Problem 2101: Detonate the Maximum Bombs

Problem Information

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

Problem Description

You are given a list of bombs. The

range

of a bomb is defined as the area where its effect can be felt. This area is in the shape of a

circle

with the center as the location of the bomb.

The bombs are represented by a

0-indexed

2D integer array

bombs

where

bombs[i] = [x

i

, y

i

, r

i

]

.

x

i

and

y

i

denote the X-coordinate and Y-coordinate of the location of the

i

th

bomb, whereas

r

i

denotes the

radius

of its range.

You may choose to detonate a

single

bomb. When a bomb is detonated, it will detonate

all bombs

that lie in its range. These bombs will further detonate the bombs that lie in their ranges.

Given the list of

bombs

, return

the

maximum

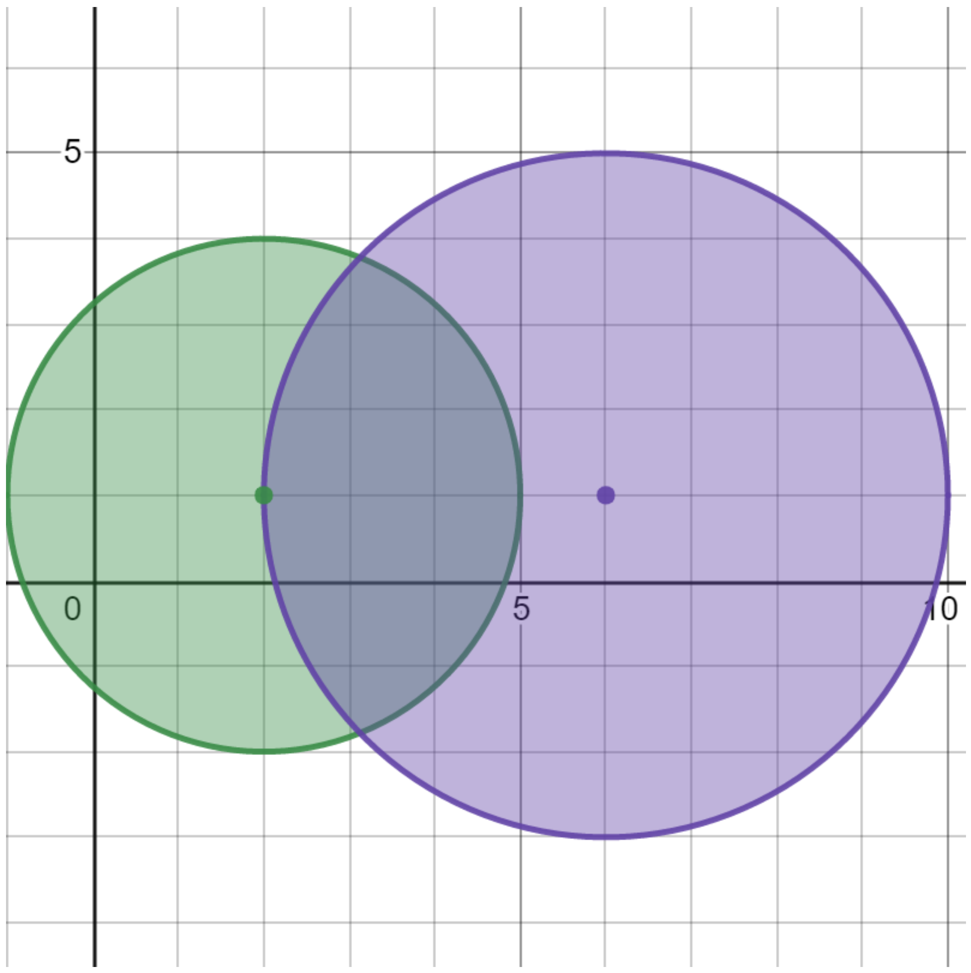
number of bombs that can be detonated if you are allowed to detonate

only one

bomb

.

Example 1:



Input:

bombs = [[2,1,3],[6,1,4]]

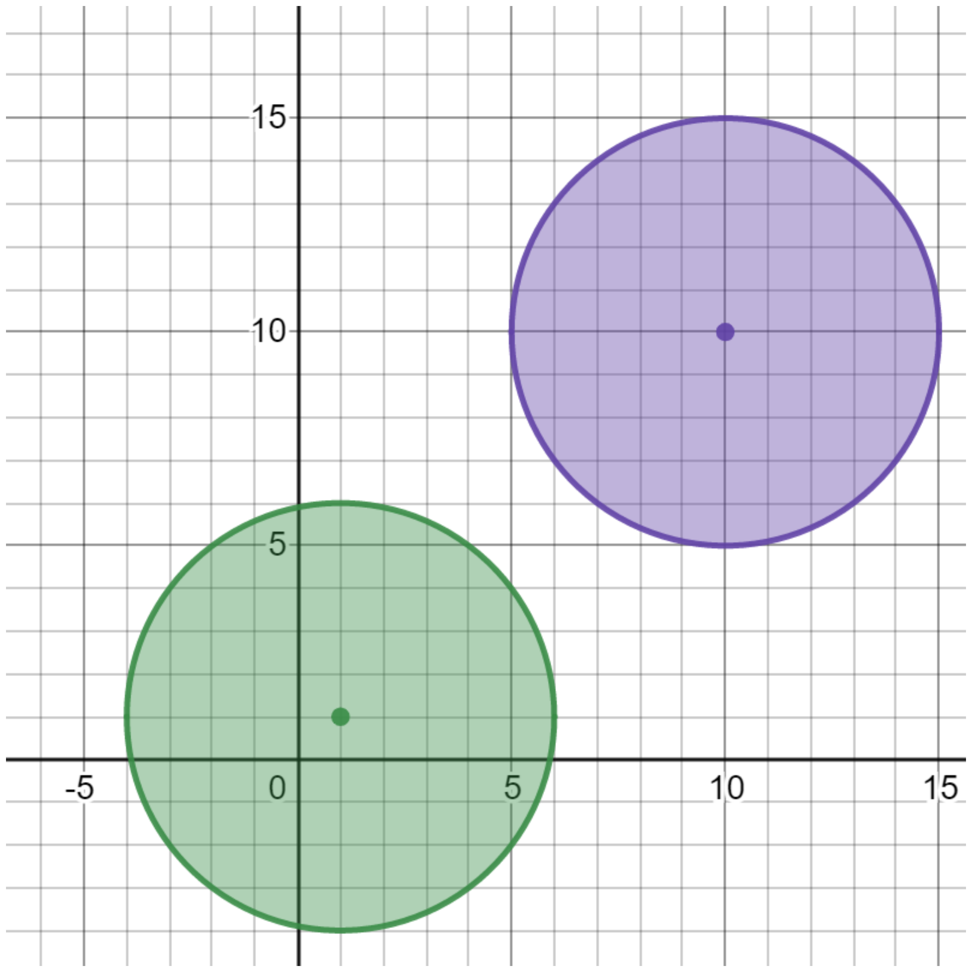
Output:

2

Explanation:

The above figure shows the positions and ranges of the 2 bombs. If we detonate the left bomb, the right bomb will not be affected. But if we detonate the right bomb, both bombs will be detonated. So the maximum bombs that can be detonated is max(1, 2) = 2.

Example 2:



Input:

bombs = [[1,1,5],[10,10,5]]

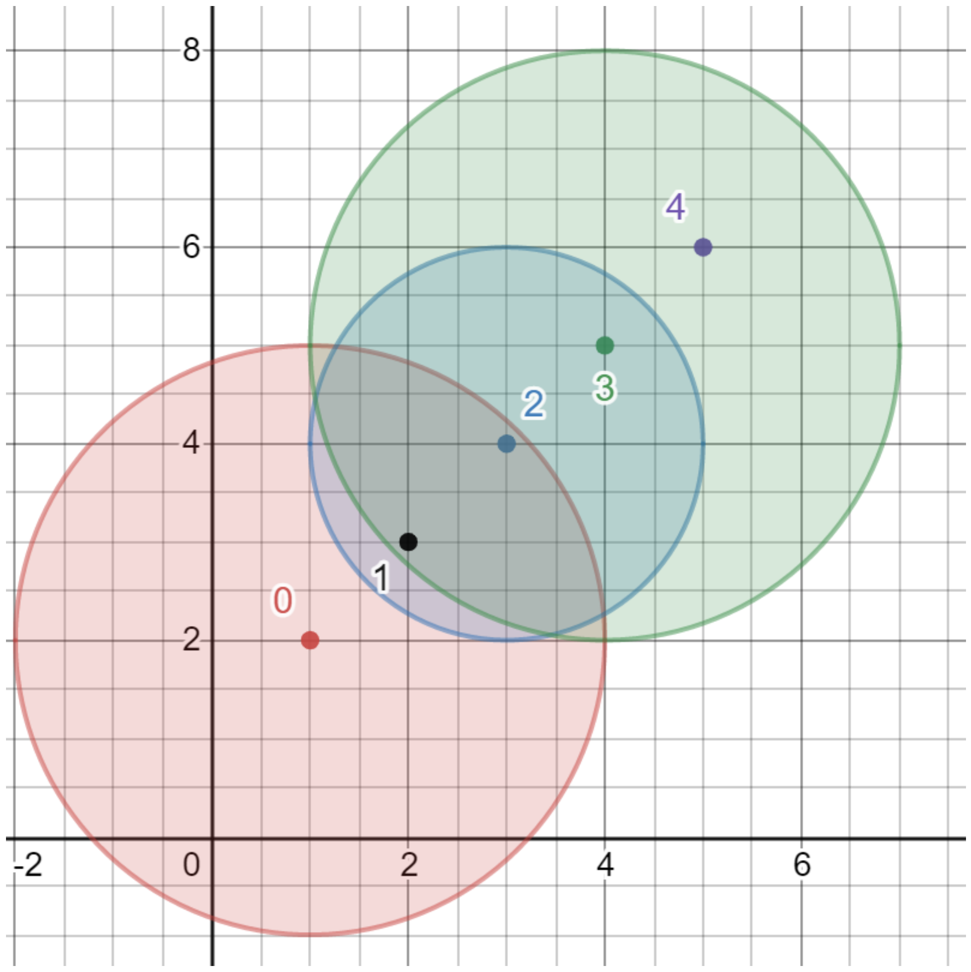
Output:

1

Explanation:

Detonating either bomb will not detonate the other bomb, so the maximum number of bombs that can be detonated is 1.

Example 3:



Input:

bombs = [[1,2,3],[2,3,1],[3,4,2],[4,5,3],[5,6,4]]

Output:

5

Explanation:

The best bomb to detonate is bomb 0 because: - Bomb 0 detonates bombs 1 and 2. The red circle denotes the range of bomb 0. - Bomb 2 detonates bomb 3. The blue circle denotes the range of bomb 2. - Bomb 3 detonates bomb 4. The green circle denotes the range of bomb 3. Thus all 5 bombs are detonated.

Constraints:

1 <= bombs.length <= 100

bombs[i].length == 3

1 <= x

i

, y

i

, r

i

<= 10

5

Code Snippets

C++:

```
class Solution {
public:
int maximumDetonation(vector<vector<int>>& bombs) {

}
};
```

Java:

```
class Solution {
public int maximumDetonation(int[][] bombs) {

}
}
```

Python3:

```
class Solution:
def maximumDetonation(self, bombs: List[List[int]]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function maximumDetonation(bombs: number[][]): number {

};
```

C#:

```
public class Solution {
public int MaximumDetonation(int[][] bombs) {

}
}
```

C:

```
int maximumDetonation(int** bombs, int bombsSize, int* bombsColSize) {

}
```

Go:

```
func maximumDetonation(bombs [][]int) int {

}
```

Kotlin:

```
class Solution {
fun maximumDetonation(bombs: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func maximumDetonation(_ bombs: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn maximum_detonation(bombs: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[][]} bombs
# @return {Integer}
def maximum_detonation(bombs)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $bombs
* @return Integer
*/
function maximumDetonation($bombs) {

}
}
```

Dart:

```
class Solution {
int maximumDetonation(List<List<int>> bombs) {

}
}
```

Scala:

```
object Solution {
def maximumDetonation(bombs: Array[Array[Int]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec maximum_detonation(bombs :: [[integer]]) :: integer
def maximum_detonation(bombs) do

end
end
```

Erlang:

```
-spec maximum_detonation(Bombs :: [[integer()]]) -> integer().
maximum_detonation(Bombs) ->
.
```

Racket:

```
(define/contract (maximum-detonation bombs)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Detonate the Maximum Bombs
* Difficulty: Medium
* Tags: array, graph, math, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int maximumDetonation(vector<vector<int>>& bombs) {

}
};
```

Java Solution:

```
/**
* Problem: Detonate the Maximum Bombs
* Difficulty: Medium
* Tags: array, graph, math, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int maximumDetonation(int[][] bombs) {

}
}
```

Python3 Solution:

```
"""
Problem: Detonate the Maximum Bombs
Difficulty: Medium
Tags: array, graph, math, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Detonate the Maximum Bombs
* Difficulty: Medium
* Tags: array, graph, math, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[][]} bombs
* @return {number}
*/
var maximumDetonation = function(bombs) {
};
```

TypeScript Solution:

```
/**
* Problem: Detonate the Maximum Bombs
* Difficulty: Medium
* Tags: array, graph, math, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function maximumDetonation(bombs: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Detonate the Maximum Bombs
* Difficulty: Medium
* Tags: array, graph, math, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int MaximumDetonation(int[][] bombs) {

}
}
```

C Solution:

```
/*
* Problem: Detonate the Maximum Bombs
* Difficulty: Medium
* Tags: array, graph, math, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int maximumDetonation(int** bombs, int bombsSize, int* bombsColSize) {

}
```

Go Solution:

```
// Problem: Detonate the Maximum Bombs
// Difficulty: Medium
// Tags: array, graph, math, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func maximumDetonation(bombs [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maximumDetonation(bombs: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func maximumDetonation(_ bombs: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Detonate the Maximum Bombs
// Difficulty: Medium
// Tags: array, graph, math, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn maximum_detonation(bombs: Vec<Vec<i32>>) -> i32 {


}
}
```

Ruby Solution:

```
# @param {Integer[][]} bombs
# @return {Integer}
def maximum_detonation(bombs)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $bombs
* @return Integer
*/
function maximumDetonation($bombs) {

}
}
```

Dart Solution:

```
class Solution {
int maximumDetonation(List<List<int>> bombs) {

}
}
```

Scala Solution:

```
object Solution {
def maximumDetonation(bombs: Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec maximum_detonation(bombs :: [[integer]]) :: integer
def maximum_detonation(bombs) do

end
end
```

Erlang Solution:

```
-spec maximum_detonation(Bombs :: [[integer()]]) -> integer().
maximum_detonation(Bombs) ->
.
```

Racket Solution:

```
(define/contract (maximum-detonation bombs)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```