HEAPSORT

Overview:

- Uses a heap as its data structure
- In-place sorting algorithm – memory efficient
- Time complexity – O(n log(n))

What is a Heap?

A heap is also known as a priority queue and can be represented by a binary tree with the following properties:

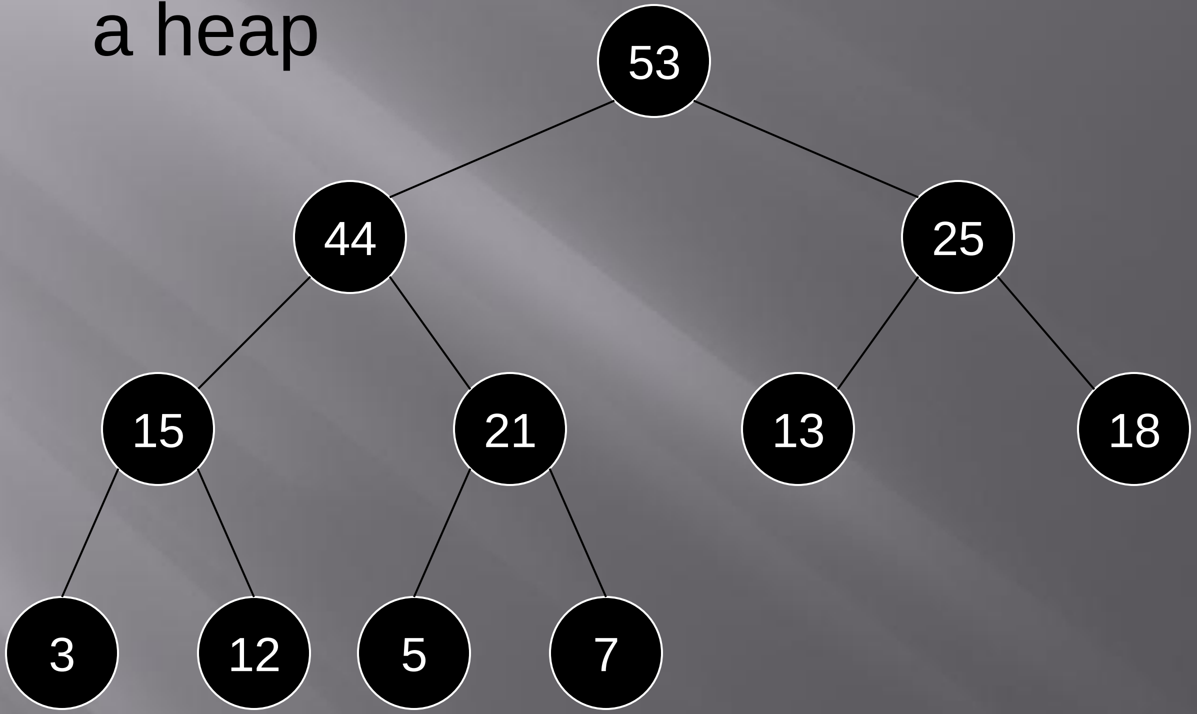
Structure property: A heap is a completely filled binary tree with the exception of the bottom row, which is filled from left to right

Heap Order property: For every node x in the heap, the parent of x greater than or equal to the value of x.

(known as a maxHeap).

Example:

a heap



Algorithm

Step 1. Build Heap – O(n)

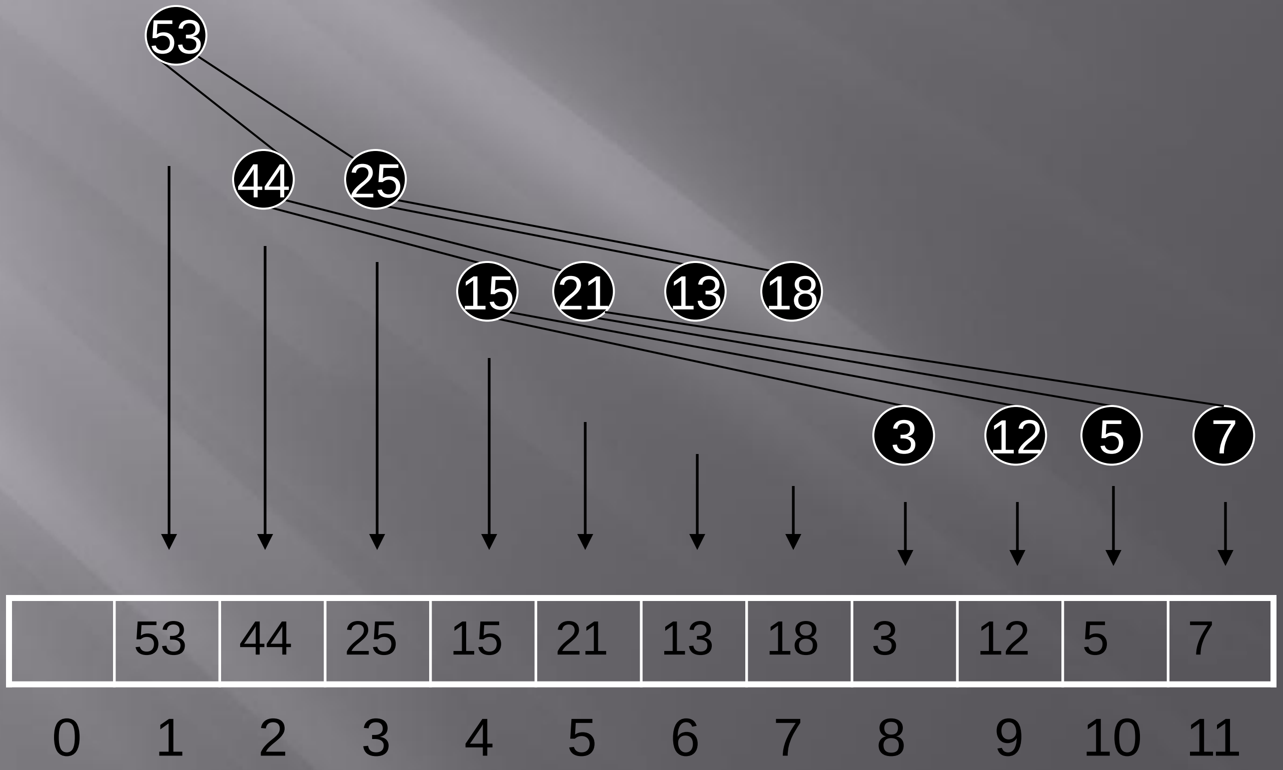
- Build binary tree taking N items as input, ensuring the heap structure property is held, in other words, build a complete binary tree.
- Heapify the binary tree making sure the binary tree satisfies the Heap Order property.

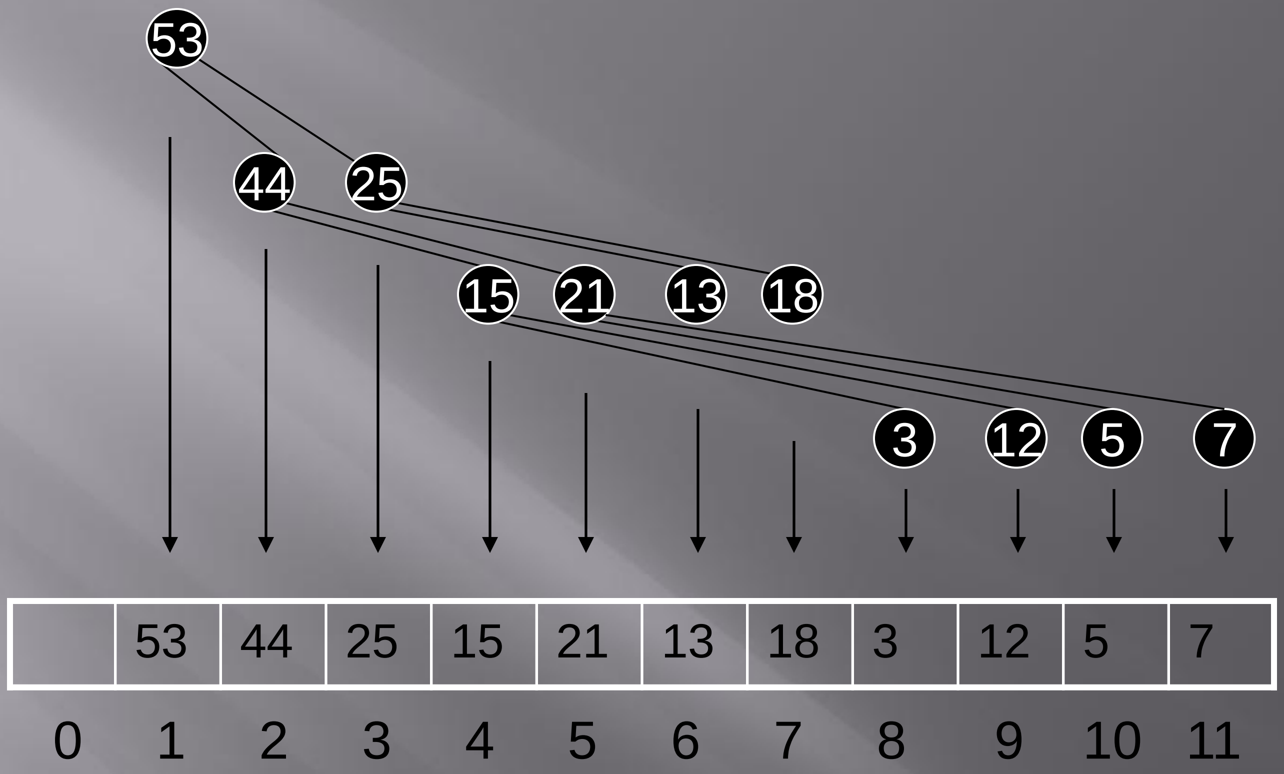
Step 2. Perform n deleteMax operations – O(log(n))

- Delete the maximum element in the heap – which is the root node, and place this element at the end of the sorted array.

Simplifying things

- For speed and efficiency we can represent the heap with an array. Place the root at array index 1, its left child at index 2, its right child at index 3, so on and so forth…





For any node i, the following formulas apply:

The index of its parent = i / 2

Index of left child = 2 * i

Index of right child = 2 * i + 1

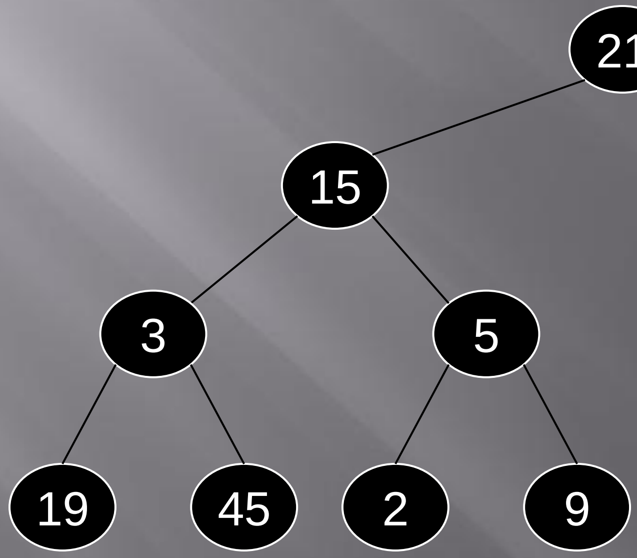
Sample Run

- Start with unordered array of data

Array representation:

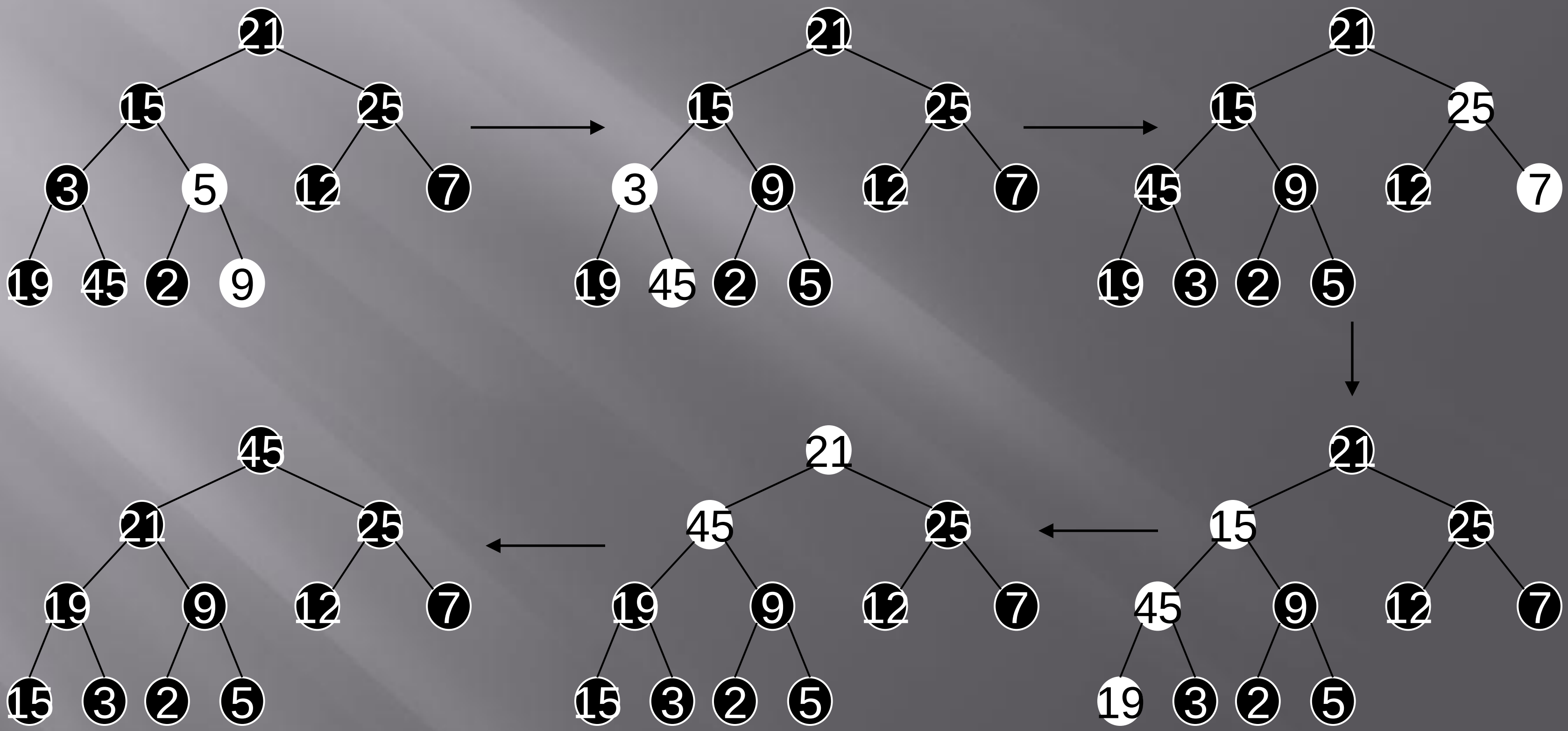
	21	15	25	3	5	12	7	19	45	2	9

Binary tree representation:

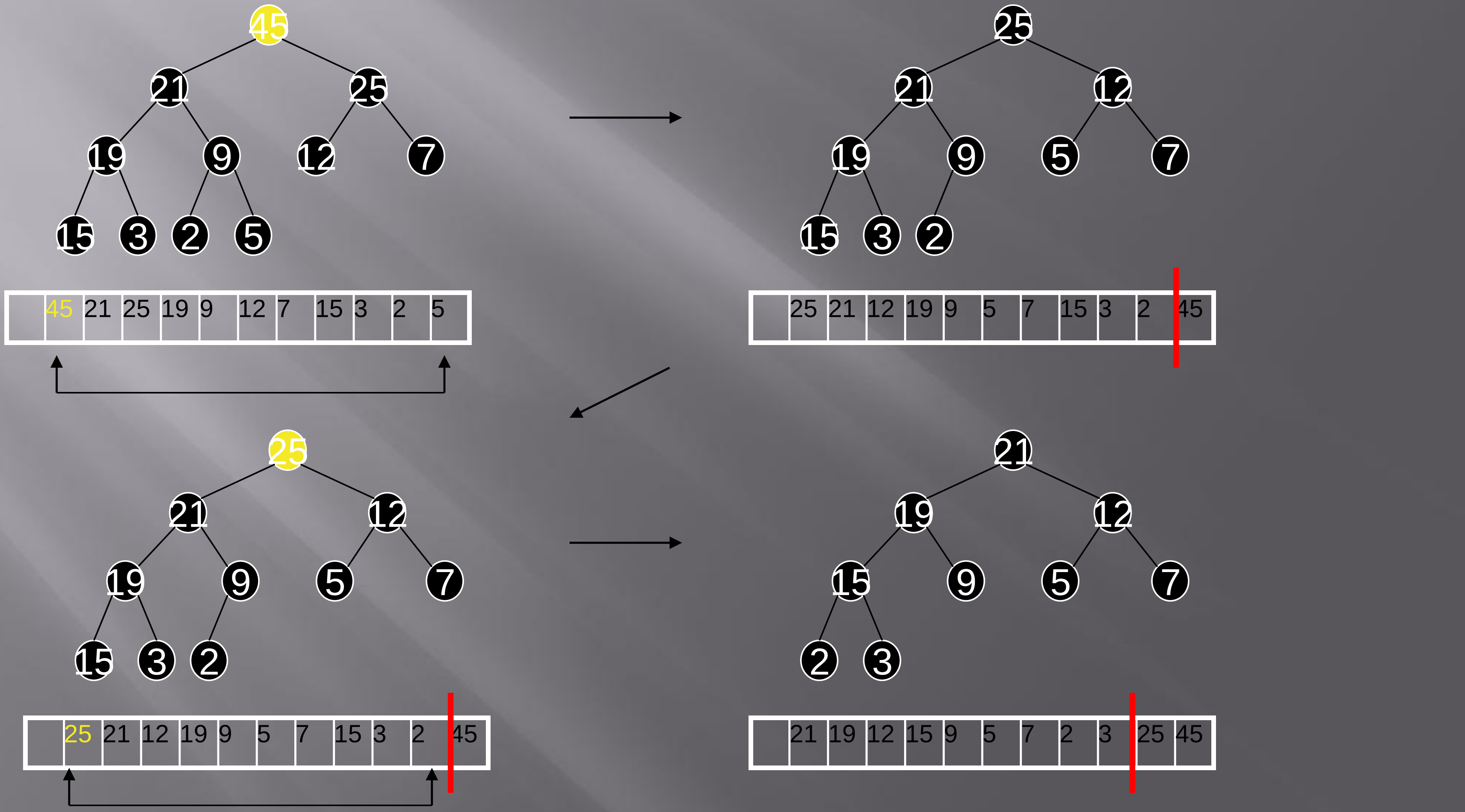


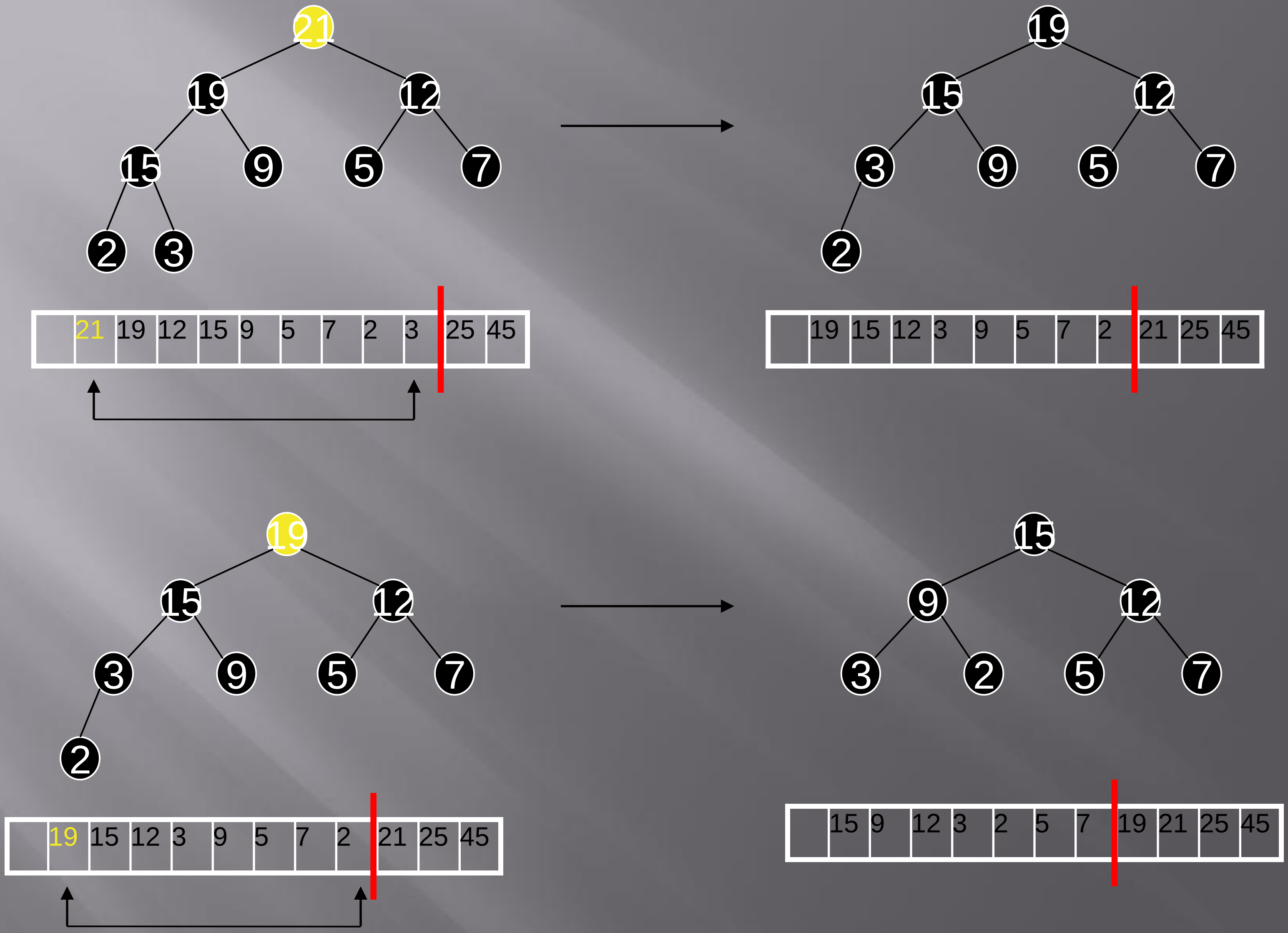
Sample Run

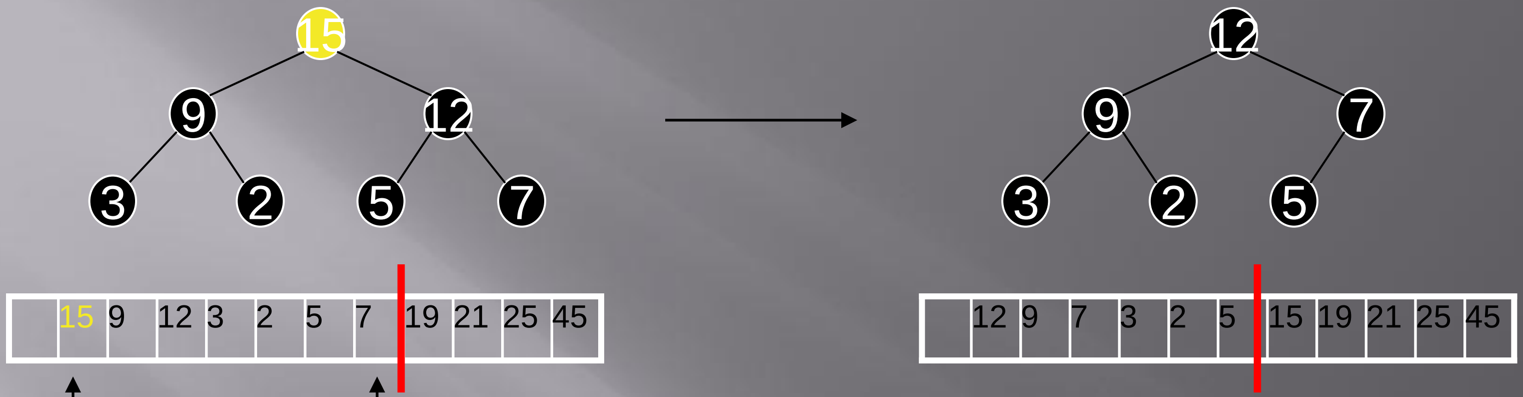
- Heapify the binary tree -

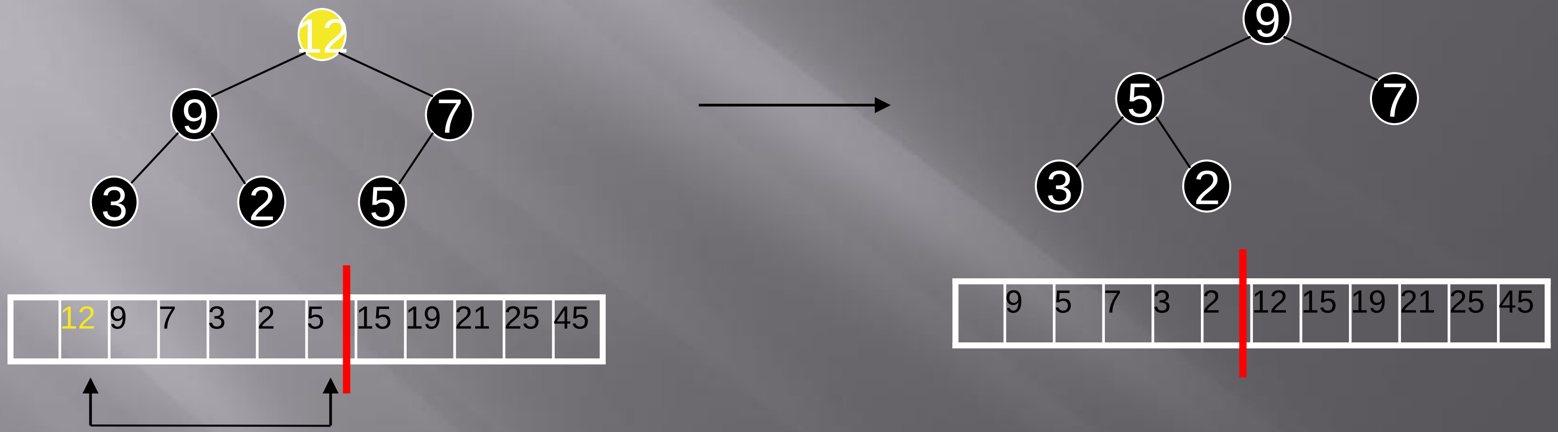


Step 2 – perform n – 1 deleteMax(es), and replace last element in heap with first, then re-heapify. Place deleted element in the last nodes position.

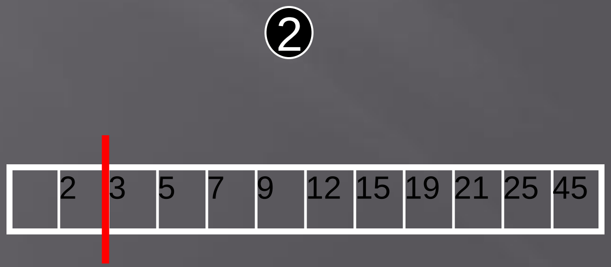



21
19
12
15
9
5
7
2
3
21 19 12 15 9 5 7 2 3 25 45
19
15
12
3
9
5
7
2
19 15 12 3 9 5 7 2 21 25 45
19
15
12
3
9
5
7
2
19 15 12 3 9 5 7 2 21 25 45
15
9
12
3
2
5
7
15 9 12 3 2 5 7 19 21 25 45


15
9
12
3
2
5
7
15 9 12 3 2 5 7 19 21 25 45
12
9
7
3
2
5
12 9 7 3 2 5 15 19 21 25 45


12
9
7
3
2
5
12 9 7 3 2 5 15 19 21 25 45
9
5
7
3
2
9 5 7 3 2 12 15 19 21 25 45

…and finally


2
2 3 5 7 9 12 15 19 21 25 45

Conclusion

- 1st Step- Build heap, O(n) time complexity
- 2nd Step – perform n deleteMax operations, each with O(log(n)) time complexity
- total time complexity = O(n log(n))
- Pros: fast sorting algorithm, memory efficient, especially for very large values of n.
- Cons: slower of the O(n log(n)) sorting algorithms