Problem 2295: Replace Elements in an Array

Problem Information

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

Problem Description

You are given a

0-indexed

array

nums

that consists of

n

distinct

positive integers. Apply

m

operations to this array, where in the

i

th

operation you replace the number

operations[i][0]

with

operations[i][1]

.

It is guaranteed that in the

i

th

operation:

operations[i][0]

exists

in

nums

.

operations[i][1]

does

not

exist in

nums

.

Return

the array obtained after applying all the operations

.

Example 1:

Input:

nums = [1,2,4,6], operations = [[1,3],[4,7],[6,1]]

Output:

[3,2,7,1]

Explanation:

We perform the following operations on nums: - Replace the number 1 with 3. nums becomes [

3

,2,4,6]. - Replace the number 4 with 7. nums becomes [3,2,

7

,6]. - Replace the number 6 with 1. nums becomes [3,2,7,

1

]. We return the final array [3,2,7,1].

Example 2:

Input:

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

Output:

[2,1]

Explanation:

We perform the following operations to nums: - Replace the number 1 with 3. nums becomes [

3

,2]. - Replace the number 2 with 1. nums becomes [3,

1

]. - Replace the number 3 with 2. nums becomes [

2

,1]. We return the array [2,1].

Constraints:

n == nums.length

m == operations.length

1 <= n, m <= 10

5

All the values of

nums

are

distinct

.

operations[i].length == 2

1 <= nums[i], operations[i][0], operations[i][1] <= 10

6

operations[i][0]

will exist in

nums

when applying the

i

th

operation.

operations[i][1]

will not exist in

nums

when applying the

i

th

operation.

Code Snippets

C++:

```
class Solution {
public:
vector<int> arrayChange(vector<int>& nums, vector<vector<int>>& operations) {

}
};
```

Java:

```
class Solution {
public int[] arrayChange(int[] nums, int[][] operations) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function arrayChange(nums: number[], operations: number[][]): number[] {

};
```

C#:

```
public class Solution {
public int[] ArrayChange(int[] nums, int[][] operations) {

}
}
```

C:

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

}
```

Go:

```
func arrayChange(nums []int, operations [][]int) []int {

}
```

Kotlin:

```
class Solution {
fun arrayChange(nums: IntArray, operations: Array<IntArray>): IntArray {

}
}
```

Swift:

```
class Solution {
func arrayChange(_ nums: [Int], _ operations: [[Int]]) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn array_change(nums: Vec<i32>, operations: Vec<Vec<i32>>) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer[][]} operations
# @return {Integer[]}
def array_change(nums, operations)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer[][] $operations
* @return Integer[]
*/
function arrayChange($nums, $operations) {

}
}
```

Dart:

```
class Solution {
List<int> arrayChange(List<int> nums, List<List<int>> operations) {

}
}
```

Scala:

```
object Solution {
def arrayChange(nums: Array[Int], operations: Array[Array[Int]]): Array[Int]
```

```
= {

}
}
```

Elixir:

```
defmodule Solution do
@spec array_change(nums :: [integer], operations :: [[integer]]) :: [integer]
def array_change(nums, operations) do

end
end
```

Erlang:

```
-spec array_change(Nums :: [integer()], Operations :: [[integer()]]) ->
[integer()].
array_change(Nums, Operations) ->
.
```

Racket:

```
(define/contract (array-change nums operations)
(-> (listof exact-integer?) (listof (listof exact-integer?)) (listof
exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Replace Elements in an Array
* Difficulty: Medium
* 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> arrayChange(vector<int>& nums, vector<vector<int>>& operations) {

}
};
```

Java Solution:

```
/**
* Problem: Replace Elements in an Array
* Difficulty: Medium
* 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[] arrayChange(int[] nums, int[][] operations) {

}
}
```

Python3 Solution:

```
"""
Problem: Replace Elements in an Array
Difficulty: Medium
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 arrayChange(self, nums: List[int], operations: List[List[int]]) ->
List[int]:
```

```
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Replace Elements in an Array
* Difficulty: Medium
* 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
* @param {number[][]} operations
* @return {number[]}
*/
var arrayChange = function(nums, operations) {

};
```

TypeScript Solution:

```
/**
* Problem: Replace Elements in an Array
* Difficulty: Medium
* 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 arrayChange(nums: number[], operations: number[][]): number[] {

};
```

C# Solution:

```
/*
* Problem: Replace Elements in an Array
* Difficulty: Medium
* 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[] ArrayChange(int[] nums, int[][] operations) {

}
}
```

C Solution:

```
/*
* Problem: Replace Elements in an Array
* Difficulty: Medium
* 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* arrayChange(int* nums, int numsSize, int** operations, int
operationsSize, int* operationsColSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Replace Elements in an Array
// Difficulty: Medium
// 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 arrayChange(nums []int, operations [][]int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun arrayChange(nums: IntArray, operations: Array<IntArray>): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func arrayChange(_ nums: [Int], _ operations: [[Int]]) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Replace Elements in an Array
// Difficulty: Medium
// 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 array_change(nums: Vec<i32>, operations: Vec<Vec<i32>>) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer[][]} operations
# @return {Integer[]}
def array_change(nums, operations)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer[][] $operations
* @return Integer[]
*/
function arrayChange($nums, $operations) {

}
}
```

Dart Solution:

```
class Solution {
List<int> arrayChange(List<int> nums, List<List<int>> operations) {

}
}
```

Scala Solution:

```
object Solution {
def arrayChange(nums: Array[Int], operations: Array[Array[Int]]): Array[Int]
= {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec array_change(nums :: [integer], operations :: [[integer]]) :: [integer]
def array_change(nums, operations) do

end
end
```

Erlang Solution:

```
-spec array_change(Nums :: [integer()], Operations :: [[integer()]]) ->
[integer()].
array_change(Nums, Operations) ->
.
```

Racket Solution:

```
(define/contract (array-change nums operations)
(-> (listof exact-integer?) (listof (listof exact-integer?)) (listof
exact-integer?))
)
```