Problem 3119: Maximum Number of Potholes That Can Be Fixed

Problem Information

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

Problem Description

You are given a string

road

, consisting only of characters

"x"

and

"."

, where each

"x"

denotes a

pothole

and each

"."

denotes a smooth road, and an integer

budget

.

In one repair operation, you can repair

n

consecutive

potholes for a price of

n + 1

.

Return the

maximum

number of potholes that can be fixed such that the sum of the prices of all of the fixes

doesn't go over

the given budget.

Example 1:

Input:

road = "..", budget = 5

Output:

0

Explanation:

There are no potholes to be fixed.

Example 2:

Input:

road = "..xxxxx", budget = 4

Output:

3

Explanation:

We fix the first three potholes (they are consecutive). The budget needed for this task is

3 + 1 = 4

.

Example 3:

Input:

road = "x.x.xxx...x", budget = 14

Output:

6

Explanation:

We can fix all the potholes. The total cost would be

(1 + 1) + (1 + 1) + (3 + 1) + (1 + 1) = 10

which is within our budget of 14.

Constraints:

1 <= road.length <= 10

5

1 <= budget <= 10

5

+ 1

road

consists only of characters

'.'

and

'x'

.

Code Snippets

C++:

```
class Solution {
public:
int maxPotholes(string road, int budget) {

}
};
```

Java:

```
class Solution {
public int maxPotholes(String road, int budget) {

}
```

```
}
```

Python3:

```
class Solution:
def maxPotholes(self, road: str, budget: int) -> int:
```

Python:

```
class Solution(object):
def maxPotholes(self, road, budget):
"""
:type road: str
:type budget: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} road
* @param {number} budget
* @return {number}
*/
var maxPotholes = function(road, budget) {

};
```

TypeScript:

```
function maxPotholes(road: string, budget: number): number {

};
```

C#:

```
public class Solution {
public int MaxPotholes(string road, int budget) {

}
}
```

C:

```
int maxPotholes(char* road, int budget) {

}
```

Go:

```
func maxPotholes(road string, budget int) int {

}
```

Kotlin:

```
class Solution {
fun maxPotholes(road: String, budget: Int): Int {

}
}
```

Swift:

```
class Solution {
func maxPotholes(_ road: String, _ budget: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_potholes(road: String, budget: i32) -> i32 {

}
}
```

Ruby:

```
# @param {String} road
# @param {Integer} budget
# @return {Integer}
def max_potholes(road, budget)
```

```
end
```

PHP:

```
class Solution {

/**
* @param String $road
* @param Integer $budget
* @return Integer
*/
function maxPotholes($road, $budget) {

}
}
```

Dart:

```
class Solution {
int maxPotholes(String road, int budget) {

}
}
```

Scala:

```
object Solution {
def maxPotholes(road: String, budget: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_potholes(road :: String.t, budget :: integer) :: integer
def max_potholes(road, budget) do

end
end
```

Erlang:

```
-spec max_potholes(Road :: unicode:unicode_binary(), Budget :: integer()) ->
integer().
max_potholes(Road, Budget) ->
.
```

Racket:

```
(define/contract (max-potholes road budget)
(-> string? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Number of Potholes That Can Be Fixed
* Difficulty: Medium
* Tags: string, greedy, sort
*
* 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 maxPotholes(string road, int budget) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Number of Potholes That Can Be Fixed
* Difficulty: Medium
* Tags: string, greedy, sort
*
* 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 maxPotholes(String road, int budget) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Number of Potholes That Can Be Fixed
Difficulty: Medium
Tags: string, greedy, sort

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 maxPotholes(self, road: str, budget: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maxPotholes(self, road, budget):
"""
:type road: str
:type budget: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Number of Potholes That Can Be Fixed
* Difficulty: Medium
```

```
* Tags: string, greedy, sort
*
* 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} road
* @param {number} budget
* @return {number}
*/
var maxPotholes = function(road, budget) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Number of Potholes That Can Be Fixed
* Difficulty: Medium
* Tags: string, greedy, sort
*
* 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 maxPotholes(road: string, budget: number): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Number of Potholes That Can Be Fixed
* Difficulty: Medium
* Tags: string, greedy, sort
*
* 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 MaxPotholes(string road, int budget) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Number of Potholes That Can Be Fixed
* Difficulty: Medium
* Tags: string, greedy, sort
*
* 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 maxPotholes(char* road, int budget) {

}
```

Go Solution:

```
// Problem: Maximum Number of Potholes That Can Be Fixed
// Difficulty: Medium
// Tags: string, greedy, sort
//
// 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 maxPotholes(road string, budget int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxPotholes(road: String, budget: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func maxPotholes(_ road: String, _ budget: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Number of Potholes That Can Be Fixed
// Difficulty: Medium
// Tags: string, greedy, sort
//
// 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 max_potholes(road: String, budget: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} road
# @param {Integer} budget
# @return {Integer}
def max_potholes(road, budget)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $road
* @param Integer $budget
* @return Integer
*/
function maxPotholes($road, $budget) {

}
}
```

Dart Solution:

```
class Solution {
int maxPotholes(String road, int budget) {

}
}
```

Scala Solution:

```
object Solution {
def maxPotholes(road: String, budget: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_potholes(road :: String.t, budget :: integer) :: integer
def max_potholes(road, budget) do

end
end
```

Erlang Solution:

```
-spec max_potholes(Road :: unicode:unicode_binary(), Budget :: integer()) ->
integer().
max_potholes(Road, Budget) ->
```

```
.
```

Racket Solution:

```
(define/contract (max-potholes road budget)
(-> string? exact-integer? exact-integer?)
)
```