Problem 179: Largest Number

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

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"

Constraints:

1 <= nums.length <= 100

0 <= nums[i] <= 10

9

Code Snippets

C++:

```
class Solution {
public:
string largestNumber(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public String largestNumber(int[] nums) {

}
}
```

Python3:

```
class Solution:
def largestNumber(self, nums: List[int]) -> str:
```

Python:

```
class Solution(object):
def largestNumber(self, nums):
"""
:type nums: List[int]
```

```
:rtype: str
"""
```

JavaScript:

```
/**
* @param {number[]} nums
* @return {string}
*/
var largestNumber = function(nums) {

};
```

TypeScript:

```
function largestNumber(nums: number[]): string {

};
```

C#:

```
public class Solution {
public string LargestNumber(int[] nums) {

}
}
```

C:

```
char* largestNumber(int* nums, int numsSize) {

}
```

Go:

```
func largestNumber(nums []int) string {

}
```

Kotlin:

```
class Solution {
fun largestNumber(nums: IntArray): String {

}
}
```

Swift:

```
class Solution {
func largestNumber(_ nums: [Int]) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn largest_number(nums: Vec<i32>) -> String {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @return {String}
def largest_number(nums)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return String
*/
function largestNumber($nums) {

}
}
```

Dart:

```
class Solution {
String largestNumber(List<int> nums) {

}
}
```

Scala:

```
object Solution {
def largestNumber(nums: Array[Int]): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec largest_number(nums :: [integer]) :: String.t
def largest_number(nums) do

end
end
```

Erlang:

```
-spec largest_number(Nums :: [integer()]) -> unicode:unicode_binary().
largest_number(Nums) ->
.
```

Racket:

```
(define/contract (largest-number nums)
(-> (listof exact-integer?) string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Largest Number
* Difficulty: Medium
* Tags: array, string, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
string largestNumber(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Largest Number
* Difficulty: Medium
* Tags: array, string, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public String largestNumber(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Largest Number
Difficulty: Medium
Tags: array, string, greedy, sort
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def largestNumber(self, nums: List[int]) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def largestNumber(self, nums):
"""
:type nums: List[int]
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Largest Number
* Difficulty: Medium
* Tags: array, string, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[]} nums
* @return {string}
*/
var largestNumber = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Largest Number
* Difficulty: Medium
* Tags: array, string, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function largestNumber(nums: number[]): string {

};
```

C# Solution:

```
/*
* Problem: Largest Number
* Difficulty: Medium
* Tags: array, string, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public string LargestNumber(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Largest Number
* Difficulty: Medium
* Tags: array, string, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

char* largestNumber(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Largest Number
// Difficulty: Medium
// Tags: array, string, greedy, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func largestNumber(nums []int) string {

}
```

Kotlin Solution:

```
class Solution {
fun largestNumber(nums: IntArray): String {

}
}
```

Swift Solution:

```
class Solution {
func largestNumber(_ nums: [Int]) -> String {

}
}
```

Rust Solution:

```
// Problem: Largest Number
// Difficulty: Medium
// Tags: array, string, greedy, sort
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn largest_number(nums: Vec<i32>) -> String {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @return {String}
def largest_number(nums)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @return String
*/
function largestNumber($nums) {

}
}
```

Dart Solution:

```
class Solution {
String largestNumber(List<int> nums) {

}
}
```

Scala Solution:

```
object Solution {
def largestNumber(nums: Array[Int]): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec largest_number(nums :: [integer]) :: String.t
def largest_number(nums) do

end
end
```

Erlang Solution:

```
-spec largest_number(Nums :: [integer()]) -> unicode:unicode_binary().
largest_number(Nums) ->
.
```

Racket Solution:

```
(define/contract (largest-number nums)
(-> (listof exact-integer?) string?)
)
```