Problem 1139: Largest 1-Bordered Square

Problem Information

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

Problem Description

Given a 2D

grid

of

0

s and

1

s, return the number of elements in the largest

square

subgrid that has all

1

s on its

border

, or

0

if such a subgrid doesn't exist in the

grid

.

Example 1:

Input:

grid = [[1,1,1],[1,0,1],[1,1,1]]

Output:

9

Example 2:

Input:

grid = [[1,1,0,0]]

Output:

1

Constraints:

1 <= grid.length <= 100

1 <= grid[0].length <= 100

grid[i][j]

is

0

or

1

Code Snippets

C++:

```
class Solution {
public:
int largest1BorderedSquare(vector<vector<int>>& grid) {

}
};
```

Java:

```
class Solution {
public int largest1BorderedSquare(int[][] grid) {

}
}
```

Python3:

```
class Solution:
def largest1BorderedSquare(self, grid: List[List[int]]) -> int:
```

Python:

```
class Solution(object):
def largest1BorderedSquare(self, grid):
"""
:type grid: List[List[int]]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[][]} grid
* @return {number}
*/
var largest1BorderedSquare = function(grid) {

};
```

TypeScript:

```
function largest1BorderedSquare(grid: number[][]): number {

};
```

C#:

```
public class Solution {
public int Largest1BorderedSquare(int[][] grid) {

}
}
```

C:

```


int largest1BorderedSquare(int** grid, int gridSize, int* gridColSize){

}
```

Go:

```
func largest1BorderedSquare(grid [][]int) int {

}
```

Kotlin:

```
class Solution {
fun largest1BorderedSquare(grid: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func largest1BorderedSquare(_ grid: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn largest1_bordered_square(grid: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[][]} grid
# @return {Integer}
def largest1_bordered_square(grid)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $grid
* @return Integer
*/
function largest1BorderedSquare($grid) {

}
}
```

Scala:

```
object Solution {
def largest1BorderedSquare(grid: Array[Array[Int]]): Int = {

}

}
```

Solutions

C++ Solution:

```
/*
* Problem: Largest 1-Bordered Square
* Difficulty: Medium
* Tags: array, 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:
int largest1BorderedSquare(vector<vector<int>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Largest 1-Bordered Square
* Difficulty: Medium
* Tags: array, 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 int largest1BorderedSquare(int[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Largest 1-Bordered Square
Difficulty: Medium
Tags: array, 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 largest1BorderedSquare(self, grid: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def largest1BorderedSquare(self, grid):
"""
:type grid: List[List[int]]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Largest 1-Bordered Square
* Difficulty: Medium
* Tags: array, 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 {number[][]} grid
* @return {number}
*/
```

```
var largest1BorderedSquare = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Largest 1-Bordered Square
* Difficulty: Medium
* Tags: array, 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 largest1BorderedSquare(grid: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Largest 1-Bordered Square
* Difficulty: Medium
* Tags: array, 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 int Largest1BorderedSquare(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Largest 1-Bordered Square
* Difficulty: Medium
* Tags: array, 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
*/



int largest1BorderedSquare(int** grid, int gridSize, int* gridColSize){

}
```

Go Solution:

```
// Problem: Largest 1-Bordered Square
// Difficulty: Medium
// Tags: array, 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 largest1BorderedSquare(grid [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun largest1BorderedSquare(grid: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func largest1BorderedSquare(_ grid: [[Int]]) -> Int {
}
}
```

Rust Solution:

```
// Problem: Largest 1-Bordered Square
// Difficulty: Medium
// Tags: array, 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 largest1_bordered_square(grid: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} grid
# @return {Integer}
def largest1_bordered_square(grid)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $grid
* @return Integer
*/
function largest1BorderedSquare($grid) {

}
}
```

Scala Solution:

```
object Solution {
def largest1BorderedSquare(grid: Array[Array[Int]]): Int = {

}
}
```