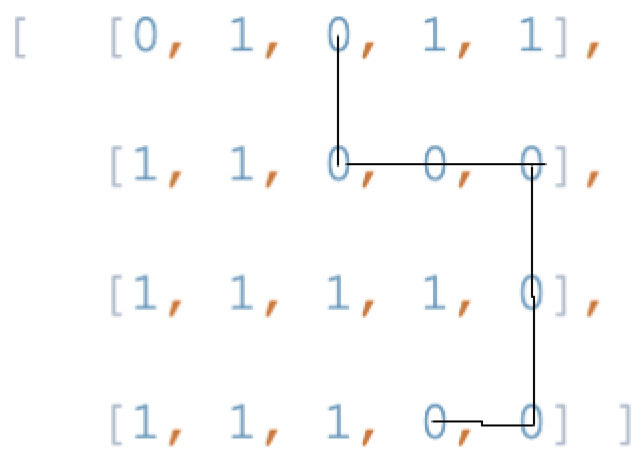
You are given n x m grid , where 1 represents land and 0 represents the water. You have to find the size of the biggest river.

A river can twist and can have branches. If there is no river in grid return 0

Input

```
[   [0, 1, 0, 1, 1],
    [1, 1, 0, 0, 0],
    [1, 1, 1, 1, 0],
    [1, 1, 1, 0, 0] ]
```

```
Output:
7
Explanation
```

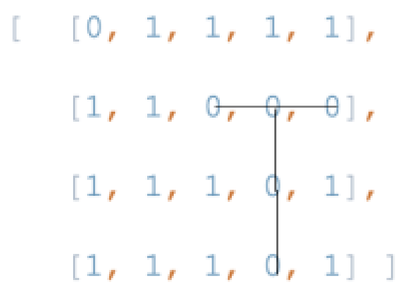


Input:

```
[   [0, 1, 1, 1, 1],

    [1, 1, 0, 0, 0],

    [1, 1, 1, 0, 1],

    [1, 1, 1, 0, 1] ]
```

```
Output:

5
```

```
Explanation:
```



```
[   [0, 1, 1, 1, 1],
    [1, 1, 0, 0, 0],
    [1, 1, 1, 0, 1],
    [1, 1, 1, 0, 1]  ]
```

Solution

https://pastebin.com/G1pjrnmN

The following question is taken from leetcode but I feel it’s a good test of python knowledge

Design a logger system that receives a stream of messages along with their timestamps. Each unique message should only be printed at most every 10 seconds (i.e. a message printed at timestamp `t` will prevent other identical messages from being printed until timestamp `t + 10`).

All messages will come in chronological order. Several messages may arrive at the same timestamp.

Implement the `Logger` class:

- `Logger()` Initializes the `logger` object.

- `bool shouldPrintMessage(int timestamp, string message)` Returns `true` if the `message` should be printed in the given `timestamp`, otherwise returns `false`.

Example 1:

```
Input

["Logger", "shouldPrintMessage", "shouldPrintMessage", "shouldPrintMessage",
"shouldPrintMessage", "shouldPrintMessage", "shouldPrintMessage"]

[[], [1, "foo"], [2, "bar"], [3, "foo"], [8, "bar"], [10, "foo"], [11, "foo"]]

Output

[null, true, true, false, false, false, true]


Explanation

Logger logger = new Logger();

logger.shouldPrintMessage(1, "foo");  // return true, next allowed timestamp
for "foo" is 1 + 10 = 11

logger.shouldPrintMessage(2, "bar");  // return true, next allowed timestamp
for "bar" is 2 + 10 = 12

logger.shouldPrintMessage(3, "foo");  // 3 < 11, return false

logger.shouldPrintMessage(8, "bar");  // 8 < 12, return false

logger.shouldPrintMessage(10, "foo"); // 10 < 11, return false

logger.shouldPrintMessage(11, "foo"); // 11 >= 11, return true, next allowed
timestamp for "foo" is

                                      // 11 + 10 = 21
```

Constraints:

- `0 <= timestamp <= 10^9`
- Every `timestamp` will be passed in non-decreasing order (chronological order).
- `1 <= message.length <= 30`
- At most `10^4` calls will be made to `shouldPrintMessage`.

Solution

1. https://pastebin.com/rkaHwFWK

2. https://pastebin.com/gbnBXMLA