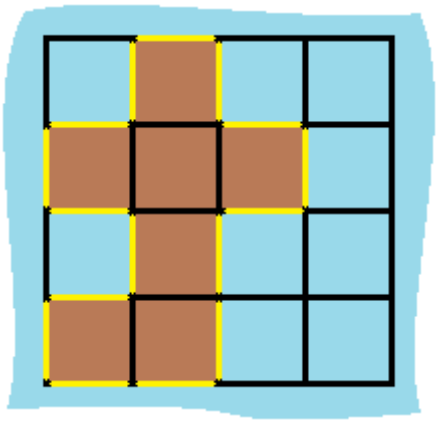
463 岛屿的周长

Label：规律

给定一个 row x col 的二维网格地图 grid ，其中：grid[i][j] = 1 表示陆地， grid[i][j] = 0 表示水域。

网格中的格子 水平和垂直 方向相连（对角线方向不相连）。整个网格被水完全包围，但其中恰好有一个岛屿（或者说，一个或多个表示陆地的格子相连组成的岛屿）。

岛屿中没有“湖”（“湖” 指水域在岛屿内部且不和岛屿周围的水相连）。格子是边长为 1 的正方形。网格为长方形，且宽度和高度均不超过 100 。计算这个岛屿的周长。



- 四边 遮盖移动

```
class Solution {
    public int islandPerimeter(int[][] grid) {
        int res=0;
        int row=grid.length;
        int col=grid[0].length;
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                if (grid[i][j] == 1) {//是陆地
                    res+=4;
                    if(i > 0 && grid[i-1][j] == 1)  res-=2; // 减去自己的和自己遮掉的边  上下
                    if(j > 0 && grid[i][j-1] == 1)  res-=2; // 减去自己的和自己遮掉的边  左右
                }
            }
        }
        return res;
    }
}
```

- 复杂条件遍历

```
class Solution {
    public int islandPerimeter(int[][] grid) {
        int count  = 0 ;

        // 第一层的 顶
        for (int j = 0; j < grid[0].length; j++) {
            if (grid[0][j] == 1) count++;
            if (j > 0  && grid[0][j-1] == 1 &&  grid[0][j] == 0)
                count++;
            if (j > 0  && grid[0][j-1] == 0 &&  grid[0][j] == 1)
                count++;
            if (j == 0 && grid[0][j] == 1)
                count++;
            if (j == grid[0].length-1  && grid[0][j] == 1)
                count++;
        }


        for (int i = 1 ; i < grid.length; i++) {
            for (int j = 0; j < grid[i].length; j++) {
                if (grid[i][j] == 1 && grid[i-1][j] == 0)  // 当前层为1 ， 上层为 0
                    count++;
                if (grid[i][j] == 0 && grid[i-1][j] == 1)  // 当前层为0 ， 上层为 1
                    count++;

                if (j > 0  && grid[i][j-1] == 1 &&  grid[i][j] == 0)
                    count++;
                if (j > 0  && grid[i][j-1] == 0 &&  grid[i][j] == 1)
                    count++;

                if (j == 0 && grid[i][j] == 1)
                    count++;
                if (j == grid[i].length-1 && grid[i][j] == 1)
                    count++;
            }
        }

        // 最后一层的 底
        for (int j = 0; j < grid[grid.length-1].length; j++) {
            if (grid[grid.length-1][j] == 1) count++;
        }
        return count;
    }
}
```

- 逻辑合并

```
class Solution {
    public int islandPerimeter(int[][] grid) {
        int count  = 0 ;

        for (int i = 0 ; i < grid.length; i++) {
            for (int j = 0; j < grid[i].length; j++) {
                if (i == 0 && grid[i][j] == 1)
                    count++;

                if (i>0 && grid[i][j] == 1 && grid[i-1][j] == 0)  // 当前层为1 ，
上层为 0
                    count++;
                if (i>0 && grid[i][j] == 0 && grid[i-1][j] == 1)  // 当前层为0 ，
上层为 1
                    count++;

                if (j > 0  && grid[i][j-1] == 1 &&  grid[i][j] == 0)
                    count++;
                if (j > 0  && grid[i][j-1] == 0 &&  grid[i][j] == 1)
                    count++;

                if (j == 0 && grid[i][j] == 1)
                    count++;
                if (j == grid[i].length-1 && grid[i][j] == 1)
                    count++;
            }
        }

        // 最后一层的 底
        for (int j = 0; j < grid[grid.length-1].length; j++) {
            if (grid[grid.length-1][j] == 1) count++;
        }
        return count;
    }
}
```