Problem 2267: Check if There Is a Valid Parentheses String Path

Problem Information

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

Problem Description

A parentheses string is a

non-empty

string consisting only of

'('

and

')'

. It is

valid

if

any

of the following conditions is

true

:

It is

()

.

It can be written as

AB

(

A

concatenated with

B

), where

A

and

B

are valid parentheses strings.

It can be written as

(A)

, where

A

is a valid parentheses string.

You are given an

m x n

matrix of parentheses

grid

. A

valid parentheses string path

in the grid is a path satisfying

all

of the following conditions:

The path starts from the upper left cell

(0, 0)

.

The path ends at the bottom-right cell

(m - 1, n - 1)

.

The path only ever moves

down

or

right

.

The resulting parentheses string formed by the path is

valid

.

Return

true

if there exists a

valid parentheses string path

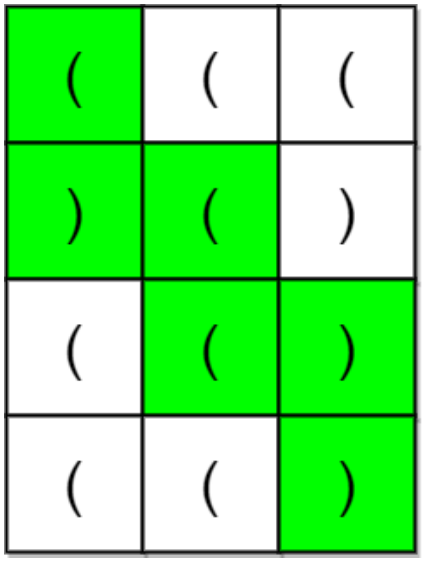
in the grid.

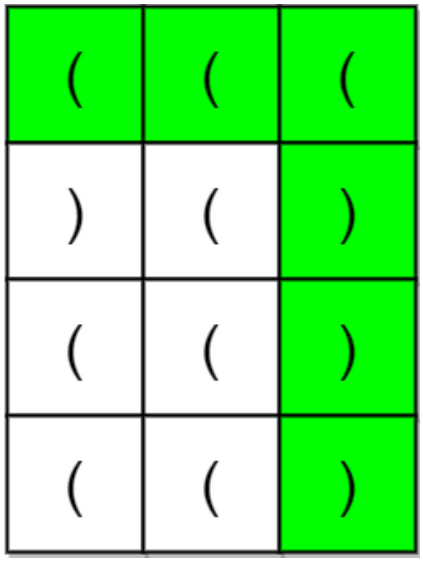
Otherwise, return

false

.

Example 1:





Input:

grid = [["(","(","("],[")","(",")"],["(","(",")"],["(","(",")"]]

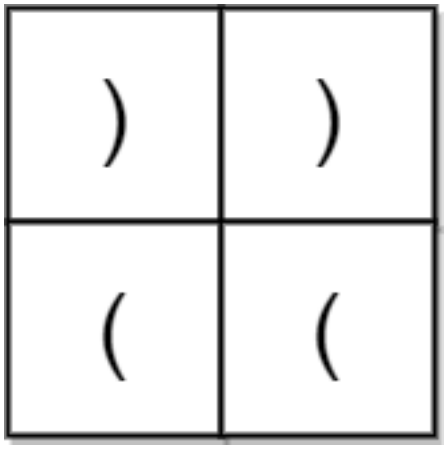
Output:

true

Explanation:

The above diagram shows two possible paths that form valid parentheses strings. The first path shown results in the valid parentheses string "()(())". The second path shown results in the valid parentheses string "((()))". Note that there may be other valid parentheses string paths.

Example 2:



Input:

grid = [[")",")"],["(","("]]

Output:

false

Explanation:

The two possible paths form the parentheses strings "))(" and ")((". Since neither of them are valid parentheses strings, we return false.

Constraints:

m == grid.length

n == grid[i].length

1 <= m, n <= 100

grid[i][j]

is either

'('

or

')'

.

Code Snippets

C++:

```
class Solution {
public:
bool hasValidPath(vector<vector<char>>& grid) {

}
};
```

Java:

```
class Solution {
public boolean hasValidPath(char[][] grid) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function hasValidPath(grid: string[][]): boolean {

};
```

C#:

```
public class Solution {
public bool HasValidPath(char[][] grid) {

}
}
```

C:

```
bool hasValidPath(char** grid, int gridSize, int* gridColSize) {

}
```

Go:

```
func hasValidPath(grid [][]byte) bool {

}
```

Kotlin:

```
class Solution {
fun hasValidPath(grid: Array<CharArray>): Boolean {

}
}
```

Swift:

```
class Solution {
func hasValidPath(_ grid: [[Character]]) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn has_valid_path(grid: Vec<Vec<char>>) -> bool {

}
}
```

Ruby:

```
# @param {Character[][]} grid
# @return {Boolean}
def has_valid_path(grid)

end
```

PHP:

```
class Solution {

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

```
*/
function hasValidPath($grid) {

}
}
```

Dart:

```
class Solution {
bool hasValidPath(List<List<String>> grid) {

}
}
```

Scala:

```
object Solution {
def hasValidPath(grid: Array[Array[Char]]): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec has_valid_path(grid :: [[char]]) :: boolean
def has_valid_path(grid) do

end
end
```

Erlang:

```
-spec has_valid_path(Grid :: [[char()]]) -> boolean().
has_valid_path(Grid) ->
.
```

Racket:

```
(define/contract (has-valid-path grid)
(-> (listof (listof char?)) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check if There Is a Valid Parentheses String Path
* Difficulty: Hard
* Tags: array, string, 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:
bool hasValidPath(vector<vector<char>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Check if There Is a Valid Parentheses String Path
* Difficulty: Hard
* Tags: array, string, 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 boolean hasValidPath(char[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Check if There Is a Valid Parentheses String Path
Difficulty: Hard
Tags: array, string, 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 hasValidPath(self, grid: List[List[str]]) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Check if There Is a Valid Parentheses String Path
* Difficulty: Hard
* Tags: array, string, 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 {character[][]} grid
* @return {boolean}
*/
var hasValidPath = function(grid) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Check if There Is a Valid Parentheses String Path
* Difficulty: Hard
* Tags: array, string, 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 hasValidPath(grid: string[][]): boolean {

};
```

C# Solution:

```
/*
* Problem: Check if There Is a Valid Parentheses String Path
* Difficulty: Hard
* Tags: array, string, 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 bool HasValidPath(char[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Check if There Is a Valid Parentheses String Path
* Difficulty: Hard
```

```
* Tags: array, string, 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
*/

bool hasValidPath(char** grid, int gridSize, int* gridColSize) {

}
```

Go Solution:

```
// Problem: Check if There Is a Valid Parentheses String Path
// Difficulty: Hard
// Tags: array, string, 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 hasValidPath(grid [][]byte) bool {

}
```

Kotlin Solution:

```
class Solution {
fun hasValidPath(grid: Array<CharArray>): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func hasValidPath(_ grid: [[Character]]) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Check if There Is a Valid Parentheses String Path
// Difficulty: Hard
// Tags: array, string, 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 has_valid_path(grid: Vec<Vec<char>>) -> bool {

}
}
```

Ruby Solution:

```
# @param {Character[][]} grid
# @return {Boolean}
def has_valid_path(grid)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[][] $grid
* @return Boolean
*/
function hasValidPath($grid) {

}
}
```

Dart Solution:

```
class Solution {
bool hasValidPath(List<List<String>> grid) {
```

```
}
}
```

Scala Solution:

```
object Solution {
def hasValidPath(grid: Array[Array[Char]]): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec has_valid_path(grid :: [[char]]) :: boolean
def has_valid_path(grid) do

end
end
```

Erlang Solution:

```
-spec has_valid_path(Grid :: [[char()]]) -> boolean().
has_valid_path(Grid) ->
.
```

Racket Solution:

```
(define/contract (has-valid-path grid)
(-> (listof (listof char?)) boolean?)
)
```