Lonely Pixel I

Given a picture consisting of black and white pixels, find the number of**black** lonely pixels.

The picture is represented by a 2D char array consisting of 'B' and 'W', which means black and white pixels respectively.

A black lonely pixel is character 'B' that located at a specific position where the same row and same column don't have any other black pixels.

Example:

```
Input:
[['W', 'W', 'B'],
 ['W', 'B', 'W'],
 ['B', 'W', 'W']]

Output: 3
Explanation: All the three 'B's are black lonely pixels.
```

Note:

1. The range of width and height of the input 2D array is [1,500].

Solution 1

O(nm) Time, O(n+m) Space Solution:

```
public int findLonelyPixel(char[][] picture) {
    int n = picture.length, m = picture[0].length;

    int[] rowCount = new int[n], colCount = new int[m];
    for (int i=0;i<n;i++)
        for (int j=0;j<m;j++)
            if (picture[i][j] == 'B') { rowCount[i]++; colCount[j]++; }

    int count = 0;
    for (int i=0;i<n;i++)
        for (int j=0;j<m;j++)
            if (picture[i][j] == 'B' && rowCount[i] == 1 && colCount[j] == 1) cou
nt++;

    return count;
}
```

O(nm) Time, O(1) Space Solution:

```
public int findLonelyPixel(char[][] picture) {
    int n = picture.length, m = picture[0].length;

    int firstRowCount = 0;
    for (int i=0;i<n;i++)
        for (int j=0;j<m;j++)
            if (picture[i][j] == 'B') {
                if (picture[0][j] < 'Y' && picture[0][j] != 'V') picture[0][j]++;
                if (i == 0) firstRowCount++;
                else if (picture[i][0] < 'Y' && picture[i][0] != 'V') picture[i][
0]++;
            }

    int count = 0;
    for (int i=0;i<n;i++)
        for (int j=0;j<m;j++)
            if (picture[i][j] < 'W' && (picture[0][j] == 'C' || picture[0][j] ==
'X')) {
                if (i == 0) count += firstRowCount == 1 ? 1 : 0;
                else if (picture[i][0] == 'C' || picture[i][0] == 'X') count++;
            }

    return count;
}
```

written by compton_scatter original link here

Solution 2

thought is very simple, we can easily count how many times B occurs in each row. But how can we know if this col has existing B?
for example, input is
W B B B
B W W W
W W W B
W W W B
we can maintain an array calls colArray[], which is used to record how many times the B occurs in each column. Then solution is simple

```
public class Solution {
    public int findLonelyPixel(char[][] picture) {
        if (picture == null || picture.length == 0 || picture[0].length == 0) ret
urn 0;

        int[] colArray = new int[picture[0].length];
        for (int i = 0; i < picture.length; i++) {
            for (int j = 0; j < picture[0].length; j++) {
                if (picture[i][j] == 'B') colArray[j]++;
            }
        }

        int ret = 0;
        for (int i = 0; i < picture.length; i++) {
            int count = 0, pos = 0;
            for (int j = 0; j < picture[0].length; j++) {
                if (picture[i][j] == 'B') {
                    count++;
                    pos = j;
                }
            }
            if (count == 1 && colArray[pos] == 1) ret++;
        }
        return ret;
    }
}
```

written by lusoul original link here

Solution 3

C++

```
/**
 * suppose matrix is m*n, there is at most min(m, n) lonely pixels, because there could be no more than 1 in each row, or column;
 * therefore, if we record num of black pixel on each row and column, we can easily tell whether each pixel is lonely or NO.
 *     _0_1_2_
 *  0 | 0 0 1   rows[0] = 1
 *  1 | 0 1 0   rows[1] = 1
 *  2 | 1 0 0   rows[2] = 1
 *
 * cols[0][1][2]
 *     1  1  1
 */
class Solution {
public:
    int findLonelyPixel(vector<vector<char>>& pic) {
        int m = pic.size();
        int n = pic[0].size();
        vector<int> rows = vector<int>(m, 0);
        vector<int> cols = vector<int>(n, 0);

        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                rows[i] += pic[i][j] == 'B';
                cols[j] += pic[i][j] == 'B';
            }
        }

        int lonely = 0;
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n && rows[i] > 0; j++) {
                lonely += pic[i][j] == 'B' && rows[i] == 1 && cols[j] == 1;
            }
        }

        return lonely;
    }
};
```

Java

```java
public class Solution {
    public int findLonelyPixel(char[][] pic) {
        int m = pic.length;
        int n = pic[0].length;
        int[] rows = new int[m];
        int[] cols = new int[n];

        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                rows[i] += pic[i][j] == 'B' ? 1 : 0;
                cols[j] += pic[i][j] == 'B' ? 1 : 0;
            }
        }

        int lonely = 0;
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n && rows[i] > 0; j++) {
                lonely += (pic[i][j] == 'B' && rows[i] == 1 && cols[j] == 1) ? 1
: 0;
            }
        }

        return lonely;
    }
}
```

written by alexander original link here