Problem 3663: Find The Least Frequent Digit

Problem Information

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

Problem Description

Given an integer

n

, find the digit that occurs

least

frequently in its decimal representation. If multiple digits have the same frequency, choose the

smallest

digit.

Return the chosen digit as an integer.

The

frequency

of a digit

x

is the number of times it appears in the decimal representation of n

.

Example 1:

Input:

n = 1553322

Output:

1

Explanation:

The least frequent digit in

n

is 1, which appears only once. All other digits appear twice.

Example 2:

Input:

n = 723344511

Output:

2

Explanation:

The least frequent digits in

n

are 7, 2, and 5; each appears only once.

Constraints:

1 <= n <= 2

31

- 1

Code Snippets

C++:

```
class Solution {
public:
int getLeastFrequentDigit(int n) {

}
};
```

Java:

```
class Solution {
public int getLeastFrequentDigit(int n) {

}
}
```

Python3:

```
class Solution:
def getLeastFrequentDigit(self, n: int) -> int:
```

Python:

```
class Solution(object):
def getLeastFrequentDigit(self, n):
"""
:type n: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @return {number}
*/
var getLeastFrequentDigit = function(n) {

};
```

TypeScript:

```
function getLeastFrequentDigit(n: number): number {

};
```

C#:

```
public class Solution {
public int GetLeastFrequentDigit(int n) {

}
}
```

C:

```
int getLeastFrequentDigit(int n) {

}
```

Go:

```
func getLeastFrequentDigit(n int) int {

}
```

Kotlin:

```
class Solution {
fun getLeastFrequentDigit(n: Int): Int {

}
}
```

Swift:

```
class Solution {
func getLeastFrequentDigit(_ n: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn get_least_frequent_digit(n: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @return {Integer}
def get_least_frequent_digit(n)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function getLeastFrequentDigit($n) {

}
}
```

Dart:

```
class Solution {
int getLeastFrequentDigit(int n) {

}
}
```

Scala:

```
object Solution {
def getLeastFrequentDigit(n: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec get_least_frequent_digit(n :: integer) :: integer
def get_least_frequent_digit(n) do

end
end
```

Erlang:

```
-spec get_least_frequent_digit(N :: integer()) -> integer().
get_least_frequent_digit(N) ->
.
```

Racket:

```
(define/contract (get-least-frequent-digit n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find The Least Frequent Digit
* Difficulty: Easy
* Tags: array, math, hash
*
```

```
* 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 getLeastFrequentDigit(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Find The Least Frequent Digit
* Difficulty: Easy
* Tags: array, math, hash
*
* 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 getLeastFrequentDigit(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Find The Least Frequent Digit
Difficulty: Easy
Tags: array, math, hash

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 getLeastFrequentDigit(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def getLeastFrequentDigit(self, n):
"""
:type n: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find The Least Frequent Digit
* Difficulty: Easy
* Tags: array, math, hash
*
* 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} n
* @return {number}
*/
var getLeastFrequentDigit = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Find The Least Frequent Digit
* Difficulty: Easy
* Tags: array, math, hash
```

```
*
* 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 getLeastFrequentDigit(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Find The Least Frequent Digit
* Difficulty: Easy
* Tags: array, math, hash
*
* 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 GetLeastFrequentDigit(int n) {

}
}
```

C Solution:

```
/*
* Problem: Find The Least Frequent Digit
* Difficulty: Easy
* Tags: array, math, hash
*
* 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 getLeastFrequentDigit(int n) {
}
```

Go Solution:

```
// Problem: Find The Least Frequent Digit
// Difficulty: Easy
// Tags: array, math, hash
//
// 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 getLeastFrequentDigit(n int) int {

}
```

Kotlin Solution:

```
class Solution {
fun getLeastFrequentDigit(n: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func getLeastFrequentDigit(_ n: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find The Least Frequent Digit
// Difficulty: Easy
// Tags: array, math, hash
//
// 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 get_least_frequent_digit(n: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @return {Integer}
def get_least_frequent_digit(n)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function getLeastFrequentDigit($n) {

}
}
```

Dart Solution:

```
class Solution {
int getLeastFrequentDigit(int n) {

}
}
```

Scala Solution:

```
object Solution {
def getLeastFrequentDigit(n: Int): Int = {
}
}
```

Elixir Solution:

```
defmodule Solution do
@spec get_least_frequent_digit(n :: integer) :: integer
def get_least_frequent_digit(n) do

end
end
```

Erlang Solution:

```
-spec get_least_frequent_digit(N :: integer()) -> integer().
get_least_frequent_digit(N) ->
.
```

Racket Solution:

```
(define/contract (get-least-frequent-digit n)
(-> exact-integer? exact-integer?)
)
```