Problem 2764: Is Array a Preorder of Some ■Binary Tree

Problem Information

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

Problem Description

Given a

0-indexed

integer

2D array

nodes

, your task is to determine if the given array represents the

preorder

traversal of some

binary

tree.

For each index

i

,

nodes[i] = [id, parentId]

, where

id

is the id of the node at the index

i

and

parentId

is the id of its parent in the tree (if the node has no parent, then

parentId == -1

).

Return

true

if the given array

represents the preorder traversal of some tree, and

false

otherwise.

Note:

the

preorder

traversal of a tree is a recursive way to traverse a tree in which we first visit the current node, then we do the preorder traversal for the left child, and finally, we do it for the right child.

Example 1:

Input:

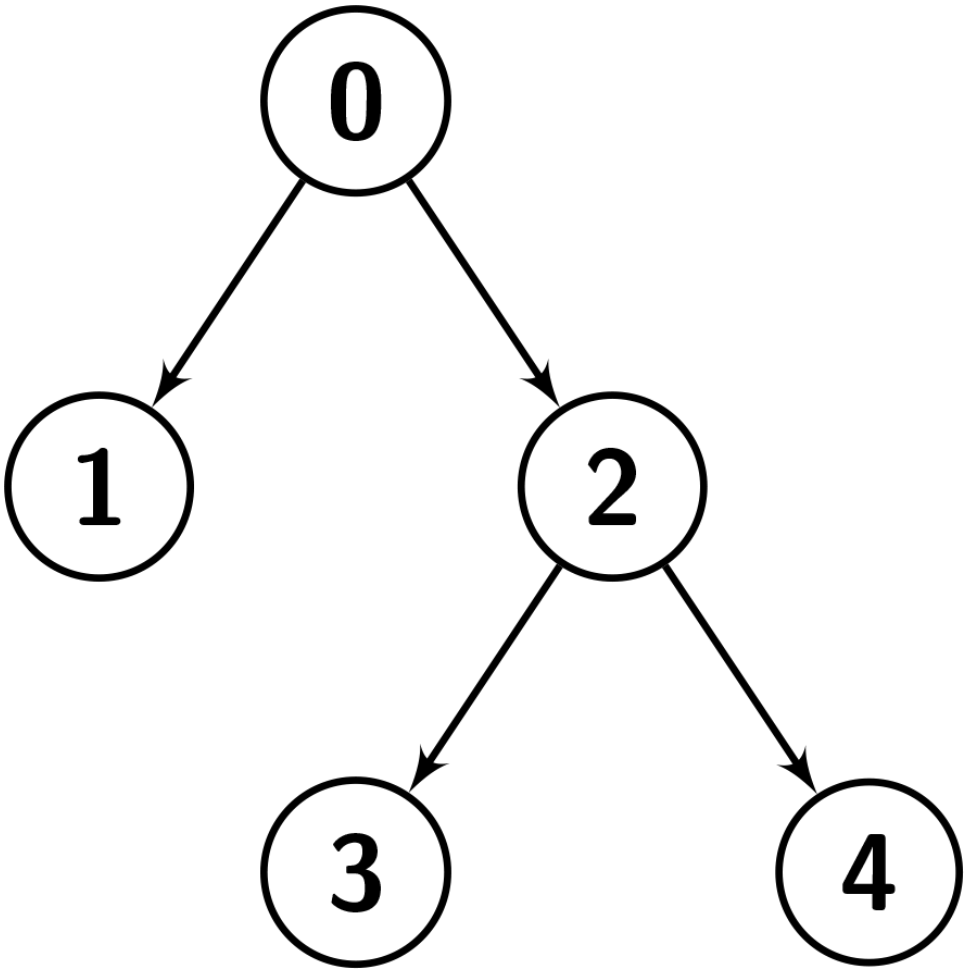
nodes = [[0,-1],[1,0],[2,0],[3,2],[4,2]]

Output:

true

Explanation:

The given nodes make the tree in the picture below. We can show that this is the preorder traversal of the tree, first we visit node 0, then we do the preorder traversal of the right child which is [1], then we do the preorder traversal of the left child which is [2,3,4].



Example 2:

Input:

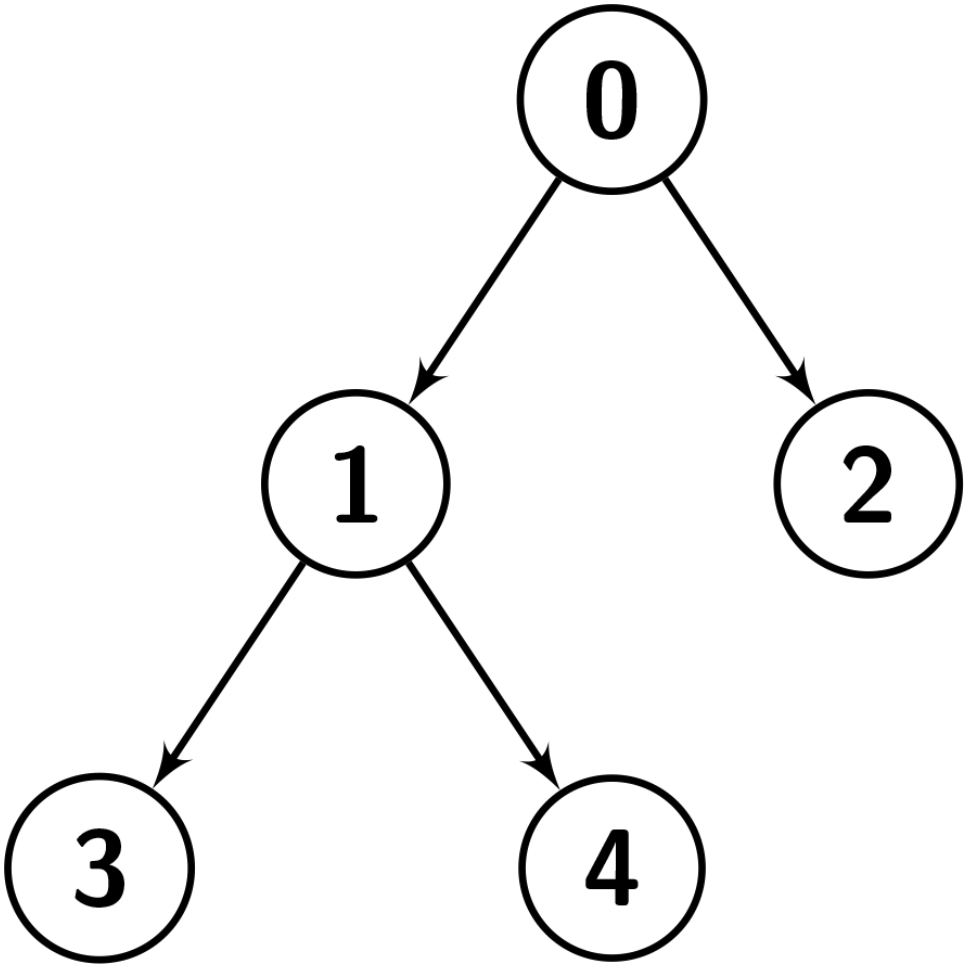
nodes = [[0,-1],[1,0],[2,0],[3,1],[4,1]]

Output:

false

Explanation:

The given nodes make the tree in the picture below. For the preorder traversal, first we visit node 0, then we do the preorder traversal of the right child which is [1,3,4], but we can see that in the given order, 2 comes between 1 and 3, so, it's not the preorder traversal of the tree.



Constraints:

1 <= nodes.length <= 10

5

nodes[i].length == 2

0 <= nodes[i][0] <= 10

5

-1 <= nodes[i][1] <= 10

5

The input is generated such that

nodes

make a binary tree.

Code Snippets

C++:

```
class Solution {
public:
bool isPreorder(vector<vector<int>>& nodes) {

}
};
```

Java:

```
class Solution {
public boolean isPreorder(List<List<Integer>> nodes) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[][]} nodes
```

```
* @return {boolean}
*/
var isPreorder = function(nodes) {

};
```

TypeScript:

```
function isPreorder(nodes: number[][]): boolean {

};
```

C#:

```
public class Solution {
public bool IsPreorder(IList<IList<int>> nodes) {

}
}
```

C:

```
bool isPreorder(int** nodes, int nodesSize, int* nodesColSize) {

}
```

Go:

```
func isPreorder(nodes [][]int) bool {

}
```

Kotlin:

```
class Solution {
fun isPreorder(nodes: List<List<Int>>): Boolean {

}
}
```

Swift:

```
class Solution {
func isPreorder(_ nodes: [[Int]]) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn is_preorder(nodes: Vec<Vec<i32>>) -> bool {

}
}
```

Ruby:

```
# @param {Integer[][]} nodes
# @return {Boolean}
def is_preorder(nodes)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $nodes
* @return Boolean
*/
function isPreorder($nodes) {

}
}
```

Dart:

```
class Solution {
bool isPreorder(List<List<int>> nodes) {

}
}
```

Scala:

```
object Solution {
def isPreorder(nodes: List[List[Int]]): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec is_preorder(nodes :: [[integer]]) :: boolean
def is_preorder(nodes) do

end
end
```

Erlang:

```
-spec is_preorder(Nodes :: [[integer()]]) -> boolean().
is_preorder(Nodes) ->
.
```

Racket:

```
(define/contract (is-preorder nodes)
(-> (listof (listof exact-integer?)) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Is Array a Preorder of Some ■Binary Tree
* Difficulty: Medium
* Tags: array, tree, search, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

```
class Solution {
public:
bool isPreorder(vector<vector<int>>& nodes) {

}
};
```

Java Solution:

```
/**
* Problem: Is Array a Preorder of Some ■Binary Tree
* Difficulty: Medium
* Tags: array, tree, search, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public boolean isPreorder(List<List<Integer>> nodes) {

}
}
```

Python3 Solution:

```
"""
Problem: Is Array a Preorder of Some ■Binary Tree
Difficulty: Medium
Tags: array, tree, search, stack

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

class Solution:
def isPreorder(self, nodes: List[List[int]]) -> bool:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Is Array a Preorder of Some ■Binary Tree
* Difficulty: Medium
* Tags: array, tree, search, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* @param {number[][]} nodes
* @return {boolean}
*/
var isPreorder = function(nodes) {

};
```

TypeScript Solution:

```
/**
* Problem: Is Array a Preorder of Some ■Binary Tree
* Difficulty: Medium
* Tags: array, tree, search, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
```

```
*/

function isPreorder(nodes: number[][]): boolean {

};
```

C# Solution:

```
/*
* Problem: Is Array a Preorder of Some ■Binary Tree
* Difficulty: Medium
* Tags: array, tree, search, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public bool IsPreorder(IList<IList<int>> nodes) {

}
}
```

C Solution:

```
/*
* Problem: Is Array a Preorder of Some ■Binary Tree
* Difficulty: Medium
* Tags: array, tree, search, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

bool isPreorder(int** nodes, int nodesSize, int* nodesColSize) {

}
```

Go Solution:

```
// Problem: Is Array a Preorder of Some ■Binary Tree
// Difficulty: Medium
// Tags: array, tree, search, stack
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func isPreorder(nodes [][]int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isPreorder(nodes: List<List<Int>>): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func isPreorder(_ nodes: [[Int]]) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Is Array a Preorder of Some ■Binary Tree
// Difficulty: Medium
// Tags: array, tree, search, stack
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn is_preorder(nodes: Vec<Vec<i32>>) -> bool {

}
```

```
}
```

Ruby Solution:

```
# @param {Integer[][]} nodes
# @return {Boolean}
def is_preorder(nodes)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $nodes
* @return Boolean
*/
function isPreorder($nodes) {

}
}
```

Dart Solution:

```
class Solution {
bool isPreorder(List<List<int>> nodes) {

}
}
```

Scala Solution:

```
object Solution {
def isPreorder(nodes: List[List[Int]]): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec is_preorder(nodes :: [[integer]]) :: boolean
def is_preorder(nodes) do

end
end
```

Erlang Solution:

```
-spec is_preorder(Nodes :: [[integer()]]) -> boolean().
is_preorder(Nodes) ->
.
```

Racket Solution:

```
(define/contract (is-preorder nodes)
(-> (listof (listof exact-integer?)) boolean?)
)
```