947. Most Stones Removed with Same Row or Column

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 `i`th 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.
```

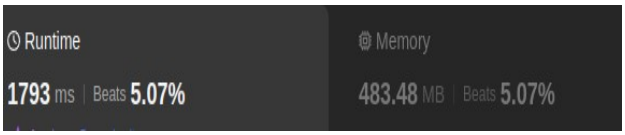
Example 3:

```
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 <= 10`4
- No two stones are at the same coordinate point.

947. Most Stones Removed with Same Row or Column




```
/*
   DSU
   Time compelxity: O(n^2.α(n)+n)
   Space complexity: O(n+n+n)=O(n)
      where:  α is the inverse Ackermann function
            n: number of stones
*/
typedef std::vector<int> vi;
typedef std::vector<vi> vvi;
class DSU{
   public:
      vi parent;
      vi group;
   public:
      DSU(int n){
         group.resize(n,1);
         parent.resize(n);
         std::iota(parent.begin(),parent.end(),0);
      }
      int find(int p){
         if(p==parent[p]) return p;
         return parent[p]=find(parent[p]);
      }
      void unify(int p,int q){
         int parent_p=find(p);
         int parent_q=find(q);
         if(parent_p==parent_q) return;
         if(group[parent_p]<group[parent_q]){
            group[parent_q]+=group[parent_p];
            group[parent_p]=1;
            parent[parent_p]=parent_q;
         }
         else{
            group[parent_p]+=group[parent_q];
            group[parent_q]=1;
            parent[parent_q]=parent_p;
         }
      }
};
```

```
class Solution {
    public:
        int n;
    public:
        int removeStones(vvi& stones){
            n=stones.size();

            DSU dsu=DSU(n);

            for(int i=0;i<n-1;++i){
                vi posi=stones[i];
                int xi=posi[0];
                int yi=posi[1];
                for(int j=i+1;j<n;++j){
                    vi posj=stones[j];
                    int xj=posj[0];
                    int yj=posj[1];

                    if(xi==xj || yi==yj) dsu.unify(i,j);
                }
            }

            int ans=0;
            for(auto& g: dsu.group) ans+=(g-1);

            return ans;
        }
};
```

947. Most Stones Removed with Same Row or Column


```
/*
   DSU
   Time compelxity: O(n.α(_N)+n._N)
   Space complexity: O(3._N)=O(_N)
      where:  α is the inverse Ackermann function
            n: number of stones
            _N=max row value+max column value+2
*/
typedef std::vector<int> vi;
typedef std::vector<vi> vvi;
class DSU{
   public:
      vi parent;
      vi group;
   public:
      DSU(int n){
         group.resize(n,1);
         parent.resize(n);
         std::iota(parent.begin(),parent.end(),0);
      }
      int find(int p){
         if(p==parent[p]) return p;
         return parent[p]=find(parent[p]);
      }
      void unify(int p,int q){
         int parent_p=find(p);
         int parent_q=find(q);
         if(parent_p==parent_q) return;
         if(group[parent_p]<group[parent_q]){
            group[parent_q]+=group[parent_p];
            group[parent_p]=1;
            parent[parent_p]=parent_q;
         }
         else{
            group[parent_p]+=group[parent_q];
            group[parent_q]=1;
            parent[parent_q]=parent_p;
         }
      }
};
```

```
class Solution {
    public:
        int n;
    public:
        int removeStones(vvi& stones){
            n=stones.size();

            vi max_row_arr=*std::max_element(stones.begin(),stones.end(),[](vi& a1,vi& a2){return a1[0]<a2[0];});
            int max_row=max_row_arr[0];

            vi max_col_arr=*std::max_element(stones.begin(),stones.end(),[](vi& a1,vi& a2){return a1[1]<a2[1];});
            int max_col=max_col_arr[1];

            int _N=max_row+1+max_col+1;

            DSU dsu=DSU(_N);

            for(int i=0;i<n;++i){
                int row=stones[i][0];
                int col=stones[i][1]+max_row+1;
                dsu.unify(row,col);
            }

            int cnt_components=0;
            for(auto&g: dsu.group) cnt_components+=g>1;

            return n-cnt_components;
        }
};
```