Problem 2341: Maximum Number of Pairs in Array

Problem Information

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

Problem Description

You are given a

0-indexed

integer array

nums

. In one operation, you may do the following:

Choose

two

integers in

nums

that are

equal

.

Remove both integers from

nums

, forming a

pair

.

The operation is done on

nums

as many times as possible.

Return

a

0-indexed

integer array

answer

of size

2

where

answer[0]

is the number of pairs that are formed and

answer[1]

is the number of leftover integers in

nums

after doing the operation as many times as possible

.

Example 1:

Input:

nums = [1,3,2,1,3,2,2]

Output:

[3,1]

Explanation:

Form a pair with nums[0] and nums[3] and remove them from nums. Now, nums = [3,2,3,2,2].
Form a pair with nums[0] and nums[2] and remove them from nums. Now, nums = [2,2,2].
Form a pair with nums[0] and nums[1] and remove them from nums. Now, nums = [2]. No more pairs can be formed. A total of 3 pairs have been formed, and there is 1 number leftover in nums.

Example 2:

Input:

nums = [1,1]

Output:

[1,0]

Explanation:

Form a pair with nums[0] and nums[1] and remove them from nums. Now, nums = []. No more pairs can be formed. A total of 1 pair has been formed, and there are 0 numbers leftover in nums.

Example 3:

Input:

nums = [0]

Output:

[0,1]

Explanation:

No pairs can be formed, and there is 1 number leftover in nums.

Constraints:

1 <= nums.length <= 100

0 <= nums[i] <= 100

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
```

```
int* numberOfPairs(int* nums, int numsSize, int* returnSize) {

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (number-of-pairs nums)
(-> (listof exact-integer?) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Number of Pairs in Array
* Difficulty: Easy
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Maximum Number of Pairs in Array
* Difficulty: Easy
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

```
}
}
```

Python3 Solution:

```
"""
Problem: Maximum Number of Pairs in Array
Difficulty: Easy
Tags: array, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Number of Pairs in Array
* Difficulty: Easy
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

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

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Number of Pairs in Array
* Difficulty: Easy
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function numberOfPairs(nums: number[]): number[] {

};
```

C# Solution:

```
/*
* Problem: Maximum Number of Pairs in Array
* Difficulty: Easy
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int[] NumberOfPairs(int[] nums) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Maximum Number of Pairs in Array
* Difficulty: Easy
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
```

Go Solution:

```
// Problem: Maximum Number of Pairs in Array
// Difficulty: Easy
// Tags: array, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func numberOfPairs(nums []int) []int {

}
```

Kotlin Solution:

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

}
```

```
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximum Number of Pairs in Array
// Difficulty: Easy
// Tags: array, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer[]
```

```
*/
function numberOfPairs($nums) {

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (number-of-pairs nums)
(-> (listof exact-integer?) (listof exact-integer?))
```

```
)
```