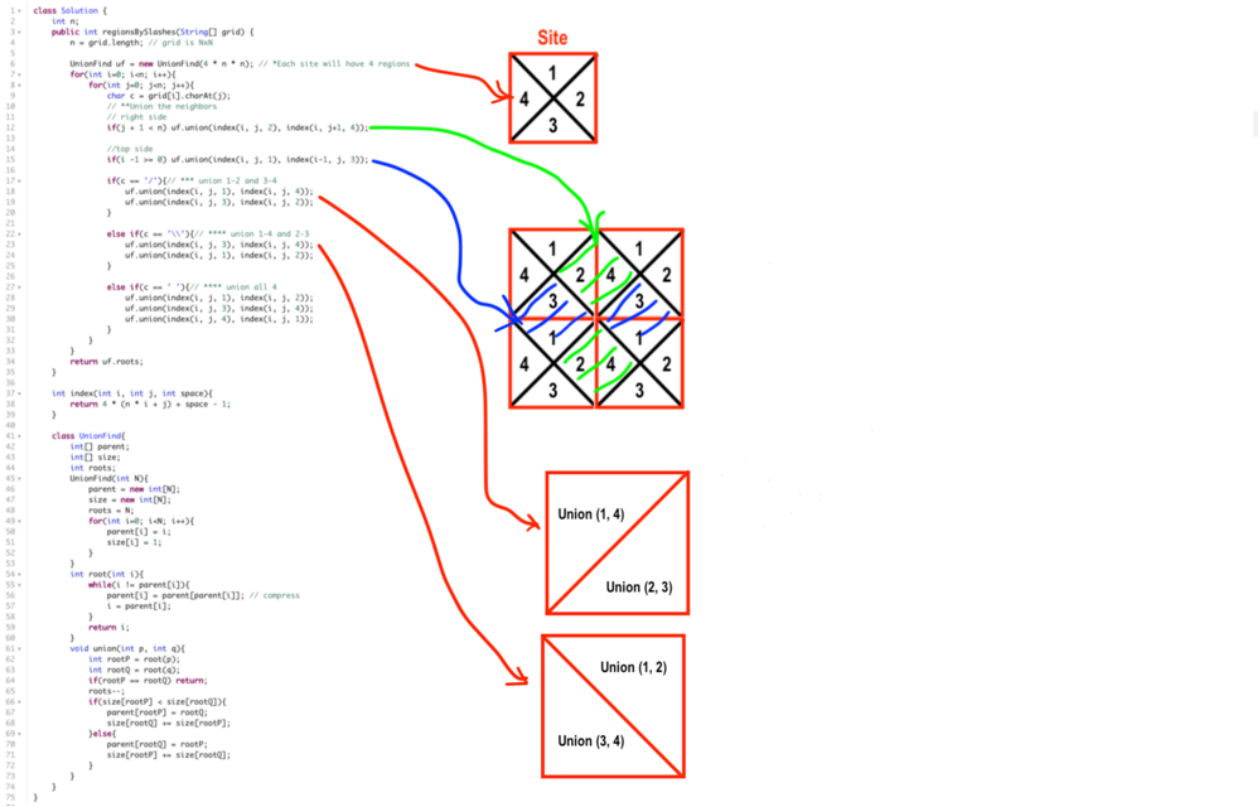
JAVA - Union Find (explanation)




```
class Solution {
    int n;
    public int regionsBySlashes(String[] grid) {
        n = grid.length;

        UnionFind uf = new UnionFind(4 * n * n);
        for(int i=0; i<n; i++){
            for(int j=0; j<n; j++){
                char c = grid[i].charAt(j);


                if(j + 1 < n) uf.union(index(i, j, 2), index(i, j+1, 4));


                if(i -1 >= 0) uf.union(index(i, j, 1), index(i-1, j, 3));

                if(c == '/'){
                    uf.union(index(i, j, 1), index(i, j, 4));
                    uf.union(index(i, j, 3), index(i, j, 2));
                }

                else if(c == '\\'){
                    uf.union(index(i, j, 3), index(i, j, 4));
                    uf.union(index(i, j, 1), index(i, j, 2));
                }

                else if(c == ' '){
                    uf.union(index(i, j, 1), index(i, j, 2));
                    uf.union(index(i, j, 3), index(i, j, 4));
                    uf.union(index(i, j, 4), index(i, j, 1));
                }
            }
        }
        return uf.roots;
    }

    int index(int i, int j, int space){
        return 4 * (n * i + j) + space - 1;
    }
```

```
    class UnionFind{
        int[] parent;
        int[] size;
        int roots;
        UnionFind(int N){
            parent = new int[N];
            size = new int[N];
            roots = N;
            for(int i=0; i<N; i++){
                parent[i] = i;
                size[i] = 1;
            }
        }
        int root(int i){
            while(i != parent[i]){
                parent[i] = parent[parent[i]];
                i = parent[i];
            }
            return i;
        }
        void union(int p, int q){
            int rootP = root(p);
            int rootQ = root(q);
            if(rootP == rootQ) return;
            roots--;
            if(size[rootP] < size[rootQ]){
                parent[rootP] = rootQ;
                size[rootQ] += size[rootP];
            }else{
                parent[rootQ] = rootP;
                size[rootP] += size[rootQ];
            }
        }
    }
}
```