On a 2D plane, we place n stones at some integer coordinate points. Each coordinate point may have at most one stone.

A stone can be removed if it shares either **the same row or the same column** as another stone that has not been removed.

Given an array stones of length n where stones[i] = [xi, yi] represents the location of the ith stone, return *the largest possible number of stones that can be removed*.

**Example 1:**

**Input:** stones = [[0,0],[0,1],[1,0],[1,2],[2,1],[2,2]]

**Output:** 5

**Explanation:** One way to remove 5 stones is as follows:

1. Remove stone [2,2] because it shares the same row as [2,1].

2. Remove stone [2,1] because it shares the same column as [0,1].

3. Remove stone [1,2] because it shares the same row as [1,0].

4. Remove stone [1,0] because it shares the same column as [0,0].

5. Remove stone [0,1] because it shares the same row as [0,0].

Stone [0,0] cannot be removed since it does not share a row/column with another stone still on the plane.

**Example 2:**

**Input:** stones = [[0,0],[0,2],[1,1],[2,0],[2,2]]

**Output:** 3

**Explanation:** One way to make 3 moves is as follows:

1. Remove stone [2,2] because it shares the same row as [2,0].

2. Remove stone [2,0] because it shares the same column as [0,0].

3. Remove stone [0,2] because it shares the same row as [0,0].

Stones [0,0] and [1,1] cannot be removed since they do not share a row/column with another stone still on the plane.

**Example 3:**

**Input:** stones = [[0,0]]

**Output:** 0

**Explanation:** [0,0] is the only stone on the plane, so you cannot remove it.

**Constraints:**

* 1 <= stones.length <= 1000
* 0 <= xi, yi <= 104
* No two stones are at the same coordinate point.