Problem 1734: Decode XORed Permutation

Problem Information

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

Problem Description

There is an integer array

perm

that is a permutation of the first

n

positive integers, where

n

is always

odd

.

It was encoded into another integer array

encoded

of length

n - 1

, such that

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

. For example, if

perm = [1,3,2]

, then

encoded = [2,1]

.

Given the

encoded

array, return

the original array

perm

. It is guaranteed that the answer exists and is unique.

Example 1:

Input:

encoded = [3,1]

Output:

[1,2,3]

Explanation:

If perm = [1,2,3], then encoded = [1 XOR 2,2 XOR 3] = [3,1]

Example 2:

Input:

encoded = [6,5,4,6]

Output:

[2,4,1,5,3]

Constraints:

3 <= n < 10

5

n

is odd.

encoded.length == n - 1

Code Snippets

C++:

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

}
};
```

Java:

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

}
```

```
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

```
class Solution {
fun decode(encoded: IntArray): IntArray {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

```
end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

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

}
};
```

Java Solution:

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

}
}
```

Python3 Solution:

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

Python Solution:

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

JavaScript Solution:

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

};
```

TypeScript Solution:

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

};
```

C# Solution:

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

}
}
```

C Solution:

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

}
```

Go Solution:

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

}
```

Kotlin Solution:

```
class Solution {
fun decode(encoded: IntArray): IntArray {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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