Problem 1720: Decode XORed Array

Problem Information

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

Problem Description

There is a

hidden

integer array

arr

that consists of

n

non-negative integers.

It was encoded into another integer array

encoded

of length

n - 1

, such that

encoded[i] = arr[i] XOR arr[i + 1]

. For example, if

arr = [1,0,2,1]

, then

encoded = [1,2,3]

.

You are given the

encoded

array. You are also given an integer

first

, that is the first element of

arr

, i.e.

arr[0]

.

Return

the original array

arr

. It can be proved that the answer exists and is unique.

Example 1:

Input:

encoded = [1,2,3], first = 1

Output:

[1,0,2,1]

Explanation:

If arr = [1,0,2,1], then first = 1 and encoded = [1 XOR 0, 0 XOR 2, 2 XOR 1] = [1,2,3]

Example 2:

Input:

encoded = [6,2,7,3], first = 4

Output:

[4,2,0,7,4]

Constraints:

2 <= n <= 10

4

encoded.length == n - 1

0 <= encoded[i] <= 10

5

0 <= first <= 10

5

Code Snippets

C++:

```
class Solution {
public:
vector<int> decode(vector<int>& encoded, int first) {

}
};
```

Java:

```
class Solution {
public int[] decode(int[] encoded, int first) {

}
}
```

Python3:

```
class Solution:
def decode(self, encoded: List[int], first: int) -> List[int]:
```

Python:

```
class Solution(object):
def decode(self, encoded, first):
"""
:type encoded: List[int]
:type first: int
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {number[]} encoded
* @param {number} first
* @return {number[]}
*/
var decode = function(encoded, first) {

};
```

TypeScript:

```
function decode(encoded: number[], first: number): number[] {

};
```

C#:

```
public class Solution {
public int[] Decode(int[] encoded, int first) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* decode(int* encoded, int encodedSize, int first, int* returnSize) {

}
```

Go:

```
func decode(encoded []int, first int) []int {

}
```

Kotlin:

```
class Solution {
fun decode(encoded: IntArray, first: Int): IntArray {

}
}
```

Swift:

```
class Solution {
func decode(_ encoded: [Int], _ first: Int) -> [Int] {

}
```

```
}
```

Rust:

```
impl Solution {
pub fn decode(encoded: Vec<i32>, first: i32) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer[]} encoded
# @param {Integer} first
# @return {Integer[]}
def decode(encoded, first)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $encoded
* @param Integer $first
* @return Integer[]
*/
function decode($encoded, $first) {

}
}
```

Dart:

```
class Solution {
List<int> decode(List<int> encoded, int first) {

}
}
```

Scala:

```
object Solution {
def decode(encoded: Array[Int], first: Int): Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec decode(encoded :: [integer], first :: integer) :: [integer]
def decode(encoded, first) do

end
end
```

Erlang:

```
-spec decode(Encoded :: [integer()], First :: integer()) -> [integer()].
decode(Encoded, First) ->
.
```

Racket:

```
(define/contract (decode encoded first)
(-> (listof exact-integer?) exact-integer? (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Decode XORed 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> decode(vector<int>& encoded, int first) {

}
};
```

Java Solution:

```
/**
* Problem: Decode XORed 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[] decode(int[] encoded, int first) {

}
}
```

Python3 Solution:

```
"""
Problem: Decode XORed 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 decode(self, encoded: List[int], first: int) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def decode(self, encoded, first):
"""
:type encoded: List[int]
:type first: int
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Decode XORed 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[]} encoded
* @param {number} first
* @return {number[]}
*/
var decode = function(encoded, first) {

};
```

TypeScript Solution:

```
/**
* Problem: Decode XORed 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 decode(encoded: number[], first: number): number[] {

};
```

C# Solution:

```
/*
* Problem: Decode XORed 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[] Decode(int[] encoded, int first) {

}
}
```

C Solution:

```
/*
* Problem: Decode XORed 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* decode(int* encoded, int encodedSize, int first, int* returnSize) {

}
```

Go Solution:

```
// Problem: Decode XORed 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 decode(encoded []int, first int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun decode(encoded: IntArray, first: Int): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func decode(_ encoded: [Int], _ first: Int) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Decode XORed 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 decode(encoded: Vec<i32>, first: i32) -> Vec<i32> {
```

```
}
}
```

Ruby Solution:

```
# @param {Integer[]} encoded
# @param {Integer} first
# @return {Integer[]}
def decode(encoded, first)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $encoded
* @param Integer $first
* @return Integer[]
*/
function decode($encoded, $first) {

}
}
```

Dart Solution:

```
class Solution {
List<int> decode(List<int> encoded, int first) {

}
}
```

Scala Solution:

```
object Solution {
def decode(encoded: Array[Int], first: Int): Array[Int] = {

}
```

```
}
```

Elixir Solution:

```
defmodule Solution do
@spec decode(encoded :: [integer], first :: integer) :: [integer]
def decode(encoded, first) do

end
end
```

Erlang Solution:

```
-spec decode(Encoded :: [integer()], First :: integer()) -> [integer()].
decode(Encoded, First) ->
.
```

Racket Solution:

```
(define/contract (decode encoded first)
(-> (listof exact-integer?) exact-integer? (listof exact-integer?))
)
```