Problem 3287: Find the Maximum Sequence Value of Array

Problem Information

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

Problem Description

You are given an integer array

nums

and a

positive

integer

k

.

The

value

of a sequence

seq

of size

2 * x

is defined as:

(seq[0] OR seq[1] OR ... OR seq[x - 1]) XOR (seq[x] OR seq[x + 1] OR ... OR seq[2 * x - 1])

.

Return the

maximum

value

of any

subsequence

of

nums

having size

2 * k

.

Example 1:

Input:

nums = [2,6,7], k = 1

Output:

5

Explanation:

The subsequence

[2, 7]

has the maximum value of

2 XOR 7 = 5

.

Example 2:

Input:

nums = [4,2,5,6,7], k = 2

Output:

2

Explanation:

The subsequence

[4, 5, 6, 7]

has the maximum value of

(4 OR 5) XOR (6 OR 7) = 2

.

Constraints:

2 <= nums.length <= 400

1 <= nums[i] < 2

7

1 <= k <= nums.length / 2

Code Snippets

C++:

```
class Solution {
public:
int maxValue(vector<int>& nums, int k) {

}
};
```

Java:

```
class Solution {
public int maxValue(int[] nums, int k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[]} nums
* @param {number} k
```

```
 * @return {number}
 */
var maxValue = function(nums, k) {

};
```

TypeScript:

```
function maxValue(nums: number[], k: number): number {

};
```

C#:

```
public class Solution {
public int MaxValue(int[] nums, int k) {

}
}
```

C:

```
int maxValue(int* nums, int numsSize, int k) {

}
```

Go:

```
func maxValue(nums []int, k int) int {

}
```

Kotlin:

```
class Solution {
fun maxValue(nums: IntArray, k: Int): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer} k
# @return {Integer}
def max_value(nums, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $k
* @return Integer
*/
function maxValue($nums, $k) {

}
}
```

Dart:

```
class Solution {
int maxValue(List<int> nums, int k) {

}
```

```
}
```

Scala:

```
object Solution {
def maxValue(nums: Array[Int], k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_value(nums :: [integer], k :: integer) :: integer
def max_value(nums, k) do

end
end
```

Erlang:

```
-spec max_value(Nums :: [integer()], K :: integer()) -> integer().
max_value(Nums, K) ->
.
```

Racket:

```
(define/contract (max-value nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Maximum Sequence Value of Array
* Difficulty: Hard
* Tags: array, dp
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int maxValue(vector<int>& nums, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Maximum Sequence Value of Array
* Difficulty: Hard
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int maxValue(int[] nums, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Maximum Sequence Value of Array
Difficulty: Hard
Tags: array, dp

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""
```

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Maximum Sequence Value of Array
* Difficulty: Hard
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {number[]} nums
* @param {number} k
* @return {number}
*/
var maxValue = function(nums, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Maximum Sequence Value of Array
```

```
* Difficulty: Hard
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function maxValue(nums: number[], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Find the Maximum Sequence Value of Array
* Difficulty: Hard
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int MaxValue(int[] nums, int k) {

}
}
```

C Solution:

```
/*
* Problem: Find the Maximum Sequence Value of Array
* Difficulty: Hard
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/
```

```
int maxValue(int* nums, int numsSize, int k) {

}
```

Go Solution:

```
// Problem: Find the Maximum Sequence Value of Array
// Difficulty: Hard
// Tags: array, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func maxValue(nums []int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxValue(nums: IntArray, k: Int): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find the Maximum Sequence Value of Array
// Difficulty: Hard
// Tags: array, dp
//
```

```
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn max_value(nums: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer} k
# @return {Integer}
def max_value(nums, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $k
* @return Integer
*/
function maxValue($nums, $k) {

}
}
```

Dart Solution:

```
class Solution {
int maxValue(List<int> nums, int k) {

}
}
```

Scala Solution:

```
object Solution {
def maxValue(nums: Array[Int], k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_value(nums :: [integer], k :: integer) :: integer
def max_value(nums, k) do

end
end
```

Erlang Solution:

```
-spec max_value(Nums :: [integer()], K :: integer()) -> integer().
max_value(Nums, K) ->
.
```

Racket Solution:

```
(define/contract (max-value nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```