Problem 2167: Minimum Time to Remove All Cars Containing Illegal Goods

Problem Information

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

Problem Description

You are given a

0-indexed

binary string

s

which represents a sequence of train cars.

s[i] = '0'

denotes that the

i

th

car does

not

contain illegal goods and

s[i] = '1'

denotes that the

i

th

car does contain illegal goods.

As the train conductor, you would like to get rid of all the cars containing illegal goods. You can do any of the following three operations

any

number of times:

Remove a train car from the

left

end (i.e., remove

s[0]

) which takes 1 unit of time.

Remove a train car from the

right

end (i.e., remove

s[s.length - 1]

) which takes 1 unit of time.

Remove a train car from

anywhere

in the sequence which takes 2 units of time.

Return

the

minimum

time to remove all the cars containing illegal goods

.

Note that an empty sequence of cars is considered to have no cars containing illegal goods.

Example 1:

Input:

s = "

11

00

1

0

1

"

Output:

5

Explanation:

One way to remove all the cars containing illegal goods from the sequence is to - remove a car from the left end 2 times. Time taken is 2 * 1 = 2. - remove a car from the right end. Time taken is 1. - remove the car containing illegal goods found in the middle. Time taken is 2. This obtains a total time of 2 + 1 + 2 = 5.

An alternative way is to - remove a car from the left end 2 times. Time taken is 2 * 1 = 2. - remove a car from the right end 3 times. Time taken is 3 * 1 = 3. This also obtains a total time of 2 + 3 = 5.

5 is the minimum time taken to remove all the cars containing illegal goods. There are no other ways to remove them with less time.

Example 2:

Input:

s = "00

1

0"

Output:

2

Explanation:

One way to remove all the cars containing illegal goods from the sequence is to - remove a car from the left end 3 times. Time taken is 3 * 1 = 3. This obtains a total time of 3.

Another way to remove all the cars containing illegal goods from the sequence is to - remove the car containing illegal goods found in the middle. Time taken is 2. This obtains a total time of 2.

Another way to remove all the cars containing illegal goods from the sequence is to - remove a car from the right end 2 times. Time taken is 2 * 1 = 2. This obtains a total time of 2.

2 is the minimum time taken to remove all the cars containing illegal goods. There are no other ways to remove them with less time.

Constraints:

1 <= s.length <= 2 * 10

5

s[i]

is either

'0'

or

'1'

.

Code Snippets

C++:

```
class Solution {
public:
int minimumTime(string s) {

}
};
```

Java:

```
class Solution {
public int minimumTime(String s) {

}
}
```

Python3:

```
class Solution:
def minimumTime(self, s: str) -> int:
```

Python:

```
class Solution(object):
def minimumTime(self, s):
"""
:type s: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {number}
*/
var minimumTime = function(s) {

};
```

TypeScript:

```
function minimumTime(s: string): number {

};
```

C#:

```
public class Solution {
public int MinimumTime(string s) {

}
}
```

C:

```
int minimumTime(char* s) {

}
```

Go:

```
func minimumTime(s string) int {

}
```

Kotlin:

```
class Solution {
fun minimumTime(s: String): Int {

}
}
```

Swift:

```
class Solution {
func minimumTime(_ s: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn minimum_time(s: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @return {Integer}
def minimum_time(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Integer
```

```
*/
function minimumTime($s) {

}
}
```

Dart:

```
class Solution {
int minimumTime(String s) {

}
}
```

Scala:

```
object Solution {
def minimumTime(s: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec minimum_time(s :: String.t) :: integer
def minimum_time(s) do

end
end
```

Erlang:

```
-spec minimum_time(S :: unicode:unicode_binary()) -> integer().
minimum_time(S) ->
.
```

Racket:

```
(define/contract (minimum-time s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Time to Remove All Cars Containing Illegal Goods
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int minimumTime(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Time to Remove All Cars Containing Illegal Goods
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int minimumTime(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Time to Remove All Cars Containing Illegal Goods
Difficulty: Hard
Tags: string, dp

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

```
class Solution(object):
def minimumTime(self, s):
"""
:type s: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Time to Remove All Cars Containing Illegal Goods
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {string} s
* @return {number}
*/
var minimumTime = function(s) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Minimum Time to Remove All Cars Containing Illegal Goods
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function minimumTime(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Time to Remove All Cars Containing Illegal Goods
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int MinimumTime(string s) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Time to Remove All Cars Containing Illegal Goods
* Difficulty: Hard
```

```
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int minimumTime(char* s) {

}
```

Go Solution:

```
// Problem: Minimum Time to Remove All Cars Containing Illegal Goods
// Difficulty: Hard
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func minimumTime(s string) int {

}
```

Kotlin Solution:

```
class Solution {
fun minimumTime(s: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func minimumTime(_ s: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Time to Remove All Cars Containing Illegal Goods
// Difficulty: Hard
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn minimum_time(s: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {Integer}
def minimum_time(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return Integer
*/
function minimumTime($s) {

}
}
```

Dart Solution:

```
class Solution {
int minimumTime(String s) {
```

```
}
}
```

Scala Solution:

```
object Solution {
def minimumTime(s: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec minimum_time(s :: String.t) :: integer
def minimum_time(s) do

end
end
```

Erlang Solution:

```
-spec minimum_time(S :: unicode:unicode_binary()) -> integer().
minimum_time(S) ->
.
```

Racket Solution:

```
(define/contract (minimum-time s)
(-> string? exact-integer?)
)
```