Problem 1481: Least Number of Unique Integers after K Removals

Problem Information

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

Problem Description

Given an array of integers

arr

and an integer

k

. Find the

least number of unique integers

after removing

exactly

k

elements

.

Example 1:

Input:

arr = [5,5,4], k = 1

Output:

1

Explanation

: Remove the single 4, only 5 is left.

Example 2:

Input:

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

Output:

2

Explanation

: Remove 4, 2 and either one of the two 1s or three 3s. 1 and 3 will be left.

Constraints:

1 <= arr.length <= 10^5

1 <= arr[i] <= 10^9

0 <= k <= arr.length

Code Snippets

C++:

```
class Solution {
public:
```

```
int findLeastNumOfUniqueInts(vector<int>& arr, int k) {

}
};
```

Java:

```
class Solution {
public int findLeastNumOfUniqueInts(int[] arr, int k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function findLeastNumOfUniqueInts(arr: number[], k: number): number {

};
```

C#:

```
public class Solution {
public int FindLeastNumOfUniqueInts(int[] arr, int k) {

}
}
```

C:

```
int findLeastNumOfUniqueInts(int* arr, int arrSize, int k){

}
```

Go:

```
func findLeastNumOfUniqueInts(arr []int, k int) int {

}
```

Kotlin:

```
class Solution {
fun findLeastNumOfUniqueInts(arr: IntArray, k: Int): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

```
# @param {Integer[]} arr
# @param {Integer} k
# @return {Integer}
def find_least_num_of_unique_ints(arr, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $arr
* @param Integer $k
* @return Integer
*/
function findLeastNumOfUniqueInts($arr, $k) {

}
}
```

Scala:

```
object Solution {
def findLeastNumOfUniqueInts(arr: Array[Int], k: Int): Int = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Least Number of Unique Integers after K Removals
* Difficulty: Medium
* Tags: array, greedy, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int findLeastNumOfUniqueInts(vector<int>& arr, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Least Number of Unique Integers after K Removals
* Difficulty: Medium
* Tags: array, greedy, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int findLeastNumOfUniqueInts(int[] arr, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Least Number of Unique Integers after K Removals
Difficulty: Medium
Tags: array, greedy, hash, sort
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Least Number of Unique Integers after K Removals
* Difficulty: Medium
* Tags: array, greedy, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Least Number of Unique Integers after K Removals
* Difficulty: Medium
* Tags: array, greedy, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function findLeastNumOfUniqueInts(arr: number[], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Least Number of Unique Integers after K Removals
* Difficulty: Medium
* Tags: array, greedy, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int FindLeastNumOfUniqueInts(int[] arr, int k) {

}
}
```

C Solution:

```
/*
* Problem: Least Number of Unique Integers after K Removals
* Difficulty: Medium
* Tags: array, greedy, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
```

```
* Space Complexity: O(n) for hash map
*/

int findLeastNumOfUniqueInts(int* arr, int arrSize, int k){

}
```

Go Solution:

```
// Problem: Least Number of Unique Integers after K Removals
// Difficulty: Medium
// Tags: array, greedy, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func findLeastNumOfUniqueInts(arr []int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun findLeastNumOfUniqueInts(arr: IntArray, k: Int): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Least Number of Unique Integers after K Removals
// Difficulty: Medium
```

```
// Tags: array, greedy, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn find_least_num_of_unique_ints(arr: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} arr
# @param {Integer} k
# @return {Integer}
def find_least_num_of_unique_ints(arr, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @param Integer $k
* @return Integer
*/
function findLeastNumOfUniqueInts($arr, $k) {

}
}
```

Scala Solution:

```
object Solution {
def findLeastNumOfUniqueInts(arr: Array[Int], k: Int): Int = {

}
```

```
}
```