Problem 2553: Separate the Digits in an Array

Problem Information

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

Problem Description

Given an array of positive integers

nums

, return

an array

answer

that consists of the digits of each integer in

nums

after separating them in

the same order

they appear in

nums

.

To separate the digits of an integer is to get all the digits it has in the same order.

For example, for the integer

10921

, the separation of its digits is

[1,0,9,2,1]

.

Example 1:

Input:

nums = [13,25,83,77]

Output:

[1,3,2,5,8,3,7,7]

Explanation:

- The separation of 13 is [1,3]. - The separation of 25 is [2,5]. - The separation of 83 is [8,3]. - The separation of 77 is [7,7]. answer = [1,3,2,5,8,3,7,7]. Note that answer contains the separations in the same order.

Example 2:

Input:

nums = [7,1,3,9]

Output:

[7,1,3,9]

Explanation:

The separation of each integer in nums is itself. answer = [7,1,3,9].

Constraints:

1 <= nums.length <= 1000

1 <= nums[i] <= 10

5

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public int[] separateDigits(int[] nums) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function separateDigits(nums: number[]): number[] {

};
```

C#:

```
public class Solution {
public int[] SeparateDigits(int[] nums) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* separateDigits(int* nums, int numsSize, int* returnSize) {

}
```

Go:

```
func separateDigits(nums []int) []int {

}
```

Kotlin:

```
class Solution {
fun separateDigits(nums: IntArray): IntArray {
```

```
}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
List<int> separateDigits(List<int> nums) {

}
}
```

Scala:

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

}
}
```

Elixir:

```
defmodule Solution do
@spec separate_digits(nums :: [integer]) :: [integer]
def separate_digits(nums) do

end
end
```

Erlang:

```
-spec separate_digits(Nums :: [integer()]) -> [integer()].
separate_digits(Nums) ->
.
```

Racket:

```
(define/contract (separate-digits nums)
(-> (listof exact-integer?) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Separate the Digits in an Array
```

```
* Difficulty: Easy
* Tags: array
*
* 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:
vector<int> separateDigits(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Separate the Digits in an Array
* Difficulty: Easy
* Tags: array
*
* 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 int[] separateDigits(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Separate the Digits in an Array
Difficulty: Easy
Tags: array

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 separateDigits(self, nums: List[int]) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Separate the Digits in an Array
* Difficulty: Easy
* Tags: array
*
* 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 {number[]}
*/
var separateDigits = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Separate the Digits in an Array
* Difficulty: Easy
* Tags: array
*
* 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 separateDigits(nums: number[]): number[] {

};
```

C# Solution:

```
/*
* Problem: Separate the Digits in an Array
* Difficulty: Easy
* Tags: array
*
* 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 int[] SeparateDigits(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Separate the Digits in an Array
* Difficulty: Easy
* Tags: array
*
* 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
```

```
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* separateDigits(int* nums, int numsSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Separate the Digits in an Array
// Difficulty: Easy
// Tags: array
//
// 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 separateDigits(nums []int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun separateDigits(nums: IntArray): IntArray {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Separate the Digits in an Array
// Difficulty: Easy
// Tags: array
//
// 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 separate_digits(nums: Vec<i32>) -> Vec<i32> {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
List<int> separateDigits(List<int> nums) {

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec separate_digits(nums :: [integer]) :: [integer]
def separate_digits(nums) do

end
end
```

Erlang Solution:

```
-spec separate_digits(Nums :: [integer()]) -> [integer()].
separate_digits(Nums) ->
.
```

Racket Solution:

```
(define/contract (separate-digits nums)
(-> (listof exact-integer?) (listof exact-integer?))
)
```