Adding an Element: Heapify Up

2 min

Sometimes you will add an element to the heap that violates the heap’s essential properties.

We’re adding 3 as a left child of 11, which violates the min-heap property that children must be larger or equal to their parent.

We need to restore the fundamental heap properties. This restoration is known as *heapify* or *heapifying*. We’re adding an element to the bottom of the tree and moving upwards, so we’re *heapifying up*.

As long as we’ve violated the heap properties, we’ll swap the offending child with its parent until we restore the properties, or until there’s no parent left. If there is no parent left, that element becomes the new root of the tree.

3 swaps with 11, but there’s still work to do because now 3 is a child of 5. One more swap and we’ve restored the heap properties. The parent value, 2, is lesser than the child, 3. We can see that 3‘s children 5 and 14 are also greater.

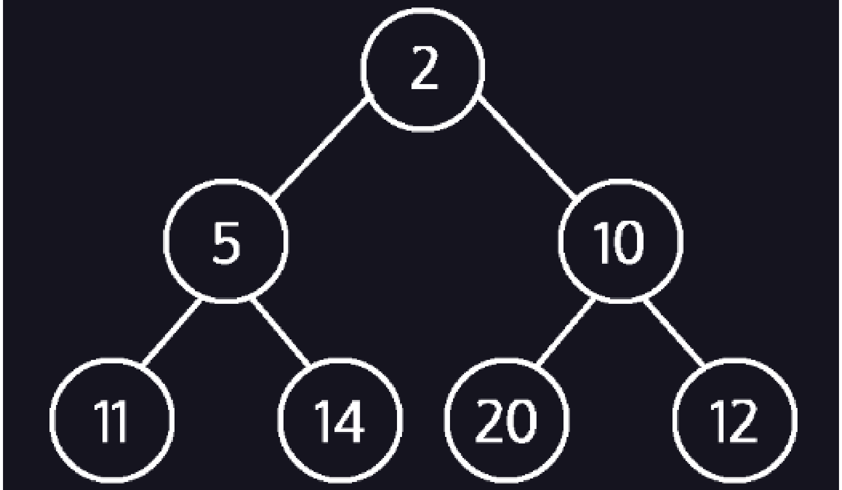
Instructions

Our heap had 8 elements when we began heapifying up. How many swaps did we make?

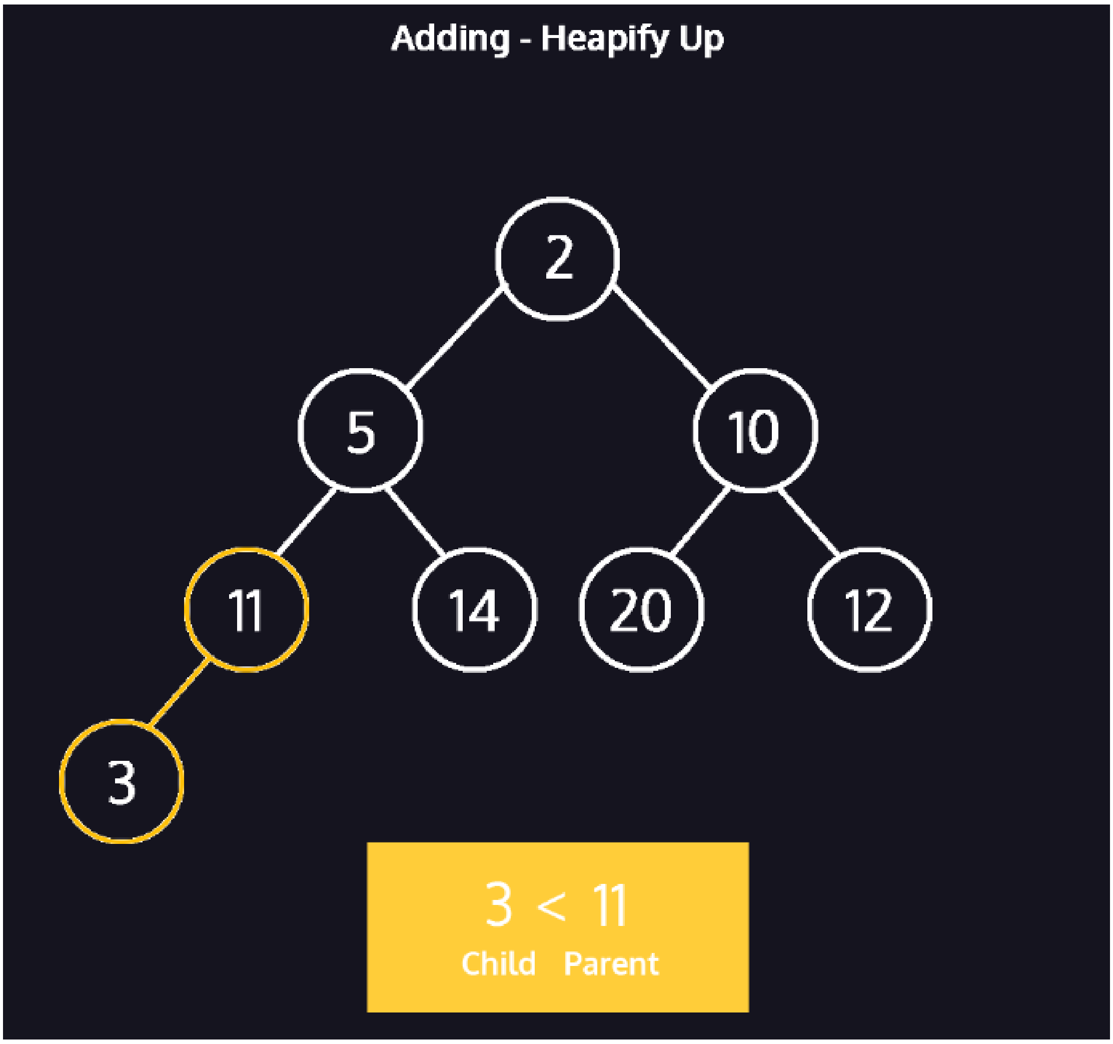
If the element had been 1 instead of 3, how many swaps would we make?

Given this 8 element heap, what is the greatest number of swaps we would make to restore the heap property?

Adding - Heapify Up



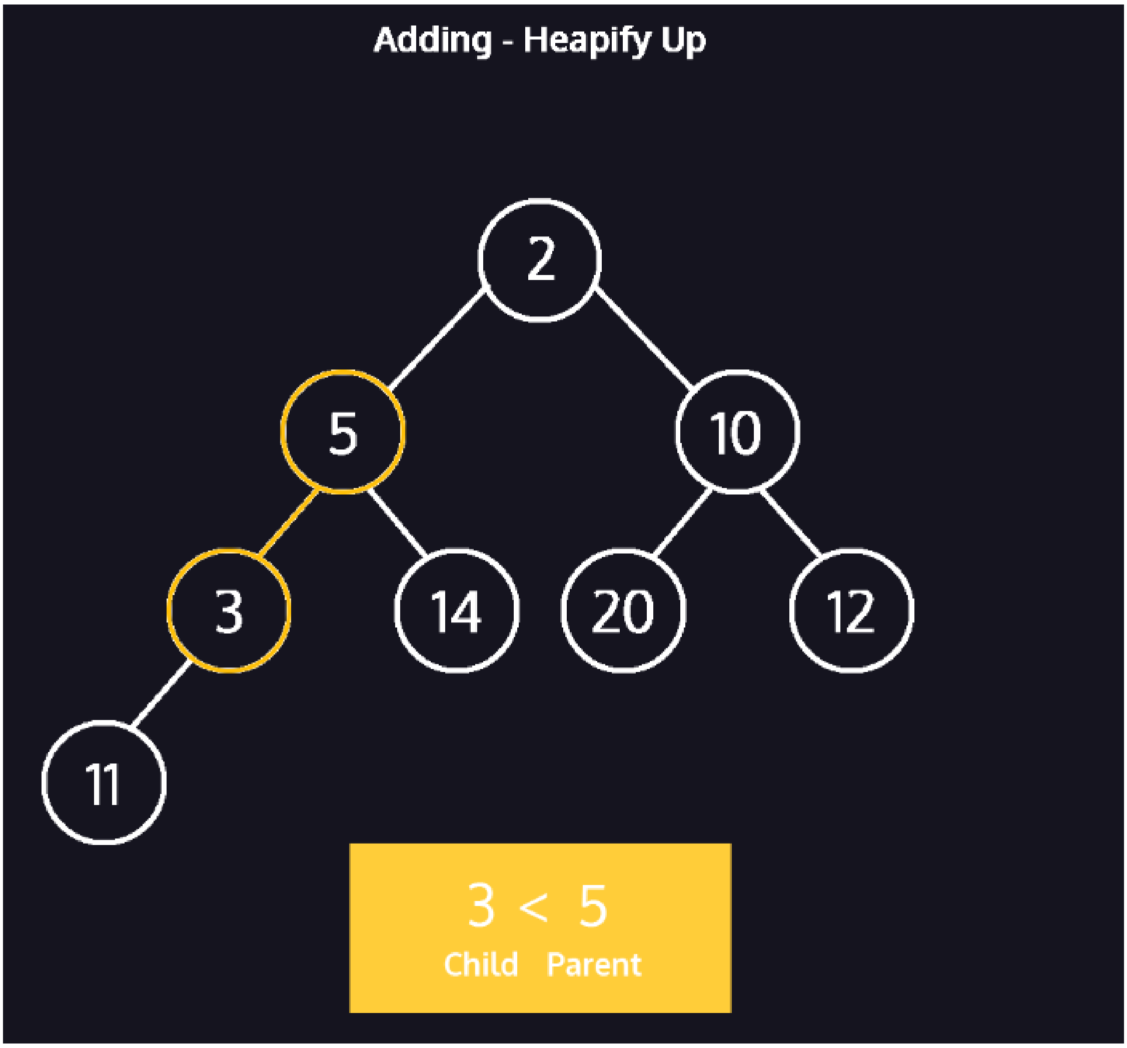
Adding 3



Adding - Heapify Up

3 < 11

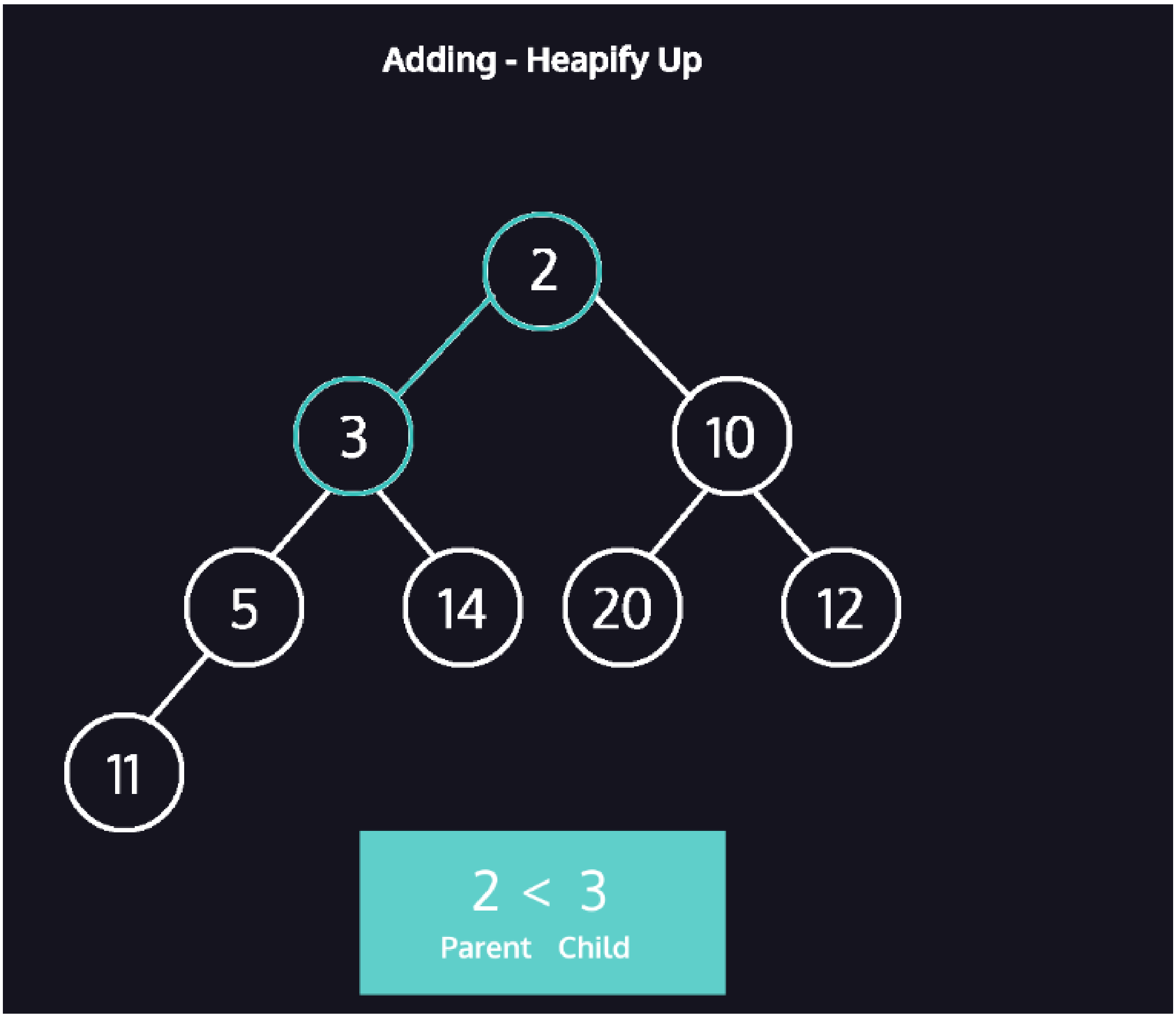
Child Parent



Adding - Heapify Up

3 < 5

Child Parent



Adding - Heapify Up

2 < 3

Parent Child