Problem 2844: Minimum Operations to Make a Special Number

Problem Information

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

Problem Description

You are given a

0-indexed

string

num

representing a non-negative integer.

In one operation, you can pick any digit of

num

and delete it. Note that if you delete all the digits of

num

,

num

becomes

0

.

Return

the

minimum number of operations

required to make

num

special

.

An integer

x

is considered

special

if it is divisible by

25

.

Example 1:

Input:

num = "2245047"

Output:

2

Explanation:

Delete digits num[5] and num[6]. The resulting number is "22450" which is special since it is divisible by 25. It can be shown that 2 is the minimum number of operations required to get a special number.

Example 2:

Input:

num = "2908305"

Output:

3

Explanation:

Delete digits num[3], num[4], and num[6]. The resulting number is "2900" which is special since it is divisible by 25. It can be shown that 3 is the minimum number of operations required to get a special number.

Example 3:

Input:

num = "10"

Output:

1

Explanation:

Delete digit num[0]. The resulting number is "0" which is special since it is divisible by 25. It can be shown that 1 is the minimum number of operations required to get a special number.

Constraints:

1 <= num.length <= 100

num

only consists of digits

'0'

through

'9'

.

num

does not contain any leading zeros.

Code Snippets

C++:

```
class Solution {
public:
int minimumOperations(string num) {

}
};
```

Java:

```
class Solution {
public int minimumOperations(String num) {

}
}
```

Python3:

```
class Solution:
def minimumOperations(self, num: str) -> int:
```

Python:

```
class Solution(object):
def minimumOperations(self, num):
"""
:type num: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} num
* @return {number}
*/
var minimumOperations = function(num) {

};
```

TypeScript:

```
function minimumOperations(num: string): number {

};
```

C#:

```
public class Solution {
public int MinimumOperations(string num) {

}
}
```

C:

```
int minimumOperations(char* num) {

}
```

Go:

```
func minimumOperations(num string) int {

}
```

Kotlin:

```
class Solution {
fun minimumOperations(num: String): Int {

}
}
```

Swift:

```
class Solution {
func minimumOperations(_ num: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn minimum_operations(num: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} num
# @return {Integer}
def minimum_operations(num)

end
```

PHP:

```
class Solution {

/**
* @param String $num
* @return Integer
*/
function minimumOperations($num) {

}
}
```

Dart:

```
class Solution {
int minimumOperations(String num) {

}
}
```

Scala:

```
object Solution {
def minimumOperations(num: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec minimum_operations(num :: String.t) :: integer
def minimum_operations(num) do

end
end
```

Erlang:

```
-spec minimum_operations(Num :: unicode:unicode_binary()) -> integer().
minimum_operations(Num) ->
.
```

Racket:

```
(define/contract (minimum-operations num)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Operations to Make a Special Number
* Difficulty: Medium
* Tags: string, greedy, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int minimumOperations(string num) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Operations to Make a Special Number
* Difficulty: Medium
* Tags: string, greedy, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int minimumOperations(String num) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Operations to Make a Special Number
Difficulty: Medium
Tags: string, greedy, math

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def minimumOperations(self, num: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def minimumOperations(self, num):
"""
:type num: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Operations to Make a Special Number
* Difficulty: Medium
* Tags: string, greedy, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
```

```
* @param {string} num
* @return {number}
*/
var minimumOperations = function(num) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Operations to Make a Special Number
* Difficulty: Medium
* Tags: string, greedy, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function minimumOperations(num: string): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Operations to Make a Special Number
* Difficulty: Medium
* Tags: string, greedy, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int MinimumOperations(string num) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Operations to Make a Special Number
* Difficulty: Medium
* Tags: string, greedy, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int minimumOperations(char* num) {

}
```

Go Solution:

```
// Problem: Minimum Operations to Make a Special Number
// Difficulty: Medium
// Tags: string, greedy, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func minimumOperations(num string) int {

}
```

Kotlin Solution:

```
class Solution {
fun minimumOperations(num: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func minimumOperations(_ num: String) -> Int {
}
}
```

Rust Solution:

```
// Problem: Minimum Operations to Make a Special Number
// Difficulty: Medium
// Tags: string, greedy, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn minimum_operations(num: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} num
# @return {Integer}
def minimum_operations(num)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $num
* @return Integer
*/
function minimumOperations($num) {

}
}
```

Dart Solution:

```
class Solution {
int minimumOperations(String num) {

}
}
```

Scala Solution:

```
object Solution {
def minimumOperations(num: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec minimum_operations(num :: String.t) :: integer
def minimum_operations(num) do

end
end
```

Erlang Solution:

```
-spec minimum_operations(Num :: unicode:unicode_binary()) -> integer().
minimum_operations(Num) ->
.
```

Racket Solution:

```
(define/contract (minimum-operations num)
(-> string? exact-integer?)
)
```