we decide that it is okay to modify `nums`), we must allocate $\mathcal{O}(n)$ space for the final return string. Therefore, the overall memory footprint is linear in the length of `nums`.

Analysis and solutions written by: @emptyset

Rate this article:

Previous Next

Comments: 32

Sort By

Type comment here... (Markdown is supported)

Preview Post

plantvsdyf ★ 50 October 15, 2018 11:59 AM

LOL. The "proof" is cute.

49 Share Reply

SHOW 1 REPLY

anmingyu11 ★ 425 February 22, 2019 7:58 PM

I cant imagine the a~b > b~a , this is so subtle, how can you figure it out ? i mean int math?or just like a idea com out of your mind ?

10 Share Reply

victorxie2013 ★ 7 June 7, 2018 12:58 PM Report

I don't get why "ab > ba and bc > cb implies ac > ca ". It seems to me some proof is needed here. Can some elaborate how to prove it?

7 Share Reply

SHOW 7 REPLIES

l0709394 ★ 15 June 20, 2019 6:35 AM

```
why do you only need to override __lt__?
I tried to flip the sign and use __gt__ but why does it fail?
```

Read More

6 Share Reply

SHOW 2 REPLIES

kalfon ★ 6 May 14, 2018 7:17 PM

I came up with a different compare function. I take the two strings representing the numbers to be compared, s1 and s2, and compare each character with wraparound. That is, I start at i = 0 up to len(s1)*len(s2) (I think lcm(len(s1), len(s2)) is actually enough) and compare s1[i % len(s1)] and s2[i % len(s2)]. The first non-equal occurence tells me which string is 'bigger'.

6 Share Reply

SHOW 7 REPLIES

the0demiurge ★ 6 March 28, 2018 8:18 PM Report

@kounkou
It is not an O(N) solution because you used sort() and sort is O(N log(N))

6 Share Reply

SHOW 3 REPLIES

iomonad ★ 43 May 14, 2019 6:05 AM

I wrote a proof of the transitivity of this comparator.
https://leetcode.com/problems/largest-number/discuss/291988/A-Proof-of-the-Concatenation-Comparator's-Transtivity

4 Share Reply

SHOW 1 REPLY

haoyangfan ★ 903 January 28, 2019 3:27 AM

offer my javascript solution:

```
/**
 * @param {number[]} nums
```

Read More

4 Share Reply

happysings ★ 2 January 24, 2019 10:10 PM

`cpp` solution faster than 100%

```
class Solution {
public:
```

Read More

2 Share Reply

g5li ★ 3 April 3, 2019 2:12 PM Report

Please check another solution with DFS Tree.
https://leetcode.com/problems/largest-number/discuss/267353/

1 Share Reply