Problem 323: Number of Connected Components in an Undirected Graph

Problem Information

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

Problem Description

You have a graph of

n

nodes. You are given an integer

n

and an array

edges

where

edges[i] = [a

i

, b

i

]

indicates that there is an edge between

a

i

and

b

i

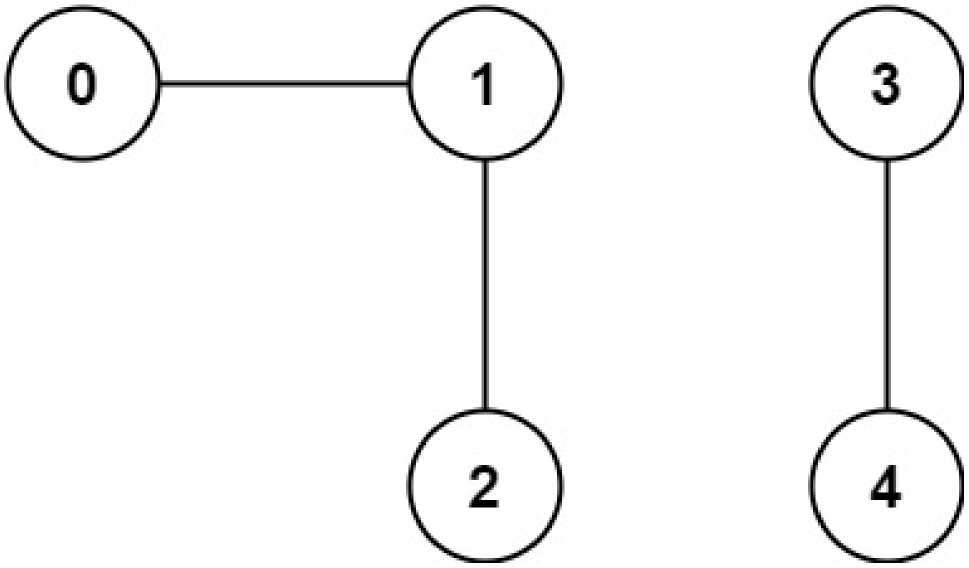
in the graph.

Return

the number of connected components in the graph

.

Example 1:



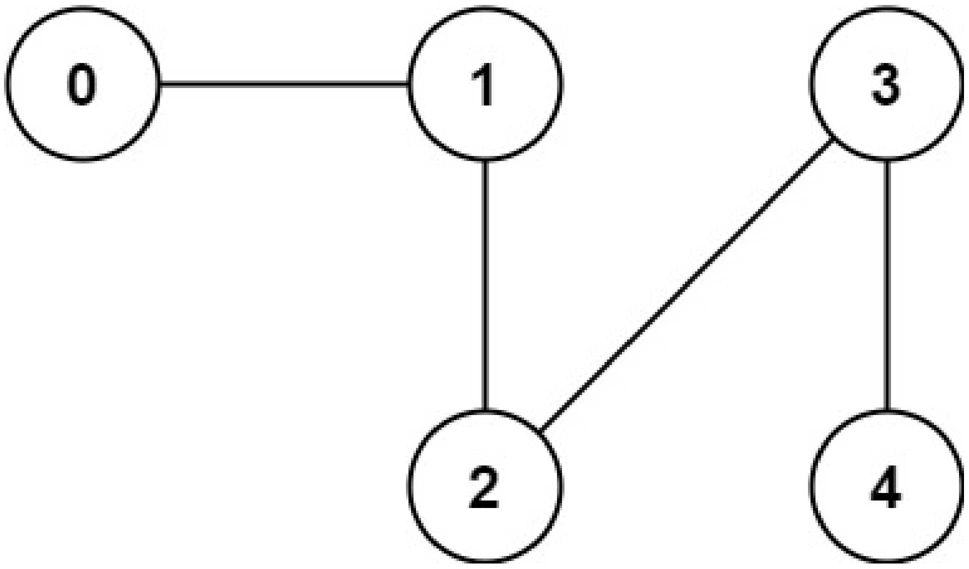
Input:

n = 5, edges = [[0,1],[1,2],[3,4]]

Output:

2

Example 2:



Input:

n = 5, edges = [[0,1],[1,2],[2,3],[3,4]]

Output:

1

Constraints:

1 <= n <= 2000

1 <= edges.length <= 5000

edges[i].length == 2

0 <= a

i

<= b

i

< n

a

i

!= b

i

There are no repeated edges.

Code Snippets

C++:

```
class Solution {
public:
int countComponents(int n, vector<vector<int>>& edges) {

}
};
```

Java:

```
class Solution {
public int countComponents(int n, int[][] edges) {

}
}
```

Python3:

```
class Solution:
def countComponents(self, n: int, edges: List[List[int]]) -> int:
```

Python:

```
class Solution(object):
def countComponents(self, n, edges):
"""
:type n: int
:type edges: List[List[int]]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number[][]} edges
* @return {number}
*/
var countComponents = function(n, edges) {

};
```

TypeScript:

```
function countComponents(n: number, edges: number[][]): number {

};
```

C#:

```
public class Solution {
public int CountComponents(int n, int[][] edges) {

}
}
```

C:

```
int countComponents(int n, int** edges, int edgesSize, int* edgesColSize) {

}
```

Go:

```
func countComponents(n int, edges [][]int) int {
}
```

Kotlin:

```
class Solution {
fun countComponents(n: Int, edges: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func countComponents(_ n: Int, _ edges: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_components(n: i32, edges: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer[][]} edges
# @return {Integer}
def count_components(n, edges)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer[][] $edges
* @return Integer
*/
function countComponents($n, $edges) {

}
}
```

Dart:

```
class Solution {
int countComponents(int n, List<List<int>> edges) {

}
}
```

Scala:

```
object Solution {
def countComponents(n: Int, edges: Array[Array[Int]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_components(n :: integer, edges :: [[integer]]) :: integer
def count_components(n, edges) do

end
end
```

Erlang:

```
-spec count_components(N :: integer(), Edges :: [[integer()]]) -> integer().
count_components(N, Edges) ->
.
```

Racket:

```
(define/contract (count-components n edges)
(-> exact-integer? (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Connected Components in an Undirected Graph
* Difficulty: Medium
* Tags: array, graph, 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 countComponents(int n, vector<vector<int>>& edges) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Connected Components in an Undirected Graph
* Difficulty: Medium
* Tags: array, graph, 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 countComponents(int n, int[][] edges) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Connected Components in an Undirected Graph
Difficulty: Medium
Tags: array, graph, 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 countComponents(self, n: int, edges: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countComponents(self, n, edges):
"""
:type n: int
:type edges: List[List[int]]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Connected Components in an Undirected Graph
* Difficulty: Medium
* Tags: array, graph, 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} n
* @param {number[][]} edges
* @return {number}
*/
var countComponents = function(n, edges) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Connected Components in an Undirected Graph
* Difficulty: Medium
* Tags: array, graph, 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 countComponents(n: number, edges: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Number of Connected Components in an Undirected Graph
* Difficulty: Medium
* Tags: array, graph, 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 CountComponents(int n, int[][] edges) {

}
}
```

C Solution:

```
/*
* Problem: Number of Connected Components in an Undirected Graph
* Difficulty: Medium
* Tags: array, graph, 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 countComponents(int n, int** edges, int edgesSize, int* edgesColSize) {

}
```

Go Solution:

```
// Problem: Number of Connected Components in an Undirected Graph
// Difficulty: Medium
// Tags: array, graph, 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 countComponents(n int, edges [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countComponents(n: Int, edges: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func countComponents(_ n: Int, _ edges: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Connected Components in an Undirected Graph
// Difficulty: Medium
// Tags: array, graph, 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 count_components(n: i32, edges: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer[][]} edges
# @return {Integer}
def count_components(n, edges)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer[][] $edges
* @return Integer
*/
function countComponents($n, $edges) {

}
}
```

Dart Solution:

```
class Solution {
int countComponents(int n, List<List<int>> edges) {

}
}
```

Scala Solution:

```
object Solution {
def countComponents(n: Int, edges: Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_components(n :: integer, edges :: [[integer]]) :: integer
def count_components(n, edges) do

end
end
```

Erlang Solution:

```
-spec count_components(N :: integer(), Edges :: [[integer()]]) -> integer().
count_components(N, Edges) ->
.
```

Racket Solution:

```
(define/contract (count-components n edges)
(-> exact-integer? (listof (listof exact-integer?)) exact-integer?)
)
```