Problem 3656: Determine if a Simple Graph Exists

Problem Information

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

Problem Description

You are given an integer array

degrees

, where

degrees[i]

represents the desired degree of the

i

th

vertex.

Your task is to determine if there exists an

undirected simple

graph with

exactly

these vertex degrees.

A

simple

graph has no self-loops or parallel edges between the same pair of vertices.

Return

true

if such a graph exists, otherwise return

false

.

Example 1:

Input:

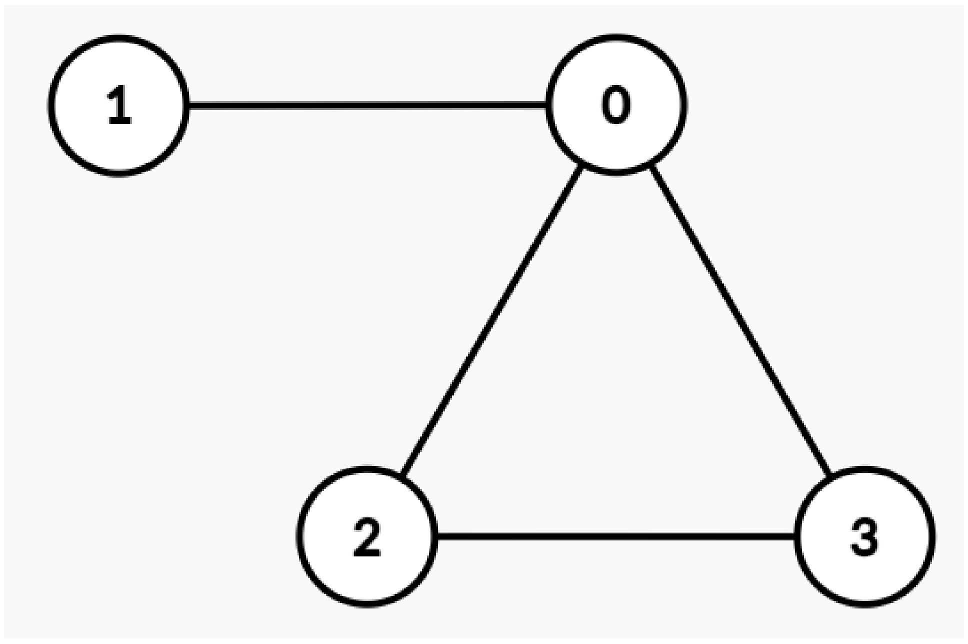
degrees = [3,1,2,2]

Output:

true

Explanation:

,



One possible undirected simple graph is:

Edges:

(0, 1), (0, 2), (0, 3), (2, 3)

Degrees:

deg(0) = 3

,

deg(1) = 1

,

deg(2) = 2

,

deg(3) = 2

.

Example 2:

Input:

degrees = [1,3,3,1]

Output:

false

Explanation:

degrees[1] = 3

and

degrees[2] = 3

means they must be connected to all other vertices.

This requires

degrees[0]

and

degrees[3]

to be at least 2, but both are equal to 1, which contradicts the requirement.

Thus, the answer is

false

.

Constraints:

1 <= n == degrees.length <= 10

5

0 <= degrees[i] <= n - 1

Code Snippets

C++:

```
class Solution {
public:
bool simpleGraphExists(vector<int>& degrees) {

}
};
```

Java:

```
class Solution {
public boolean simpleGraphExists(int[] degrees) {

}
}
```

Python3:

```
class Solution:
def simpleGraphExists(self, degrees: List[int]) -> bool:
```

Python:

```
class Solution(object):
def simpleGraphExists(self, degrees):
"""
:type degrees: List[int]
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {number[]} degrees
* @return {boolean}
*/
var simpleGraphExists = function(degrees) {

};
```

TypeScript:

```
function simpleGraphExists(degrees: number[]): boolean {

};
```

C#:

```
public class Solution {
public bool SimpleGraphExists(int[] degrees) {

}
}
```

C:

```
bool simpleGraphExists(int* degrees, int degreesSize) {

}
```

Go:

```
func simpleGraphExists(degrees []int) bool {

}
```

Kotlin:

```
class Solution {
fun simpleGraphExists(degrees: IntArray): Boolean {

}
}
```

Swift:

```
class Solution {
func simpleGraphExists(_ degrees: [Int]) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn simple_graph_exists(degrees: Vec<i32>) -> bool {

}
}
```

Ruby:

```
# @param {Integer[]} degrees
# @return {Boolean}
def simple_graph_exists(degrees)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $degrees
* @return Boolean
*/
function simpleGraphExists($degrees) {

}
}
```

Dart:

```
class Solution {
bool simpleGraphExists(List<int> degrees) {

}
}
```

Scala:

```
object Solution {
def simpleGraphExists(degrees: Array[Int]): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec simple_graph_exists(degrees :: [integer]) :: boolean
def simple_graph_exists(degrees) do

end
end
```

Erlang:

```
-spec simple_graph_exists(Degrees :: [integer()]) -> boolean().
simple_graph_exists(Degrees) ->
.
```

Racket:

```
(define/contract (simple-graph-exists degrees)
(-> (listof exact-integer?) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Determine if a Simple Graph Exists
* Difficulty: Medium
* Tags: array, graph, sort, 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:
bool simpleGraphExists(vector<int>& degrees) {

}
};
```

Java Solution:

```
/**
* Problem: Determine if a Simple Graph Exists
* Difficulty: Medium
* Tags: array, graph, sort, 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 boolean simpleGraphExists(int[] degrees) {

}
}
```

Python3 Solution:

```
"""
Problem: Determine if a Simple Graph Exists
Difficulty: Medium
Tags: array, graph, sort, 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 simpleGraphExists(self, degrees: List[int]) -> bool:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
def simpleGraphExists(self, degrees):
"""
:type degrees: List[int]
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Determine if a Simple Graph Exists
* Difficulty: Medium
* Tags: array, graph, sort, 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[]} degrees
* @return {boolean}
*/
var simpleGraphExists = function(degrees) {

};
```

TypeScript Solution:

```
/**
* Problem: Determine if a Simple Graph Exists
* Difficulty: Medium
* Tags: array, graph, sort, 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 simpleGraphExists(degrees: number[]): boolean {

};
```

C# Solution:

```
/*
* Problem: Determine if a Simple Graph Exists
* Difficulty: Medium
* Tags: array, graph, sort, 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 bool SimpleGraphExists(int[] degrees) {

}
}
```

C Solution:

```
/*
* Problem: Determine if a Simple Graph Exists
* Difficulty: Medium
* Tags: array, graph, sort, 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
*/

bool simpleGraphExists(int* degrees, int degreesSize) {

}
```

Go Solution:

```
// Problem: Determine if a Simple Graph Exists
// Difficulty: Medium
// Tags: array, graph, sort, 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 simpleGraphExists(degrees []int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun simpleGraphExists(degrees: IntArray): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func simpleGraphExists(_ degrees: [Int]) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Determine if a Simple Graph Exists
// Difficulty: Medium
// Tags: array, graph, sort, 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 simple_graph_exists(degrees: Vec<i32>) -> bool {

}

}
```

Ruby Solution:

```
# @param {Integer[]} degrees
# @return {Boolean}
def simple_graph_exists(degrees)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $degrees
* @return Boolean
*/
function simpleGraphExists($degrees) {

}
}
```

Dart Solution:

```
class Solution {
bool simpleGraphExists(List<int> degrees) {

}
}
```

Scala Solution:

```
object Solution {
def simpleGraphExists(degrees: Array[Int]): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec simple_graph_exists(degrees :: [integer]) :: boolean
def simple_graph_exists(degrees) do

end
end
```

Erlang Solution:

```
-spec simple_graph_exists(Degrees :: [integer()]) -> boolean().
simple_graph_exists(Degrees) ->
.
```

Racket Solution:

```
(define/contract (simple-graph-exists degrees)
(-> (listof exact-integer?) boolean?)
)
```