THE BINARY HEAP

INSERT

INSERT AN ELEMENT AS A LEAF NODE IN THE HEAP

IN AN ARRAY IMPLEMENTATION THAT WOULD BE AT THE VERY END - THE NEWLY INSERTED ELEMENT WOULD BE THE LAST ELEMENT IN THE ARRAY

THE ELEMENT MIGHT BE IN THE WRONG POSITION WITH RESPECT TO ALL NODES ABOVE IT

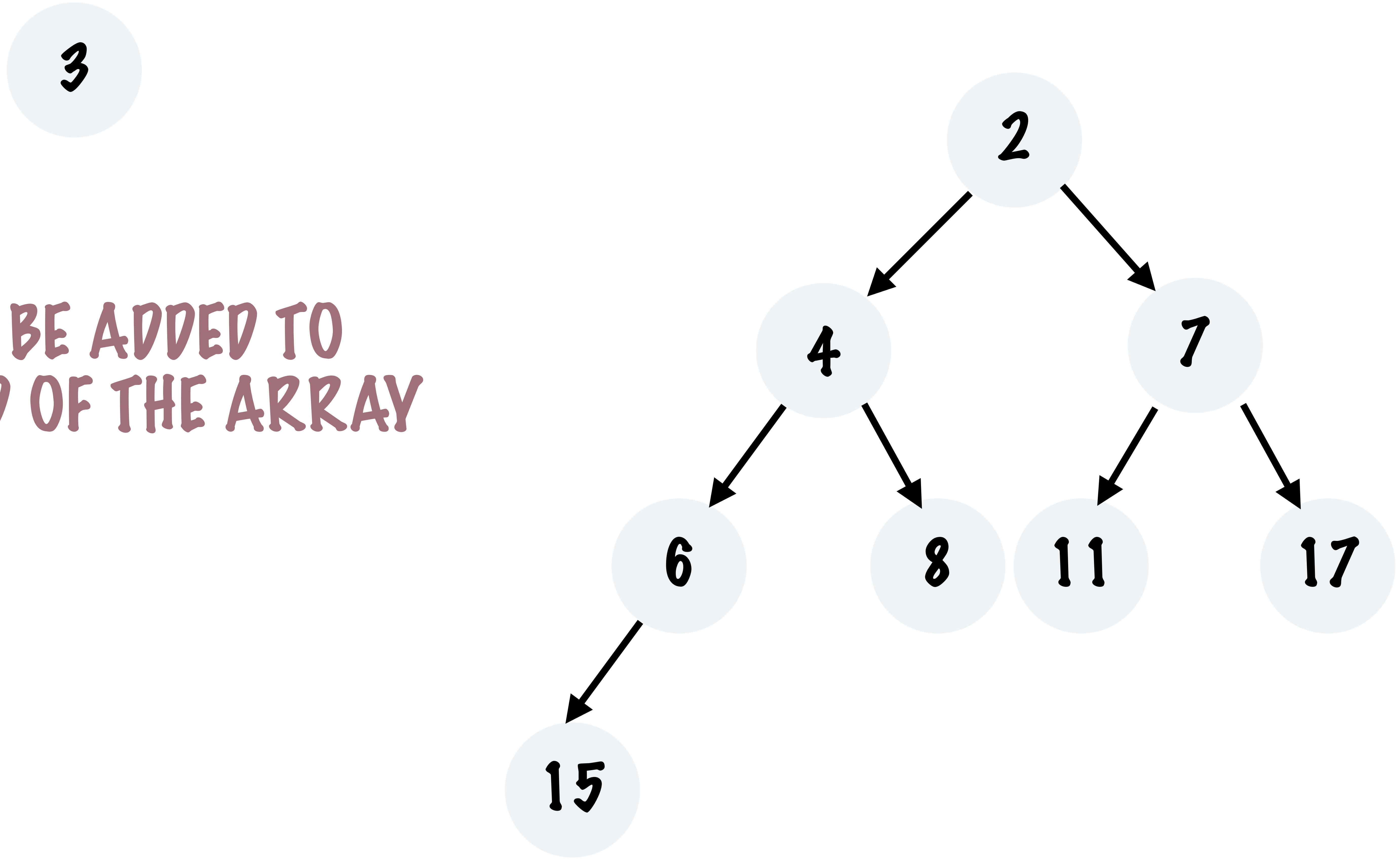
IT HAS TO BE MOVED UPWARDS IN THE HEAP TOWARDS THE ROOT NODE TO FIND IT'S RIGHT POSITION

SIFT UP

THE BINARY HEAP

INSERT

INSERT THE ELEMENT 3
INTO THIS HEAP



3 WILL BE ADDED TO
THE END OF THE ARRAY

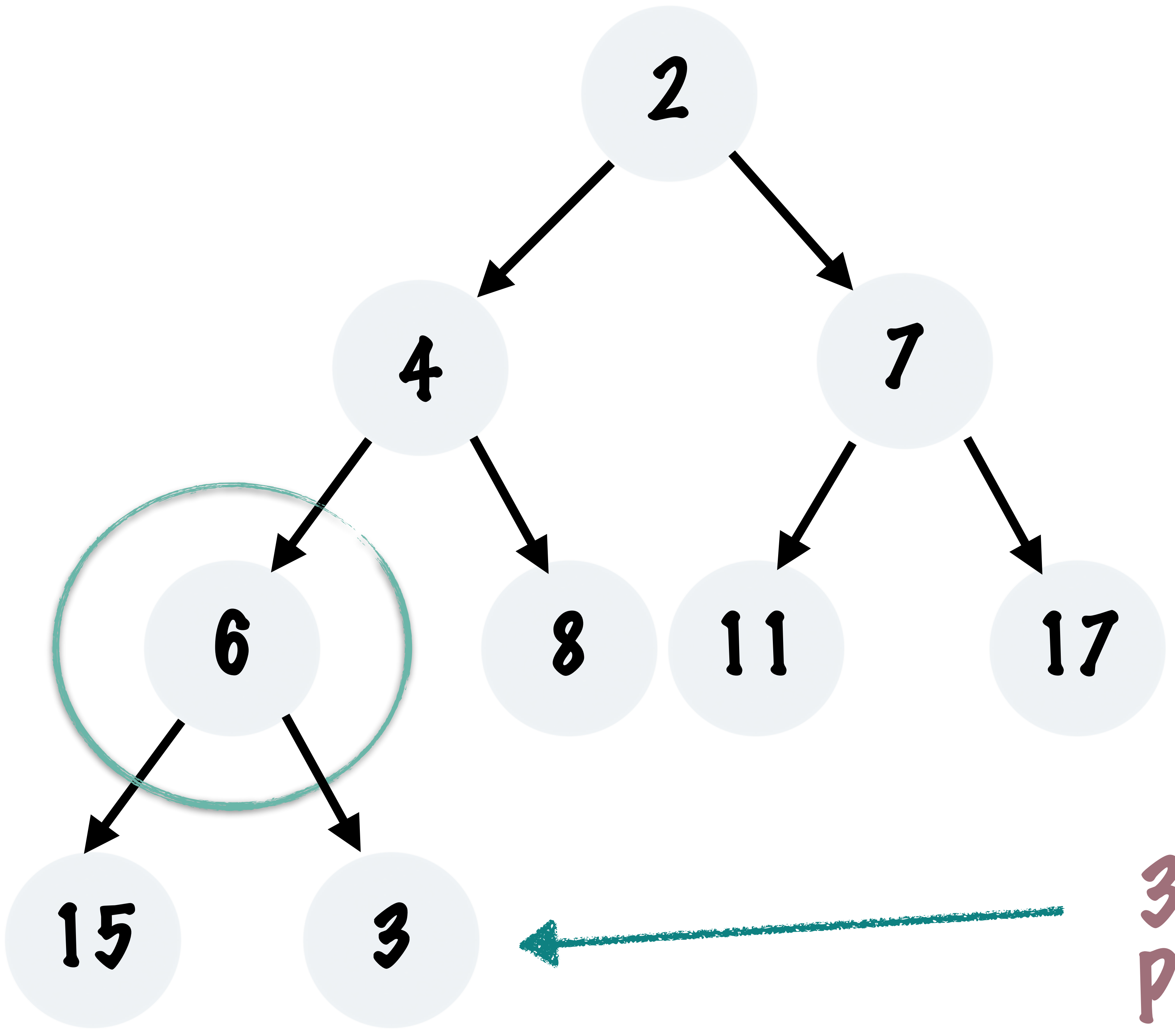
THE BINARY HEAP

INSERT

NOW SIFT UP THE ELEMENT 3 TO ITS CORRECT POSITION

COMPARE 3 WITH ITS PARENT

3 < 6 SO SWAP THE TWO



3 IS IN THE WRONG POSITION

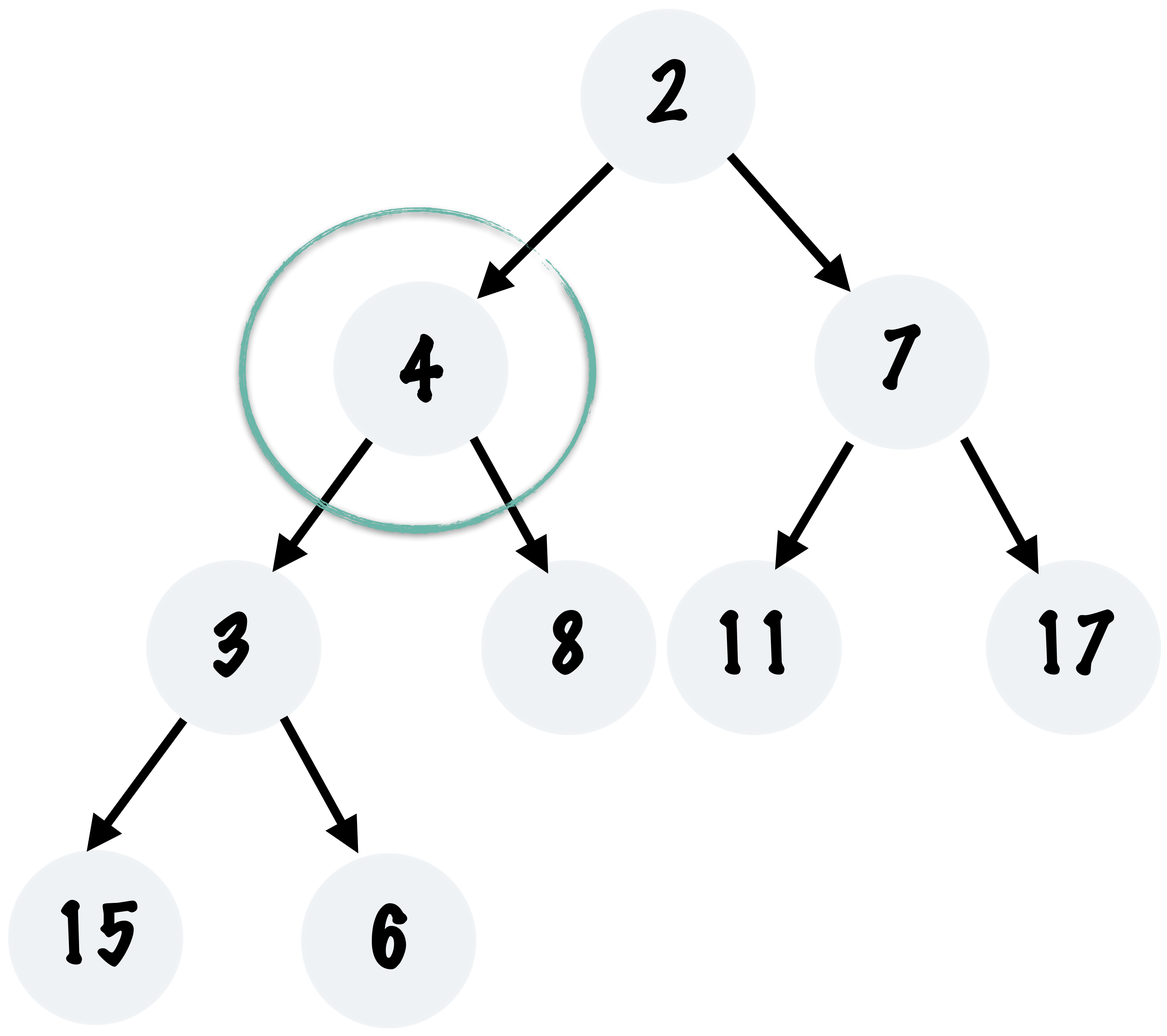
THE BINARY HEAP

INSERT

COMPARE 3 WITH
ITS PARENT AGAIN

3 < 4 SO SWAP AGAIN

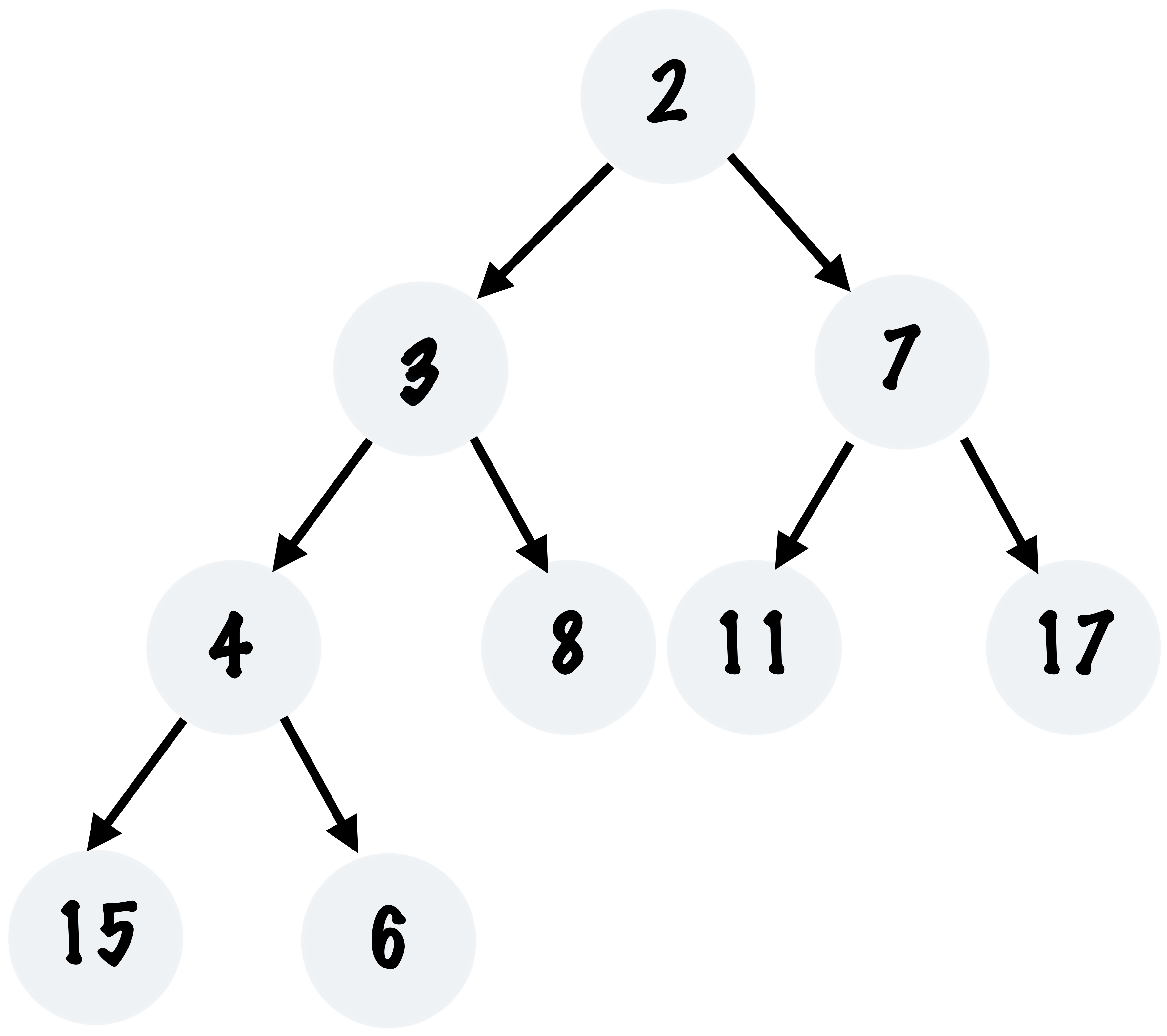
SIFT UP THE ELEMENT 3
ONCE AGAIN



THE BINARY HEAP

INSERT

THIS IS THE CORRECT POSITION FOR ELEMENT 3



THE BINARY HEAP

INSERT

NOW LET'S SEE SOME CODE...

INSERT

```
public void insert(T value) throws HeapFullException {
    if (count >= array.length) {
        throw new HeapFullException();
    }

    array[count] = value;
    siftUp(count);

    count++;
}
```

ENSURE THE HEAP IS NOT FULL BEFORE INSERTING THE ELEMENT

ADD THE NEW ELEMENT TO THE END OF THE ARRAY - THAT IS THE LAST LEAF NODE IN THE HEAP

SIFT THE ELEMENT UP TO THE RIGHT POSITION

INCREMENT THE COUNT OF THE NUMBER OF ELEMENTS

THE BINARY HEAP

REMOVE

REMOVE THE HIGHEST PRIORITY ELEMENT IN THE HEAP I.E THE MINIMUM ELEMENT

IN AN ARRAY IMPLEMENTATION THAT WOULD BE THE ELEMENT AT INDEX 0

COPY OVER THE LAST ELEMENT IN THE ARRAY TO INDEX 0

THE ELEMENT MIGHT BE IN THE WRONG POSITION WITH RESPECT TO ALL NODES BELOW IT

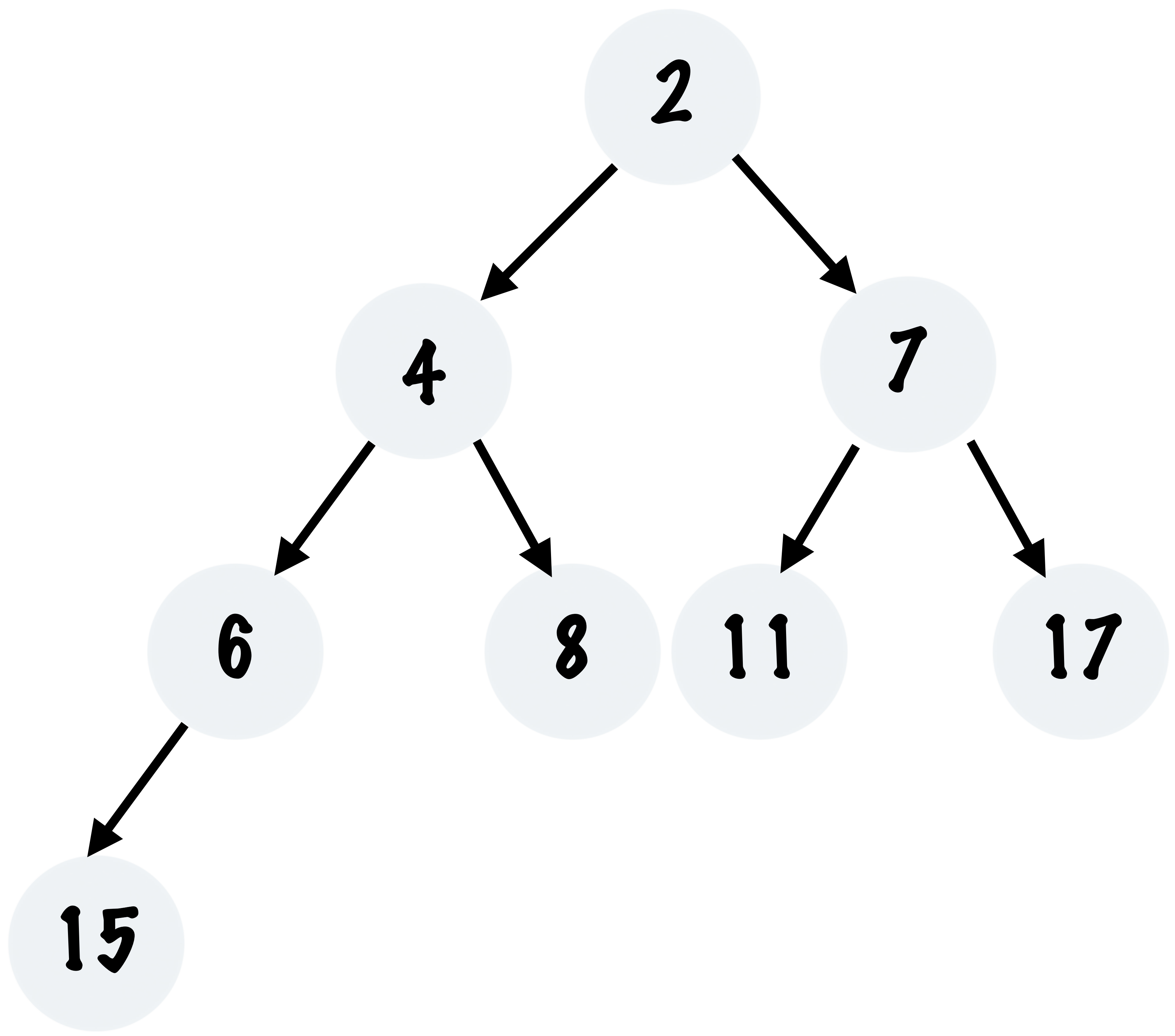
IT HAS TO BE MOVED DOWNWARDS IN THE HEAP TOWARDS THE LEAF NODES TO FIND IT'S RIGHT POSITION

SIFT DOWN

THE BINARY HEAP

REMOVE

REMOVE THE ELEMENT 2
FROM THIS HEAP

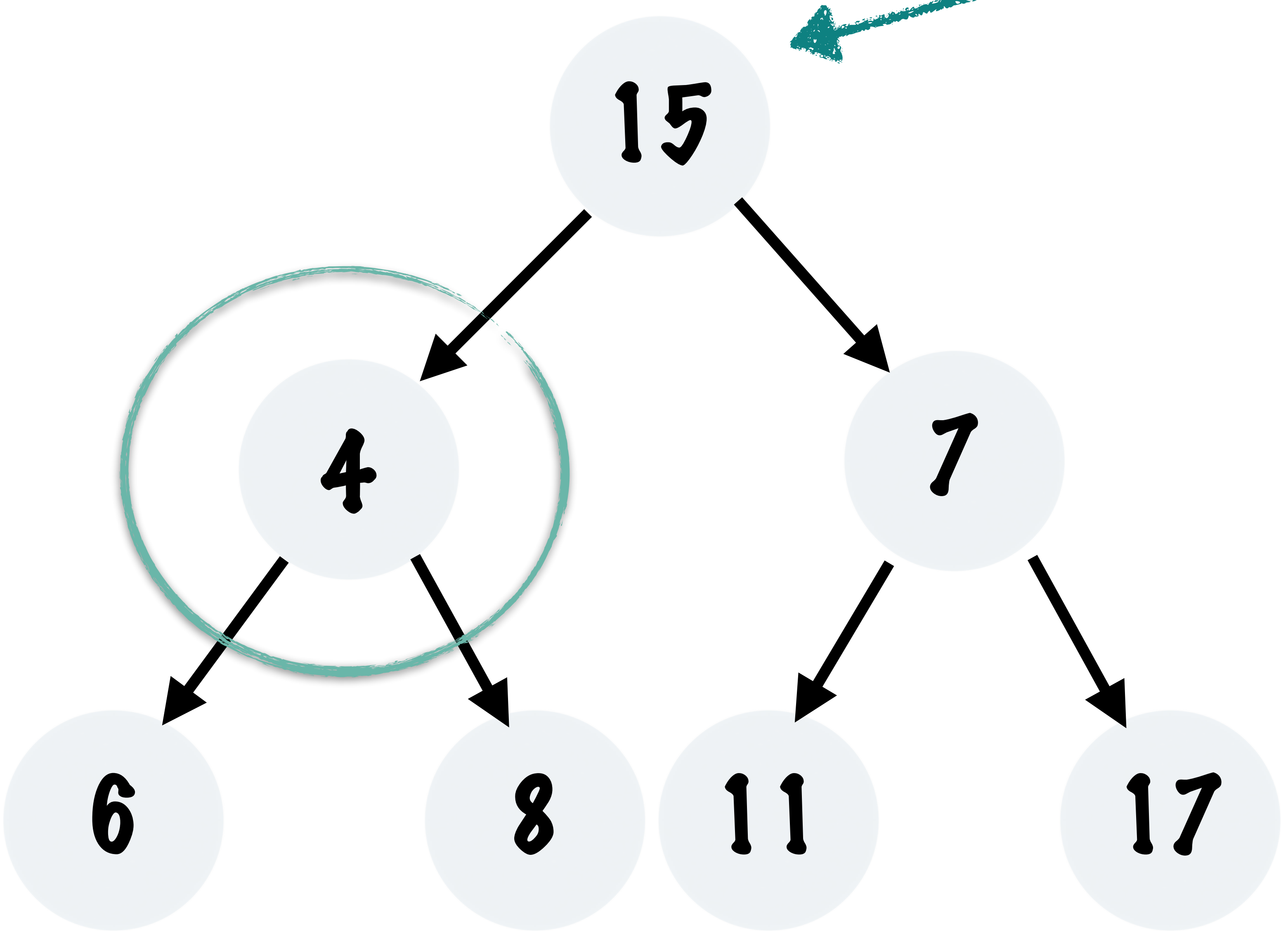


COPY OVER THE LAST
ELEMENT FROM THE
HEAP TO THE EMPTY
FIRST POSITION

THE BINARY HEAP

REMOVE

NOW SIFT DOWN THE ELEMENT 15 TO ITS CORRECT POSITION



THIS ELEMENT IS IN THE WRONG POSITION WITH RESPECT TO NODES BELOW IT IN THE HEAP

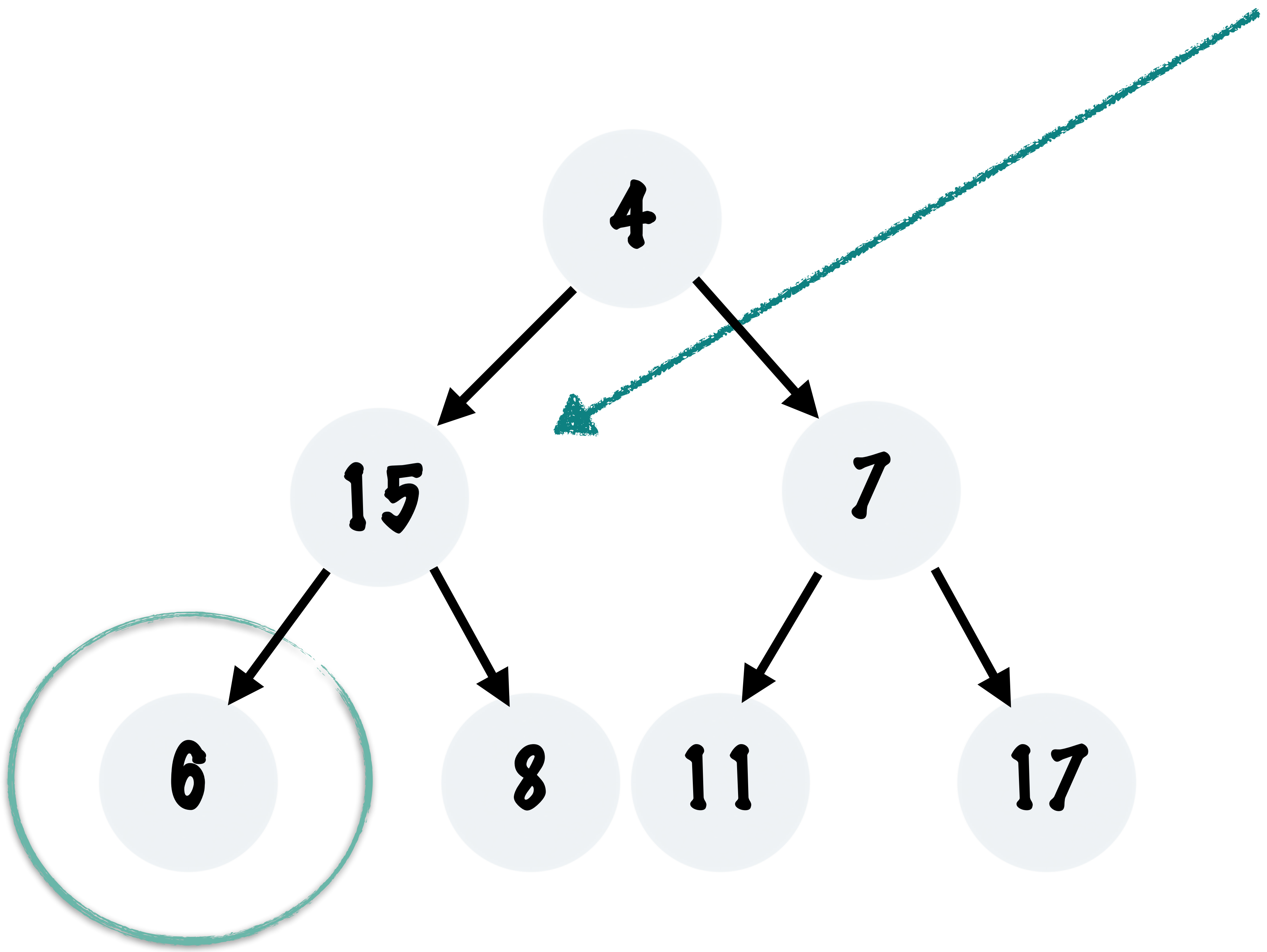
SWAP 15 WITH THE MINIMUM OF ITS LEFT AND RIGHT CHILD

THE BINARY HEAP

REMOVE

SIFT DOWN THE ELEMENT 15 TO ITS CORRECT POSITION

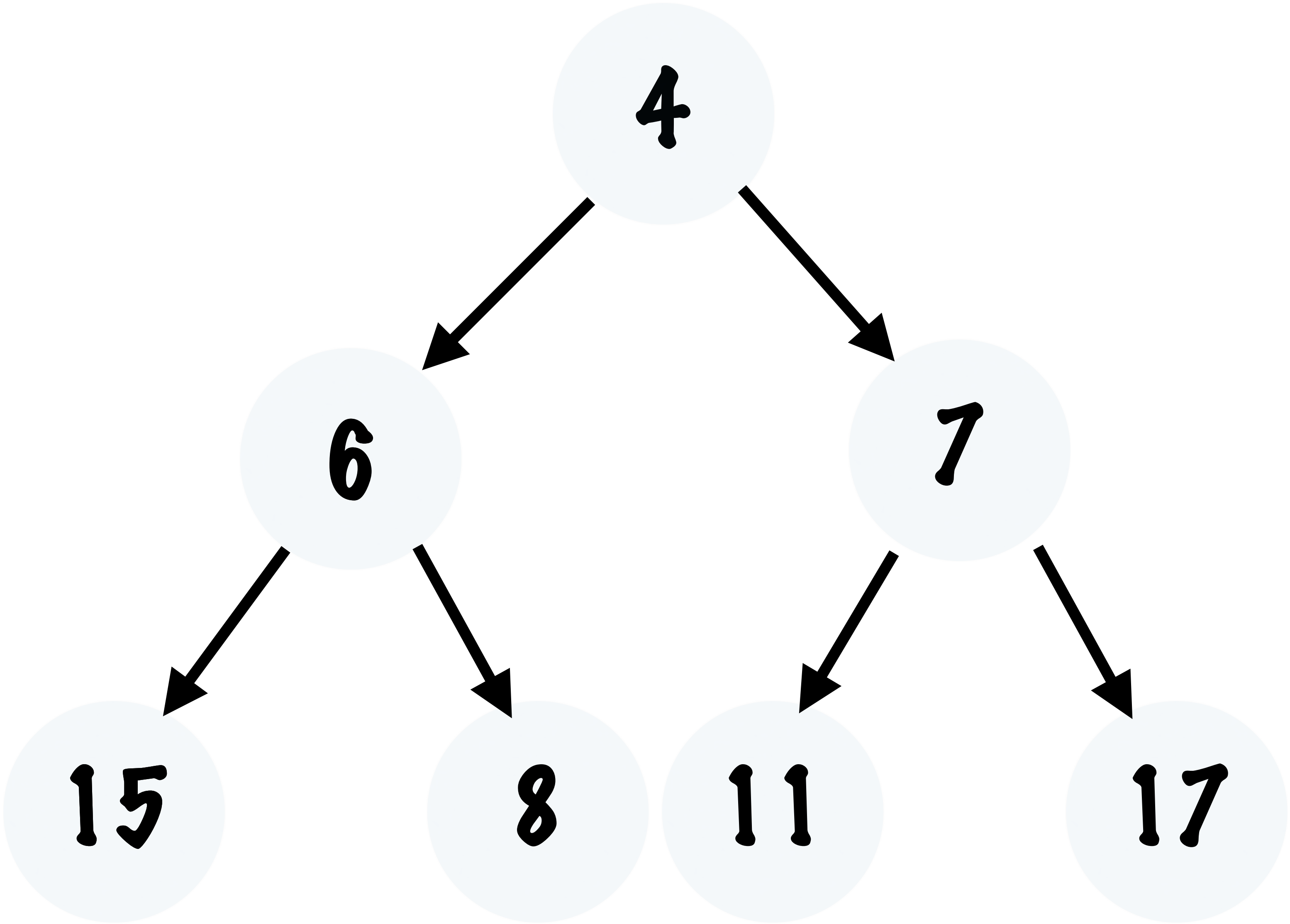
THIS ELEMENT IS STILL IN THE WRONG POSITION WITH RESPECT TO NODES BELOW IT IN THE HEAP



SWAP 15 WITH THE MINIMUM OF ITS LEFT AND RIGHT CHILD

6 < 8 SO SWAP 15 AND 6

THE BINARY HEAP
REMOVE



15 IS NOW IN THE CORRECT POSITION

THE BINARY HEAP
REMOVE

NOW LET'S SEE SOME CODE...

REMOVE HIGHEST PRIORITY

```java
public T removeHighestPriority() throws HeapEmptyException {
    T min = getHighestPriority();

    array[0] = array[count - 1];
    count--;
    siftDown(0);

    return min;
}
```

STORE THE MINIMUM DATA TO RETURN THE VALUE

COPY OVER THE LAST ELEMENT TO THE VERY FIRST INDEX IN THE ARRAY

DECREMENT THE NUMBER OF ELEMENTS IN THE HEAP

PERCOLATE THE ELEMENT DOWN TO THE RIGHT POSITION

GET HIGHEST PRIORITY

CHECK FOR AN EMPTY HEAP

```
public T getHighestPriority() throws HeapEmptyException {
    if (count == 0) {
        throw new HeapEmptyException();
    }

    return array[0];
}
```

RETURN THE FIRST ELEMENT IN THE ARRAY

THE BINARY HEAP

COMPLEXITY

INSERTION

INSERTING A NEW ELEMENT - COMPLEXITY O(LG N)

ACCESS

ACCESSING THE HIGHEST PRIORITY ELEMENT IS FAST - O(1)

REMOVE

REMOVING THE HIGHEST PRIORITY ELEMENT IS O(LG N)