Problem 2562: Find the Array Concatenation Value

Problem Information

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

Problem Description

You are given a

0-indexed

integer array

nums

.

The

concatenation

of two numbers is the number formed by concatenating their numerals.

For example, the concatenation of

15

,

49

is

1549

.

The

concatenation value

of

nums

is initially equal to

0

. Perform this operation until

nums

becomes empty:

If

nums

has a size greater than one, add the value of the concatenation of the first and the last element to the

concatenation value

of

nums

, and remove those two elements from

nums

. For example, if the

nums

was

[1, 2, 4, 5, 6]

, add 16 to the

concatenation value

.

If only one element exists in

nums

, add its value to the

concatenation value

of

nums

, then remove it.

Return

the concatenation value of

nums

.

Example 1:

Input:

nums = [7,52,2,4]

Output:

596

Explanation:

Before performing any operation, nums is [7,52,2,4] and concatenation value is 0. - In the first operation: We pick the first element, 7, and the last element, 4. Their concatenation is 74, and we add it to the concatenation value, so it becomes equal to 74. Then we delete them from nums, so nums becomes equal to [52,2]. - In the second operation: We pick the first element, 52, and the last element, 2. Their concatenation is 522, and we add it to the concatenation value, so it becomes equal to 596. Then we delete them from the nums, so nums becomes empty. Since the concatenation value is 596 so the answer is 596.

Example 2:

Input:

nums = [5,14,13,8,12]

Output:

673

Explanation:

Before performing any operation, nums is [5,14,13,8,12] and concatenation value is 0. - In the first operation: We pick the first element, 5, and the last element, 12. Their concatenation is 512, and we add it to the concatenation value, so it becomes equal to 512. Then we delete them from the nums, so nums becomes equal to [14,13,8]. - In the second operation: We pick the first element, 14, and the last element, 8. Their concatenation is 148, and we add it to the concatenation value, so it becomes equal to 660. Then we delete them from the nums, so nums becomes equal to [13]. - In the third operation: nums has only one element, so we pick 13 and add it to the concatenation value, so it becomes equal to 673. Then we delete it from nums, so nums become empty. Since the concatenation value is 673 so the answer is 673.

Constraints:

1 <= nums.length <= 1000

1 <= nums[i] <= 10

4

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public long findTheArrayConcVal(int[] nums) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public long FindTheArrayConcVal(int[] nums) {

}
}
```

C:

```
long long findTheArrayConcVal(int* nums, int numsSize) {

}
```

Go:

```
func findTheArrayConcVal(nums []int) int64 {

}
```

Kotlin:

```
class Solution {
fun findTheArrayConcVal(nums: IntArray): Long {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn find_the_array_conc_val(nums: Vec<i32>) -> i64 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}

}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (find-the-array-conc-val nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Array Concatenation Value
* 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:
long long findTheArrayConcVal(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Array Concatenation Value
* 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 long findTheArrayConcVal(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Array Concatenation Value
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 findTheArrayConcVal(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Array Concatenation Value
* 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 findTheArrayConcVal = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Array Concatenation Value
* 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 findTheArrayConcVal(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Find the Array Concatenation Value
* 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 long FindTheArrayConcVal(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Find the Array Concatenation Value
* 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
*/

long long findTheArrayConcVal(int* nums, int numsSize) {
}
```

Go Solution:

```
// Problem: Find the Array Concatenation Value
// 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 findTheArrayConcVal(nums []int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun findTheArrayConcVal(nums: IntArray): Long {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find the Array Concatenation Value
// 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 find_the_array_conc_val(nums: Vec<i32>) -> i64 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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



}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (find-the-array-conc-val nums)
(-> (listof exact-integer?) exact-integer?)
)
```