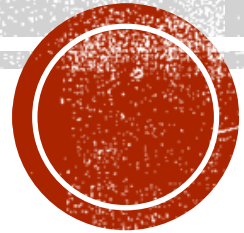
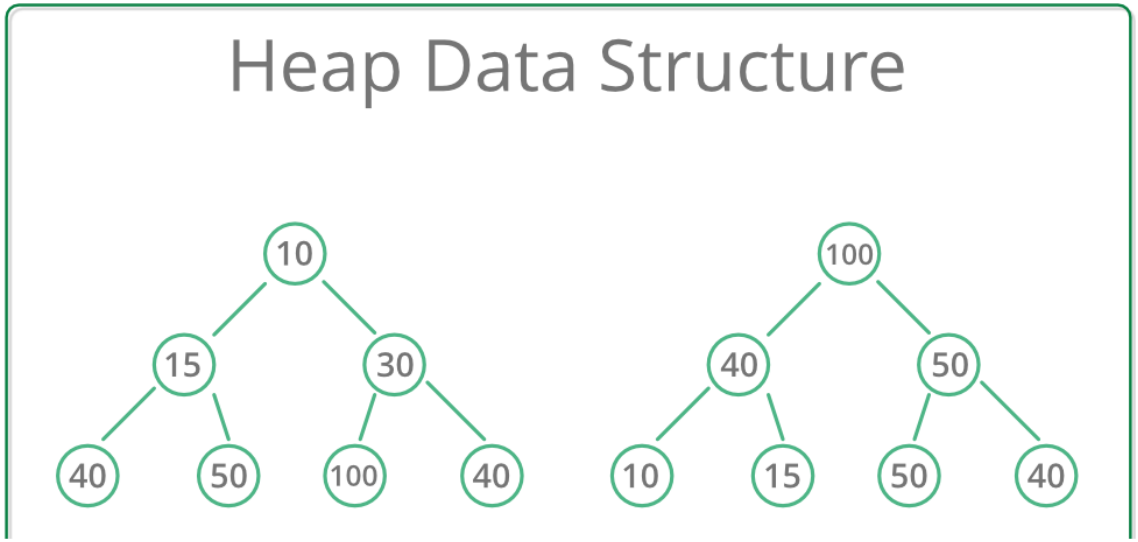
DATA STRUCTURES AND ALGORITHMS



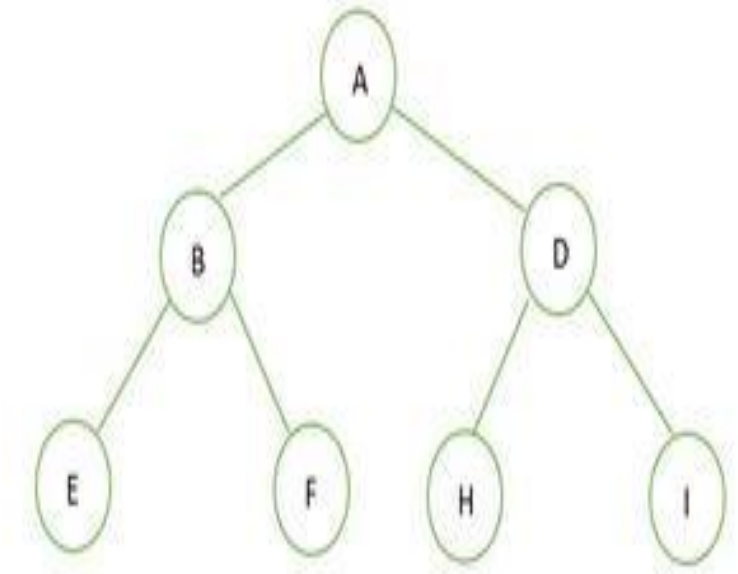
HEAP

- Heap is a tree based data structure
- It follows the following properties:
  - Structural Property
  - Ordering Property

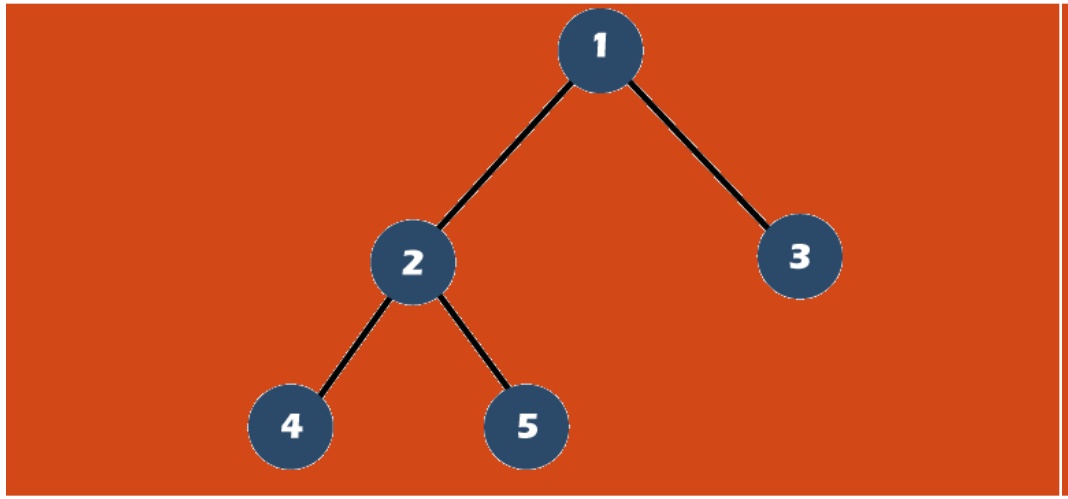


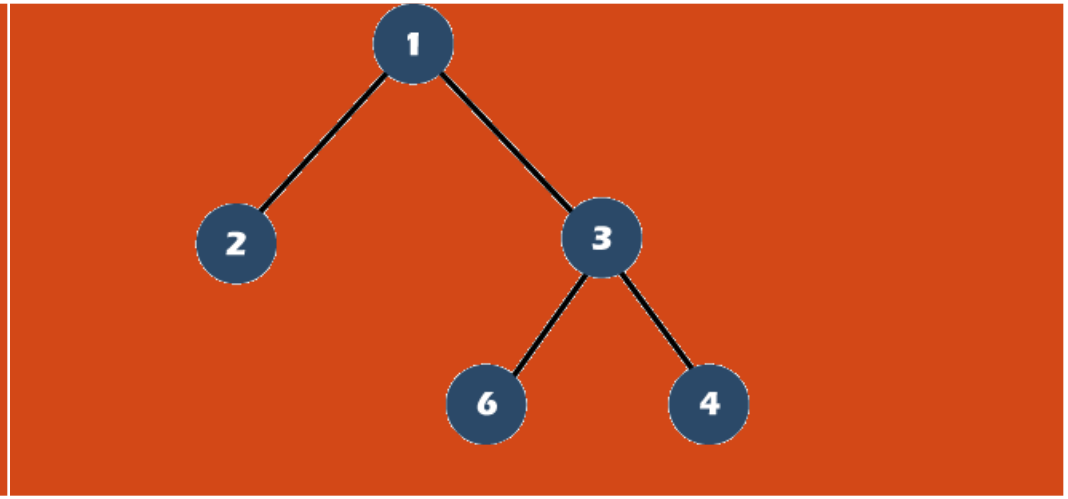
STRUCTURAL PROPERTY OF HEAP

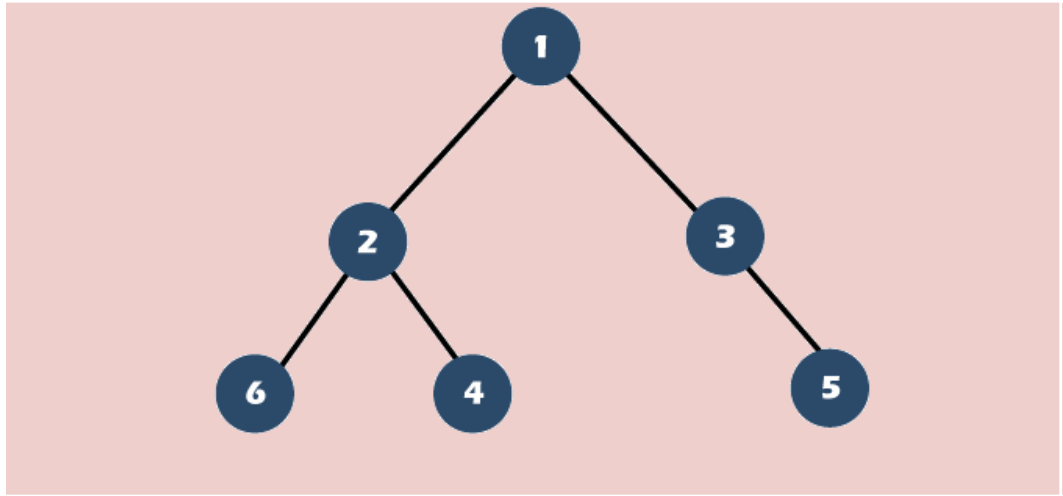
- Structural property of heap says that the tree must be a complete binary tree.
- A complete binary tree is a binary tree in which all the levels of binary tree is completely filled except possibly the last level, which is filled from left side.
- All the levels should be filled except for the last one. The last level may or may not be filled.
- All the nodes should be filled from left to right.

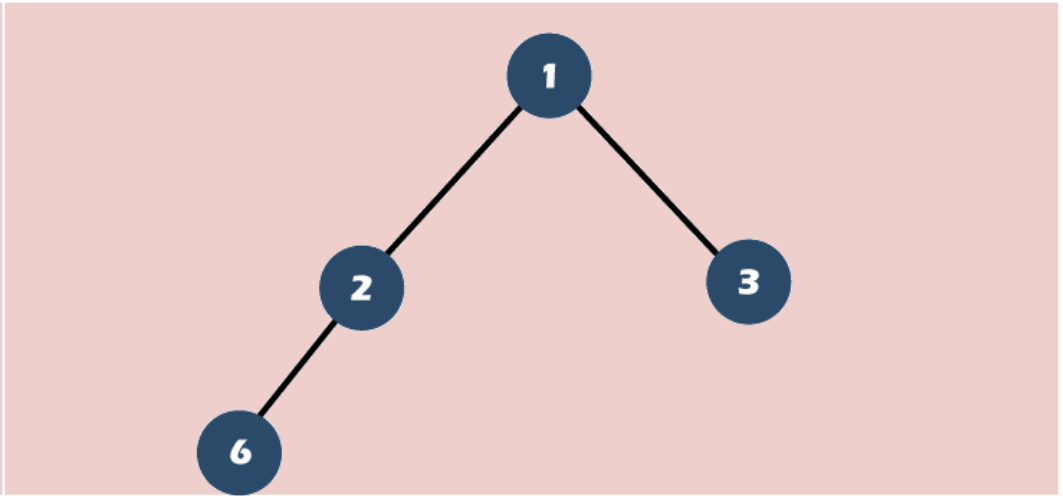


COMPLETE BINARY TREE







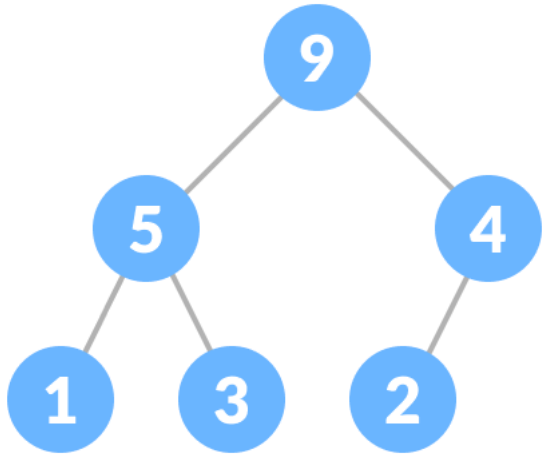


ORDERING PROPERTY OF HEAP

- Generally, heaps are of two orders/ types.
  - Min Heap
  - Max Heap

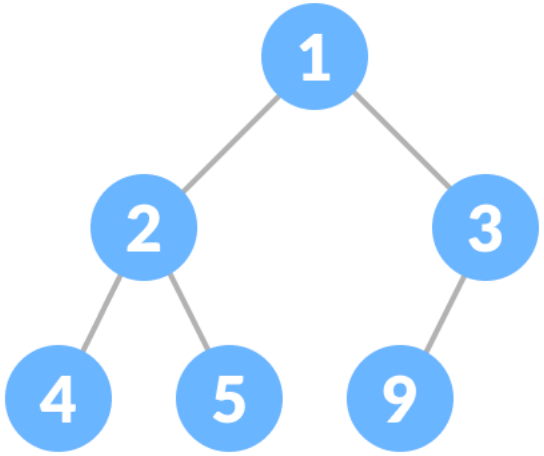
MAX HEAP

- In this heap, the value of the root node must be the greatest among all its child nodes and the same thing must be done for its left and right sub-tree also.



MIN HEAP

- In this heap, the value of the root node must be the smallest among all its child nodes and the same thing must be done for its left and right sub-tree also.



HEAP TREE CONSTRUCTION

- There can be two ways to create a heap tree:

1. Insert each key one by one
   - Time complexity is O(n log n)
2. Heapify Method
   - Time complexity is O(n)

HEAP CONSTRUCTION (ONE BY ONE)

- In this method, a heap tree is constructed according to the following points:
  - Insert the root node
  - Insert the next key as left child. After insertion compare it with its parent node
  - If value of parent node is greater than/ less than the value of left child then swap those
  - Insert the next key as right child. After insertion compare it with its parent node
  - If value of parent node is greater than/ less than the value of right child then swap those

HEAP CONSTRUCTION (ONE BY ONE)

- Insert key one by one in empty space takes O(1)
- To insert a key into already constructed heap takes O(log n) in worst case
- It takes log n comparisons and log n swapping
- If there are n elements total then it would be O(n log n)

HEAP CONSTRUCTION (ONE BY ONE)

- 14, 24, 12, 11, 25, 8, 35

HEAP CONSTRUCTION (HEAPIFY)

- 145, 40, 25, 65, 12, 48, 18, 1, 100, 27, 7, 3, 45, 9, 30

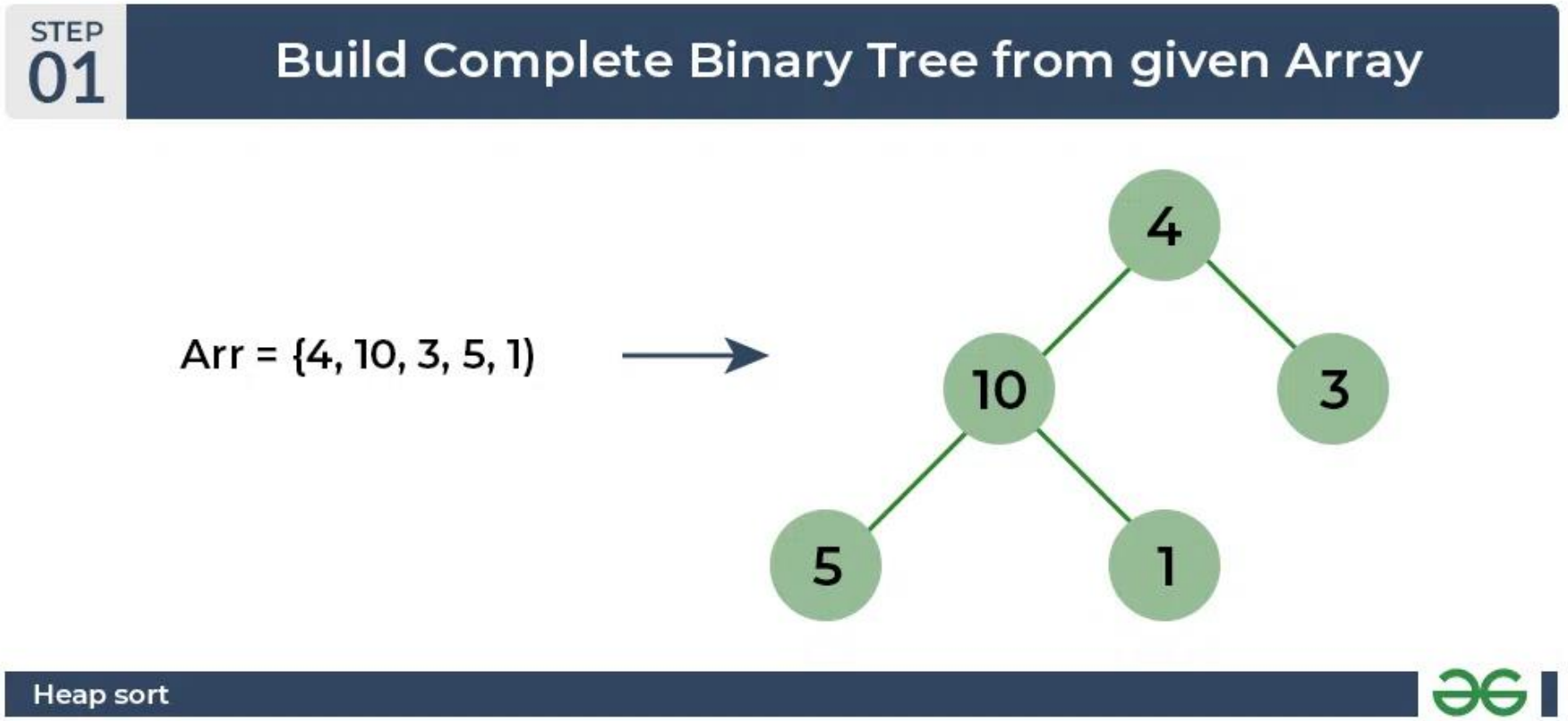
DELETION OF AN ELEMENT

- Deleting an element in heap tree is not simple
- You cannot delete an element directly from heap tree except that it is rightmost leaf element
- Since deleting an element at any intermediary position in the heap can be costly, so we can simply replace the element to be deleted by the last element and delete the last element of the Heap
- Replace the root or element to be deleted by the last element
- Delete the last element from the Heap
- Since, the last element is now placed at the position of the root node. So, it may not follow the heap property. Therefore, **heapify** the last node placed at the position of root
- Time complexity is O(log n) where n is no of elements in the heap

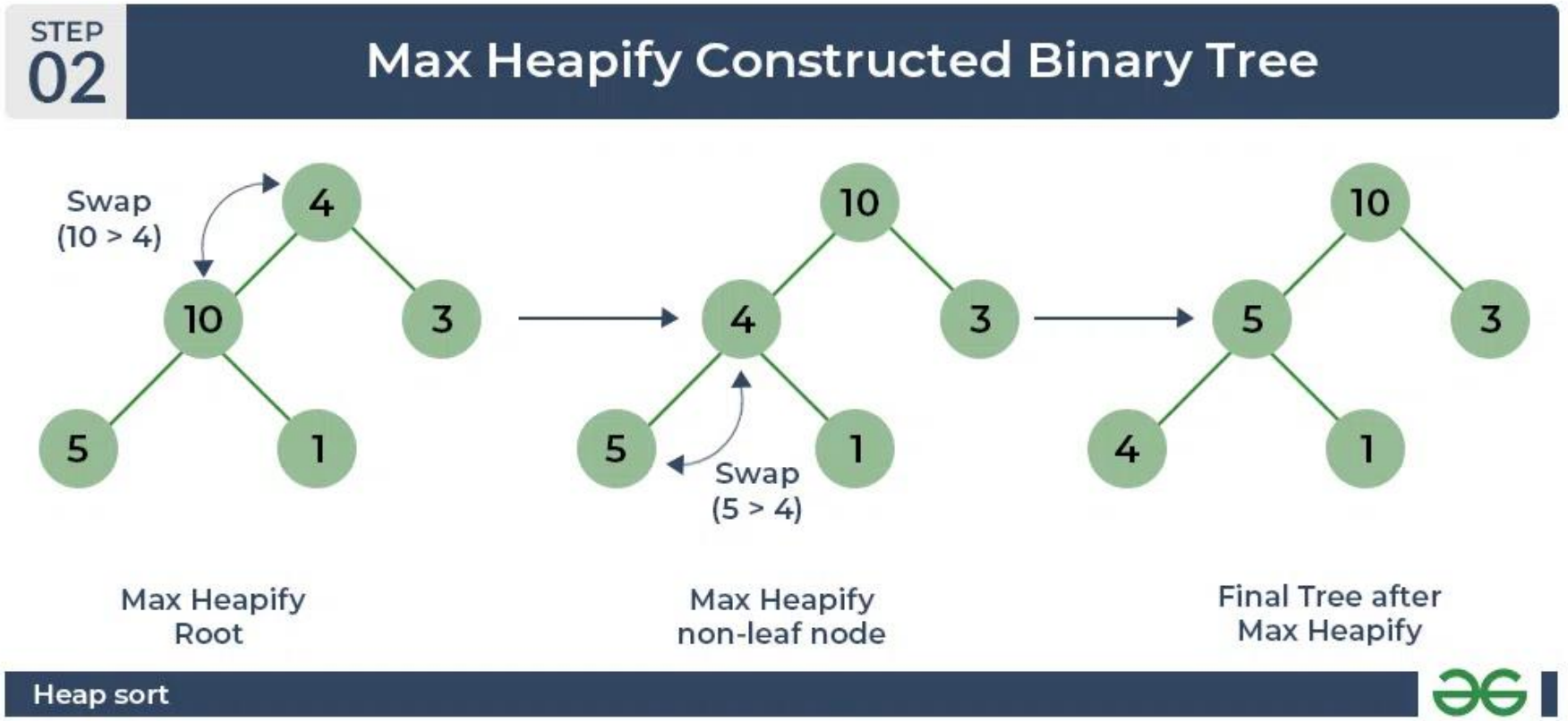
HEAP SORT

- Heap sort is a comparison-based sorting technique based on binary heap data structure.
- It is similar to the selection sort where we first find the minimum element and place the minimum element at the beginning. Repeat the same process for the remaining elements.
- In case of max heap:
  - One by one delete the root node of the Max-heap and replace it with the last node in the heap and then heapify the root of the heap.
- In case of min heap:
  - One by one delete the root node of the Min-heap and replace it with the first node in the heap and then heapify the root of the heap.
- Time Complexity: O(n log n) → O(n) + O(n log n)

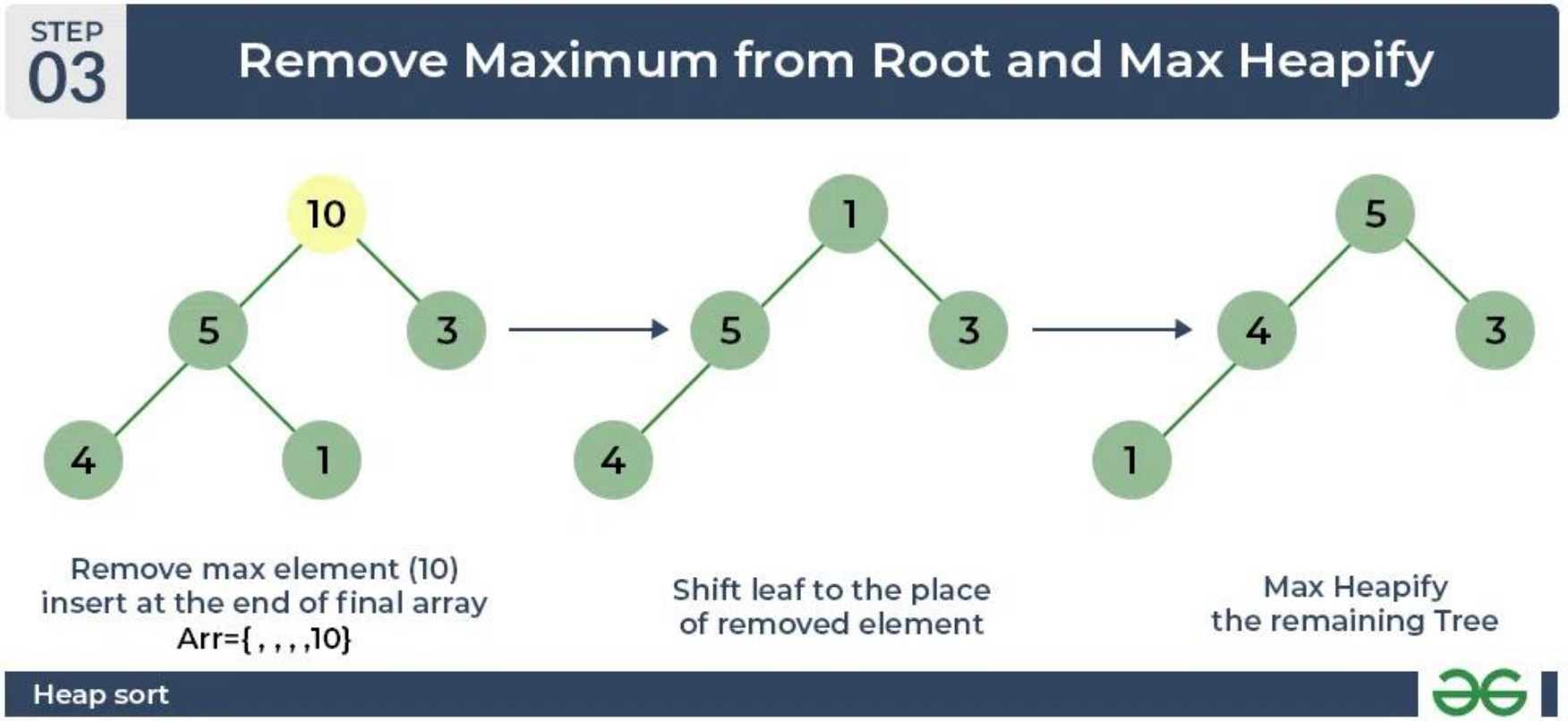
EXAMPLE: HEAP SORT



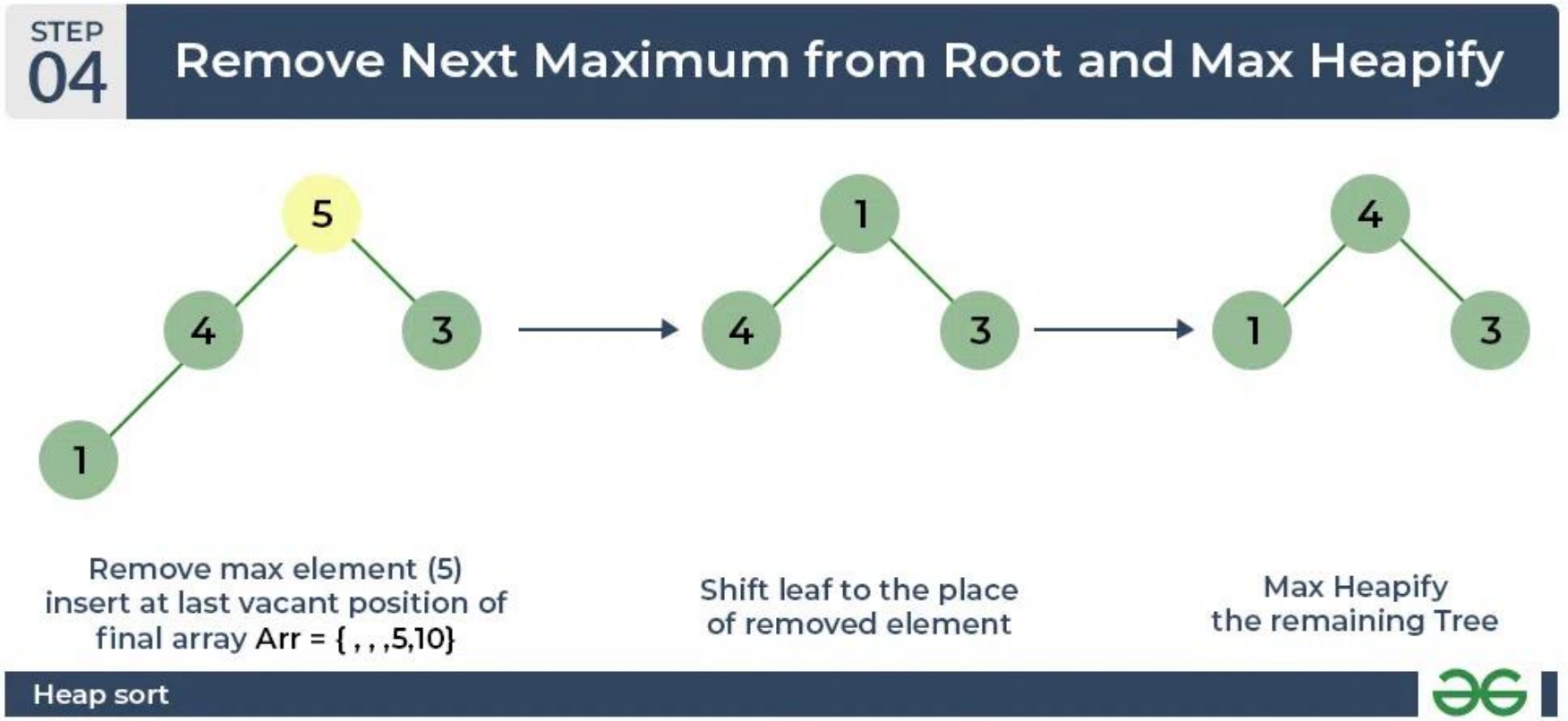
EXAMPLE: HEAP SORT



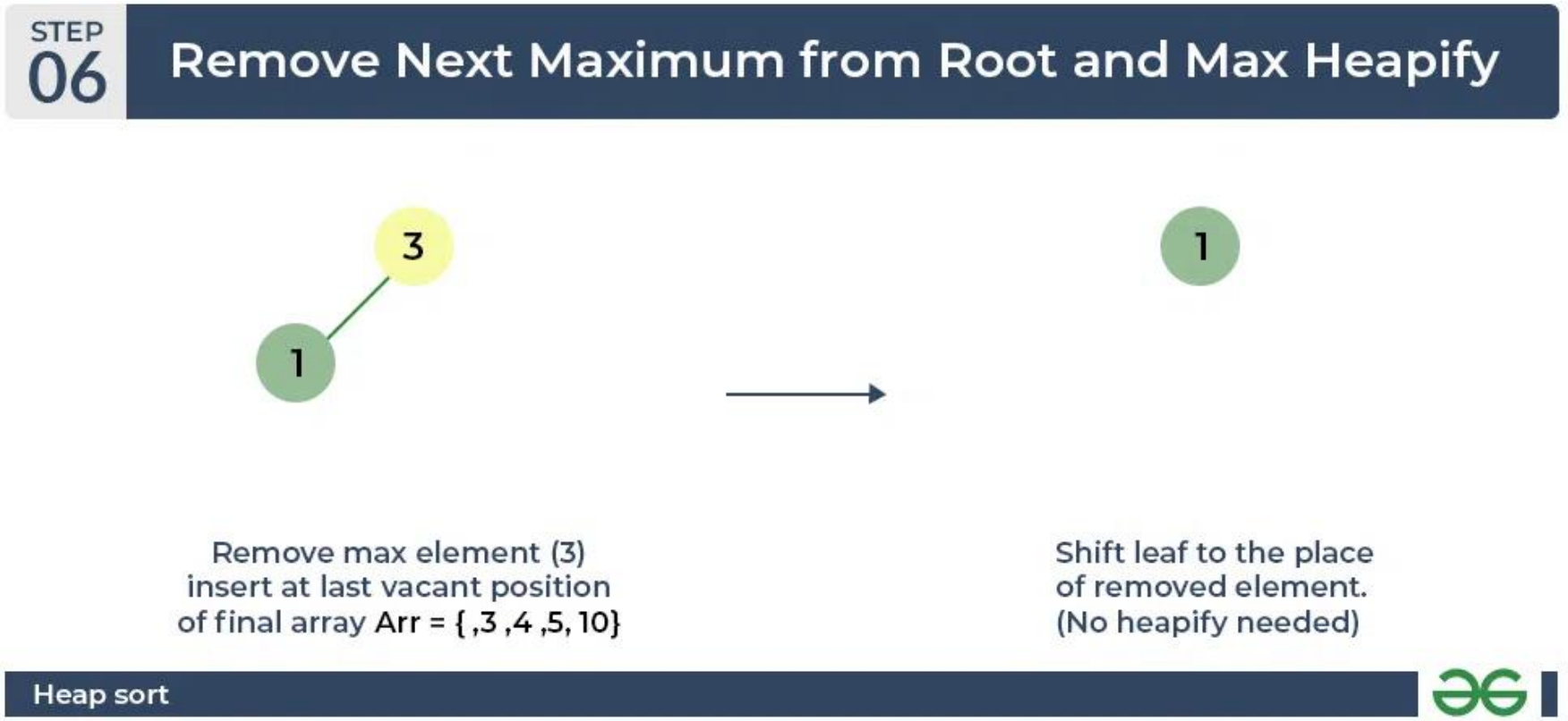
EXAMPLE: HEAP SORT



EXAMPLE: HEAP SORT



EXAMPLE: HEAP SORT



EXAMPLE: HEAP SORT