Problem 1791: Find Center of Star Graph

Problem Information

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

Problem Description

There is an undirected

star

graph consisting of

n

nodes labeled from

1

to

n

. A star graph is a graph where there is one

center

node and

exactly

n - 1

edges that connect the center node with every other node.

You are given a 2D integer array

edges

where each

edges[i] = [u

i

, v

i

]

indicates that there is an edge between the nodes

u

i

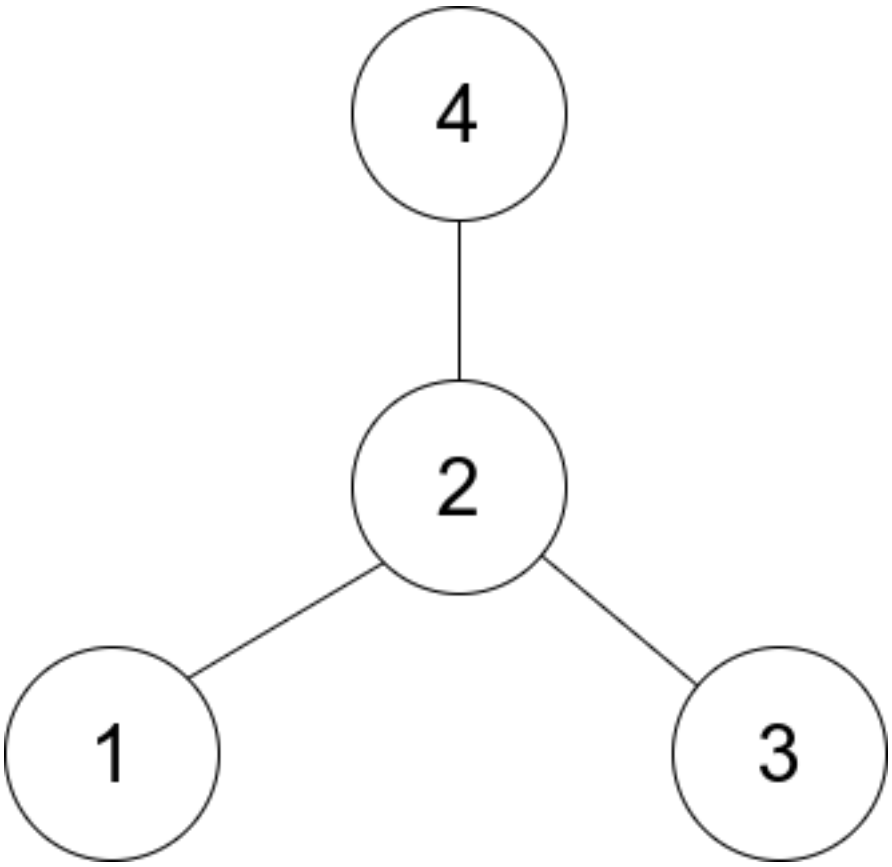
and

v

i

. Return the center of the given star graph.

Example 1:



Input:

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

Output:

2

Explanation:

As shown in the figure above, node 2 is connected to every other node, so 2 is the center.

Example 2:

Input:

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

Output:

1

Constraints:

3 <= n <= 10

5

edges.length == n - 1

edges[i].length == 2

1 <= u

i,

v

i

<= n

u

i

!= v

i

The given

edges

represent a valid star graph.

Code Snippets

C++:

```
class Solution {
public:
int findCenter(vector<vector<int>>& edges) {

}
};
```

Java:

```
class Solution {
public int findCenter(int[][] edges) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function findCenter(edges: number[][]): number {

};
```

C#:

```
public class Solution {
public int FindCenter(int[][] edges) {

}
}
```

C:

```
int findCenter(int** edges, int edgesSize, int* edgesColSize) {

}
```

Go:

```
func findCenter(edges [][]int) int {

}
```

Kotlin:

```
class Solution {
fun findCenter(edges: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func findCenter(_ edges: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn find_center(edges: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[][]} edges
# @return {Integer}
def find_center(edges)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $edges
* @return Integer
*/
function findCenter($edges) {

}
}
```

Dart:

```
class Solution {
int findCenter(List<List<int>> edges) {

}
}
```

Scala:

```
object Solution {
def findCenter(edges: Array[Array[Int]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec find_center(edges :: [[integer]]) :: integer
def find_center(edges) do

end
end
```

Erlang:

```
-spec find_center(Edges :: [[integer()]]) -> integer().
find_center(Edges) ->
.
```

Racket:

```
(define/contract (find-center edges)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Center of Star Graph
* Difficulty: Easy
* Tags: array, graph
*
* 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 findCenter(vector<vector<int>>& edges) {

}
};
```

Java Solution:

```
/**
* Problem: Find Center of Star Graph
* Difficulty: Easy
* Tags: array, graph
*
* 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 findCenter(int[][] edges) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Center of Star Graph
Difficulty: Easy
Tags: array, graph

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

Python Solution:

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

```
"""
```

JavaScript Solution:

```
/**
* Problem: Find Center of Star Graph
* Difficulty: Easy
* Tags: array, graph
*
* 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[][]} edges
* @return {number}
*/
var findCenter = function(edges) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Center of Star Graph
* Difficulty: Easy
* Tags: array, graph
*
* 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 findCenter(edges: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Find Center of Star Graph
* Difficulty: Easy
* Tags: array, graph
*
* 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 FindCenter(int[][] edges) {

}
}
```

C Solution:

```
/*
* Problem: Find Center of Star Graph
* Difficulty: Easy
* Tags: array, graph
*
* 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 findCenter(int** edges, int edgesSize, int* edgesColSize) {

}
```

Go Solution:

```
// Problem: Find Center of Star Graph
// Difficulty: Easy
// Tags: array, graph
//
// 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 findCenter(edges [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun findCenter(edges: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func findCenter(_ edges: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find Center of Star Graph
// Difficulty: Easy
// Tags: array, graph
//
// 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 find_center(edges: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} edges
# @return {Integer}
def find_center(edges)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $edges
* @return Integer
*/
function findCenter($edges) {

}
}
```

Dart Solution:

```
class Solution {
int findCenter(List<List<int>> edges) {

}
}
```

Scala Solution:

```
object Solution {
def findCenter(edges: Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec find_center(edges :: [[integer]]) :: integer
def find_center(edges) do

end
end
```

Erlang Solution:

```
-spec find_center(Edges :: [[integer()]]) -> integer().
find_center(Edges) ->
.
```

Racket Solution:

```
(define/contract (find-center edges)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```