# Problem 947: Most Stones Removed with Same Row or Column

## Problem Information

**Difficulty:** Medium
**Acceptance Rate:** 0.00%
**Paid Only:** No

## Problem Description

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] = [x

i

, y

i

]

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 <= x

i

, y

i

<= 10

4

No two stones are at the same coordinate point.

## Code Snippets

**C++:**

```
class Solution {
public:
int removeStones(vector<vector<int>>& stones) {

}
};
```

**Java:**

```
class Solution {
public int removeStones(int[][] stones) {

}
}
```

**Python3:**

```
class Solution:
def removeStones(self, stones: List[List[int]]) -> int:
```

**Python:**

```
class Solution(object):
def removeStones(self, stones):
"""
:type stones: List[List[int]]
:rtype: int
"""
```

**JavaScript:**

```
/**
* @param {number[][]} stones
* @return {number}
*/
var removeStones = function(stones) {

};
```

**TypeScript:**

```
function removeStones(stones: number[][]): number {

};
```

**C#:**

```
public class Solution {
public int RemoveStones(int[][] stones) {

}
}
```

**C:**

```
int removeStones(int** stones, int stonesSize, int* stonesColSize) {

}
```

**Go:**

```
func removeStones(stones [][]int) int {

}
```

**Kotlin:**

```
class Solution {
fun removeStones(stones: Array<IntArray>): Int {

}
}
```

**Swift:**

```
class Solution {
func removeStones(_ stones: [[Int]]) -> Int {

}
}
```

**Rust:**

```
impl Solution {
pub fn remove_stones(stones: Vec<Vec<i32>>) -> i32 {

}
}
```

**Ruby:**

```
# @param {Integer[][]} stones
# @return {Integer}
def remove_stones(stones)

end
```

**PHP:**

```
class Solution {

/**
* @param Integer[][] $stones
* @return Integer
*/
function removeStones($stones) {

}
}
```

**Dart:**

```
class Solution {
int removeStones(List<List<int>> stones) {

}
```

```
}
```

**Scala:**

```
object Solution {
def removeStones(stones: Array[Array[Int]]): Int = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec remove_stones(stones :: [[integer]]) :: integer
def remove_stones(stones) do

end
end
```

**Erlang:**

```
-spec remove_stones(Stones :: [[integer()]]) -> integer().
remove_stones(Stones) ->
.
```

**Racket:**

```
(define/contract (remove-stones stones)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Most Stones Removed with Same Row or Column
* Difficulty: Medium
* Tags: array, graph, hash, search
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int removeStones(vector<vector<int>>& stones) {

}
};
```

### Java Solution:

```
/**
* Problem: Most Stones Removed with Same Row or Column
* Difficulty: Medium
* Tags: array, graph, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int removeStones(int[][] stones) {

}
}
```

### Python3 Solution:

```
"""
Problem: Most Stones Removed with Same Row or Column
Difficulty: Medium
Tags: array, graph, hash, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""
```

```
class Solution:
def removeStones(self, stones: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

### Python Solution:

```
class Solution(object):
def removeStones(self, stones):
"""
:type stones: List[List[int]]
:rtype: int
"""
```

### JavaScript Solution:

```
/**
* Problem: Most Stones Removed with Same Row or Column
* Difficulty: Medium
* Tags: array, graph, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {number[][]} stones
* @return {number}
*/
var removeStones = function(stones) {

};
```

### TypeScript Solution:

```
/**
* Problem: Most Stones Removed with Same Row or Column
* Difficulty: Medium
* Tags: array, graph, hash, search
```

```
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function removeStones(stones: number[][]): number {

};
```

### C# Solution:

```
/*
* Problem: Most Stones Removed with Same Row or Column
* Difficulty: Medium
* Tags: array, graph, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int RemoveStones(int[][] stones) {

}
}
```

### C Solution:

```
/*
* Problem: Most Stones Removed with Same Row or Column
* Difficulty: Medium
* Tags: array, graph, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int removeStones(int** stones, int stonesSize, int* stonesColSize) {
```

```
}
```

### Go Solution:

```
// Problem: Most Stones Removed with Same Row or Column
// Difficulty: Medium
// Tags: array, graph, hash, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func removeStones(stones [][]int) int {

}
```

### Kotlin Solution:

```
class Solution {
fun removeStones(stones: Array<IntArray>): Int {

}
}
```

### Swift Solution:

```
class Solution {
func removeStones(_ stones: [[Int]]) -> Int {

}
}
```

### Rust Solution:

```
// Problem: Most Stones Removed with Same Row or Column
// Difficulty: Medium
// Tags: array, graph, hash, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
```

```
// Space Complexity: O(n) for hash map

impl Solution {
pub fn remove_stones(stones: Vec<Vec<i32>>) -> i32 {

}
}
```

**Ruby Solution:**

```
# @param {Integer[][]} stones
# @return {Integer}
def remove_stones(stones)

end
```

**PHP Solution:**

```
class Solution {

/**
* @param Integer[][] $stones
* @return Integer
*/
function removeStones($stones) {

}
}
```

**Dart Solution:**

```
class Solution {
int removeStones(List<List<int>> stones) {

}
}
```

**Scala Solution:**

```
object Solution {
def removeStones(stones: Array[Array[Int]]): Int = {
```

```
}
}
```

### Elixir Solution:

```
defmodule Solution do
@spec remove_stones(stones :: [[integer]]) :: integer
def remove_stones(stones) do

end
end
```

### Erlang Solution:

```
-spec remove_stones(Stones :: [[integer()]]) -> integer().
remove_stones(Stones) ->
.
```

### Racket Solution:

```
(define/contract (remove-stones stones)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```