Problem 85: Maximal Rectangle

Problem Information

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

Problem Description

Given a

rows x cols

binary

matrix

filled with

0

's and

1

's, find the largest rectangle containing only

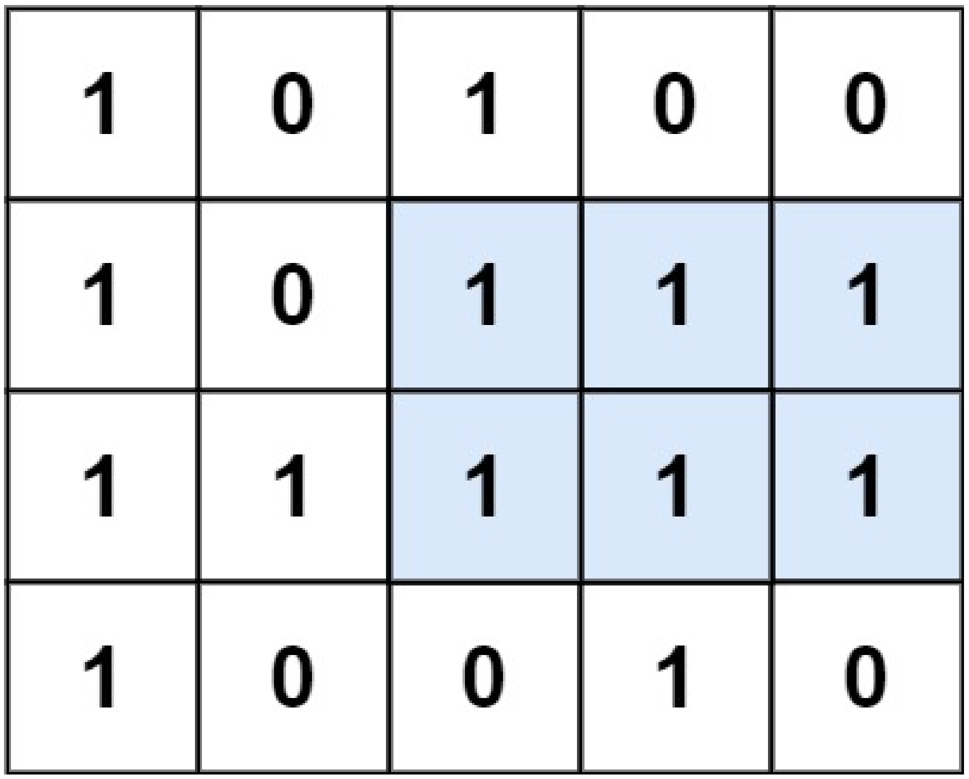
1

's and return

its area

.

Example 1:



1	0	1	0	0
1	0	1	1	1
1	1	1	1	1
1	0	0	1	0

Input:

matrix = [["1","0","1","0","0"],["1","0","1","1","1"],["1","1","1","1","1"],["1","0","0","1","0"]]

Output:

6

Explanation:

The maximal rectangle is shown in the above picture.

Example 2:

Input:

matrix = [["0"]]

Output:

0

Example 3:

Input:

matrix = [["1"]]

Output:

1

Constraints:

rows == matrix.length

cols == matrix[i].length

1 <= rows, cols <= 200

matrix[i][j]

is

'0'

or

'1'

.

Code Snippets

C++:

```
class Solution {
public:
```

```
int maximalRectangle(vector<vector<char>>& matrix) {

}
};
```

Java:

```
class Solution {
public int maximalRectangle(char[][] matrix) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function maximalRectangle(matrix: string[][]): number {

};
```

C#:

```
public class Solution {
public int MaximalRectangle(char[][] matrix) {

}
}
```

C:

```
int maximalRectangle(char** matrix, int matrixSize, int* matrixColSize) {

}
```

Go:

```
func maximalRectangle(matrix [][]byte) int {

}
```

Kotlin:

```
class Solution {
fun maximalRectangle(matrix: Array<CharArray>): Int {

}
}
```

Swift:

```
class Solution {
func maximalRectangle(_ matrix: [[Character]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn maximal_rectangle(matrix: Vec<Vec<char>>) -> i32 {

}
}
```

Ruby:

```
# @param {Character[][]} matrix
# @return {Integer}
def maximal_rectangle(matrix)

end
```

PHP:

```
class Solution {

/**
* @param String[][] $matrix
* @return Integer
*/
function maximalRectangle($matrix) {

}
}
```

Dart:

```
class Solution {
int maximalRectangle(List<List<String>> matrix) {

}
}
```

Scala:

```
object Solution {
def maximalRectangle(matrix: Array[Array[Char]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec maximal_rectangle(matrix :: [[char]]) :: integer
def maximal_rectangle(matrix) do
```

```
end
end
```

Erlang:

```
-spec maximal_rectangle(Matrix :: [[char()]]) -> integer().
maximal_rectangle(Matrix) ->
.
```

Racket:

```
(define/contract (maximal-rectangle matrix)
(-> (listof (listof char?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximal Rectangle
* Difficulty: Hard
* Tags: array, dp, stack
*
* 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 maximalRectangle(vector<vector<char>>& matrix) {

}
};
```

Java Solution:

```
/**
* Problem: Maximal Rectangle
* Difficulty: Hard
* Tags: array, dp, stack
*
* 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 maximalRectangle(char[][] matrix) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximal Rectangle
Difficulty: Hard
Tags: array, dp, stack

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximal Rectangle
* Difficulty: Hard
* Tags: array, dp, stack
*
* 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[][]} matrix
* @return {number}
*/
var maximalRectangle = function(matrix) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximal Rectangle
* Difficulty: Hard
* Tags: array, dp, stack
*
* 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 maximalRectangle(matrix: string[][]): number {

};
```

C# Solution:

```
/*
* Problem: Maximal Rectangle
* Difficulty: Hard
* Tags: array, dp, stack
*
* 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 MaximalRectangle(char[][] matrix) {

}
}
```

C Solution:

```
/*
* Problem: Maximal Rectangle
* Difficulty: Hard
* Tags: array, dp, stack
*
* 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 maximalRectangle(char** matrix, int matrixSize, int* matrixColSize) {

}
```

Go Solution:

```
// Problem: Maximal Rectangle
// Difficulty: Hard
// Tags: array, dp, stack
//
// 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 maximalRectangle(matrix [][]byte) int {

}
```

Kotlin Solution:

```
class Solution {
fun maximalRectangle(matrix: Array<CharArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func maximalRectangle(_ matrix: [[Character]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximal Rectangle
// Difficulty: Hard
// Tags: array, dp, stack
//
// 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 maximal_rectangle(matrix: Vec<Vec<char>>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Character[][]} matrix
# @return {Integer}
def maximal_rectangle(matrix)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param String[][] $matrix
* @return Integer
*/
function maximalRectangle($matrix) {

}
}
```

Dart Solution:

```
class Solution {
int maximalRectangle(List<List<String>> matrix) {

}
}
```

Scala Solution:

```
object Solution {
def maximalRectangle(matrix: Array[Array[Char]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec maximal_rectangle(matrix :: [[char]]) :: integer
def maximal_rectangle(matrix) do

end
end
```

Erlang Solution:

```
-spec maximal_rectangle(Matrix :: [[char()]]) -> integer().
maximal_rectangle(Matrix) ->
.
```

Racket Solution:

```
(define/contract (maximal-rectangle matrix)
(-> (listof (listof char?)) exact-integer?)
)
```