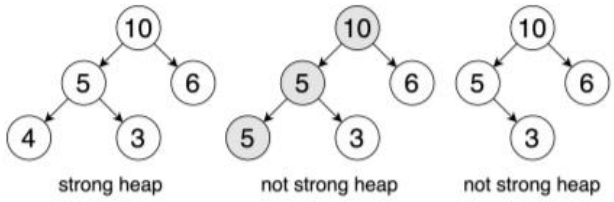
We will call a binary tree a strong heap1 if it is a complete binary tree and it satisfies the “strong (max) heap property” as:

• for all nodes x, we have x < parent(x), and

• for nodes x in level 2 or below, we also have x + parent(x) < parent(parent(x)).

In StrongHeap.java, implement isStrongHeap. This takes one argument, the root of a binary tree, and returns whether the given binary tree is a strong max heap. State the worst case time and space complexity (in Big-O notation) in the method’s Javadoc.

These are some examples of trees which are and are not strong heaps. The second is not a strong heap because 5 + 5 ≮ 10. The third is not a strong heap because the tree is not complete, despite satisfying the strong heap property.



Hints and notes:

• As a reminder, the root node of a tree is at level 0.

• A tree with a single node is trivially a strong binary heap.

• Strong binary heaps are a subset of binary heaps.

• If a binary tree is not complete, it cannot be a strong heap. If a tree is complete, you must check the strong heap property to determine whether it is a strong heap.

• Node values can be negative but not null. Negatives should not be treated specially