Problem 2249: Count Lattice Points Inside a Circle

Problem Information

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

Problem Description

Given a 2D integer array

circles

where

circles[i] = [x

i

, y

i

, r

i

]

represents the center

(x

i

, y

i

)

and radius

r

i

of the

i

th

circle drawn on a grid, return

the

number of lattice points

that are present inside

at least one

circle

.

Note:

A

lattice point

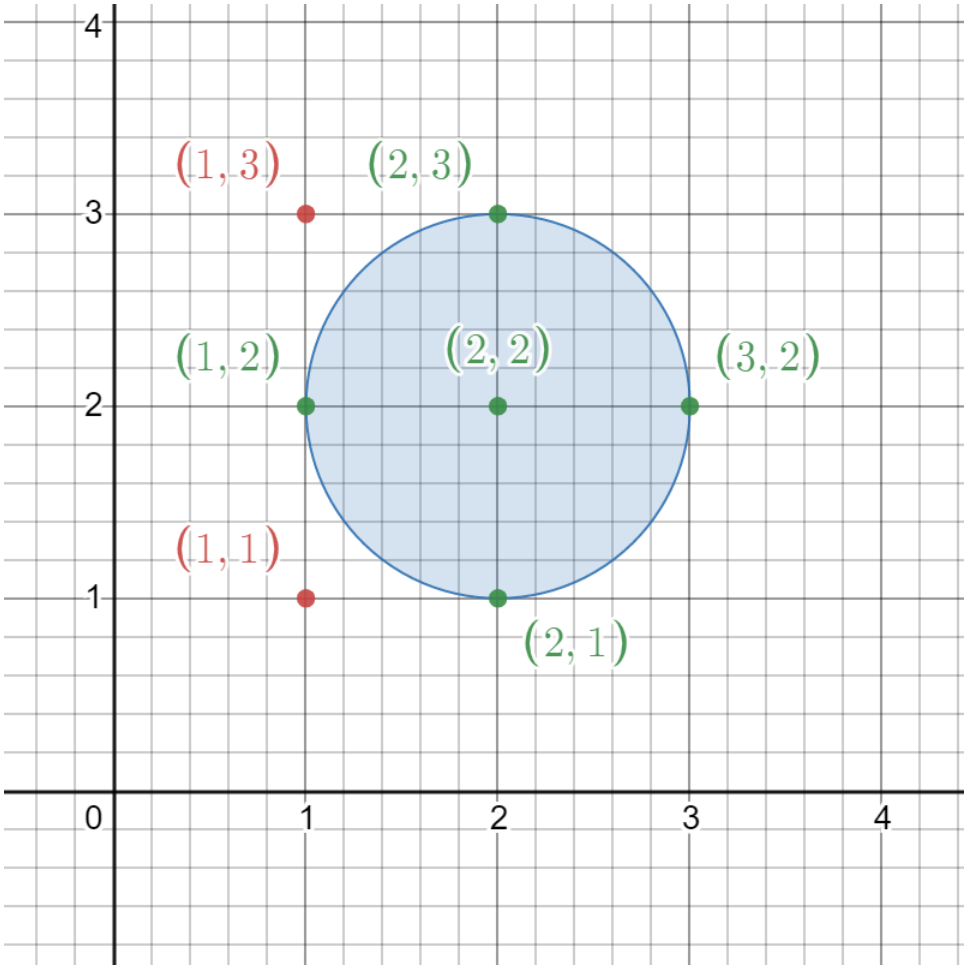
is a point with integer coordinates.

Points that lie

on the circumference of a circle

are also considered to be inside it.

Example 1:



Input:

circles = [[2,2,1]]

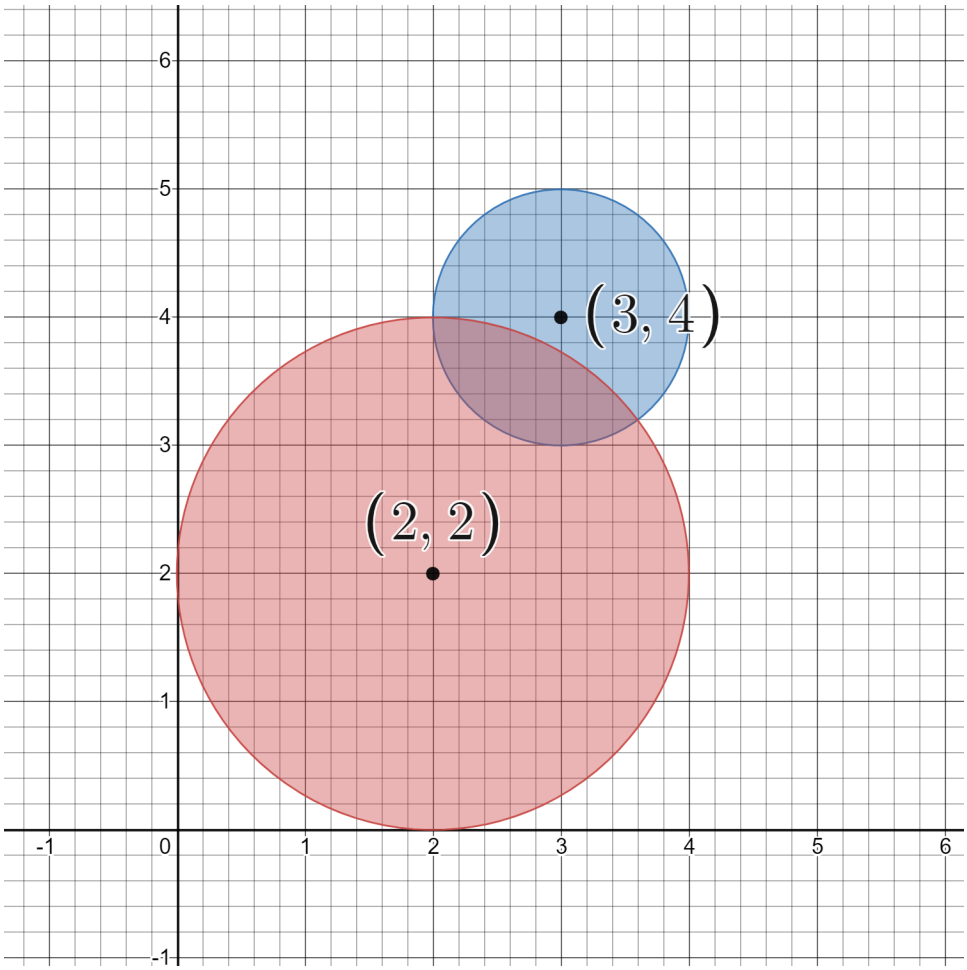
Output:

5

Explanation:

The figure above shows the given circle. The lattice points present inside the circle are (1, 2), (2, 1), (2, 2), (2, 3), and (3, 2) and are shown in green. Other points such as (1, 1) and (1, 3), which are shown in red, are not considered inside the circle. Hence, the number of lattice points present inside at least one circle is 5.

Example 2:



Input:

circles = [[2,2,2],[3,4,1]]

Output:

16

Explanation:

The figure above shows the given circles. There are exactly 16 lattice points which are present inside at least one circle. Some of them are (0, 2), (2, 0), (2, 4), (3, 2), and (4, 4).

Constraints:

1 <= circles.length <= 200

circles[i].length == 3

1 <= x

i

, y

i

<= 100

1 <= r

i

<= min(x

i

, y

i

)

Code Snippets

C++:

```
class Solution {
public:
int countLatticePoints(vector<vector<int>>& circles) {

}
};
```

Java:

```
class Solution {
public int countLatticePoints(int[][] circles) {

}
}
```

Python3:

```
class Solution:
def countLatticePoints(self, circles: List[List[int]]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function countLatticePoints(circles: number[][]): number {
```

```
};
```

C#:

```
public class Solution {
public int CountLatticePoints(int[][] circles) {

}
}
```

C:

```
int countLatticePoints(int** circles, int circlesSize, int* circlesColSize) {

}
```

Go:

```
func countLatticePoints(circles [][]int) int {

}
```

Kotlin:

```
class Solution {
fun countLatticePoints(circles: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func countLatticePoints(_ circles: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_lattice_points(circles: Vec<Vec<i32>>) -> i32 {
```

```
}
}
```

Ruby:

```
# @param {Integer[][]} circles
# @return {Integer}
def count_lattice_points(circles)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $circles
* @return Integer
*/
function countLatticePoints($circles) {

}
}
```

Dart:

```
class Solution {
int countLatticePoints(List<List<int>> circles) {

}
}
```

Scala:

```
object Solution {
def countLatticePoints(circles: Array[Array[Int]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_lattice_points(circles :: [[integer]]) :: integer
def count_lattice_points(circles) do

end
end
```

Erlang:

```
-spec count_lattice_points(Circles :: [[integer()]]) -> integer().
count_lattice_points(Circles) ->
.
```

Racket:

```
(define/contract (count-lattice-points circles)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Lattice Points Inside a Circle
* Difficulty: Medium
* Tags: array, math, hash
*
* 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 countLatticePoints(vector<vector<int>>& circles) {

}
};
```

Java Solution:

```
/**
* Problem: Count Lattice Points Inside a Circle
* Difficulty: Medium
* Tags: array, math, hash
*
* 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 countLatticePoints(int[][] circles) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Lattice Points Inside a Circle
Difficulty: Medium
Tags: array, math, hash

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 countLatticePoints(self, circles: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Lattice Points Inside a Circle
* Difficulty: Medium
* Tags: array, math, hash
*
* 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[][]} circles
* @return {number}
*/
var countLatticePoints = function(circles) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Lattice Points Inside a Circle
* Difficulty: Medium
* Tags: array, math, hash
*
* 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 countLatticePoints(circles: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Count Lattice Points Inside a Circle
* Difficulty: Medium
* Tags: array, math, hash
*
```

```
* 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 CountLatticePoints(int[][] circles) {

}
}
```

C Solution:

```
/*
* Problem: Count Lattice Points Inside a Circle
* Difficulty: Medium
* Tags: array, math, hash
*
* 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 countLatticePoints(int** circles, int circlesSize, int* circlesColSize) {

}
```

Go Solution:

```
// Problem: Count Lattice Points Inside a Circle
// Difficulty: Medium
// Tags: array, math, hash
//
// 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 countLatticePoints(circles [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countLatticePoints(circles: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func countLatticePoints(_ circles: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count Lattice Points Inside a Circle
// Difficulty: Medium
// Tags: array, math, hash
//
// 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 count_lattice_points(circles: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} circles
# @return {Integer}
def count_lattice_points(circles)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $circles
* @return Integer
*/
function countLatticePoints($circles) {

}
}
```

Dart Solution:

```
class Solution {
int countLatticePoints(List<List<int>> circles) {

}
}
```

Scala Solution:

```
object Solution {
def countLatticePoints(circles: Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_lattice_points(circles :: [[integer]]) :: integer
def count_lattice_points(circles) do

end
end
```

Erlang Solution:

```
-spec count_lattice_points(Circles :: [[integer()]]) -> integer().
count_lattice_points(Circles) ->
.
```

Racket Solution:

```
(define/contract (count-lattice-points circles)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```