UNIVERSITY INSTITUTE OF COMPUTING

MASTER OF COMPUTER APPLICATIONS

DESIGN AND ANALYSIS OF ALGORITHMS

24CAT-611

UNIT-2

DISCOVER . LEARN . EMPOWER

DESIGN AND ANALYSIS OF ALGORITHMS

Course Outcome

CO Number	Title	Level
CO3	Apply and analyze important algorithmic design paradigms and their applications	Understand
CO4	Implement the major graph algorithms to model engineering problems	Understand

- **Divide and Conquer**: General method, Binary search, Advantages and disadvantages of divide and conquer, Decrease and conquer approach: Topological sort

Topics to be covered

- Dijkstra's Algorithm
- Huffman Trees and Codes
- Heaps and Heap Sort

Heap Sort Algorithm

- Heap Sort is a popular and efficient sorting algorithm in computer programming. Learning how to write the heap sort algorithm requires knowledge of two types of data structures - arrays and trees.

- The initial set of numbers that we want to sort is stored in an array e.g. [10, 3, 76, 34, 23, 32] and after sorting, we get a sorted array [3,10,23,32,34,76]

- Heap sort works by visualizing the elements of the array as a special kind of complete binary tree called a heap.

Relationship between Array Indexes and Tree Elements

- A complete binary tree has an interesting property that we can use to find the children and parents of any node.
- If the index of any element in the array is i, the element in the index 2i+1 will become the left child and element in 2i+2 index will become the right child. Also, the parent of any element at index i is given by the lower bound of (i-1)/2.

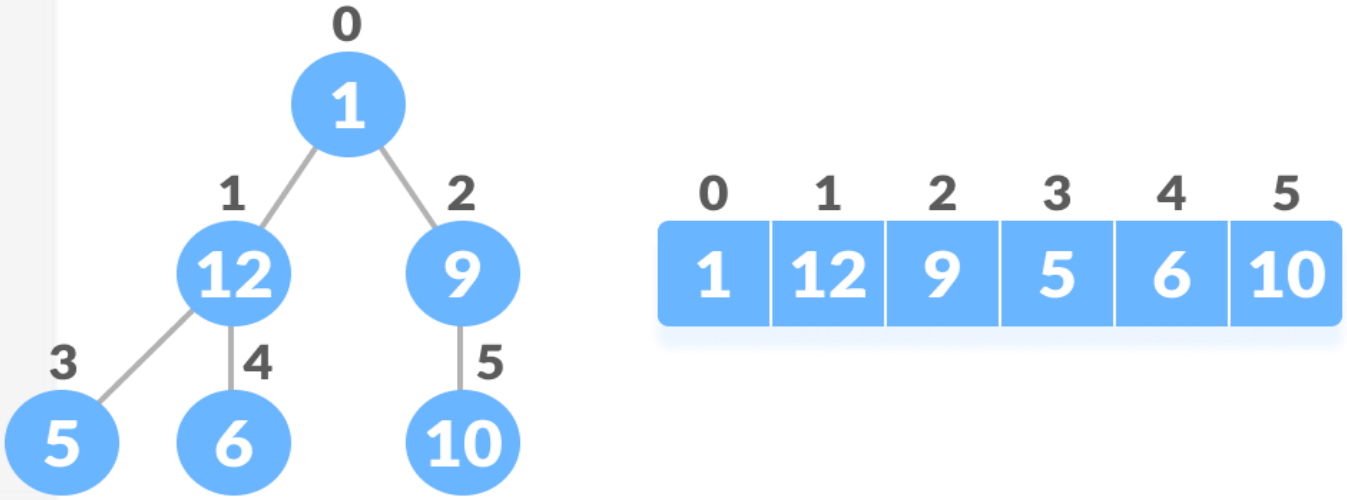


Fig: Relationship between array and heap indices

Let's test it out

Left child of 1 (index 0)

= element in (2*0+1) index

= element in 1 index

= 12

Right child of 1

= element in (2*0+2) index

= element in 2 index

= 9

Similarly,

Left child of 12 (index 1)

= element in (2*1+1) index

= element in 3 index

= 5

Right child of 12

= element in (2*1+2) index

= element in 4 index

= 6

For finding parent of any node

- Let us also confirm that the rules hold for finding parent of any node
- Parent of 9 (position 2)
- = (2-1)/2
- = ½
- = 0.5
- ~ 0 index
- = 1

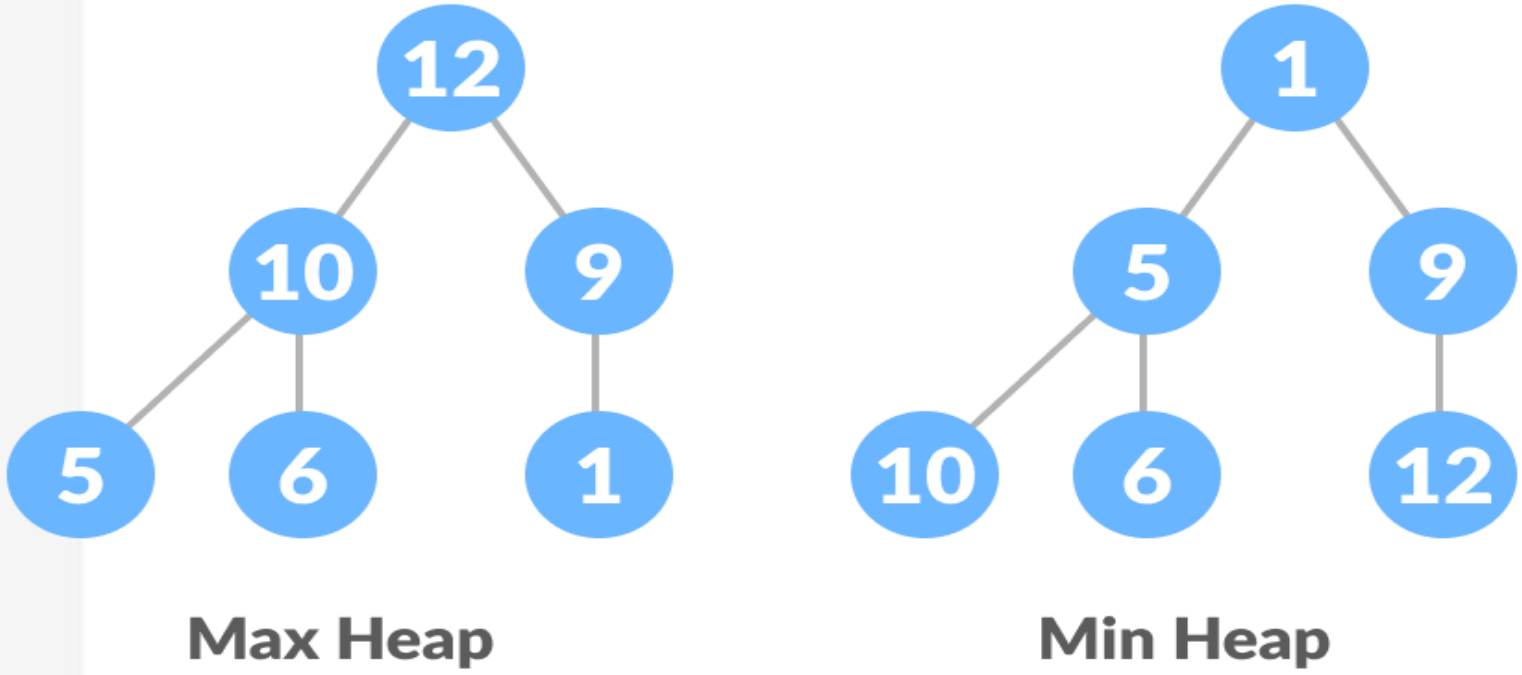
- Parent of 12 (position 1)
- = (1-1)/2
- = 0 index
- = 1

What is Heap Data Structure?

- Heap is a special tree-based data structure. A binary tree is said to follow a heap data structure if
- it is a complete binary tree
- All nodes in the tree follow the property that they are greater than their children i.e. the largest element is at the root and both its children and smaller than the root and so on. Such a heap is called a max-heap. If instead, all nodes are smaller than their children, it is called a min-heap

Min Heap and Max Heap

- The following example diagram shows Max-Heap and Min-Heap.

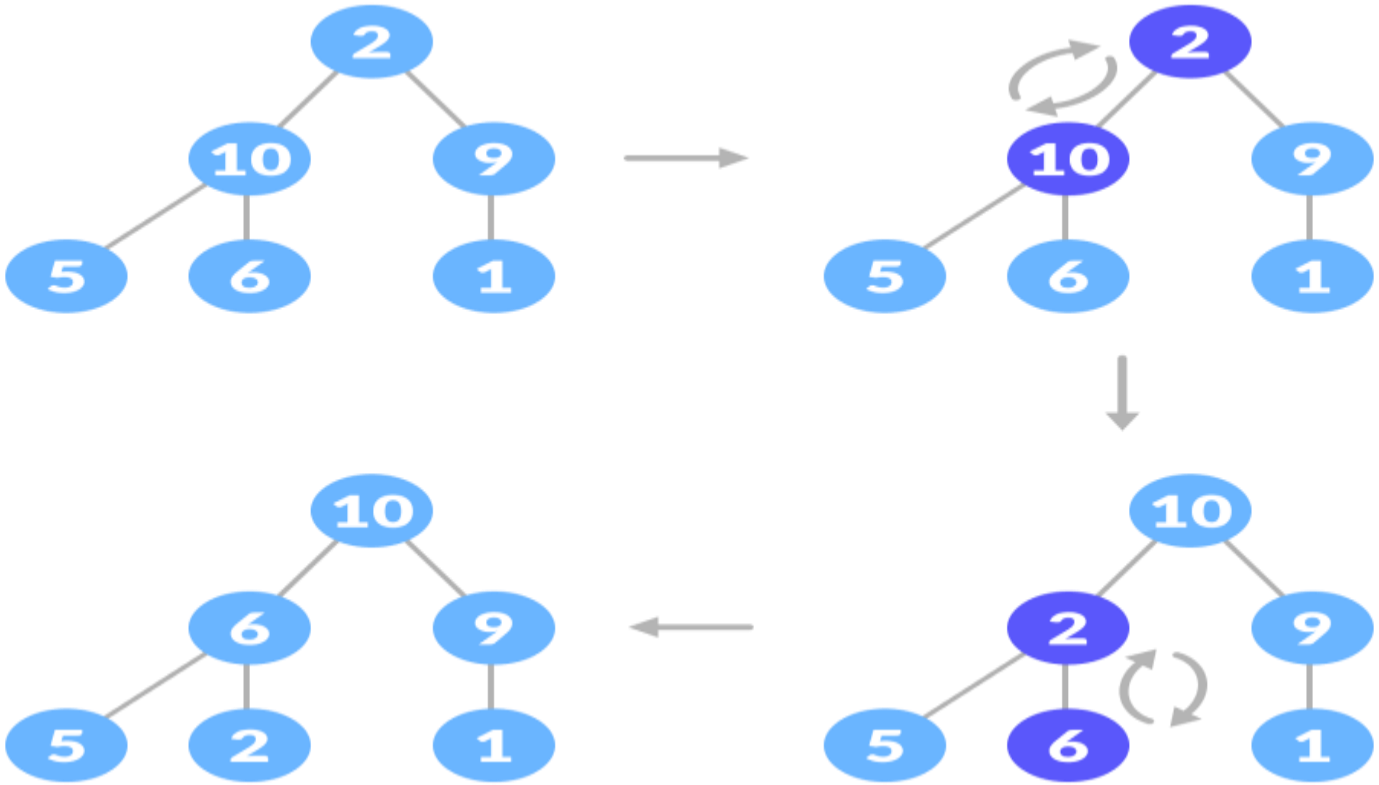


How to "heapify" a tree

- Starting from a complete binary tree, we can modify it to become a Max-Heap by running a function called heapify on all the non-leaf elements of the heap.
- Since heapify uses recursion, it can be difficult to grasp. So let's first think about how you would heapify a tree with just three elements.

How to heapify root element when its subtrees are max-heaps

- The top element isn't a max-heap but all the sub-trees are max-heaps.
- To maintain the max-heap property for the entire tree, we will have to keep pushing 2 downwards until it reaches its correct position.



Contd..

- Thus, to maintain the max-heap property in a tree where both sub-trees are max-heaps, we need to run heapify on the root element repeatedly until it is larger than its children or it becomes a leaf node.

Build max-heap

- To build a max-heap from any tree, we can thus start heapifying each sub-tree from the bottom up and end up with a max-heap after the function is applied to all the elements including the root element.

- In the case of a complete tree, the first index of a non-leaf node is given by n/2 - 1. All other nodes after that are leaf-nodes and thus don't need to be heapified.

Create array and calculate i

	0	1	2	3	4	5
arr	1	12	9	5	6	10

n = 6

i = 6/2 - 1 = 2 # loop runs from 2 to 0

Figure 1: Given array

Steps to build max heap for heap sort

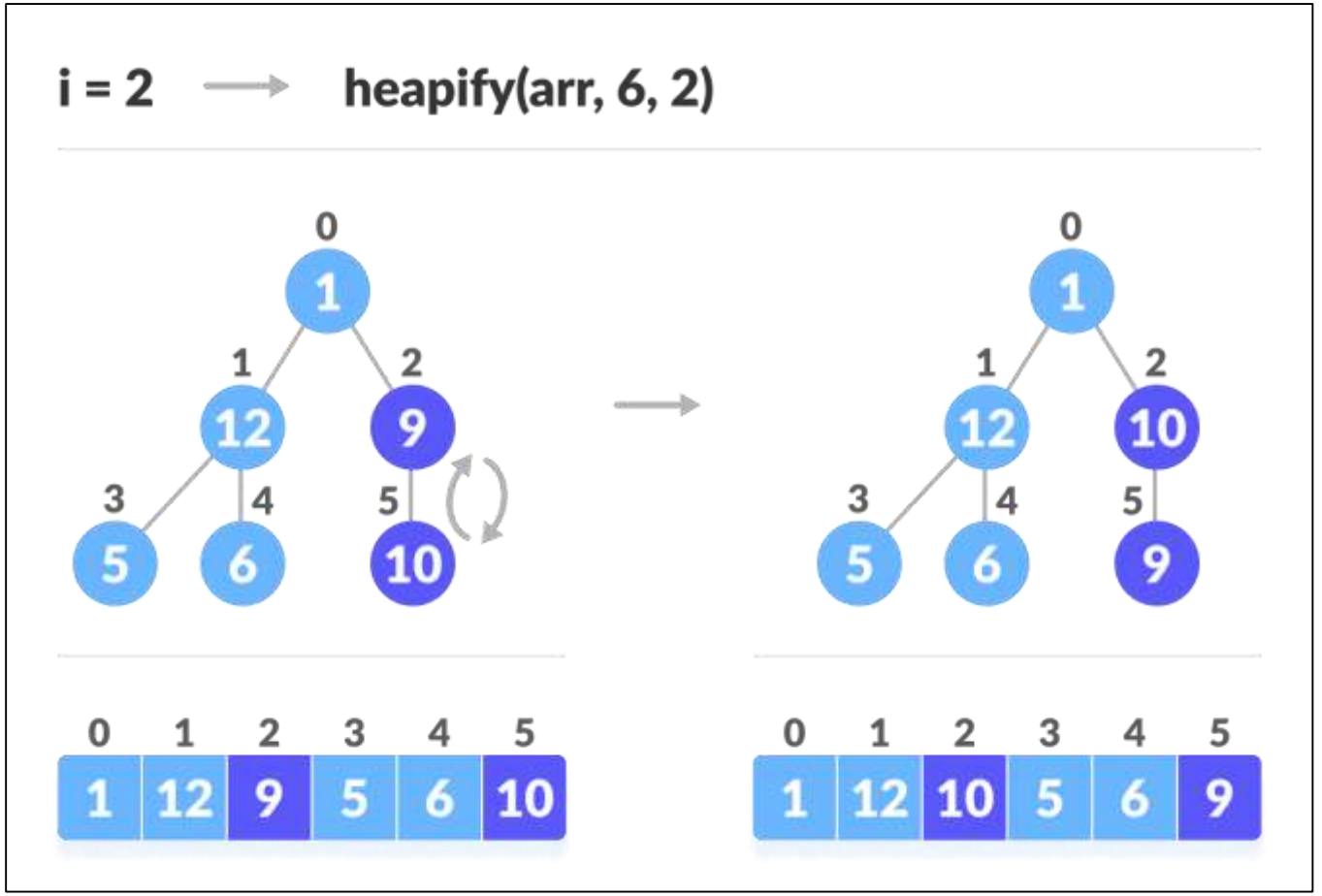


Figure 2: Heapify

Steps to build max heap for heap sort

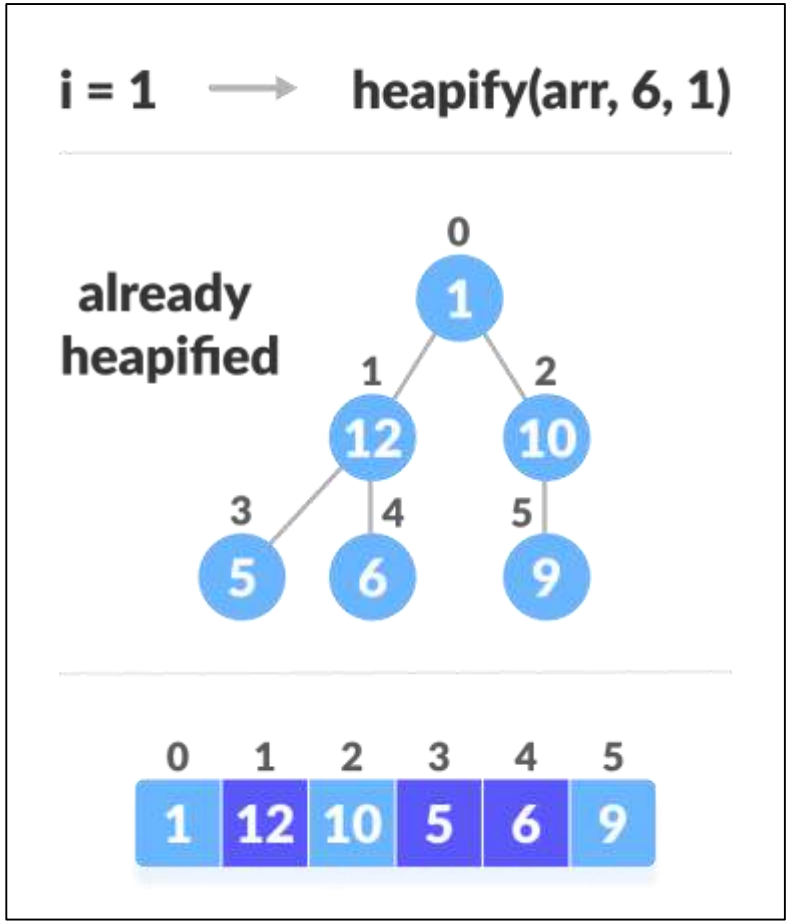
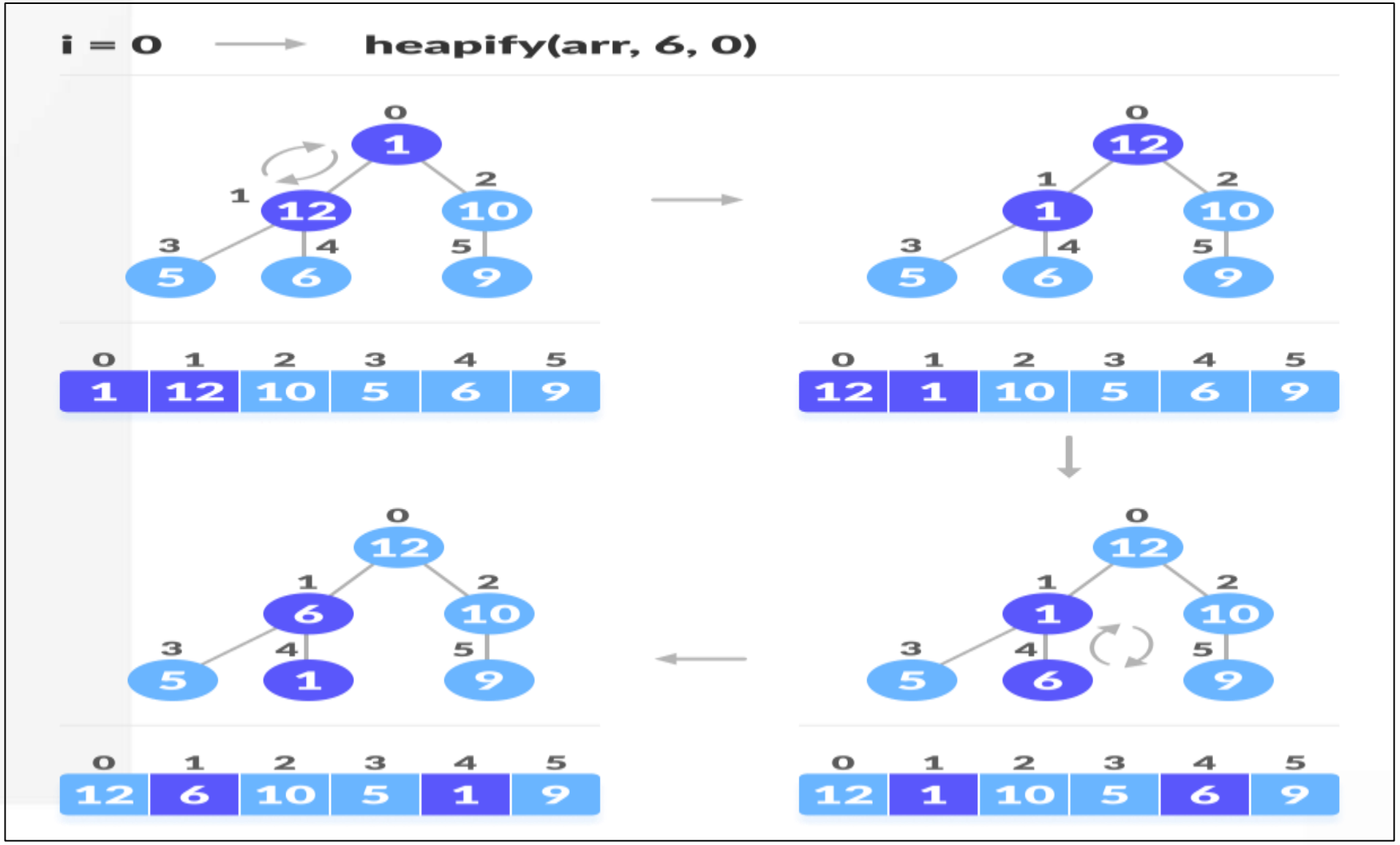


Figure 3: Heapify

Steps to build max heap for heap sort



Contd..

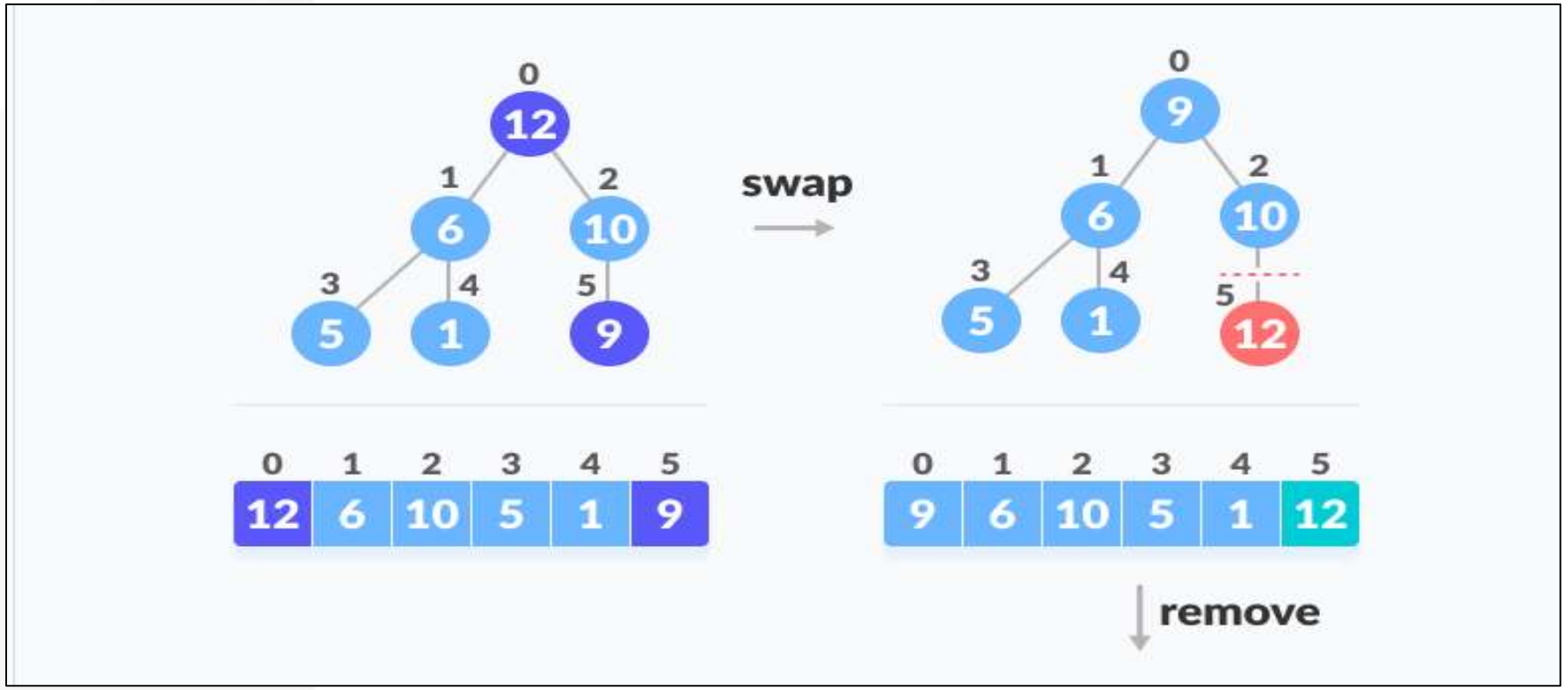


Figure 4: Swapping

Contd..

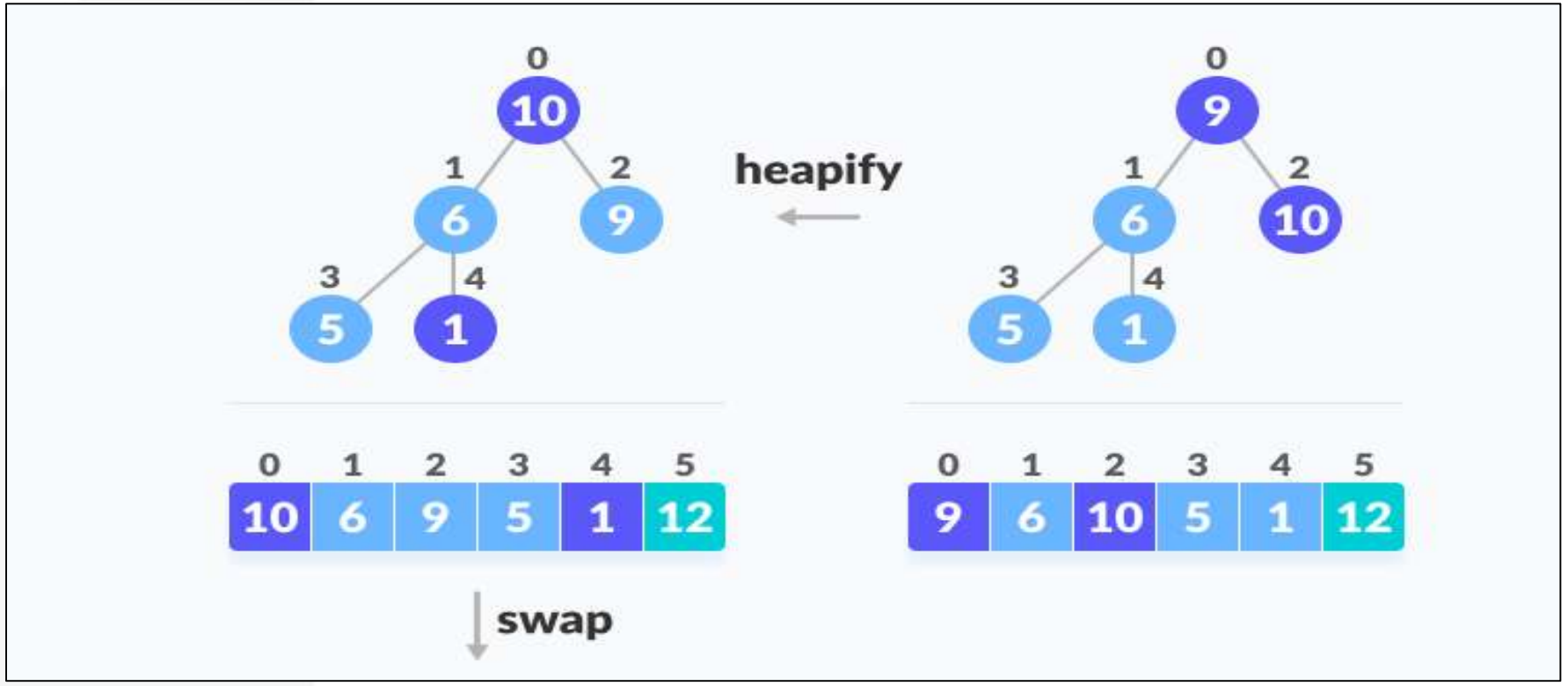


Figure 5: Heapify

Contd..

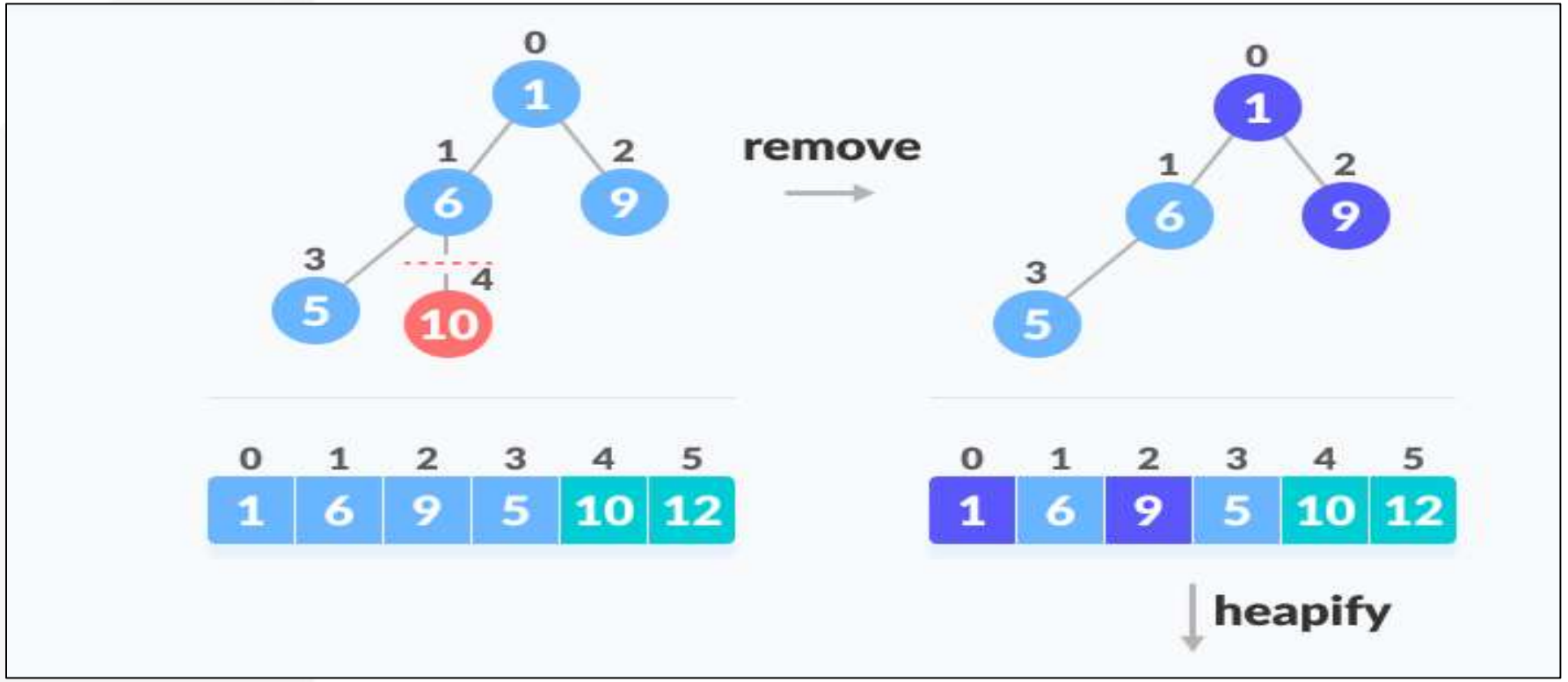


Figure 6: Removing element

Contd..

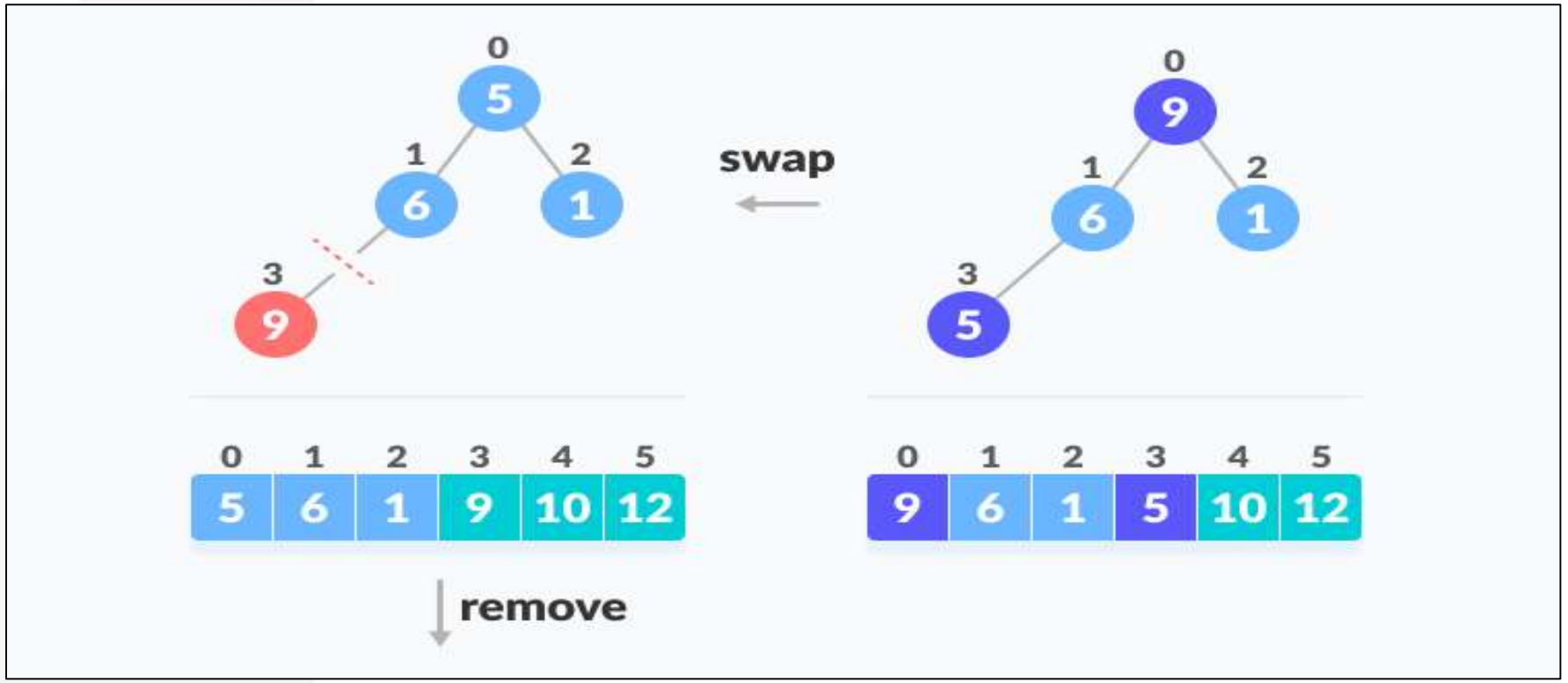


Figure 7: Swapping

Contd..

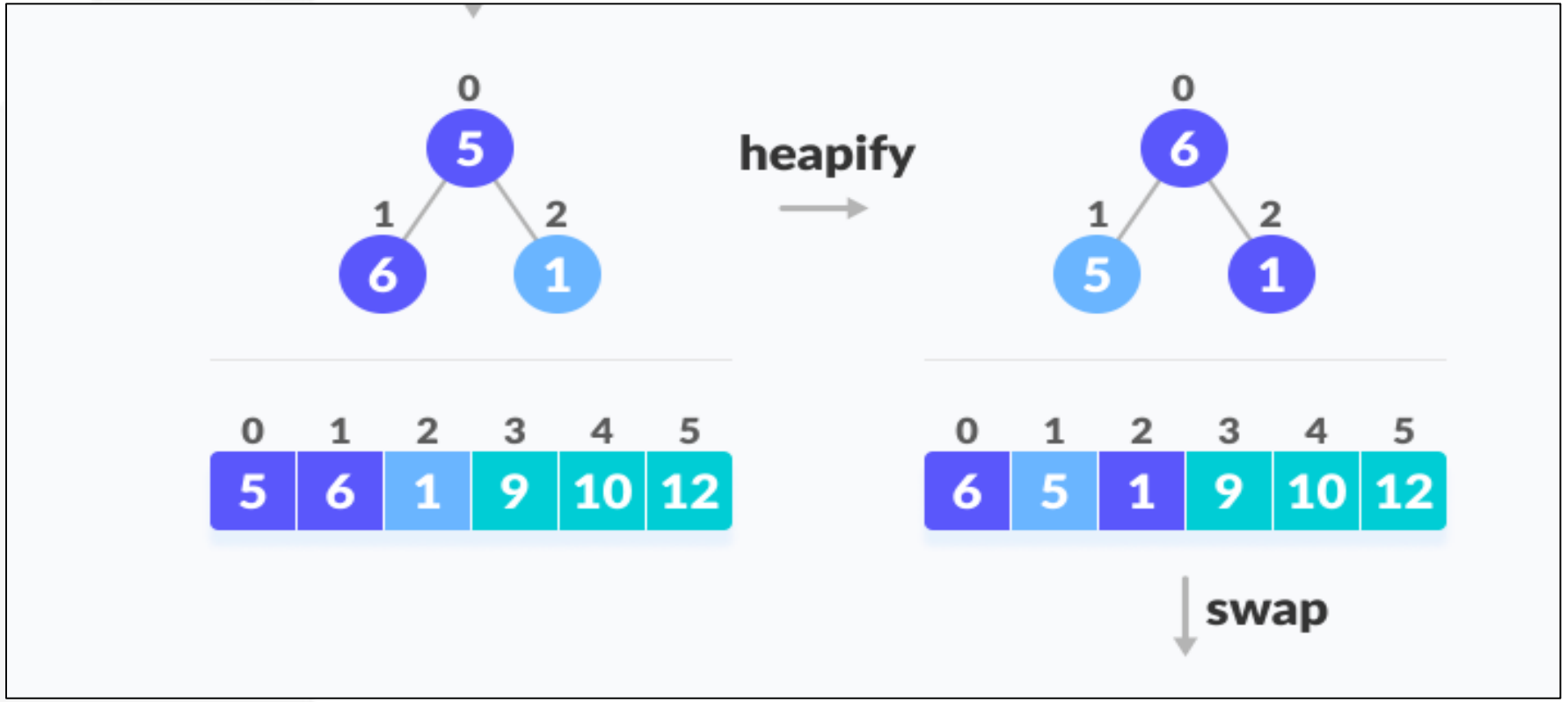


Figure 8: Heapify

Contd..

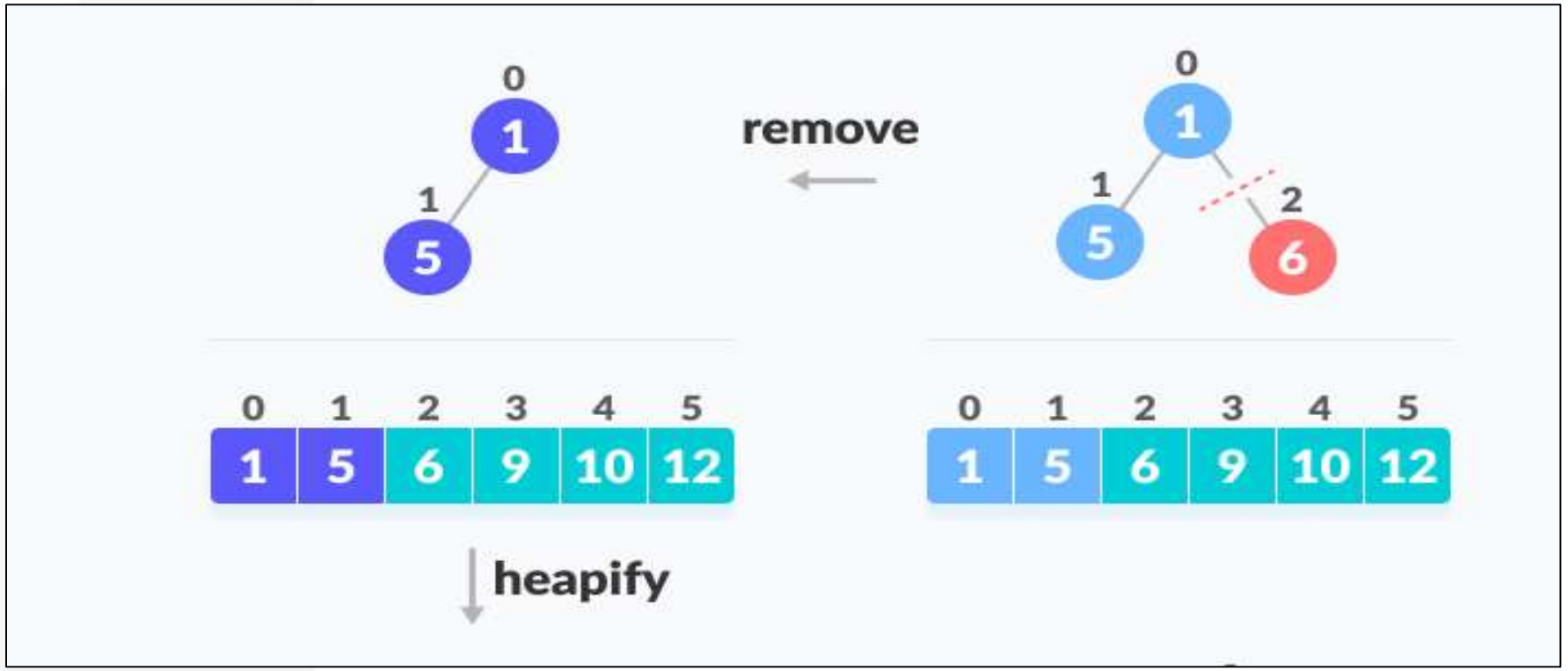
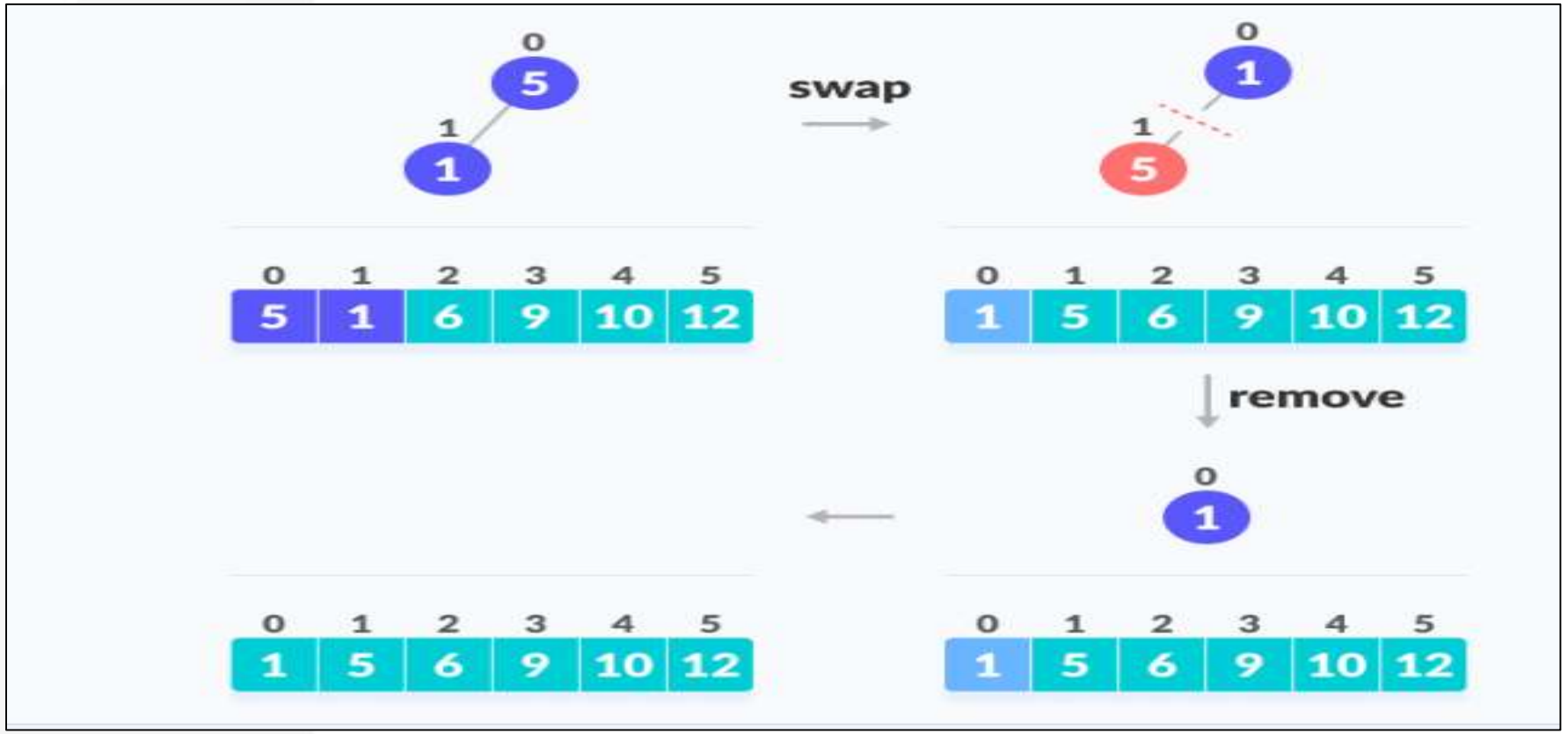


Figure 9: Removing element

Swap, Remove and Heapify



References

1) http://epgp.inflibnet.ac.in/epgpdata/uploads/epgp_content/S000007CS/P001064/M014960/ET/1459249858E17.pdf
2) https://www.studytonight.com/data-structures/heap-sort
3) https://www.interviewbit.com/tutorial/heap-sort-algorithm/

Books:

1. Introduction to Algorithms by Coreman, Leiserson, Rivest, Stein.
2. Fundamentals of Algorithms by Ellis Horwitz, Sartaj Sahni, Sanguthevar Rajasekaran

THANK YOU