Problem 2859: Sum of Values at Indices With K Set Bits

Problem Information

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

Problem Description

You are given a

0-indexed

integer array

nums

and an integer

k

.

Return

an integer that denotes the

sum

of elements in

nums

whose corresponding

indices

have

exactly

k

set bits in their binary representation.

The

set bits

in an integer are the

1

's present when it is written in binary.

For example, the binary representation of

21

is

10101

, which has

3

set bits.

Example 1:

Input:

nums = [5,10,1,5,2], k = 1

Output:

13

Explanation:

The binary representation of the indices are: 0 = 000

2

1 = 001

2

2 = 010

2

3 = 011

2

4 = 100

2

Indices 1, 2, and 4 have k = 1 set bits in their binary representation. Hence, the answer is nums[1] + nums[2] + nums[4] = 13.

Example 2:

Input:

nums = [4,3,2,1], k = 2

Output:

1

Explanation:

The binary representation of the indices are: 0 = 00

2

1 = 01

2

2 = 10

2

3 = 11

2

Only index 3 has k = 2 set bits in its binary representation. Hence, the answer is nums[3] = 1.

Constraints:

1 <= nums.length <= 1000

1 <= nums[i] <= 10

5

0 <= k <= 10

Code Snippets

C++:

```
class Solution {
public:
```

```
int sumIndicesWithKSetBits(vector<int>& nums, int k) {

}
};
```

Java:

```
class Solution {
public int sumIndicesWithKSetBits(List<Integer> nums, int k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public int SumIndicesWithKSetBits(IList<int> nums, int k) {

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

```
class Solution {
fun sumIndicesWithKSetBits(nums: List<Int>, k: Int): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

```
object Solution {
def sumIndicesWithKSetBits(nums: List[Int], k: Int): Int = {

}
```

```
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (sum-indices-with-k-set-bits nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Sum of Values at Indices With K Set Bits
* 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 sumIndicesWithKSetBits(vector<int>& nums, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Sum of Values at Indices With K Set Bits
* 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 sumIndicesWithKSetBits(List<Integer> nums, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Sum of Values at Indices With K Set Bits
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 sumIndicesWithKSetBits(self, nums: List[int], k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Sum of Values at Indices With K Set Bits
* 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[]} nums
* @param {number} k
* @return {number}
*/
var sumIndicesWithKSetBits = function(nums, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Sum of Values at Indices With K Set Bits
* 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 sumIndicesWithKSetBits(nums: number[], k: number): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Sum of Values at Indices With K Set Bits
* 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 SumIndicesWithKSetBits(IList<int> nums, int k) {

}
}
```

C Solution:

```
/*
* Problem: Sum of Values at Indices With K Set Bits
* 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
*/

int sumIndicesWithKSetBits(int* nums, int numsSize, int k) {

}
```

Go Solution:

```
// Problem: Sum of Values at Indices With K Set Bits
// 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 sumIndicesWithKSetBits(nums []int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun sumIndicesWithKSetBits(nums: List<Int>, k: Int): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Sum of Values at Indices With K Set Bits
// 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 sum_indices_with_k_set_bits(nums: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

```
object Solution {
def sumIndicesWithKSetBits(nums: List[Int], k: Int): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (sum-indices-with-k-set-bits nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```