# Problem 1591: Strange Printer II

## Problem Information

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

## Problem Description

There is a strange printer with the following two special requirements:

On each turn, the printer will print a solid rectangular pattern of a single color on the grid. This will cover up the existing colors in the rectangle.

Once the printer has used a color for the above operation,

the same color cannot be used again

.

You are given a

m x n

matrix

targetGrid

, where

targetGrid[row][col]

is the color in the position

(row, col)

of the grid.

Return

true

if it is possible to print the matrix

targetGrid

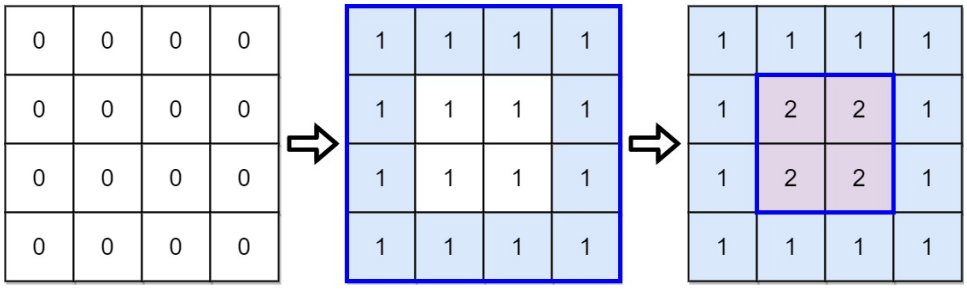
,

otherwise, return

false

.

Example 1:



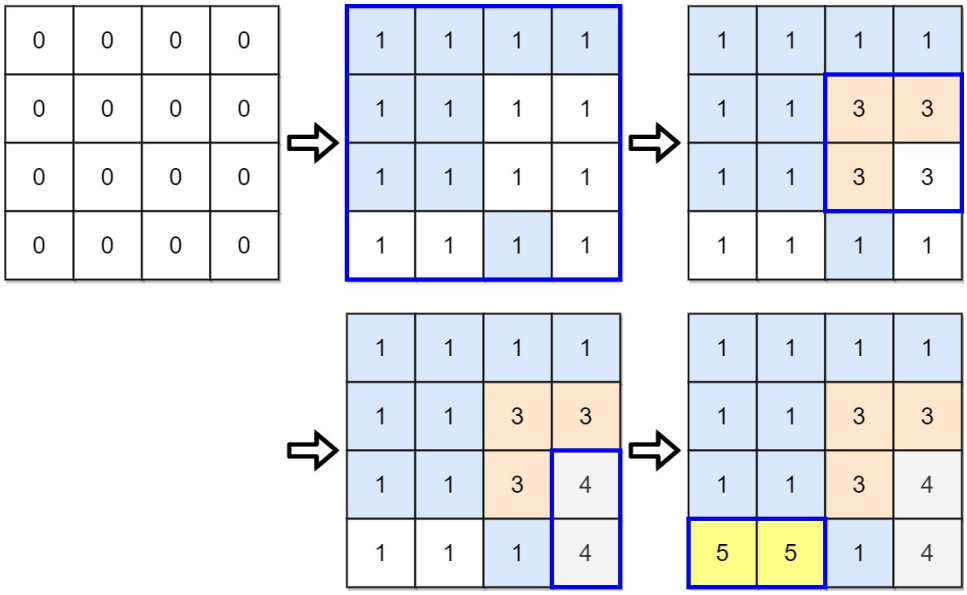

Input:

targetGrid = [[1,1,1,1],[1,2,2,1],[1,2,2,1],[1,1,1,1]]

Output:

true

Example 2:




Input:

targetGrid = [[1,1,1,1],[1,1,3,3],[1,1,3,4],[5,5,1,4]]

Output:

true

Example 3:

Input:

targetGrid = [[1,2,1],[2,1,2],[1,2,1]]

Output:

false

Explanation:

It is impossible to form targetGrid because it is not allowed to print the same color in different turns.

Constraints:

m == targetGrid.length

n == targetGrid[i].length

1 <= m, n <= 60

1 <= targetGrid[row][col] <= 60

## Code Snippets

**C++:**

```
class Solution {
public:
bool isPrintable(vector<vector<int>>& targetGrid) {

}
};
```

**Java:**

```
class Solution {
public boolean isPrintable(int[][] targetGrid) {

}
}
```

**Python3:**

```
class Solution:
def isPrintable(self, targetGrid: List[List[int]]) -> bool:
```

**Python:**

```
class Solution(object):
def isPrintable(self, targetGrid):
"""
:type targetGrid: List[List[int]]
:rtype: bool
"""
```

**JavaScript:**

```
/**
* @param {number[][]} targetGrid
* @return {boolean}
*/
var isPrintable = function(targetGrid) {

};
```

**TypeScript:**

```
function isPrintable(targetGrid: number[][]): boolean {

};
```

**C#:**

```
public class Solution {
public bool IsPrintable(int[][] targetGrid) {

}
}
```

**C:**

```
bool isPrintable(int** targetGrid, int targetGridSize, int*
targetGridColSize) {

}
```

**Go:**

```
func isPrintable(targetGrid [][]int) bool {

}
```

**Kotlin:**

```
class Solution {
fun isPrintable(targetGrid: Array<IntArray>): Boolean {

}
}
```

**Swift:**

```
class Solution {
func isPrintable(_ targetGrid: [[Int]]) -> Bool {

}
}
```

**Rust:**

```
impl Solution {
pub fn is_printable(target_grid: Vec<Vec<i32>>) -> bool {

}
}
```

**Ruby:**

```
# @param {Integer[][]} target_grid
# @return {Boolean}
def is_printable(target_grid)

end
```

**PHP:**

```
class Solution {

/**
* @param Integer[][] $targetGrid
* @return Boolean
*/
function isPrintable($targetGrid) {

}
}
```

**Dart:**

```
class Solution {
bool isPrintable(List<List<int>> targetGrid) {

}
```

```
}
```

**Scala:**

```
object Solution {
def isPrintable(targetGrid: Array[Array[Int]]): Boolean = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec is_printable(target_grid :: [[integer]]) :: boolean
def is_printable(target_grid) do

end
end
```

**Erlang:**

```
-spec is_printable(TargetGrid :: [[integer()]]) -> boolean().
is_printable(TargetGrid) ->
.
```

**Racket:**

```
(define/contract (is-printable targetGrid)
(-> (listof (listof exact-integer?)) boolean?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Strange Printer II
* Difficulty: Hard
* Tags: array, graph, sort
*
```

```
* 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:
bool isPrintable(vector<vector<int>>& targetGrid) {

}
};
```

### Java Solution:

```
/**
* Problem: Strange Printer II
* Difficulty: Hard
* Tags: array, graph, sort
*
* 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 boolean isPrintable(int[][] targetGrid) {

}
}
```

### Python3 Solution:

```
"""
Problem: Strange Printer II
Difficulty: Hard
Tags: array, graph, sort

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 isPrintable(self, targetGrid: List[List[int]]) -> bool:
# TODO: Implement optimized solution
pass
```

**Python Solution:**

```
class Solution(object):
def isPrintable(self, targetGrid):
"""
:type targetGrid: List[List[int]]
:rtype: bool
"""
```

**JavaScript Solution:**

```
/**
* Problem: Strange Printer II
* Difficulty: Hard
* Tags: array, graph, sort
*
* 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 {number[][]} targetGrid
* @return {boolean}
*/
var isPrintable = function(targetGrid) {

};
```

**TypeScript Solution:**

```
/**
* Problem: Strange Printer II
* Difficulty: Hard
* Tags: array, graph, sort
```

```
*
* 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 isPrintable(targetGrid: number[][]): boolean {

};
```

### C# Solution:

```
/*
* Problem: Strange Printer II
* Difficulty: Hard
* Tags: array, graph, sort
*
* 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 bool IsPrintable(int[][] targetGrid) {

}
}
```

### C Solution:

```
/*
* Problem: Strange Printer II
* Difficulty: Hard
* Tags: array, graph, sort
*
* 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
*/

bool isPrintable(int** targetGrid, int targetGridSize, int*
```

```
targetGridColSize) {

}
```

### Go Solution:

```
// Problem: Strange Printer II
// Difficulty: Hard
// Tags: array, graph, sort
//
// 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 isPrintable(targetGrid [][]int) bool {

}
```

### Kotlin Solution:

```
class Solution {
fun isPrintable(targetGrid: Array<IntArray>): Boolean {

}
}
```

### Swift Solution:

```
class Solution {
func isPrintable(_ targetGrid: [[Int]]) -> Bool {

}
}
```

### Rust Solution:

```
// Problem: Strange Printer II
// Difficulty: Hard
// Tags: array, graph, sort
//
// 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 is_printable(target_grid: Vec<Vec<i32>>) -> bool {

}
}
```

**Ruby Solution:**

```
# @param {Integer[][]} target_grid
# @return {Boolean}
def is_printable(target_grid)

end
```

**PHP Solution:**

```
class Solution {

/**
* @param Integer[][] $targetGrid
* @return Boolean
*/
function isPrintable($targetGrid) {

}
}
```

**Dart Solution:**

```
class Solution {
bool isPrintable(List<List<int>> targetGrid) {

}
}
```

**Scala Solution:**

```
object Solution {
def isPrintable(targetGrid: Array[Array[Int]]): Boolean = {
```

```
}
}
```

### Elixir Solution:

```
defmodule Solution do
@spec is_printable(target_grid :: [[integer]]) :: boolean
def is_printable(target_grid) do

end
end
```

### Erlang Solution:

```
-spec is_printable(TargetGrid :: [[integer()]]) -> boolean().
is_printable(TargetGrid) ->
.
```

### Racket Solution:

```
(define/contract (is-printable targetGrid)
(-> (listof (listof exact-integer?)) boolean?)
)
```