Problem 3212: Count Submatrices With Equal Frequency of X and Y

Problem Information

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

Problem Description

Given a 2D character matrix

grid

, where

grid[i][j]

is either

'X'

,

'Y'

, or

'.'

, return the number of

submatrices

that contain:

grid[0][0]

an

equal

frequency of

'X'

and

'Y'

.

at least

one

'X'

.

Example 1:

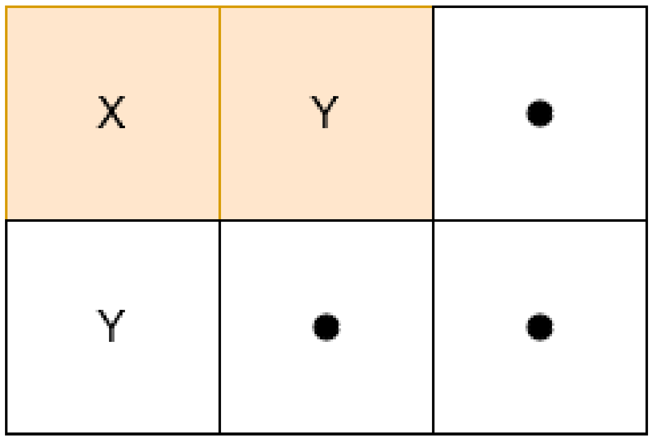
Input:

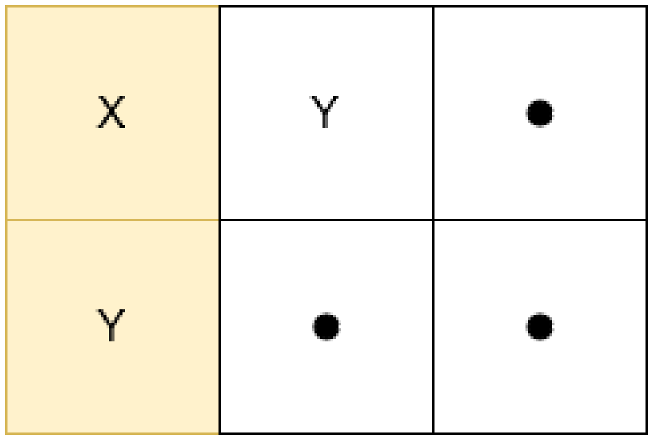
grid = [["X","Y","."],["Y",".","."]]

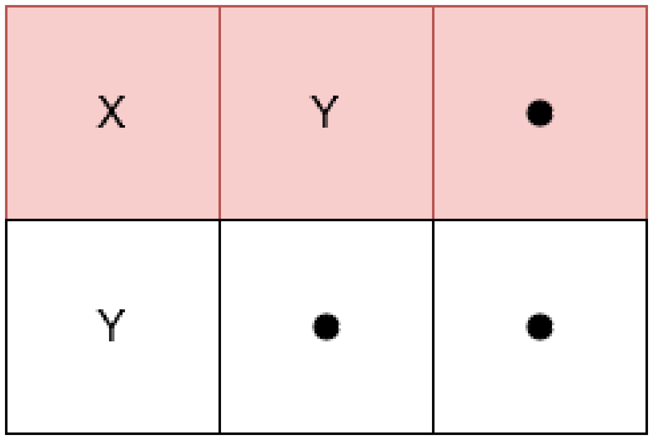
Output:

3

Explanation:







Example 2:

Input:

grid = [["X","X"],["X","Y"]]

Output:

0

Explanation:

No submatrix has an equal frequency of

'X'

and

'Y'

.

Example 3:

Input:

grid = [[".","."],[".","."]]

Output:

0

Explanation:

No submatrix has at least one

'X'

.

Constraints:

1 <= grid.length, grid[i].length <= 1000

grid[i][j]

is either

'X'

,

'Y'

, or

'.'

.

Code Snippets

C++:

```
class Solution {
public:
int numberOfSubmatrices(vector<vector<char>>& grid) {

}
};
```

Java:

```
class Solution {
public int numberOfSubmatrices(char[][] grid) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public int NumberOfSubmatrices(char[][] grid) {

}
}
```

C:

```
int numberOfSubmatrices(char** grid, int gridSize, int* gridColSize) {

}
```

Go:

```
func numberOfSubmatrices(grid [][]byte) int {
}
```

Kotlin:

```
class Solution {
fun numberOfSubmatrices(grid: Array<CharArray>): Int {

}
}
```

Swift:

```
class Solution {
func numberOfSubmatrices(_ grid: [[Character]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn number_of_submatrices(grid: Vec<Vec<char>>) -> i32 {

}
}
```

Ruby:

```
# @param {Character[][]} grid
# @return {Integer}
def number_of_submatrices(grid)

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

```
object Solution {
def numberOfSubmatrices(grid: Array[Array[Char]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_submatrices(grid :: [[char]]) :: integer
def number_of_submatrices(grid) do

end
end
```

Erlang:

```
-spec number_of_submatrices(Grid :: [[char()]]) -> integer().
number_of_submatrices(Grid) ->
.
```

Racket:

```
(define/contract (number-of-submatrices grid)
(-> (listof (listof char?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Submatrices With Equal Frequency of X and Y
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int numberOfSubmatrices(vector<vector<char>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Count Submatrices With Equal Frequency of X and Y
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int numberOfSubmatrices(char[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Submatrices With Equal Frequency of X and Y
Difficulty: Medium
Tags: array

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Submatrices With Equal Frequency of X and Y
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {character[][]} grid
* @return {number}
*/
var numberOfSubmatrices = function(grid) {
};
```

TypeScript Solution:

```
/**
* Problem: Count Submatrices With Equal Frequency of X and Y
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function numberOfSubmatrices(grid: string[][]): number {

};
```

C# Solution:

```
/*
* Problem: Count Submatrices With Equal Frequency of X and Y
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int NumberOfSubmatrices(char[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Count Submatrices With Equal Frequency of X and Y
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int numberOfSubmatrices(char** grid, int gridSize, int* gridColSize) {

}
```

Go Solution:

```
// Problem: Count Submatrices With Equal Frequency of X and Y
// Difficulty: Medium
// Tags: array
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func numberOfSubmatrices(grid [][]byte) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfSubmatrices(grid: Array<CharArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func numberOfSubmatrices(_ grid: [[Character]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count Submatrices With Equal Frequency of X and Y
// Difficulty: Medium
// Tags: array
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn number_of_submatrices(grid: Vec<Vec<char>>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Character[][]} grid
# @return {Integer}
def number_of_submatrices(grid)

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

Scala Solution:

```
object Solution {
def numberOfSubmatrices(grid: Array[Array[Char]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_submatrices(grid :: [[char]]) :: integer
def number_of_submatrices(grid) do

end
end
```

Erlang Solution:

```
-spec number_of_submatrices(Grid :: [[char()]]) -> integer().
number_of_submatrices(Grid) ->
.
```

Racket Solution:

```
(define/contract (number-of-submatrices grid)
(-> (listof (listof char?)) exact-integer?)
)
```