179. Largest Number

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

Note: 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"
```

Example 3:

```
Input: nums = [1]
Output: "1"
```

Example 4:

```
Input: nums = [10]
Output: "10"
```

Constraints:

- `1 <= nums.length <= 100`
- `0 <= nums[i] <= 10<sup>9</sup>`

```
from functools import cmp_to_key
class Solution:
    def largestNumber(self, nums: List[int]) -> str:
        def compare(a,b):
            n1 = a+b
            n2 = b+a
            return int(n1)-int(n2)



        nums = [str(x) for x in nums]
        nums.sort(key = cmp_to_key(compare),reverse = True)
        temp = ''.join(nums)
        if int(temp)==0:
            return '0'
        else:
            return temp
        # print(nums)
```

```
def largestNumber1(self, nums):
    if not any(nums):
        return "0"
    return "".join(sorted(map(str, nums), cmp=lambda n1, n2: -1 if
n1+n2>n2+n1 else (1 if n1+n2<n2+n1 else 0)))

# bubble sort
def largestNumber2(self, nums):
    for i in xrange(len(nums), 0, -1):
        for j in xrange(i-1):
            if not self.compare(nums[j], nums[j+1]):
                nums[j], nums[j+1] = nums[j+1], nums[j]
    return str(int("".join(map(str, nums))))

def compare(self, n1, n2):
    return str(n1) + str(n2) > str(n2) + str(n1)

# selection sort
def largestNumber3(self, nums):
    for i in xrange(len(nums), 0, -1):
        tmp = 0
        for j in xrange(i):
            if not self.compare(nums[j], nums[tmp]):
                tmp = j
        nums[tmp], nums[i-1] = nums[i-1], nums[tmp]
    return str(int("".join(map(str, nums))))

# insertion sort
def largestNumber4(self, nums):
    for i in xrange(len(nums)):
        pos, cur = i, nums[i]
        while pos > 0 and not self.compare(nums[pos-1], cur):
            nums[pos] = nums[pos-1]  # move one-step forward
            pos -= 1
        nums[pos] = cur
    return str(int("".join(map(str, nums))))

# merge sort
```

```python
def largestNumber5(self, nums):
    nums = self.mergeSort(nums, 0, len(nums)-1)
    return str(int("".join(map(str, nums))))

def mergeSort(self, nums, l, r):
    if l > r:
        return
    if l == r:
        return [nums[l]]
    mid = l + (r-l)//2
    left = self.mergeSort(nums, l, mid)
    right = self.mergeSort(nums, mid+1, r)
    return self.merge(left, right)

def merge(self, l1, l2):
    res, i, j = [], 0, 0
    while i < len(l1) and j < len(l2):
        if not self.compare(l1[i], l2[j]):
            res.append(l2[j])
            j += 1
        else:
            res.append(l1[i])
            i += 1
    res.extend(l1[i:] or l2[j:])
    return res

# quick sort, in-place
def largestNumber(self, nums):
    self.quickSort(nums, 0, len(nums)-1)
    return str(int("".join(map(str, nums))))

def quickSort(self, nums, l, r):
    if l >= r:
        return
    pos = self.partition(nums, l, r)
    self.quickSort(nums, l, pos-1)
    self.quickSort(nums, pos+1, r)

def partition(self, nums, l, r):
    low = l
    while l < r:
        if self.compare(nums[l], nums[r]):
            nums[l], nums[low] = nums[low], nums[l]
            low += 1
        l += 1
    nums[low], nums[r] = nums[r], nums[low]
    return low
```