Sorting Algorithms

Heap Sort

Time Complexity

Best, Average and Worst Case:

O(N * LogN)

Space Complexity

No additional space required
(In-place algorithm)

O(1)

Original Array

We are going to use the following relation to represent our array as a complete binary tree (our Heap):

parent = index

left child = 2*index+1

right child = 2*index+2

parent = index

left child = 2*index+1

right child = 2*index+2

Parent number 0: index 0

left child = 2*0+1 = index 1 (number 3)

right child = 2*0+2 = index 2 (number 8)

parent = index

left child = 2*index+1

right child = 2*index+2

Parent number 3: index 1

left child = 2*1+1 = index 3 (number 4)

right child = 2*1+2 = index 4 (number 1)

parent = index

left child = 2*index+1

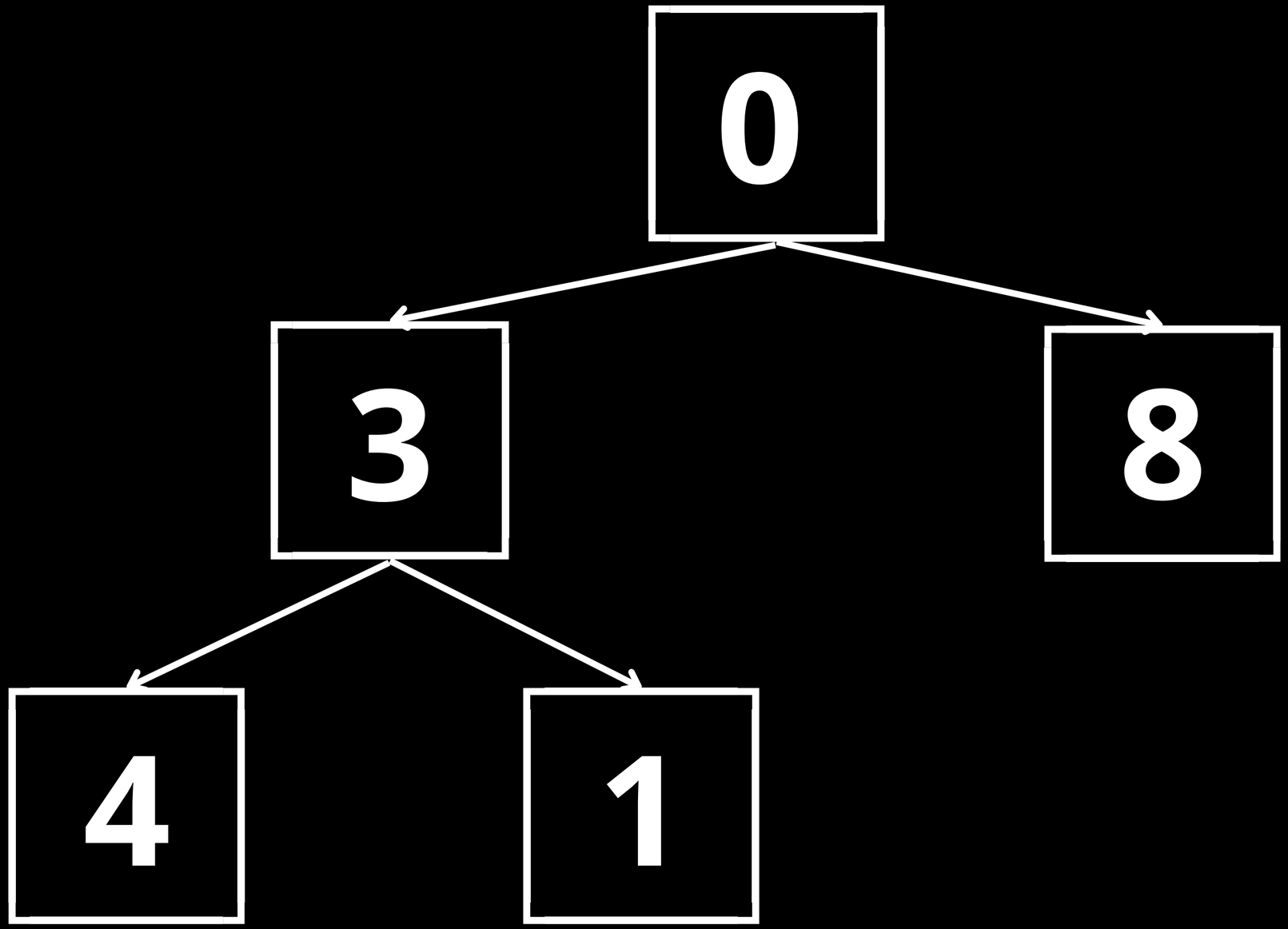
right child = 2*index+2

Parent number 8: index 2

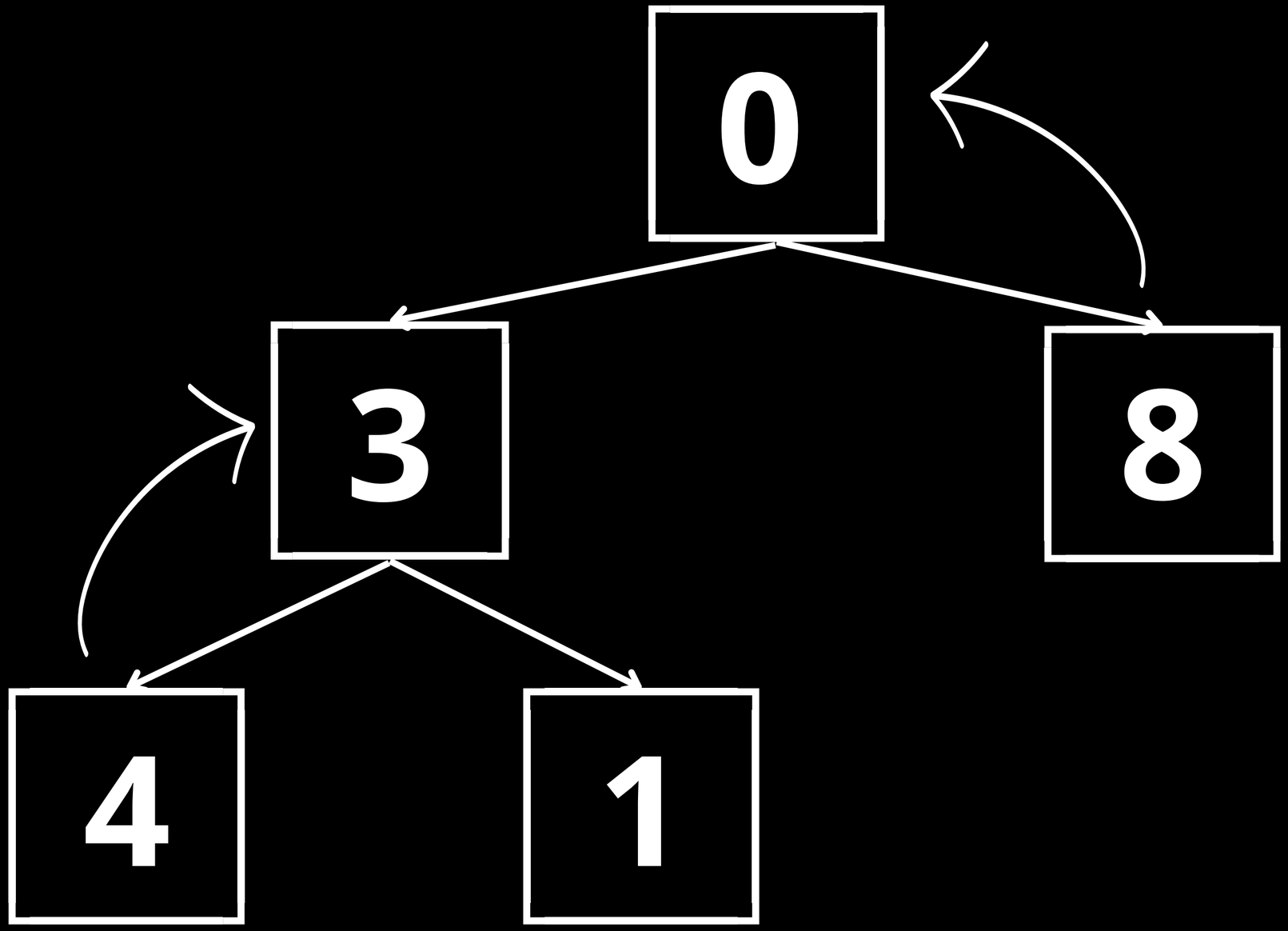
left child = 2*2+1 = index 5 (out of bounds)

right child = 2*2+2 = index 6 (out of bounds)

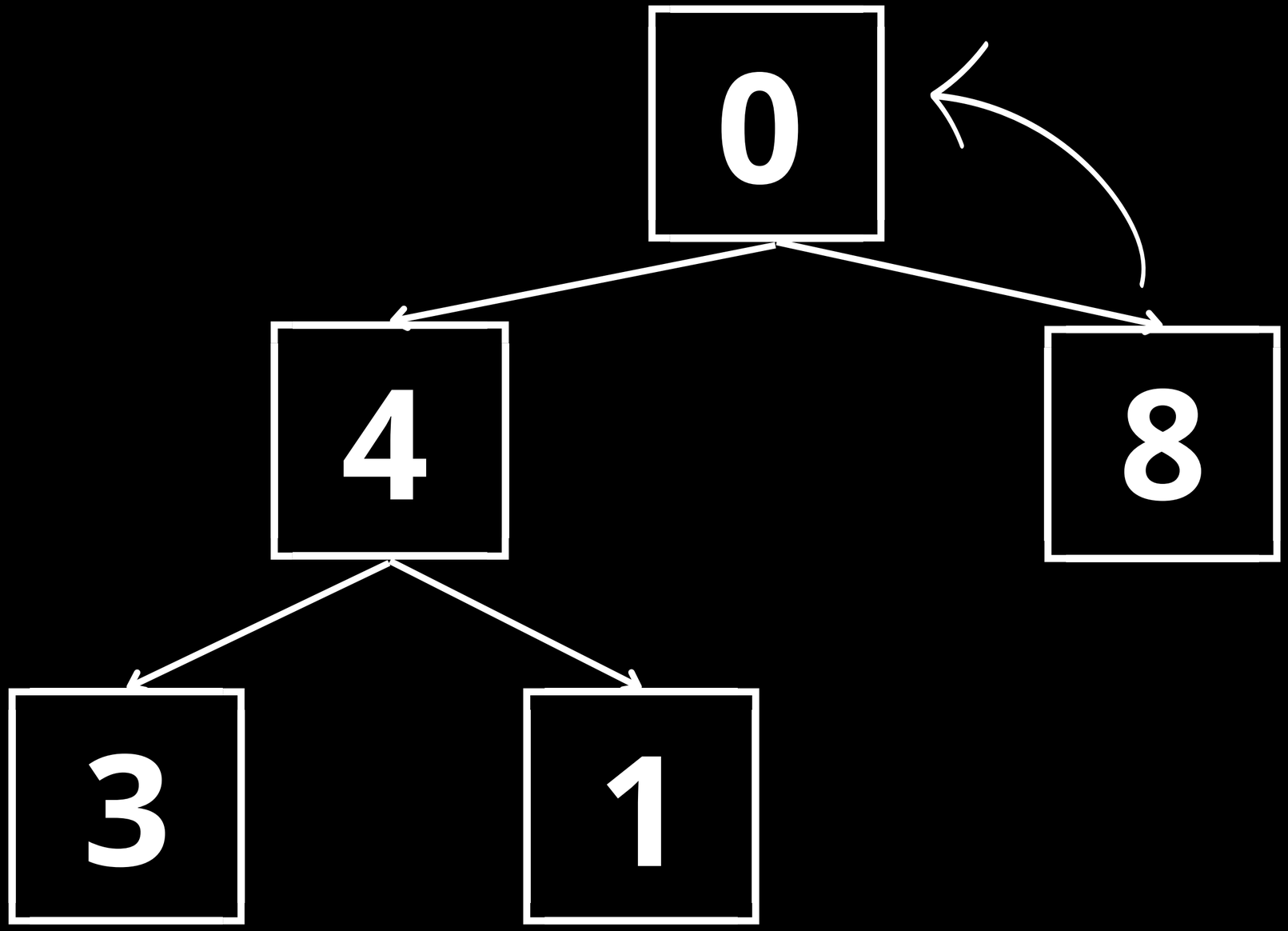
Create a Complete Binary Tree with the Array



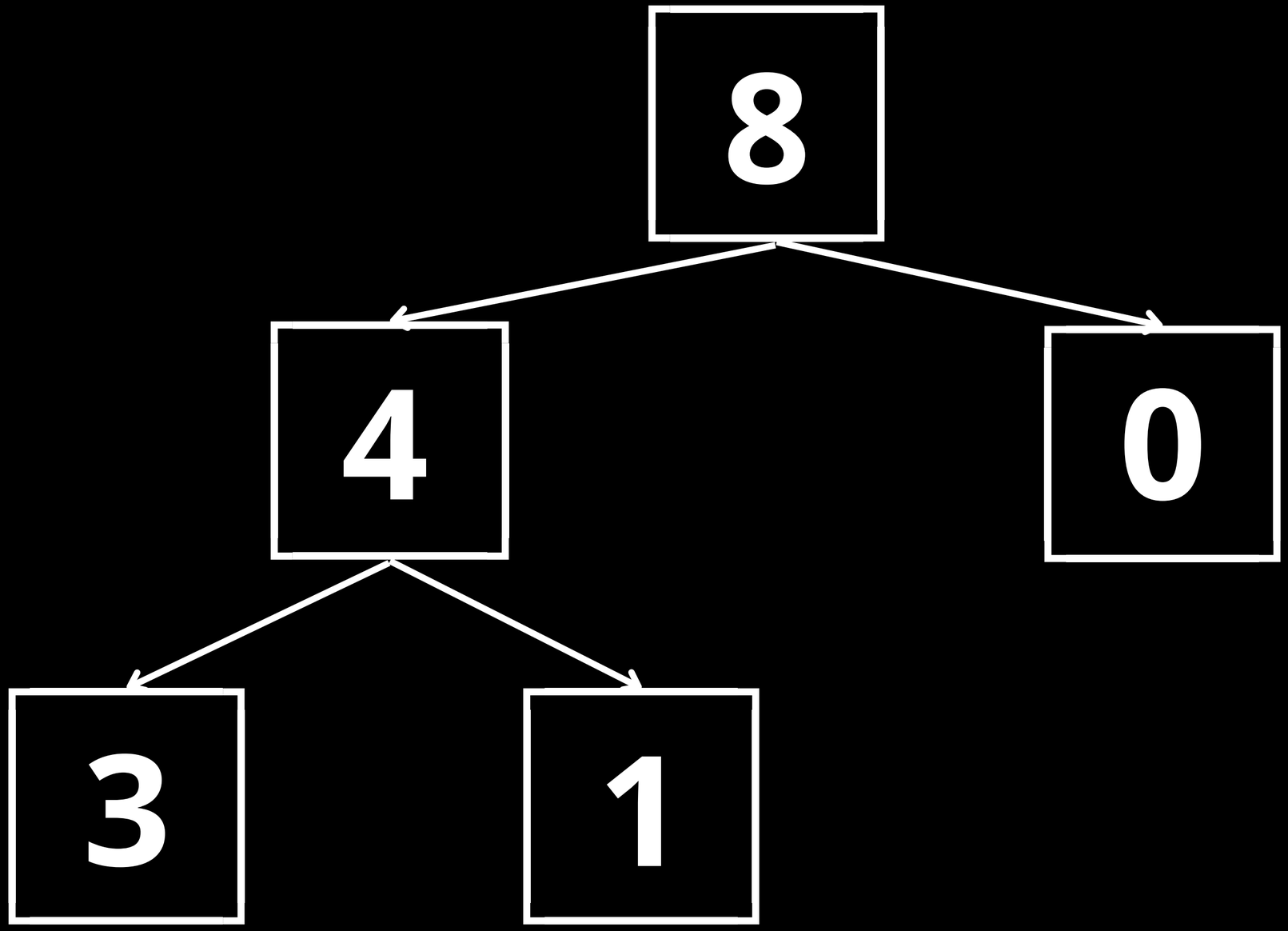
Now transform it into a Max Heap



Now transform it into a Max Heap



This is our Max Heap

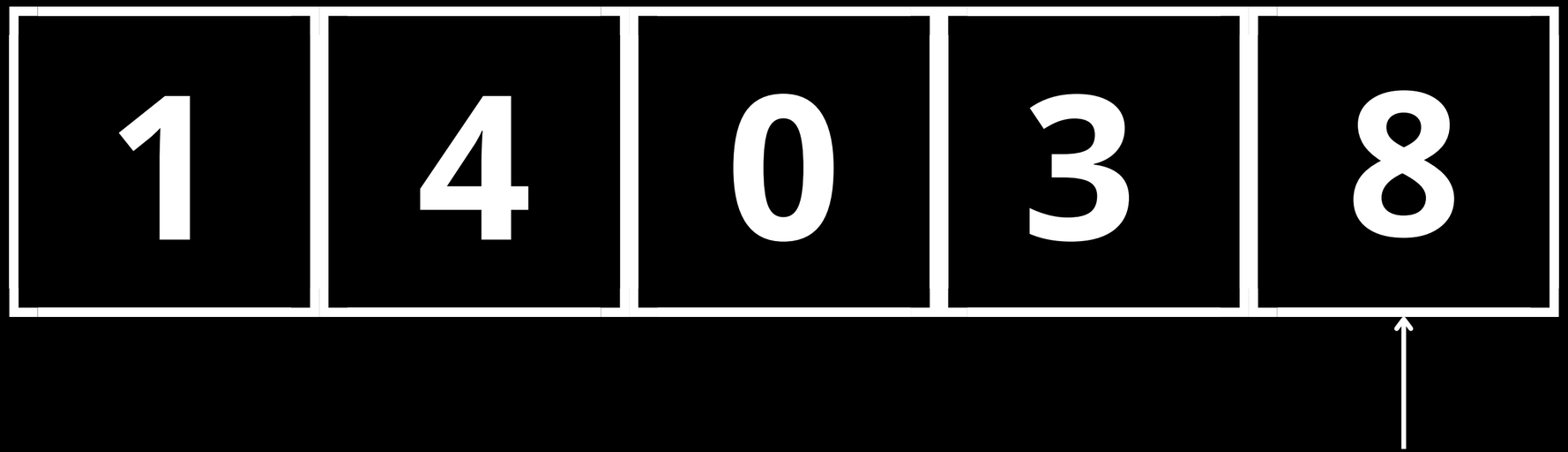


Now that we have a Max Heap, we can remove the root. To do so, we can simply swap it with the last current element in the array, and re-generate a Max Heap from the remaining elements up to size-1;

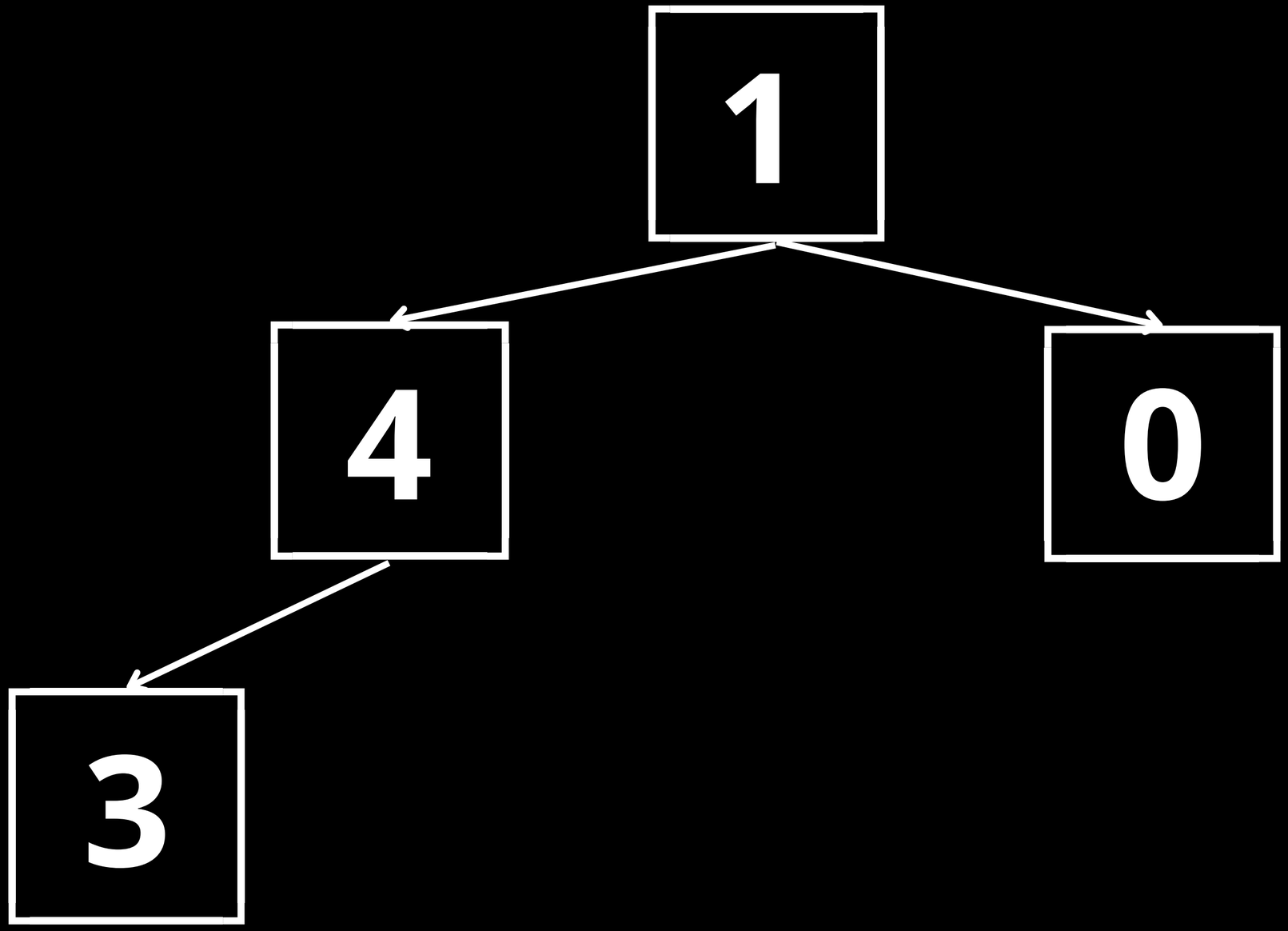
Current Array:

Now that we have removed the root, we can regenerate our Max Heap from indexes 0 to size-1

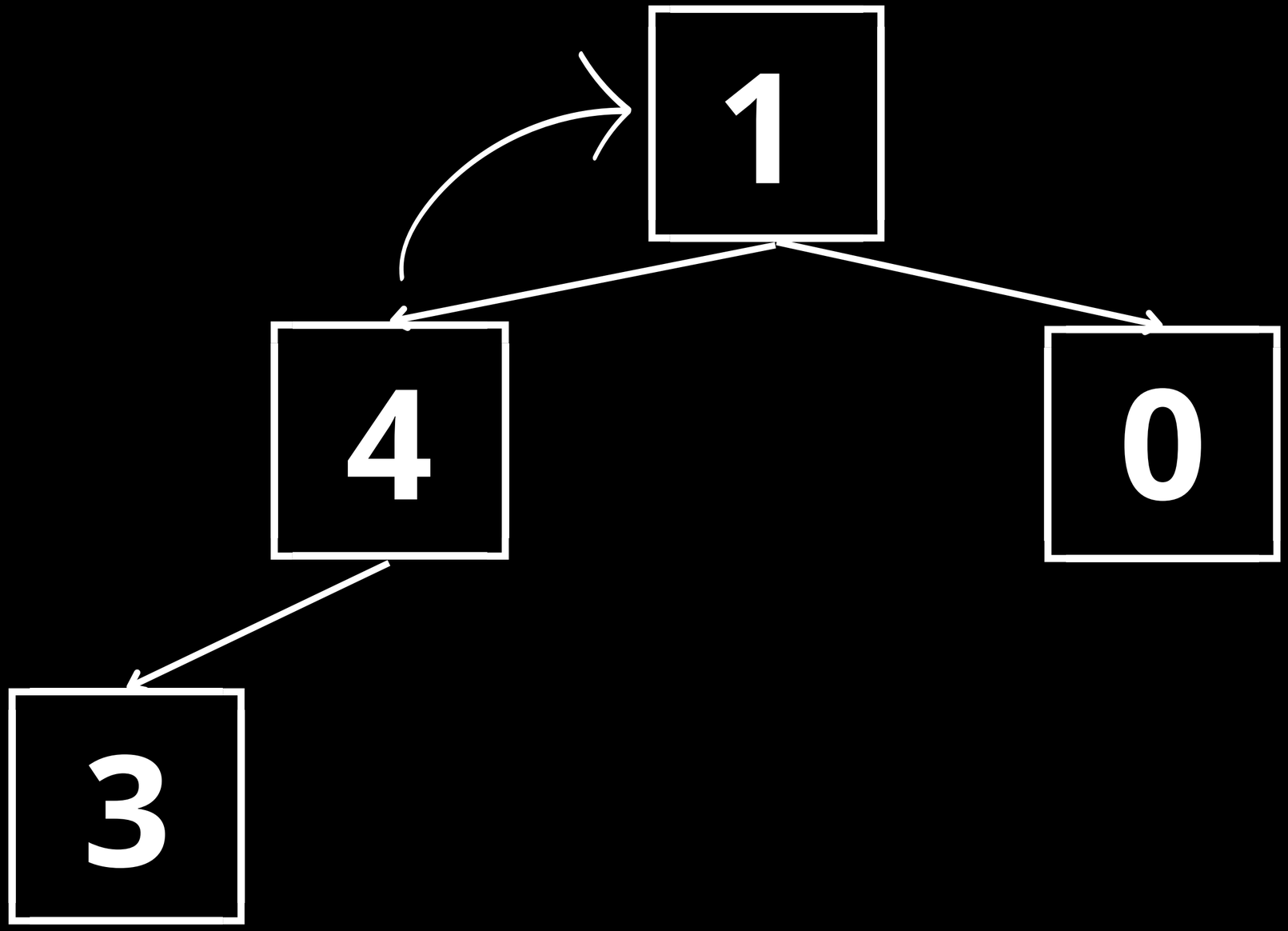
Current Array:



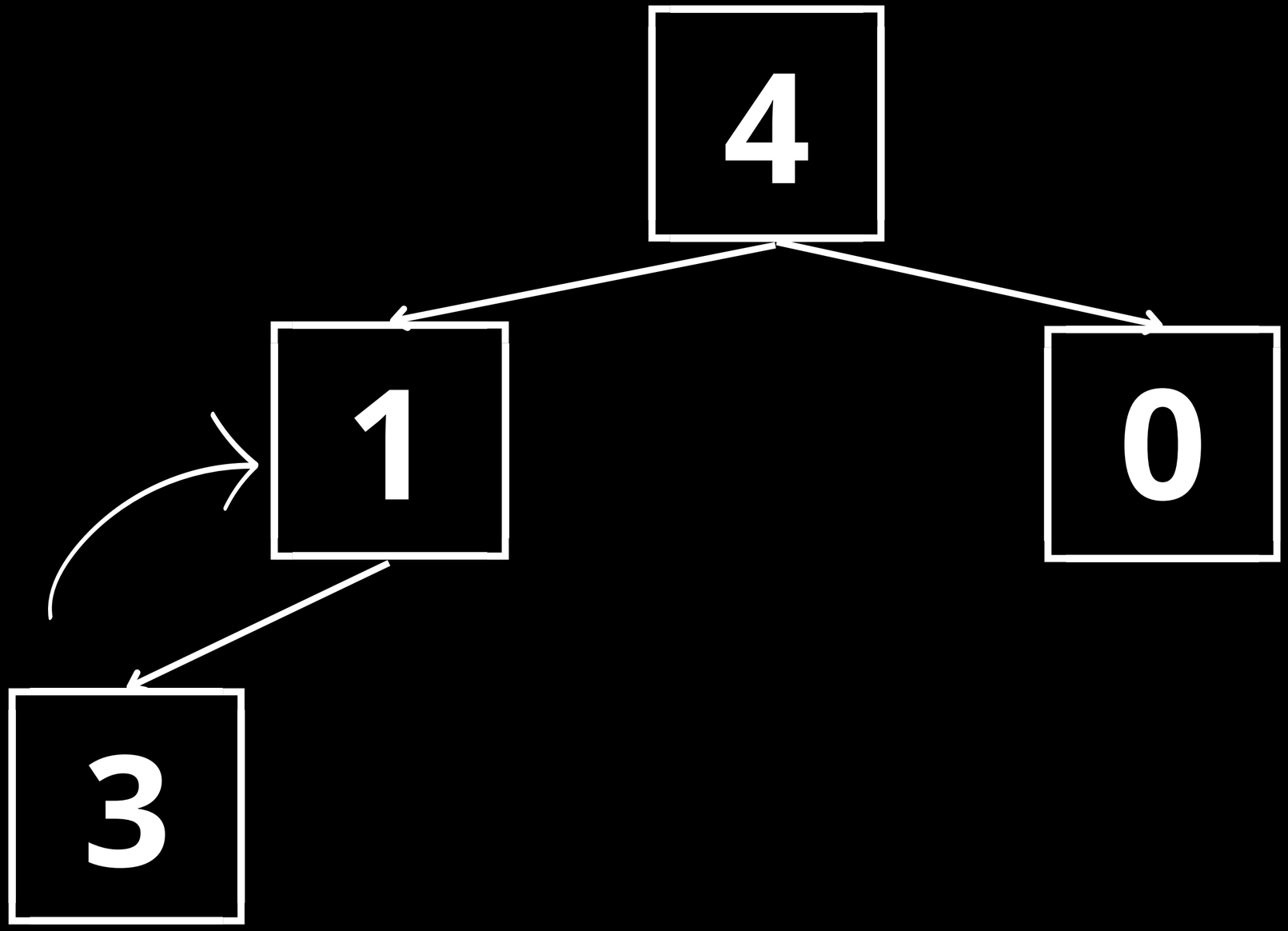
Next Binary Tree is:



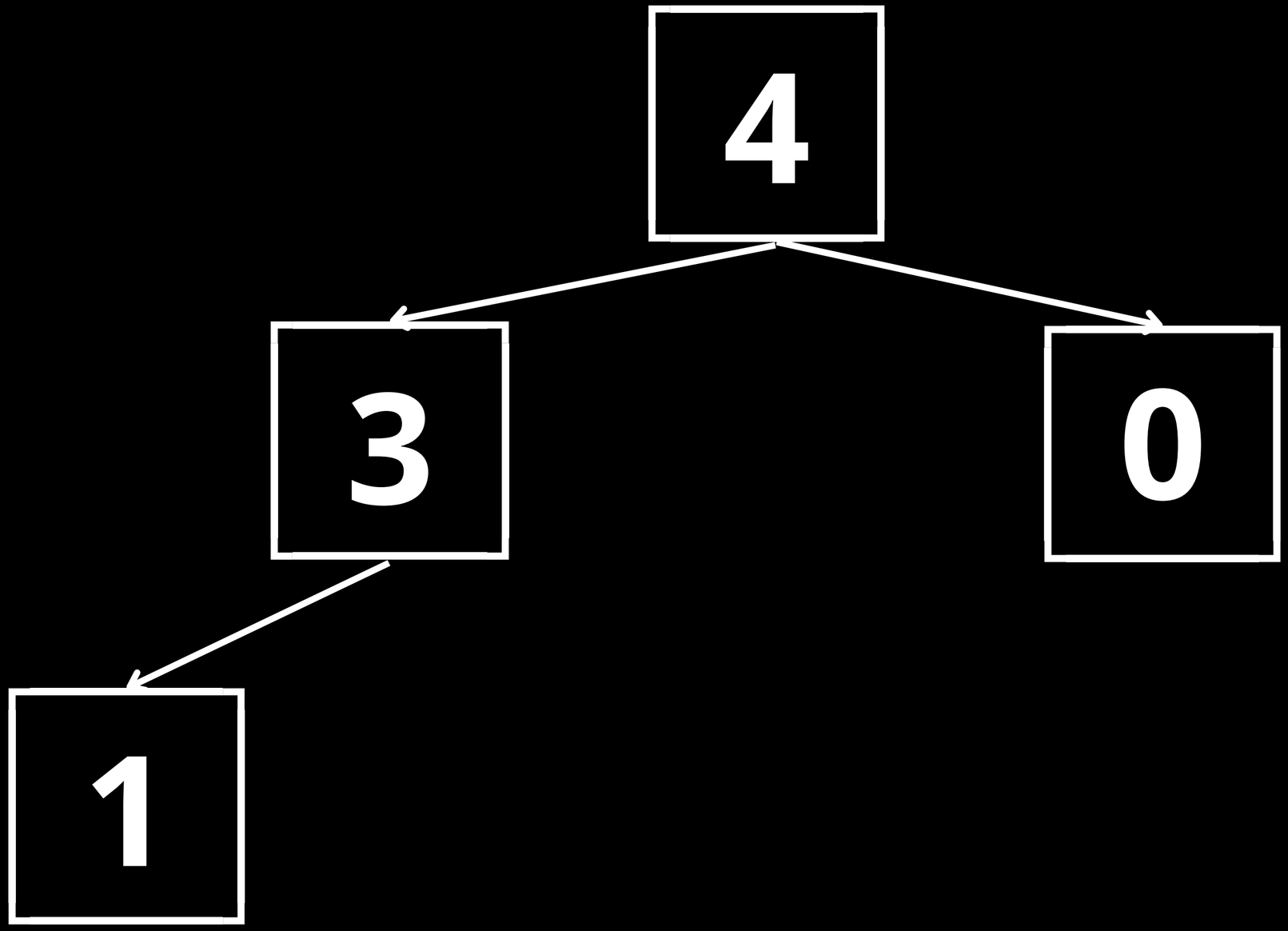
Now transform it into a Max Heap



Now transform it into a Max Heap

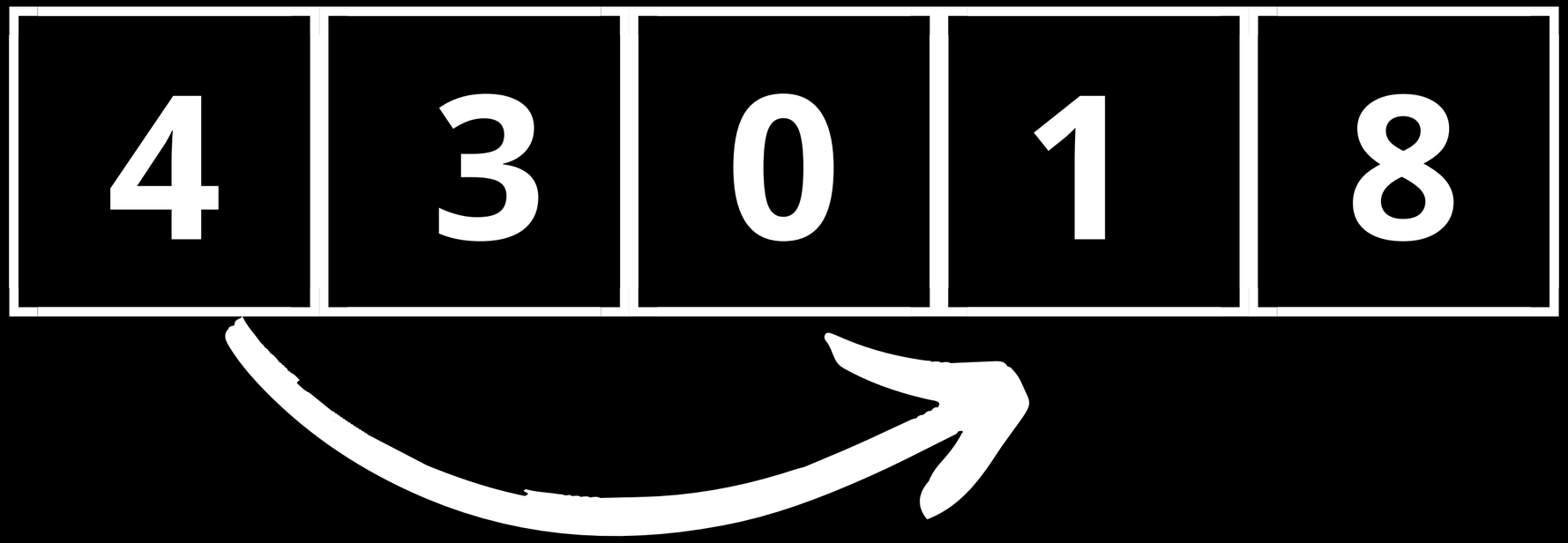


This is our new Max Heap



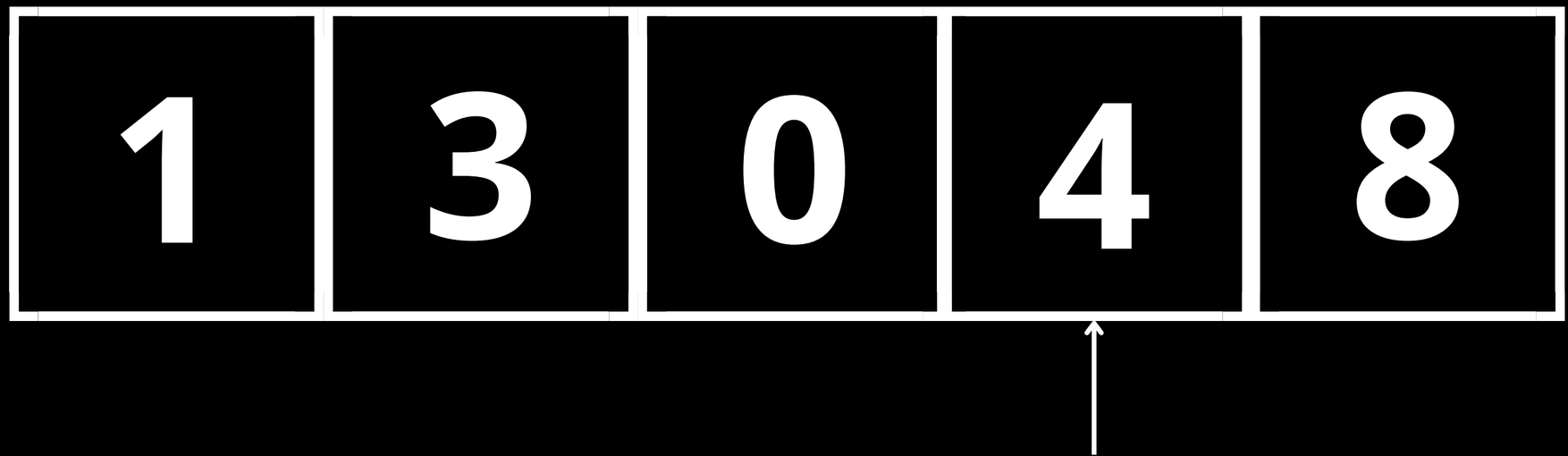
Now that we have a Max Heap, we can remove the root. To do so, we can simply swap it with the last current element in the array, and re-generate a Max Heap from the remaining elements up to size-2;

Current Array:

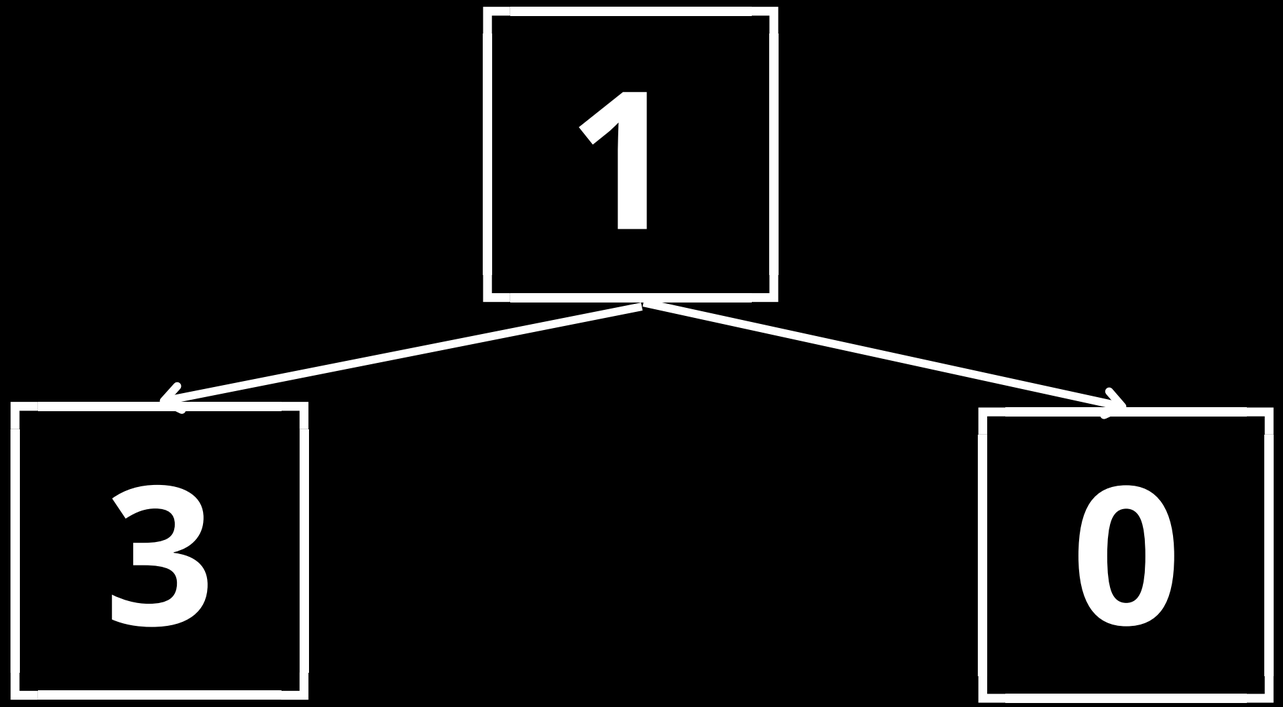


Now that we have removed the root, we can regenerate our Max Heap from indexes 0 to size-2

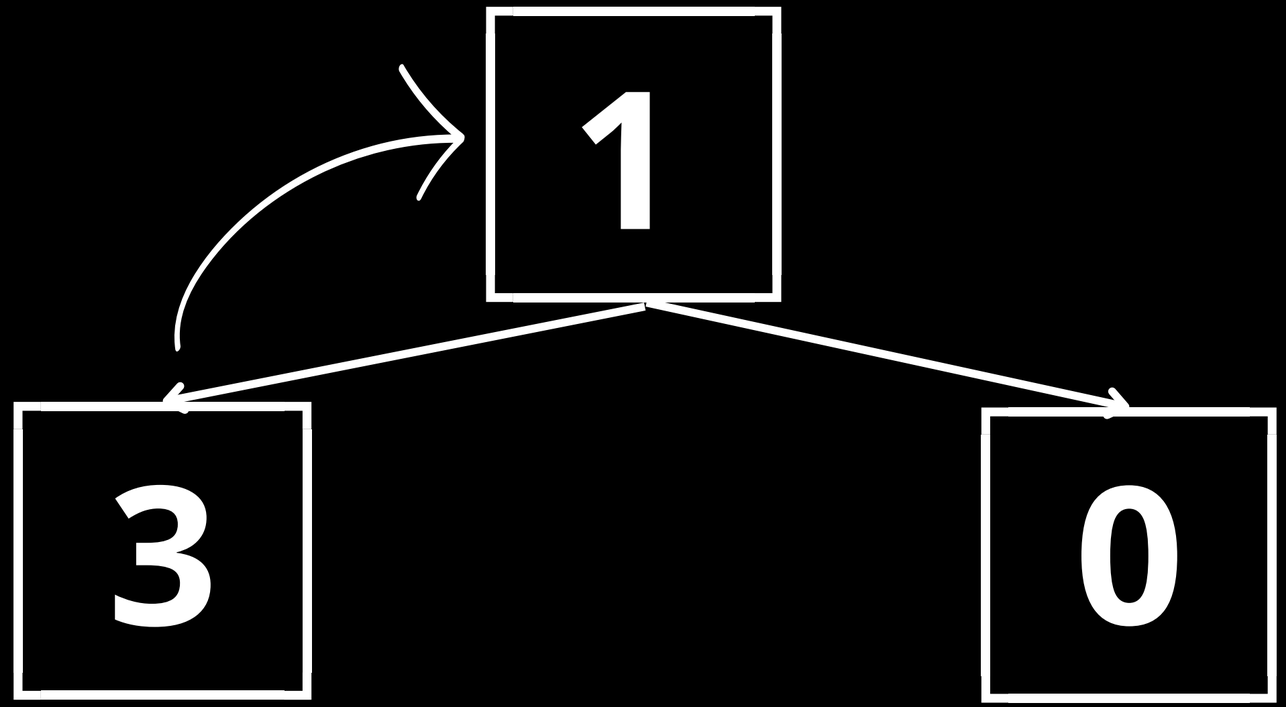
Current Array:



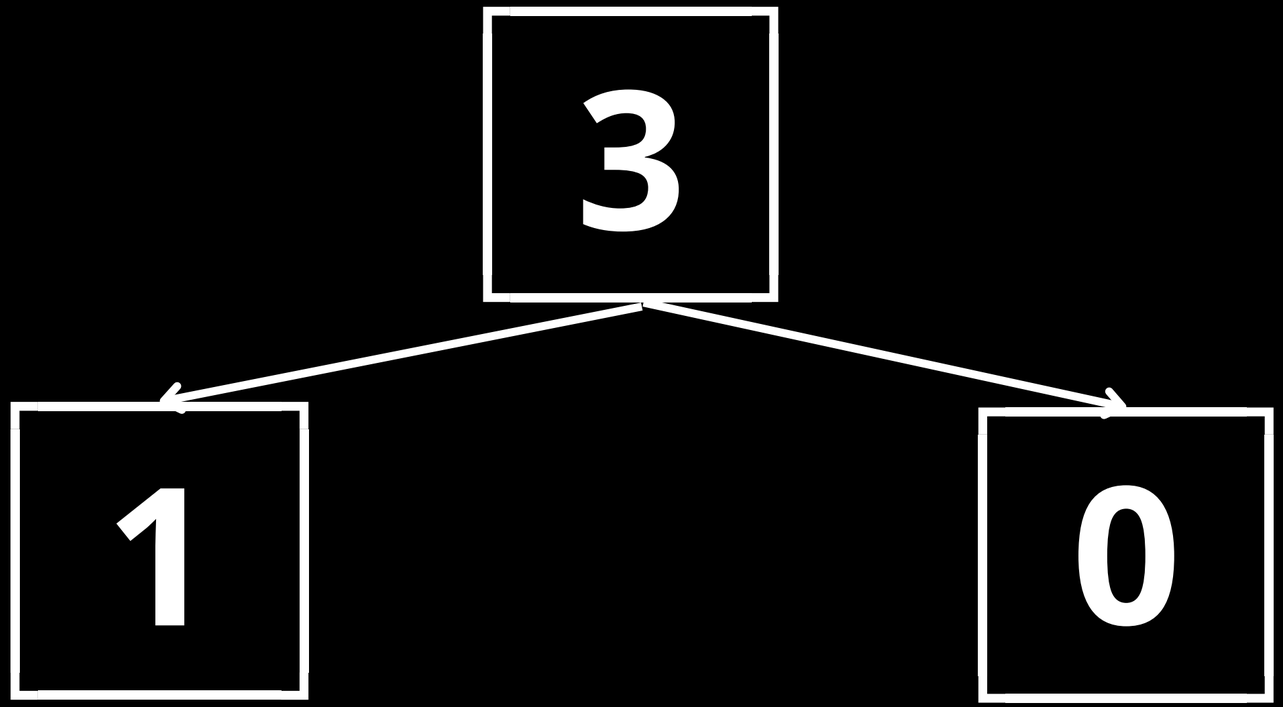
Next Binary Tree is:



Now transform it into a Max Heap

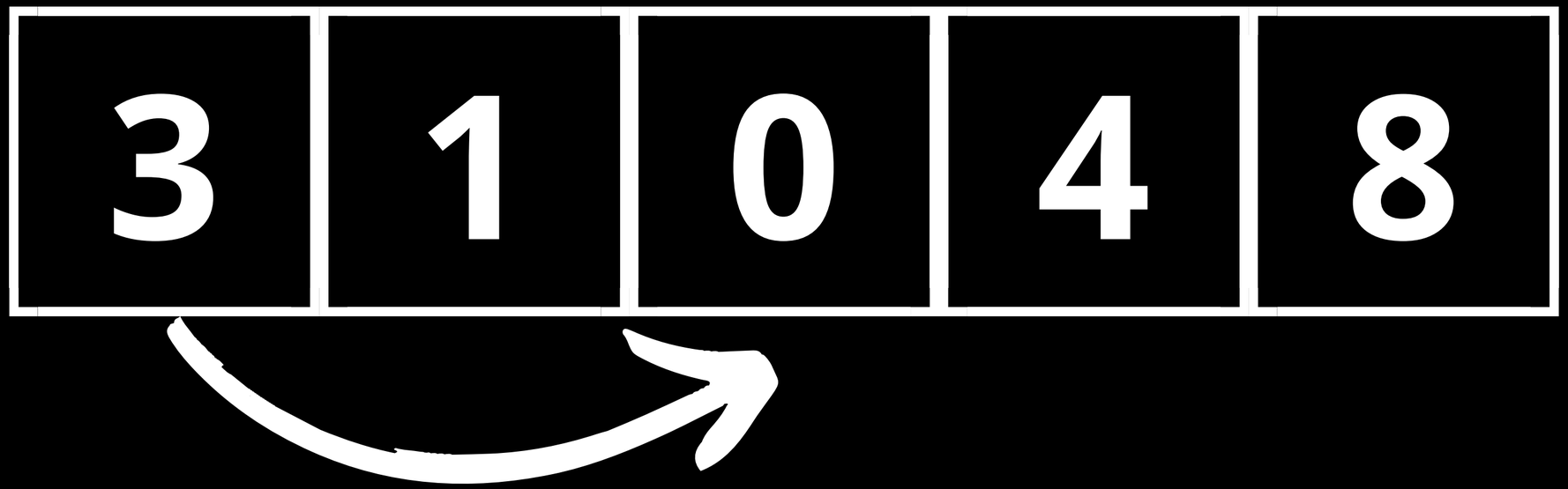


This is our new Max Heap



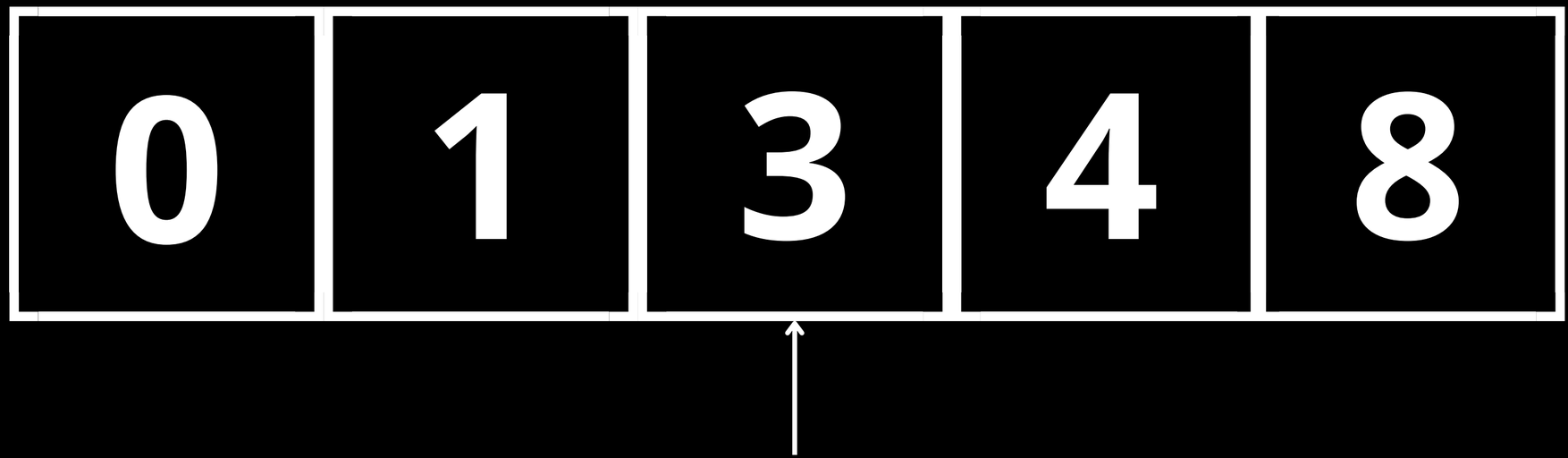
Now that we have a Max Heap, we can remove the root. To do so, we can simply swap it with the last current element in the array, and re-generate a Max Heap from the remaining elements up to size-3;

Current Array:

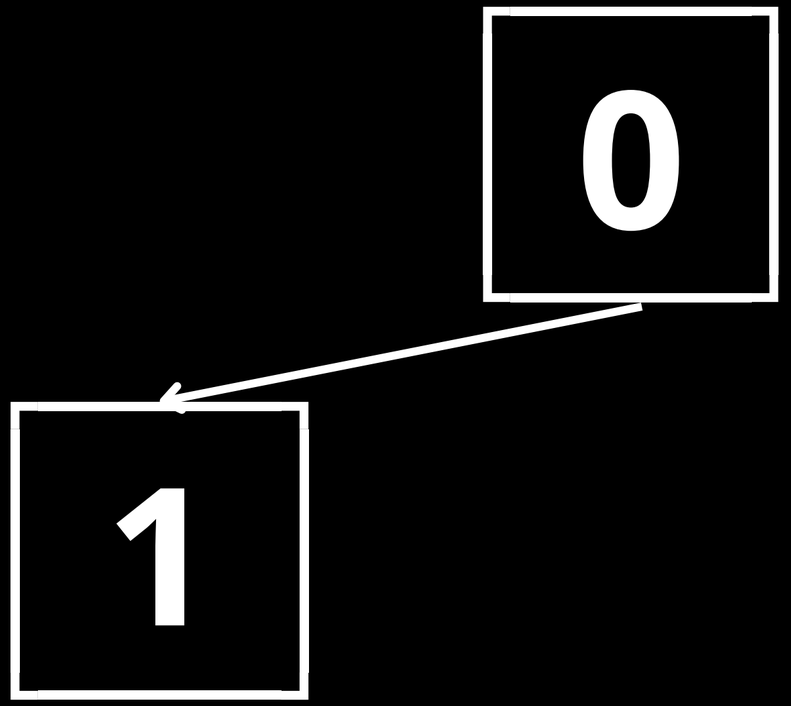


Now that we have removed the root, we can regenerate our Max Heap from indexes 0 to size-3

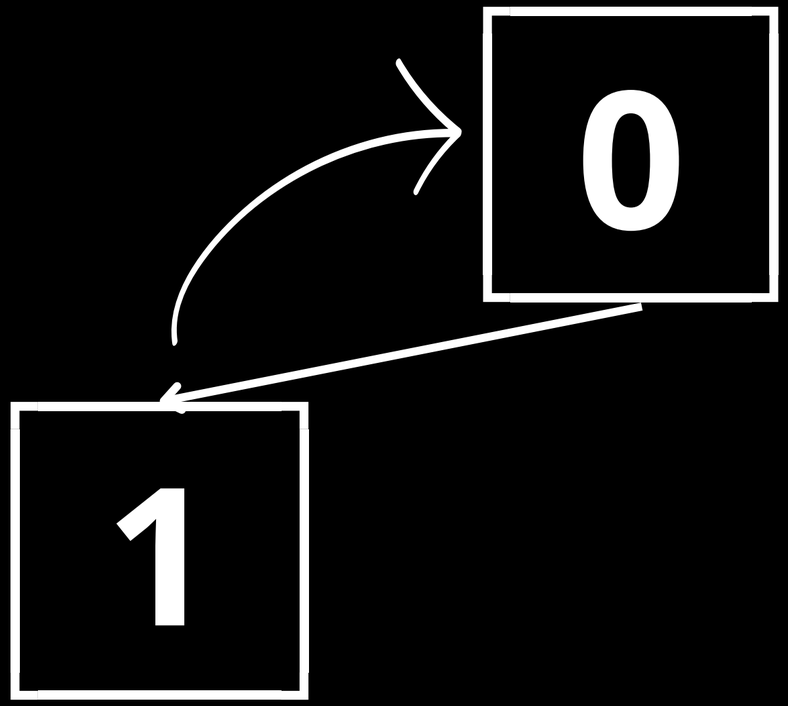
Current Array:



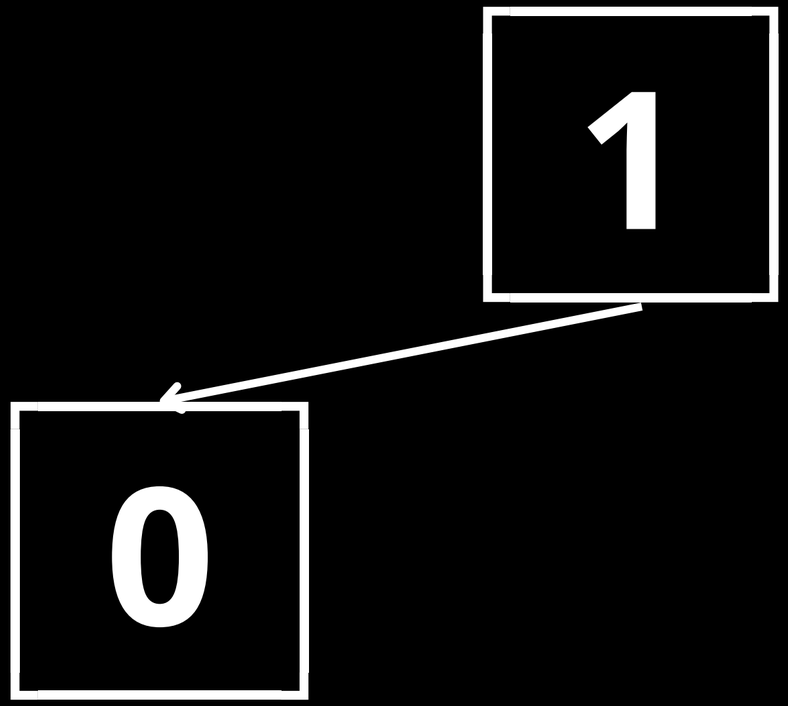
Next Binary Tree is:



Now transform it into a Max Heap

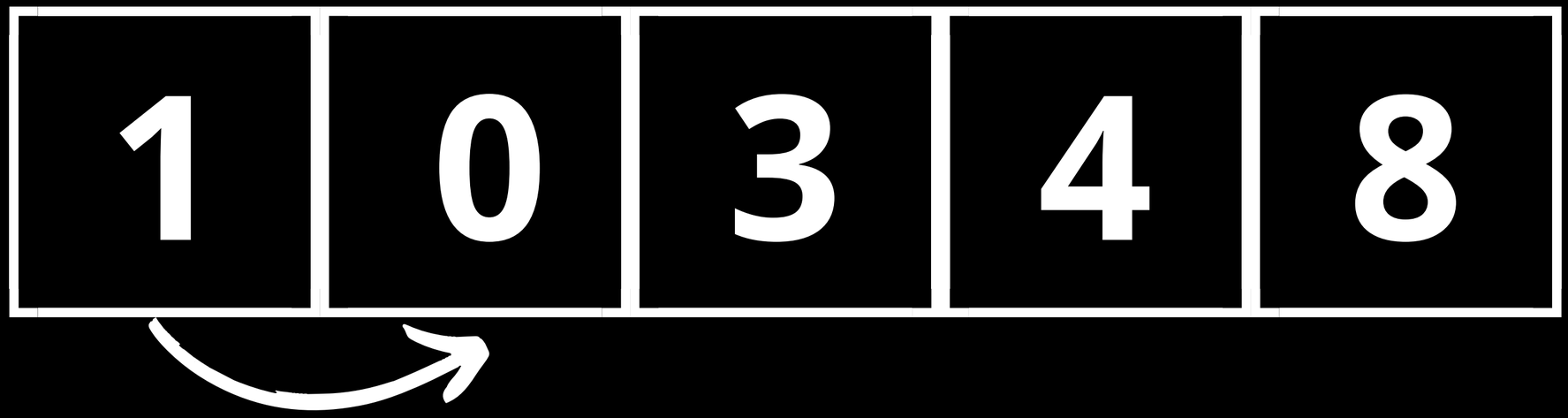


This is our new Max Heap



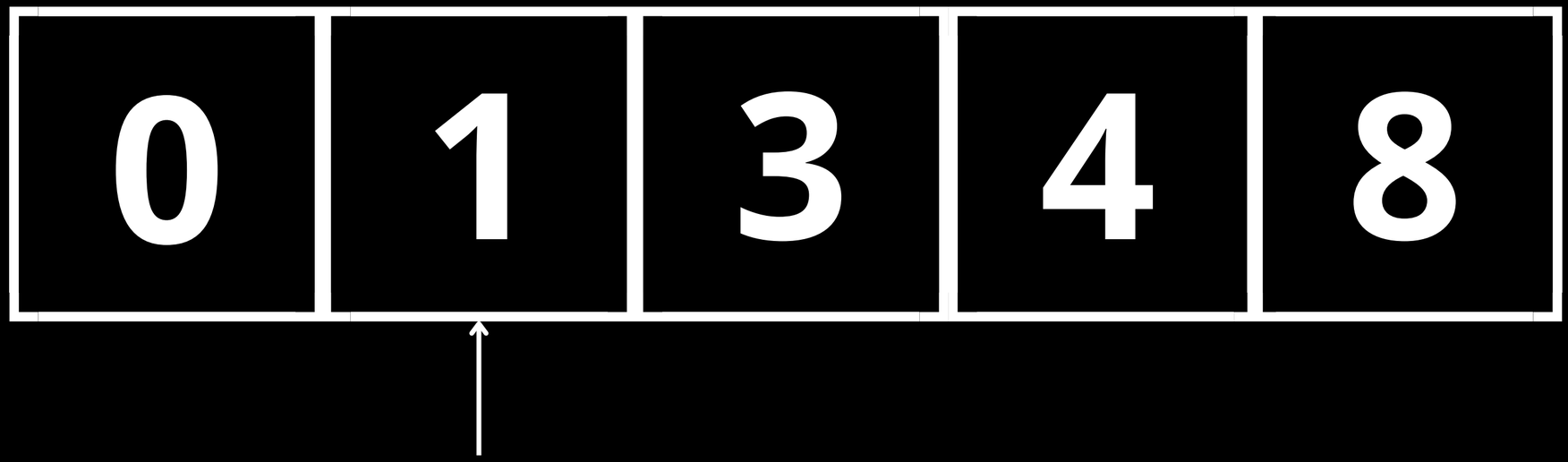
Now that we have a Max Heap, we can remove the root. To do so, we can simply swap it with the last current element in the array, and re-generate a Max Heap from the remaining elements up to size-4;

Current Array:



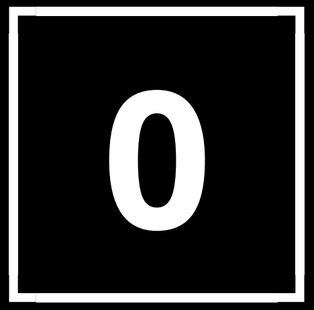
Now that we have removed the root, we can regenerate our Max Heap from indexes 0 to size-4

Current Array:



Ignore up to this index on next Max Heap iteration

Next Binary Tree is:



As there is only one element left on the tree, it’s already a Max Heap and it’s already in it’s correct position

Sorted Array