Problem 2222: Number of Ways to Select Buildings

Problem Information

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

Problem Description

You are given a

0-indexed

binary string

s

which represents the types of buildings along a street where:

s[i] = '0'

denotes that the

i

th

building is an office and

s[i] = '1'

denotes that the

i

th

building is a restaurant.

As a city official, you would like to

select

3 buildings for random inspection. However, to ensure variety,

no two consecutive

buildings out of the

selected

buildings can be of the same type.

For example, given

s = "0

0

1

1

0

1

"

, we cannot select the

1

st

,

3

rd

, and

5

th

buildings as that would form

"0

11

"

which is

not

allowed due to having two consecutive buildings of the same type.

Return

the

number of valid ways

to select 3 buildings.

Example 1:

Input:

s = "001101"

Output:

6

Explanation:

The following sets of indices selected are valid: - [0,2,4] from ''

0

0

1

1

0

1'' forms "010" - [0,3,4] from ''

0

01

10

1'' forms "010" - [1,2,4] from "0

01

1

0

1'' forms "010" - [1,3,4] from "0

0

1

10

1" forms "010" - [2,4,5] from "00

1

1

01

" forms "101" - [3,4,5] from "001

101

" forms "101" No other selection is valid. Thus, there are 6 total ways.

Example 2:

Input:

s = "11100"

Output:

0

Explanation:

It can be shown that there are no valid selections.

Constraints:

3 <= s.length <= 10

5

s[i]

is either

'0'

or

'1'

.

Code Snippets

C++:

```
class Solution {
public:
long long numberOfWays(string s) {

}
};
```

Java:

```
class Solution {
public long numberOfWays(String s) {

}
}
```

Python3:

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

Python:

```
class Solution(object):
def numberOfWays(self, s):
```

```
"""
:type s: str
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public long NumberOfWays(string s) {

}
}
```

C:

```
long long numberOfWays(char* s) {

}
```

Go:

```
func numberOfWays(s string) int64 {

}
```

Kotlin:

```
class Solution {
fun numberOfWays(s: String): Long {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn number_of_ways(s: String) -> i64 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

```
object Solution {
def numberOfWays(s: String): Long = {

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (number-of-ways s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Ways to Select Buildings
* Difficulty: Medium
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
long long numberOfWays(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Ways to Select Buildings
* Difficulty: Medium
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public long numberOfWays(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Ways to Select Buildings
Difficulty: Medium
Tags: array, string, tree, dp
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Number of Ways to Select Buildings
* Difficulty: Medium
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* 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 numberOfWays = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Ways to Select Buildings
* Difficulty: Medium
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function numberOfWays(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Number of Ways to Select Buildings
* Difficulty: Medium
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public long NumberOfWays(string s) {

}
}
```

C Solution:

```
/*
* Problem: Number of Ways to Select Buildings
* Difficulty: Medium
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

long long numberOfWays(char* s) {

}
```

Go Solution:

```
// Problem: Number of Ways to Select Buildings
// Difficulty: Medium
// Tags: array, string, tree, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func numberOfWays(s string) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfWays(s: String): Long {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Number of Ways to Select Buildings
// Difficulty: Medium
// Tags: array, string, tree, dp
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn number_of_ways(s: String) -> i64 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

```
object Solution {
def numberOfWays(s: String): Long = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (number-of-ways s)
(-> string? exact-integer?)
)
```