# Problem 2638: Count the Number of K-Free Subsets

## Problem Information

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

## Problem Description

You are given an integer array

nums

, which contains

distinct

elements and an integer

k

.

A subset is called a

k-Free

subset if it contains

no

two elements with an absolute difference equal to

k

. Notice that the empty set is a

k-Free

subset.

Return

the number of

k-Free

subsets of

nums

.

A

subset

of an array is a selection of elements (possibly none) of the array.

Example 1:

Input:

nums = [5,4,6], k = 1

Output:

5

Explanation:

There are 5 valid subsets: {}, {5}, {4}, {6} and {4, 6}.

Example 2:

Input:

nums = [2,3,5,8], k = 5

Output:

12

Explanation:

There are 12 valid subsets: {}, {2}, {3}, {5}, {8}, {2, 3}, {2, 3, 5}, {2, 5}, {2, 5, 8}, {2, 8}, {3, 5} and {5, 8}.

Example 3:

Input:

nums = [10,5,9,11], k = 20

Output:

16

Explanation:

All subsets are valid. Since the total count of subsets is 2

4

= 16, so the answer is 16.

Constraints:

1 <= nums.length <= 50

1 <= nums[i] <= 1000

1 <= k <= 1000

## Code Snippets

**C++:**

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

}
};
```

**Java:**

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

}
}
```

**Python3:**

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

**Python:**

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

**JavaScript:**

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

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

};
```

**TypeScript:**

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

};
```

**C#:**

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

}
}
```

**C:**

```
long long countTheNumOfKFreeSubsets(int* nums, int numsSize, int k) {

}
```

**Go:**

```
func countTheNumOfKFreeSubsets(nums []int, k int) int64 {

}
```

**Kotlin:**

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

}
}
```

**Swift:**

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

}
}
```

**Rust:**

```
impl Solution {
pub fn count_the_num_of_k_free_subsets(nums: Vec<i32>, k: i32) -> i64 {

}
}
```

**Ruby:**

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

end
```

**PHP:**

```
class Solution {

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

}
}
```

**Dart:**

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

}
```

```
}
```

**Scala:**

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

}
}
```

**Elixir:**

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

end
end
```

**Erlang:**

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

**Racket:**

```
(define/contract (count-the-num-of-k-free-subsets nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

## Solutions

**C++ Solution:**

```
/*
* Problem: Count the Number of K-Free Subsets
* Difficulty: Medium
```

```
* Tags: array, dp, math, sort
*
* 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:
long long countTheNumOfKFreeSubsets(vector<int>& nums, int k) {

}
};
```

### Java Solution:

```
/**
* Problem: Count the Number of K-Free Subsets
* Difficulty: Medium
* Tags: array, dp, math, sort
*
* 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 long countTheNumOfKFreeSubsets(int[] nums, int k) {

}
}
```

### Python3 Solution:

```
"""
Problem: Count the Number of K-Free Subsets
Difficulty: Medium
Tags: array, dp, math, sort

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

**Python Solution:**

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

**JavaScript Solution:**

```
/**
* Problem: Count the Number of K-Free Subsets
* Difficulty: Medium
* Tags: array, dp, math, sort
*
* 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 countTheNumOfKFreeSubsets = function(nums, k) {

};
```

**TypeScript Solution:**

```
/**
* Problem: Count the Number of K-Free Subsets
* Difficulty: Medium
* Tags: array, dp, math, sort
*
* 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 countTheNumOfKFreeSubsets(nums: number[], k: number): number {

};
```

### C# Solution:

```
/*
* Problem: Count the Number of K-Free Subsets
* Difficulty: Medium
* Tags: array, dp, math, sort
*
* 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 long CountTheNumOfKFreeSubsets(int[] nums, int k) {

}
}
```

### C Solution:

```
/*
* Problem: Count the Number of K-Free Subsets
* Difficulty: Medium
* Tags: array, dp, math, sort
*
* 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
```

```
*/

long long countTheNumOfKFreeSubsets(int* nums, int numsSize, int k) {

}
```

### Go Solution:

```
// Problem: Count the Number of K-Free Subsets
// Difficulty: Medium
// Tags: array, dp, math, sort
//
// 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 countTheNumOfKFreeSubsets(nums []int, k int) int64 {

}
```

### Kotlin Solution:

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

}
}
```

### Swift Solution:

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

}
}
```

### Rust Solution:

```
// Problem: Count the Number of K-Free Subsets
// Difficulty: Medium
// Tags: array, dp, math, sort
```

```
//
// 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 count_the_num_of_k_free_subsets(nums: Vec<i32>, k: i32) -> i64 {

}
}
```

### Ruby Solution:

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

end
```

### PHP Solution:

```
class Solution {

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

}
}
```

### Dart Solution:

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

}
}
```

### Scala Solution:

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

}
}
```

### Elixir Solution:

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

end
end
```

### Erlang Solution:

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

### Racket Solution:

```
(define/contract (count-the-num-of-k-free-subsets nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```