Problem 1338: Reduce Array Size to The Half

Problem Information

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

Problem Description

You are given an integer array

arr

. You can choose a set of integers and remove all the occurrences of these integers in the array.

Return

the minimum size of the set so that

at least

half of the integers of the array are removed

.

Example 1:

Input:

arr = [3,3,3,3,5,5,5,2,2,7]

Output:

2

Explanation:

Choosing {3,7} will make the new array [5,5,5,2,2] which has size 5 (i.e equal to half of the size of the old array). Possible sets of size 2 are {3,5},{3,2},{5,2}. Choosing set {2,7} is not possible as it will make the new array [3,3,3,3,5,5,5] which has a size greater than half of the size of the old array.

Example 2:

Input:

arr = [7,7,7,7,7,7]

Output:

1

Explanation:

The only possible set you can choose is {7}. This will make the new array empty.

Constraints:

2 <= arr.length <= 10

5

arr.length

is even.

1 <= arr[i] <= 10

5

Code Snippets

C++:

```
class Solution {
public:
int minSetSize(vector<int>& arr) {

}
};
```

Java:

```
class Solution {
public int minSetSize(int[] arr) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function minSetSize(arr: number[]): number {

};
```

C#:

```
public class Solution {
public int MinSetSize(int[] arr) {

}
}
```

C:

```
int minSetSize(int* arr, int arrSize) {

}
```

Go:

```
func minSetSize(arr []int) int {

}
```

Kotlin:

```
class Solution {
fun minSetSize(arr: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func minSetSize(_ arr: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_set_size(arr: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} arr
# @return {Integer}
def min_set_size(arr)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $arr
* @return Integer
*/
function minSetSize($arr) {

}
}
```

Dart:

```
class Solution {
int minSetSize(List<int> arr) {

}
}
```

Scala:

```
object Solution {
def minSetSize(arr: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_set_size(arr :: [integer]) :: integer
def min_set_size(arr) do

end
end
```

Erlang:

```
-spec min_set_size(Arr :: [integer()]) -> integer().
min_set_size(Arr) ->
.
```

Racket:

```
(define/contract (min-set-size arr)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Reduce Array Size to The Half
* Difficulty: Medium
* Tags: array, greedy, hash, sort, queue, heap
*
* 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 minSetSize(vector<int>& arr) {

}
};
```

Java Solution:

```
/**
* Problem: Reduce Array Size to The Half
* Difficulty: Medium
* Tags: array, greedy, hash, sort, queue, heap
*
* 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 minSetSize(int[] arr) {

}
}
```

Python3 Solution:

```
"""
Problem: Reduce Array Size to The Half
Difficulty: Medium
Tags: array, greedy, hash, sort, queue, heap

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 minSetSize(self, arr: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

```
"""
```

JavaScript Solution:

```
/**
* Problem: Reduce Array Size to The Half
* Difficulty: Medium
* Tags: array, greedy, hash, sort, queue, heap
*
* 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
* @return {number}
*/
var minSetSize = function(arr) {

};
```

TypeScript Solution:

```
/**
* Problem: Reduce Array Size to The Half
* Difficulty: Medium
* Tags: array, greedy, hash, sort, queue, heap
*
* 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 minSetSize(arr: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Reduce Array Size to The Half
* Difficulty: Medium
* Tags: array, greedy, hash, sort, queue, heap
*
* 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 MinSetSize(int[] arr) {

}
}
```

C Solution:

```
/*
* Problem: Reduce Array Size to The Half
* Difficulty: Medium
* Tags: array, greedy, hash, sort, queue, heap
*
* 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 minSetSize(int* arr, int arrSize) {

}
```

Go Solution:

```
// Problem: Reduce Array Size to The Half
// Difficulty: Medium
// Tags: array, greedy, hash, sort, queue, heap
//
// 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 minSetSize(arr []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minSetSize(arr: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func minSetSize(_ arr: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Reduce Array Size to The Half
// Difficulty: Medium
// Tags: array, greedy, hash, sort, queue, heap
//
// 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 min_set_size(arr: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} arr
# @return {Integer}
def min_set_size(arr)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @return Integer
*/
function minSetSize($arr) {

}
}
```

Dart Solution:

```
class Solution {
int minSetSize(List<int> arr) {

}
}
```

Scala Solution:

```
object Solution {
def minSetSize(arr: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_set_size(arr :: [integer]) :: integer
def min_set_size(arr) do

end
end
```

Erlang Solution:

```
-spec min_set_size(Arr :: [integer()]) -> integer().
min_set_size(Arr) ->
.
```

Racket Solution:

```
(define/contract (min-set-size arr)
(-> (listof exact-integer?) exact-integer?)
)
```