Problem 2433: Find The Original Array of Prefix Xor

Problem Information

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

Problem Description

You are given an

integer

array

pref

of size

n

. Find and return

the array

arr

of size

n

that satisfies

:

pref[i] = arr[0] ^ arr[1] ^ ... ^ arr[i]

.

Note that

^

denotes the

bitwise-xor

operation.

It can be proven that the answer is

unique

.

Example 1:

Input:

pref = [5,2,0,3,1]

Output:

[5,7,2,3,2]

Explanation:

From the array [5,7,2,3,2] we have the following: - pref[0] = 5. - pref[1] = 5 ^ 7 = 2. - pref[2] = 5 ^ 7 ^ 2 = 0. - pref[3] = 5 ^ 7 ^ 2 ^ 3 = 3. - pref[4] = 5 ^ 7 ^ 2 ^ 3 ^ 2 = 1.

Example 2:

Input:

pref = [13]

Output:

[13]

Explanation:

We have pref[0] = arr[0] = 13.

Constraints:

1 <= pref.length <= 10

5

0 <= pref[i] <= 10

6

Code Snippets

C++:

```
class Solution {
public:
vector<int> findArray(vector<int>& pref) {

}
};
```

Java:

```
class Solution {
public int[] findArray(int[] pref) {

}
}
```

Python3:

```
class Solution:
def findArray(self, pref: List[int]) -> List[int]:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function findArray(pref: number[]): number[] {

};
```

C#:

```
public class Solution {
public int[] FindArray(int[] pref) {

}
}
```

C:

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

```
int* findArray(int* pref, int prefSize, int* returnSize) {

}
```

Go:

```
func findArray(pref []int) []int {

}
```

Kotlin:

```
class Solution {
fun findArray(pref: IntArray): IntArray {

}
}
```

Swift:

```
class Solution {
func findArray(_ pref: [Int]) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn find_array(pref: Vec<i32>) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer[]} pref
# @return {Integer[]}
def find_array(pref)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $pref
* @return Integer[]
*/
function findArray($pref) {

}
}
```

Dart:

```
class Solution {
List<int> findArray(List<int> pref) {

}
}
```

Scala:

```
object Solution {
def findArray(pref: Array[Int]): Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec find_array(pref :: [integer]) :: [integer]
def find_array(pref) do

end
end
```

Erlang:

```
-spec find_array(Pref :: [integer()]) -> [integer()].
find_array(Pref) ->
.
```

Racket:

```
(define/contract (find-array pref)
(-> (listof exact-integer?) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find The Original Array of Prefix Xor
* Difficulty: Medium
* 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> findArray(vector<int>& pref) {

}
};
```

Java Solution:

```
/**
* Problem: Find The Original Array of Prefix Xor
* Difficulty: Medium
* 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[] findArray(int[] pref) {
}
}
```

Python3 Solution:

```
"""
Problem: Find The Original Array of Prefix Xor
Difficulty: Medium
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 findArray(self, pref: List[int]) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find The Original Array of Prefix Xor
* Difficulty: Medium
* 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[]} pref
* @return {number[]}
*/
var findArray = function(pref) {

};
```

TypeScript Solution:

```
/**
* Problem: Find The Original Array of Prefix Xor
* Difficulty: Medium
* 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 findArray(pref: number[]): number[] {

};
```

C# Solution:

```
/*
* Problem: Find The Original Array of Prefix Xor
* Difficulty: Medium
* 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[] FindArray(int[] pref) {

}
}
```

C Solution:

```
/*
* Problem: Find The Original Array of Prefix Xor
* Difficulty: Medium
* 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* findArray(int* pref, int prefSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Find The Original Array of Prefix Xor
// Difficulty: Medium
// 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 findArray(pref []int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun findArray(pref: IntArray): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func findArray(_ pref: [Int]) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Find The Original Array of Prefix Xor
// Difficulty: Medium
// 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_array(pref: Vec<i32>) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer[]} pref
# @return {Integer[]}
def find_array(pref)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $pref
* @return Integer[]
*/
function findArray($pref) {

}
}
```

Dart Solution:

```
class Solution {
List<int> findArray(List<int> pref) {

}
}
```

Scala Solution:

```
object Solution {
def findArray(pref: Array[Int]): Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec find_array(pref :: [integer]) :: [integer]
def find_array(pref) do

end
end
```

Erlang Solution:

```
-spec find_array(Pref :: [integer()]) -> [integer()].
find_array(Pref) ->
.
```

Racket Solution:

```
(define/contract (find-array pref)
(-> (listof exact-integer?) (listof exact-integer?))
)
```