Problem:

First, some preliminary jargon: recall that the depth of a node of a binary tree is the length of the path from the root to the node.

Furthermore, a node of a binary tree is called a leaf if it has no children. Otherwise, the node is said to be an internal node.

A tree is said to be full if every internal node has two children (in other words, no node has exactly one child).

Now, you are given an array that contains the depth of each leaf in the tree in arbitrary order. From this array, determine whether or not the tree is full. Return `true` if the tree is full, and `false` otherwise.

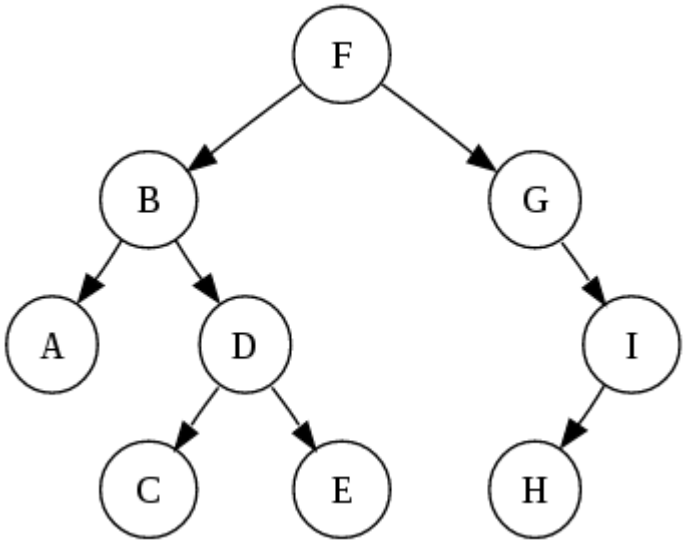
Example

- For `leaf_array = [0]`, the output should be `treeIsFull(leaf_array) = true`.

  The given array represents a tree with a single node.

- For `leaf_arry = [3, 2, 3, 3]`, the output should be `treeIsFull(leaf_array) = false`.

  The given array may represent this tree (or similar to it with the same nodes configuration):



Note, that the answer would be the same for the same array with the elements ordered differently.

Input/Output

- **[time limit] 3000ms (java)**

- **[input] array.integer leaf_array**

  Array containing the depth of each leaf in a random order.

  Constraints:
  $1 \leq leaf\_array.length \leq 30$,
  $0 \leq leaf\_array[i] \leq 10$.

- **[output] boolean**

  `true` if the tree is full, `false` otherwise.