959. Regions Cut By Slashes

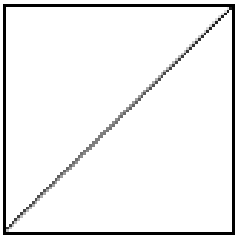
An `n x n` grid is composed of `1 x 1` squares where each `1 x 1` square consists of a `'/'`, `'\'`, or blank space `' '`. These characters divide the square into contiguous regions.

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 `'\\'`.

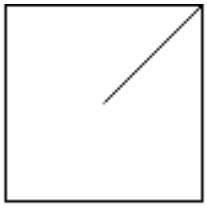
Example 1:

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



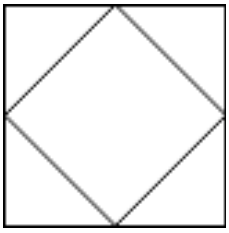
Example 2:

```
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 `' '`.

959. Regions Cut By Slashes


```
/*
   Union-Find
   Time complexity: O(α(n^2)n^2)
   Space complexity: O(n^2)
*/
class DSU{
   public:
      int nb_points;
      std::vector<int> parent;
   public:
      DSU(int n){
         nb_points=n+1;
         int m=nb_points*nb_points;
         parent.resize(m);
         for(int i=0;i<m;++i) parent[i]=i;
      }
      int find(int p){
         int root=p;
         while(root!=parent[root]) root=parent[root];
         // Path compression
         while(p!=root){
            int next=parent[p];
            parent[p]=root;
            p=next;
         }
         return root;
      }
      int find_rec(int p){
         if(p==parent[p]) return p;
         return parent[p]=find_rec(parent[p]); // Path compression
      }
      void unify(int p,int q){
         int parent_p=find(p);
         int parent_q=find(q);
         if(parent_p==parent_q) return;
         parent[parent_p]=parent_q;
      }
};
```

```
class Solution {
public:
    int regionsBySlashes(std::vector<std::string>& grid) {
        int n=grid.size();
        DSU dsu=DSU(n);
        int m=dsu.nb_points;
        for (int i=0;i<m;i++){
            for (int j=0;j<m;j++){
                if (i==0 || j==0 || i==m-1 || j==m-1){
                    int point=i*m+j;
                    if(point!=0) dsu.unify(0,point);
                }
            }
        }
        int ans=1;
        for(int i=0;i<n;++i){
            std::string s=grid[i];
            for(int j=0;j<s.size();++j){
                if(s[j]==' ') continue;
                int point1,point2;
                if(s[j]=='/'){
                    point1=i*m+(j+1);
                    point2=(i+1)*m+j;
                }
                else if(s[j]=='\\'){
                    point1=i*m+j;
                    point2=(i+1)*m+(j+1);
                }
                if(dsu.find_rec(point1)==dsu.find_rec(point2)) ans++;
                else dsu.unify(point1,point2);
            }
        }
        return ans;
    }
};
```