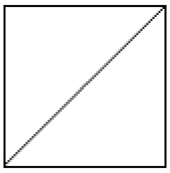
959. Regions Cut By Slashes

An `n x n` grid is composed of `1 x 1` squares where each `1 x 1` square consists of a `'/'`, `'\'`, or blank space `' '`. These characters divide the square into contiguous regions.

Given the grid `grid` represented as a string array, return *the number of regions*.

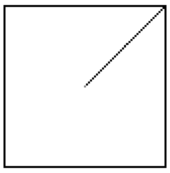
Note that backslash characters are escaped, so a `'\'` is represented as `'\\'`.

Example 1:



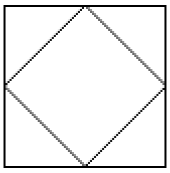
```
Input: grid = [" /","/ "]
Output: 2
```

Example 2:



```
Input: grid = [" /","  "]
Output: 1
```

Example 3:



```
Input: grid = ["/\\","\\/"]
Output: 5
Explanation: Recall that because \ characters are escaped, "\\/" refers to
\/, and "/\\" refers to /\.
```

Constraints:

- `n == grid.length == grid[i].length`
- `1 <= n <= 30`

- `grid[i][j]` is either `'/'`, `'\'`, or `' '`.

```
class Solution:
    def regionsBySlashes(self, grid: List[str]) -> int:
        n = len(grid)+1

        parent = [i for i in range(n*n)]
        rank = [1]*(n*n)
        count = 1
        for i in range(n):
            for j in range(n):
                cellNumber = i*n+j
                if i==0 or j==0 or i==n-1 or j==n-1:
                    if cellNumber!=0:
                        if self.union(0,cellNumber,parent,rank):
                            count+=1

        for i in range(len(grid)):
            string = grid[i]
            for j in range(len(string)):
                if string[j]=='/':
                    cell1 = i*n+j+1
                    cell2 = (i+1)*n+j
                    if self.union(cell1,cell2,parent,rank):
                        count+=1
                elif string[j]=='\\':
                    cell1 = i*n+j
                    cell2 = (i+1)*n+j+1
                    if self.union(cell1,cell2,parent,rank):
                        count+=1
        return count










    def find(self,parent,x):
        if parent[x]==x:
            return x
        temp=self.find(parent,parent[x])
        parent[x]=temp
        return temp

    def union(self,x,y,parent,rank):
        lx = self.find(parent,x)
        ly = self.find(parent,y)
        if lx!=ly:
            if rank[lx]>rank[ly]:
                parent[ly] = lx
            elif rank[lx]<rank[ly]:
                parent[lx] = ly
            else:
                parent[lx]=ly
                rank[ly] = rank[ly]+1
            return False
        return True
```