Bomb Enemy

Given a 2D grid, each cell is either a wall `'W'`, an enemy `'E'` or empty `'0'` (the number zero), return the maximum enemies you can kill using one bomb.
The bomb kills all the enemies in the same row and column from the planted point until it hits the wall since the wall is too strong to be destroyed.
Note that you can only put the bomb at an empty cell.

Example:

```
For the given grid

0 E 0 0
E 0 W E
0 E 0 0

return 3. (Placing a bomb at (1,1) kills 3 enemies)
```

Credits:
Special thanks to @memoryless for adding this problem and creating all test cases.

Solution 1

Walk through the matrix. At the start of each non-wall-streak (row-wise or column-wise), count the number of hits in that streak and remember it. O(mn) time, O(n) space.

```
int maxKilledEnemies(vector<vector<char>>& grid) {
    int m = grid.size(), n = m ? grid[0].size() : 0;
    int result = 0, rowhits, colhits[n];
    for (int i=0; i<m; i++) {
        for (int j=0; j<n; j++) {
            if (!j || grid[i][j-1] == 'W') {
                rowhits = 0;
                for (int k=j; k<n && grid[i][k] != 'W'; k++)
                    rowhits += grid[i][k] == 'E';
            }
            if (!i || grid[i-1][j] == 'W') {
                colhits[j] = 0;
                for (int k=i; k<m && grid[k][j] != 'W'; k++)
                    colhits[j] += grid[k][j] == 'E';
            }
            if (grid[i][j] == '0')
                result = max(result, rowhits + colhits[j]);
        }
    }
    return result;
}
```

written by StefanPochmann original link here

Solution 2

Pretty straight forward O(mn2) solution. Cacheing the last search result could save a lot of time. Welcome to discuss.

```
public class Solution {
public int maxKilledEnemies(char[][] grid) {
    if (grid == null || grid.length == 0 || grid[0].length == 0) {
        return 0;
    }
    int ret = 0;
    int row = grid.length;
    int col = grid[0].length;
    int rowCache = 0;
    int[] colCache = new int[col];
    for (int i = 0;i < row; i++) {
        for (int j = 0; j < col; j++) {
            if (j == 0 || grid[i][j-1] == 'W') {
                rowCache = 0;
                for (int k = j; k < col && grid[i][k] != 'W'; k++) {
                    rowCache += grid[i][k] == 'E' ? 1 : 0;
                }
            }
            if (i == 0 || grid[i-1][j] == 'W') {
                colCache[j] = 0;
                for (int k = i;k < row && grid[k][j] != 'W'; k++) {
                    colCache[j] += grid[k][j] == 'E' ? 1 :0;
                }
            }
            if (grid[i][j] == '0') {
                ret = Math.max(ret, rowCache + colCache[j]);
            }
        }
    }
    return ret;
}
```

}

written by francis.yang1991 original link here

Solution 3

Compute a matrix for row-wise hits and one for column-wise hits. Then find the maximum.

```
def maxKilledEnemies(self, grid):
    def hits(grid):
        return [[h
                 for block in ''.join(row).split('W')
                 for h in [block.count('E')] * len(block) + [0]]
                for row in grid]
    rowhits = hits(grid)
    colhits = zip(*hits(zip(*grid)))
    return max([rh + ch
                for row in zip(grid, rowhits, colhits)
                for cell, rh, ch in zip(*row)
                if cell == '0'] or [0])
```

written by StefanPochmann original link here