Problem 959: Regions Cut By Slashes

Problem Information

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

Problem Description

An

n x n

grid is composed of

1 x 1

squares where each

1 x 1

square consists of a

'/'

,

'\'

, or blank space

' '

. These characters divide the square into contiguous regions.

Given the grid

grid

represented as a string array, return

the number of regions

.

Note that backslash characters are escaped, so a

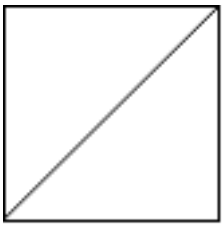
'\'

is represented as

'\\'

.

Example 1:



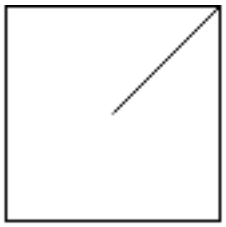
Input:

grid = [" /","/ "]

Output:

2

Example 2:



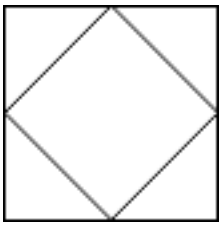
Input:

grid = [" /"," "]

Output:

1

Example 3:



Input:

grid = ["/\\","\\/"]

Output:

5

Explanation:

Recall that because \ characters are escaped, "\\/" refers to \/, and "/\\" refers to /\.

Constraints:

n == grid.length == grid[i].length

1 <= n <= 30

grid[i][j]

is either

'/'

,

'\'

, or

' '

.

Code Snippets

C++:

```
class Solution {
public:
int regionsBySlashes(vector<string>& grid) {

}
};
```

Java:

```
class Solution {
public int regionsBySlashes(String[] grid) {

}
}
```

Python3:

```
class Solution:
def regionsBySlashes(self, grid: List[str]) -> int:
```

Python:

```
class Solution(object):
def regionsBySlashes(self, grid):
"""
:type grid: List[str]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string[]} grid
* @return {number}
*/
var regionsBySlashes = function(grid) {

};
```

TypeScript:

```
function regionsBySlashes(grid: string[]): number {

};
```

C#:

```
public class Solution {
public int RegionsBySlashes(string[] grid) {

}
}
```

C:

```
int regionsBySlashes(char** grid, int gridSize) {

}
```

Go:

```
func regionsBySlashes(grid []string) int {
```

```
}
```

Kotlin:

```
class Solution {
fun regionsBySlashes(grid: Array<String>): Int {

}
}
```

Swift:

```
class Solution {
func regionsBySlashes(_ grid: [String]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn regions_by_slashes(grid: Vec<String>) -> i32 {

}
}
```

Ruby:

```
# @param {String[]} grid
# @return {Integer}
def regions_by_slashes(grid)

end
```

PHP:

```
class Solution {

/**
* @param String[] $grid
* @return Integer
*/
```

```
function regionsBySlashes($grid) {

}
}
```

Dart:

```
class Solution {
int regionsBySlashes(List<String> grid) {

}
}
```

Scala:

```
object Solution {
def regionsBySlashes(grid: Array[String]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec regions_by_slashes(grid :: [String.t]) :: integer
def regions_by_slashes(grid) do

end
end
```

Erlang:

```
-spec regions_by_slashes(Grid :: [unicode:unicode_binary()]) -> integer().
regions_by_slashes(Grid) ->
.
```

Racket:

```
(define/contract (regions-by-slashes grid)
(-> (listof string?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Regions Cut By Slashes
* Difficulty: Medium
* Tags: array, string, graph, hash, search
*
* 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 regionsBySlashes(vector<string>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Regions Cut By Slashes
* Difficulty: Medium
* Tags: array, string, graph, hash, search
*
* 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 regionsBySlashes(String[] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Regions Cut By Slashes
Difficulty: Medium
Tags: array, string, graph, hash, search

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 regionsBySlashes(self, grid: List[str]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def regionsBySlashes(self, grid):
"""
:type grid: List[str]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Regions Cut By Slashes
* Difficulty: Medium
* Tags: array, string, graph, hash, search
*
* 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 {string[]} grid
* @return {number}
*/
var regionsBySlashes = function(grid) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Regions Cut By Slashes
* Difficulty: Medium
* Tags: array, string, graph, hash, search
*
* 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 regionsBySlashes(grid: string[]): number {

};
```

C# Solution:

```
/*
* Problem: Regions Cut By Slashes
* Difficulty: Medium
* Tags: array, string, graph, hash, search
*
* 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 RegionsBySlashes(string[] grid) {

}
}
```

C Solution:

```
/*
* Problem: Regions Cut By Slashes
* Difficulty: Medium
```

```
* Tags: array, string, graph, hash, search
*
* 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 regionsBySlashes(char** grid, int gridSize) {

}
```

Go Solution:

```
// Problem: Regions Cut By Slashes
// Difficulty: Medium
// Tags: array, string, graph, hash, search
//
// 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 regionsBySlashes(grid []string) int {

}
```

Kotlin Solution:

```
class Solution {
fun regionsBySlashes(grid: Array<String>): Int {

}
}
```

Swift Solution:

```
class Solution {
func regionsBySlashes(_ grid: [String]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Regions Cut By Slashes
// Difficulty: Medium
// Tags: array, string, graph, hash, search
//
// 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 regions_by_slashes(grid: Vec<String>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String[]} grid
# @return {Integer}
def regions_by_slashes(grid)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $grid
* @return Integer
*/
function regionsBySlashes($grid) {

}
}
```

Dart Solution:

```
class Solution {
int regionsBySlashes(List<String> grid) {
```

```
}
}
```

Scala Solution:

```
object Solution {
def regionsBySlashes(grid: Array[String]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec regions_by_slashes(grid :: [String.t]) :: integer
def regions_by_slashes(grid) do

end
end
```

Erlang Solution:

```
-spec regions_by_slashes(Grid :: [unicode:unicode_binary()]) -> integer().
regions_by_slashes(Grid) ->
.
```

Racket Solution:

```
(define/contract (regions-by-slashes grid)
(-> (listof string?) exact-integer?)
)
```