# Challenge 7: Check if a Given Undirected Graph is Tree or not?

In this lesson, we will learn the difference between a graph and a tree. You will use this knowledge for the challenge below.

We'll cover the following

## Problem statement#

The next section will tackle the tree data structure. For now, here's the basic difference between a graph and a tree. A graph can only be a tree under two conditions:

- There are **no cycles**.
- The graph is **connected**.

> A graph is connected when there is a path between every pair of vertices. In a connected graph, there are no unreachable vertices. Each

> vertex must be connected to every other vertex through ei
> edge or a graph traversal.

You have to implement `is_tree()` function which will take a graph as an input and find out if it is a tree.

## Input#

An undirected graph.

## Output#

Returns `True` if the given graph is a tree. Otherwise, it returns `False` .

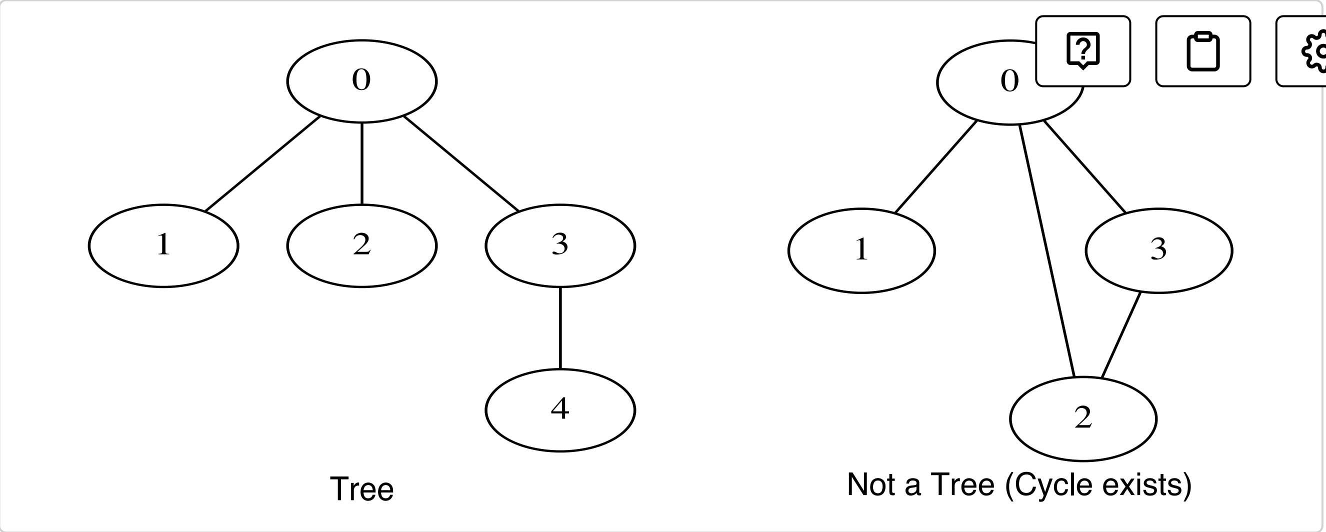
## Sample input#

```
graph = {
    0 - 1
    0 - 2
    0 - 3
    3 - 4
}
```

## Sample output#

```
True
```

Take a look at the illustration below to understand better.



Tree

Not a Tree (Cycle exists)

## Coding exercise#

Take a close look and design a step-by-step algorithm first before jumping on to the implementation.

The point of this exercise is to understand the difference between a tree and a graph. Other than that, the task is fairly easy.

If you get stuck, you can always refer to the solution provided in the solution section.

Good luck!

> One of the test cases passes by default. You need to pass all test cases for the solution to be considered correct.

main.py

Graph.py

Stack.py

Queue.py

LinkedList.py

Node.py

```
from Graph import Graph
from Queue import MyQueue
from Stack import MyStack
# You can check the input graph in console tab

# Create Stack => stack = MyStack()
# Functions of Stack => push(int), pop(), top(), is_empty()
# Create Queue => queue = MyQueue()
# Functions of Queue => enqueue(int), dequeue(), size(), front(), is_empty()
# class Graph => {int vertices, linkedList[] array}
# class linkedList => {Node head_node}
# class Node => {int data, Node next_element}


def is_tree(g):
    # Write your code here
    pass
```

← Back

Solution Review: Check if a Path Exist...

Next →

Solution Review: Check if Given Undir...

☑ Mark as Completed

Report an Issue