Given a binary tree `t`, determine whether it is *symmetric* around its center, i.e. each side mirrors the other.

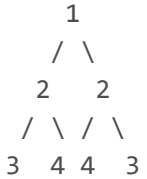
Example

- For

```
t = {
    "value": 1,
    "left": {
        "value": 2,
        "left": {
            "value": 3,
            "left": null,
            "right": null
        },
        "right": {
            "value": 4,
            "left": null,
            "right": null
        }
    },
    "right": {
        "value": 2,
        "left": {
            "value": 4,
            "left": null,
            "right": null
        },
        "right": {
            "value": 3,
            "left": null,
            "right": null
        }
    }
}
```

the output should be `isTreeSymmetric(t) = true`.

Here's what the tree in this example looks like:



As you can see, it is symmetric.

- For

```
t = {
    "value": 1,
    "left": {
        "value": 2,
        "left": null,
        "right": {
            "value": 3,
            "left": null,
            "right": null
        }
    },
    "right": {
        "value": 2,
        "left": null,
        "right": {
            "value": 3,
            "left": null,
            "right": null
        }
    }
}
```

  the output should be `isTreeSymmetric(t) = false`.

  Here's what the tree in this example looks like:

```
    1
   / \
  2   2
   \   \
   3    3
```

  As you can see, it is not symmetric.

Input/Output

- **[time limit] 4000ms (py3)**
- **[input] tree.integer t**

  A binary tree of integers.

  Guaranteed constraints:
  $0 \le$ tree size $< 5 \cdot 10^4$,
  $-1000 \le$ node value ≤ 1000.

- **[output] boolean**

Return `true` if `t` is symmetric and `false`otherwise.