DATA STRUCTURES

UNIT-4

Priority Queues, Heaps

What is a priority queue?

- A priority queue is an abstract data type that behaves similarly to the normal queue except that each element has some priority, i.e., the element with the highest priority would come first in a priority queue.

- The priority of the elements in a priority queue will determine the order in which elements are removed from the priority queue.

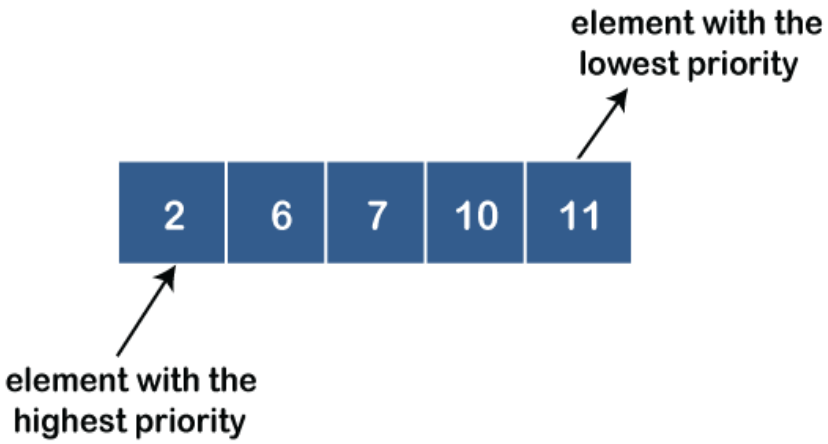
- For example, suppose we have some values like 1, 3, 4, 8, 14, 22 inserted in a priority queue with an ordering imposed on the values is from least to the greatest. Therefore, the 1 number would be having the highest priority while 22 will be having the lowest priority.

Characteristics of a Priority queue

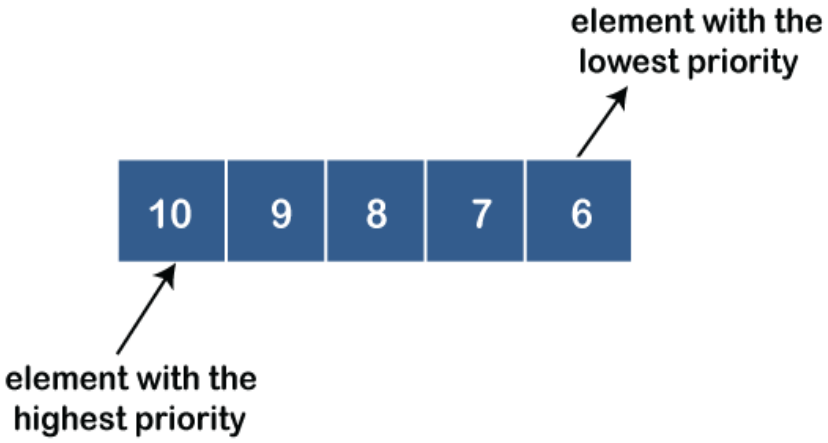
- A priority queue is an extension of a queue that contains the following characteristics:
  - Every element in a priority queue has some priority associated with it.
  - An element with the higher priority will be deleted before the deletion of the lesser priority.
  - If two elements in a priority queue have the same priority, they will be arranged using the FIFO principle.

Types of Priority Queue

- **Ascending order priority queue:** In ascending order priority queue, a lower priority number is given as a higher priority in a priority. For example, we take the numbers from 1 to 5 arranged in an ascending order like 1,2,3,4,5; therefore, the smallest number, i.e., 1 is given as the highest priority in a priority queue.



- **Descending order priority queue:** In descending order priority queue, a higher priority number is given as a higher priority in a priority. For example, we take the numbers from 1 to 5 arranged in descending order like 5, 4, 3, 2, 1; therefore, the largest number, i.e., 5 is given as the highest priority in a priority queue.



Implementation of Priority Queue

- The priority queue can be implemented in three ways:
  - Arrays
  - Linked List
  - Heap Data Structure

Priority Queues-Arrays

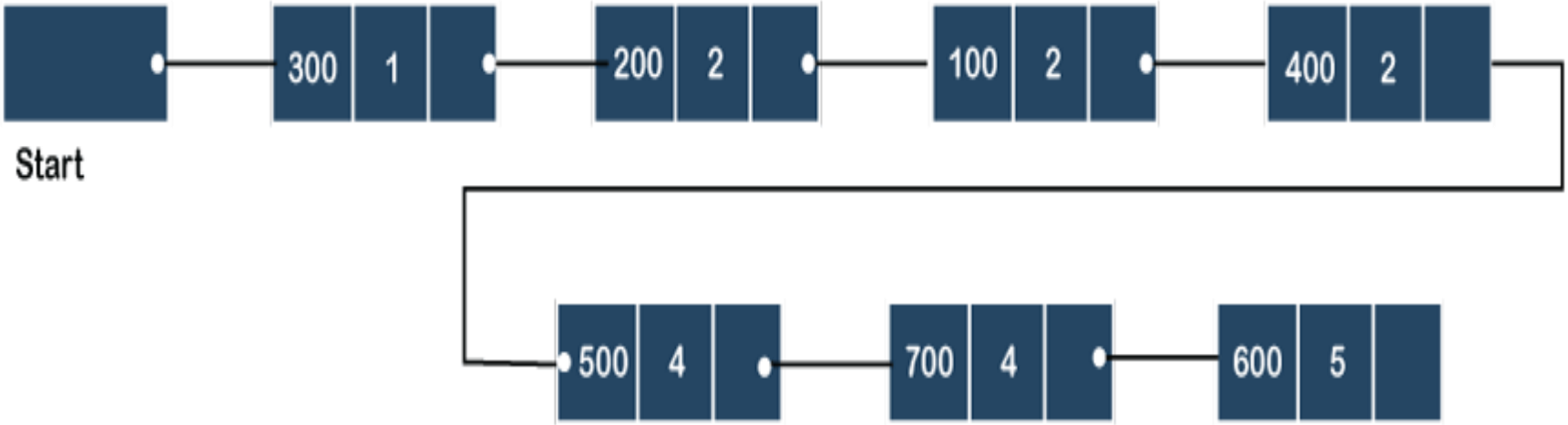
- Now, we will see how to represent the priority queue through a one-way list.
- We will create the priority queue by using the list given below in which INFO list contains the data elements, PRN list contains the priority numbers of each data element available in the INFO list.

	INFO	PNR
0	200	2
1	400	4
2	500	4
3	300	1
4	100	2
5	600	3
6	700	4

Priority Queues-Linked List

For the same example which is represented with arrays:

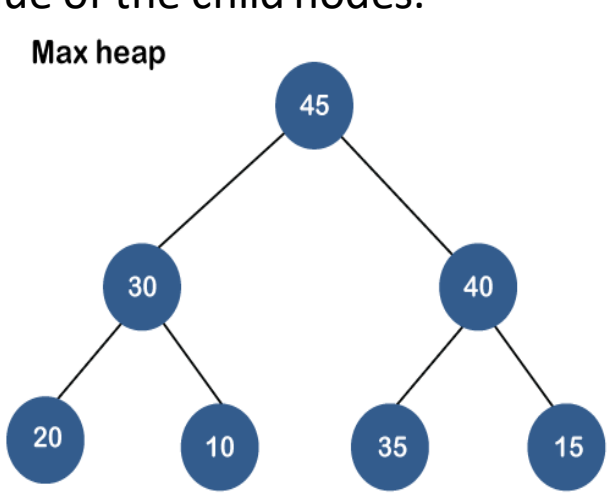
	INFO	PNR
0	200	2
1	400	4
2	500	4
3	300	1
4	100	2
5	600	3
6	700	4

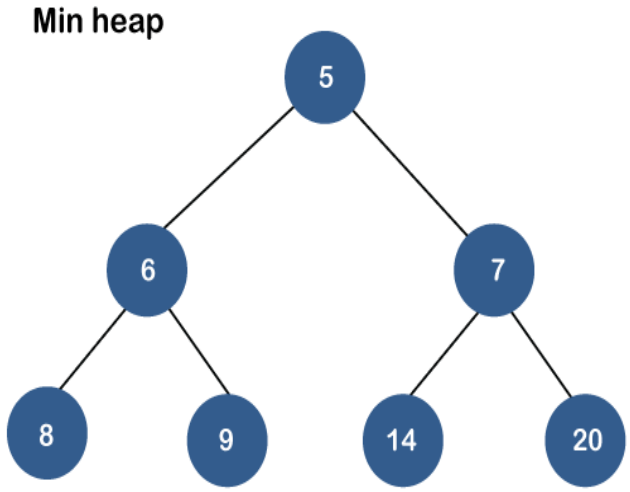


Priority Queues-Binary Heaps

- A heap is a tree-based data structure that forms a complete binary tree, and satisfies the heap property. If A is a parent node of B, then A is ordered with respect to the node B for all nodes A and B in a heap.
- It means that the value of the parent node could be more than or equal to the value of the child node, or the value of the parent node could be less than or equal to the value of the child node. Therefore, we can say that there are two types of heaps:
  - **Max heap:** The max heap is a heap in which the value of the parent node is greater than the value of the child nodes.

  - **Min heap:** The min heap is a heap in which the value of the parent node is less than the value of the child nodes.



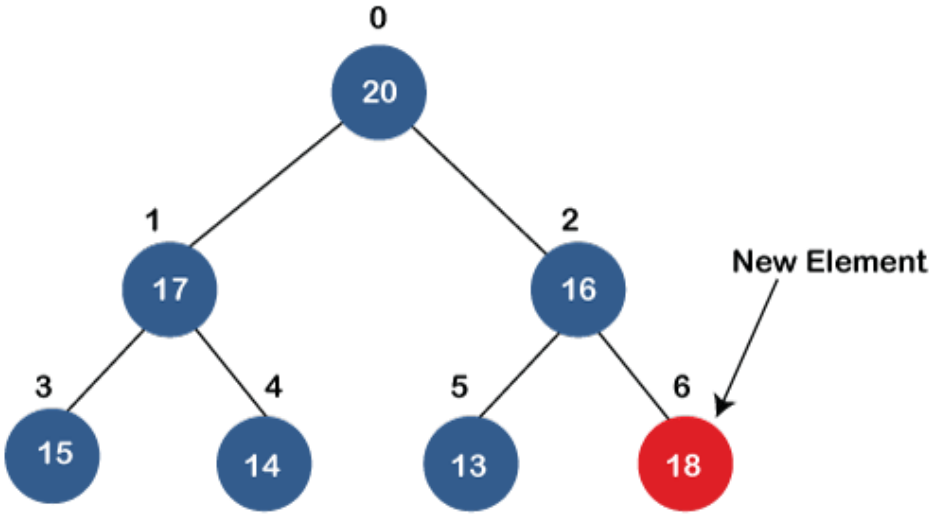


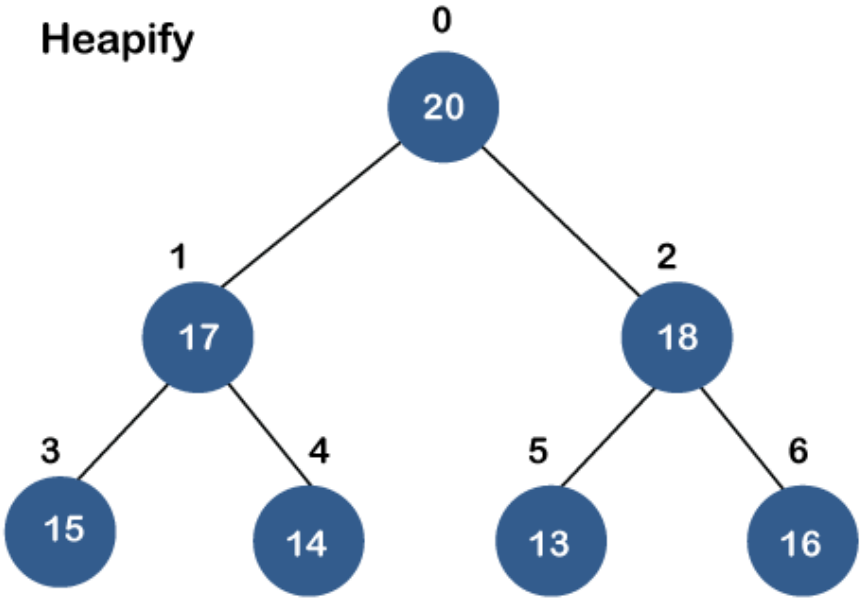
Operations on Max Heaps

- The common operations that we can perform on a priority queue are
  - Insertion and
  - Deletion.

Inserting the element in a max heap

- If we insert an element in a priority queue, it will move to the empty slot by looking from top to bottom and left to right.
- After the insertion, then it is compared with the parent node; if it is found out of order, elements are swapped. This process continues until the element is placed in a correct position.



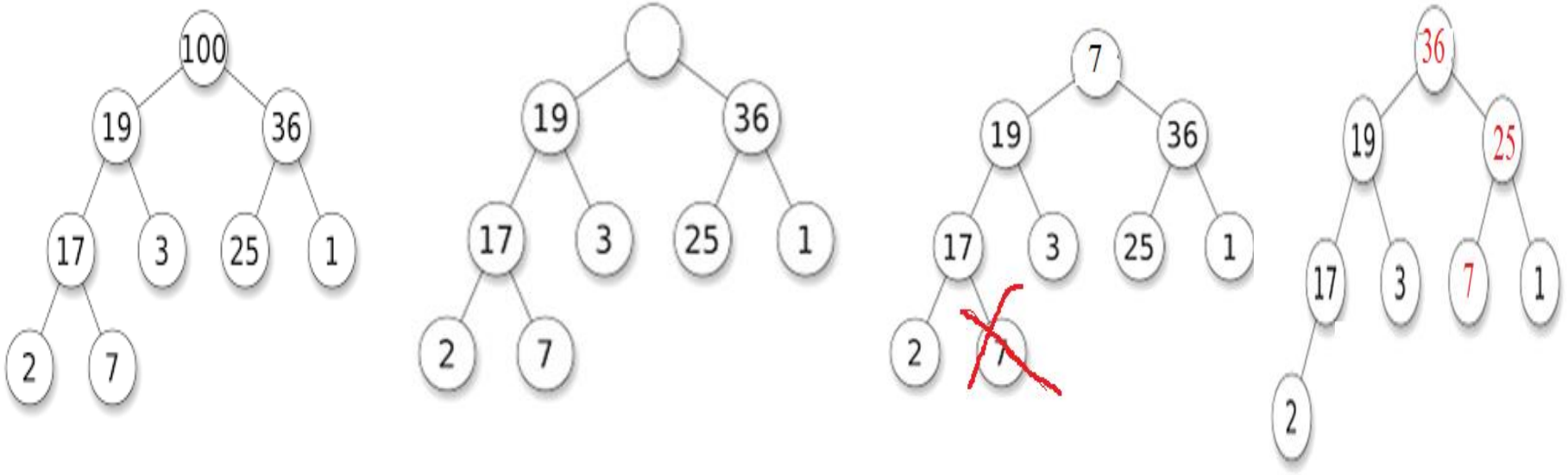


MAX Heap Construction/Insertion

- **Step 1** – Create a new node at the end of heap.
- **Step 2** – Assign new value to the node.
- **Step 3** – Compare the value of this child node with its parent.
- **Step 4** – If value of parent is less than child, then swap them.
- **Step 5** – Repeat step 3 & 4 until Heap property holds.

Delete Element from Heap

- As we know that in a max heap, the maximum element is the root node. When we remove the root node, it creates an empty slot.
- The last inserted element will be added in this empty slot.
- Then, this element is compared with the child nodes, i.e., left-child and right child, and swap with the smaller of the two.
- It keeps moving down the tree until the heap property is restored.



MAX Heap Deletion

- Deletion in Max (or Min) Heap always happens at the root to remove the Maximum (or minimum) value.
- **Step 1** – Remove root node.
- **Step 2** – Move the last element of last level to root.
- **Step 3** – Compare the value of this child node with its parent.
- **Step 4** – If value of parent is less than child, then swap them.
- **Step 5** – Repeat step 3 & 4 until Heap property holds.

Example 2 on MAX Heap

- Create a MAX HEAP for Input
  - 35 33 42 10 14 19 27 44 26 31
  - Apply 4 delete operations on the constructed heap.

Heap Sort

Heap Sort Algorithm

- For sorting in increasing order:

- **Step 1:** Build a max heap from the input data.
- **Step 2:** At this point, the largest item is stored at the root of the heap. Replace it with the last item of the heap followed by reducing the size of heap by 1.
- **Step 3:** Finally, heapify the root of the tree.
- **Step 4:** Repeat step 2 & 3 while size of heap is greater than 1.

Heap Sort Example

- Sort the following elements in ascending order:

  1, 12, 9, 5, 6, 10

- Step 1: Build Max Heap

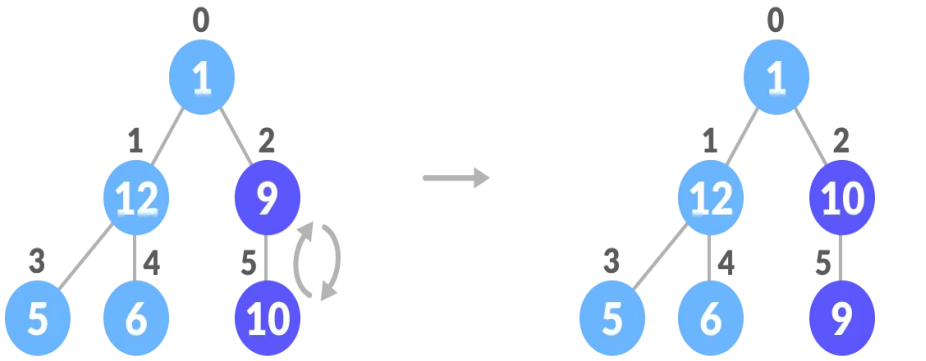
Heap Sort Example

n = 6

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

Heap Sort Example

i = 2 → heapify(arr, 6, 2)

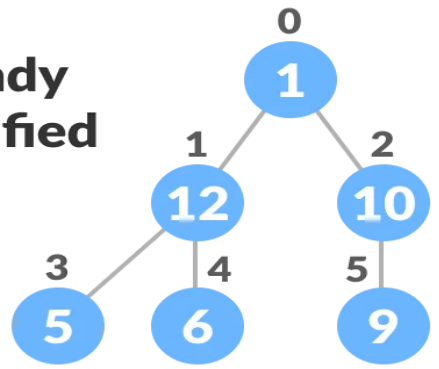


0	1	2	3	4	5
1	12	9	5	6	10

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

i = 1 → heapify(arr, 6, 1)

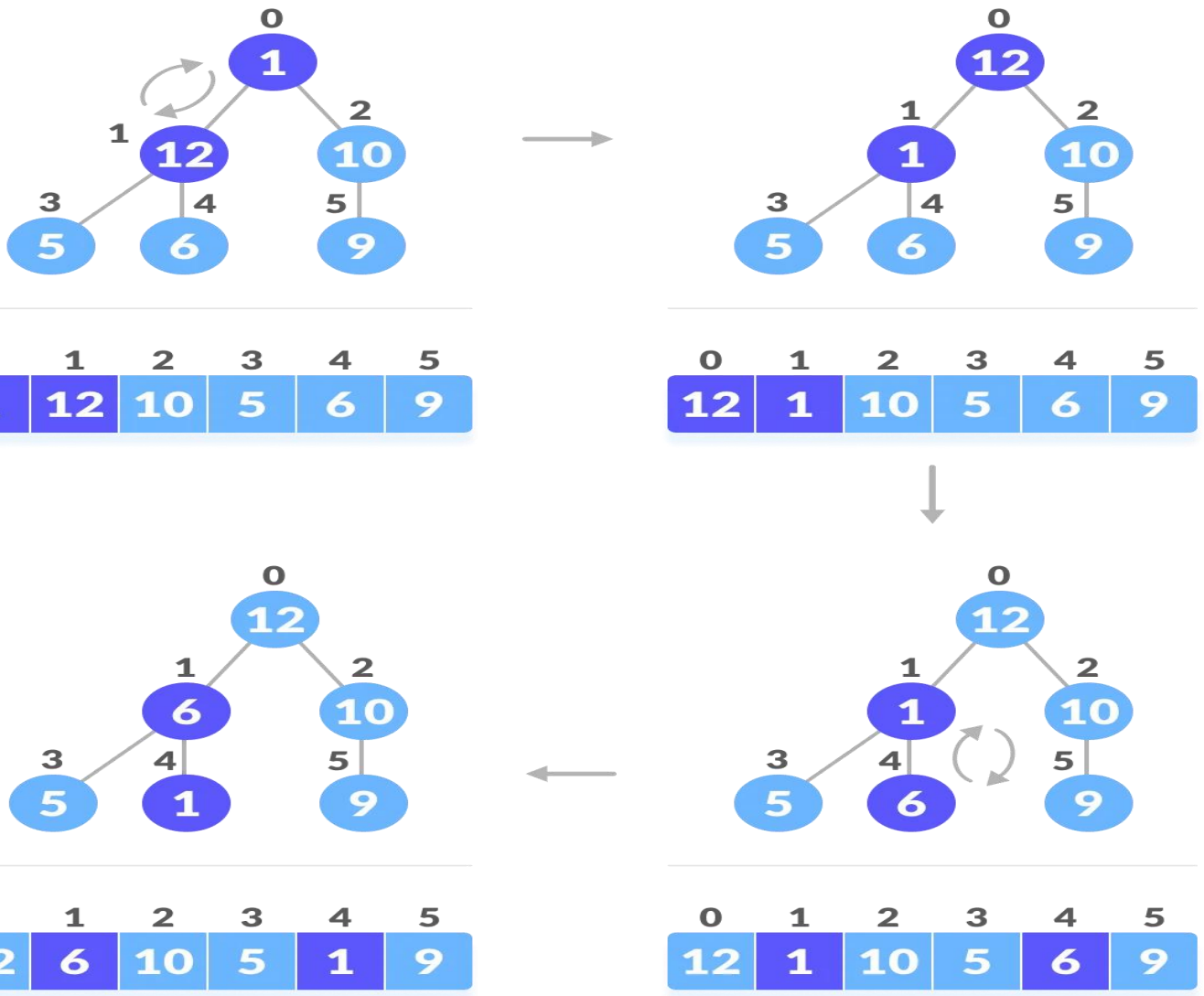
already heapified



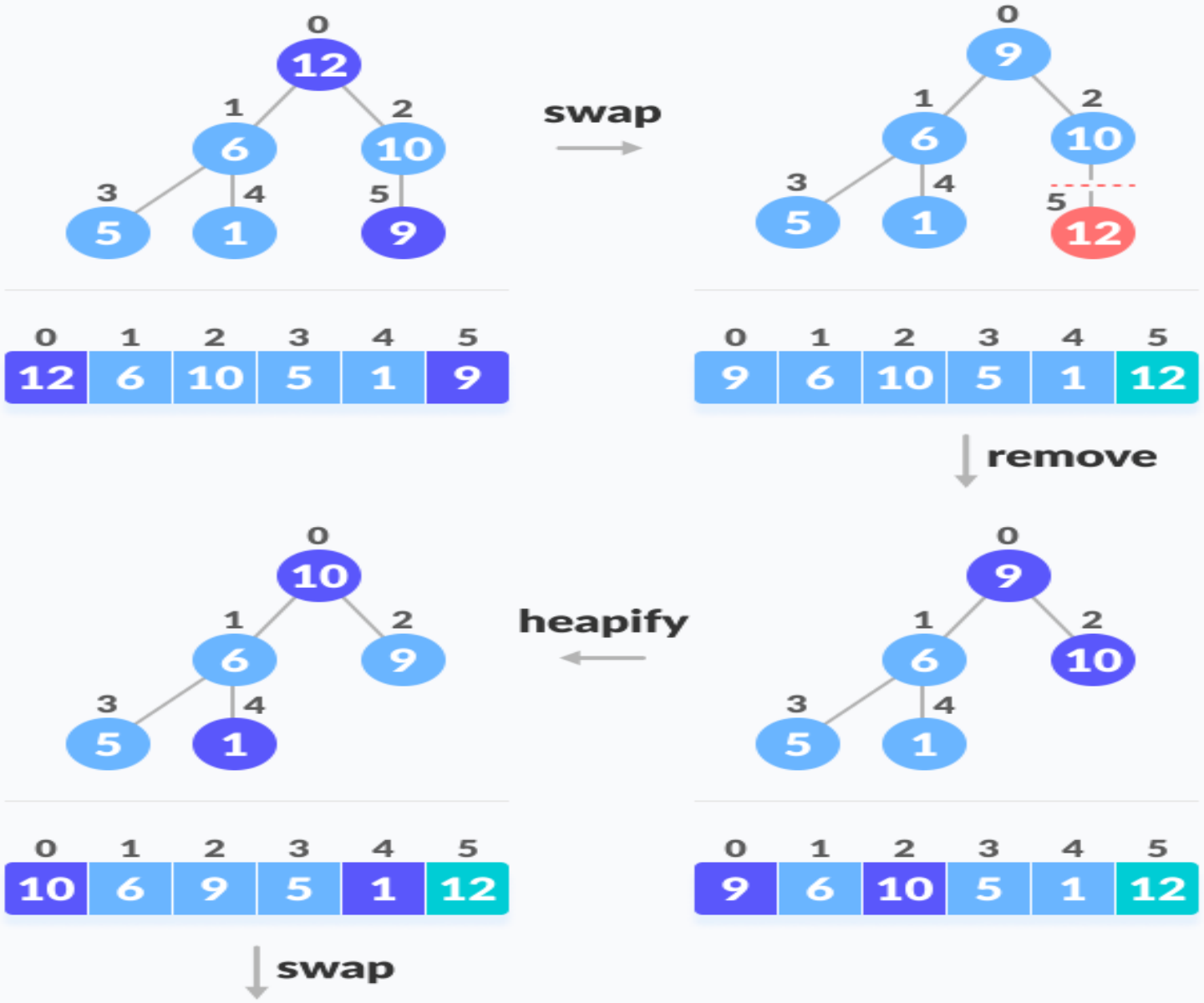
0	1	2	3	4	5
1	12	10	5	6	9

Heap Sort Example

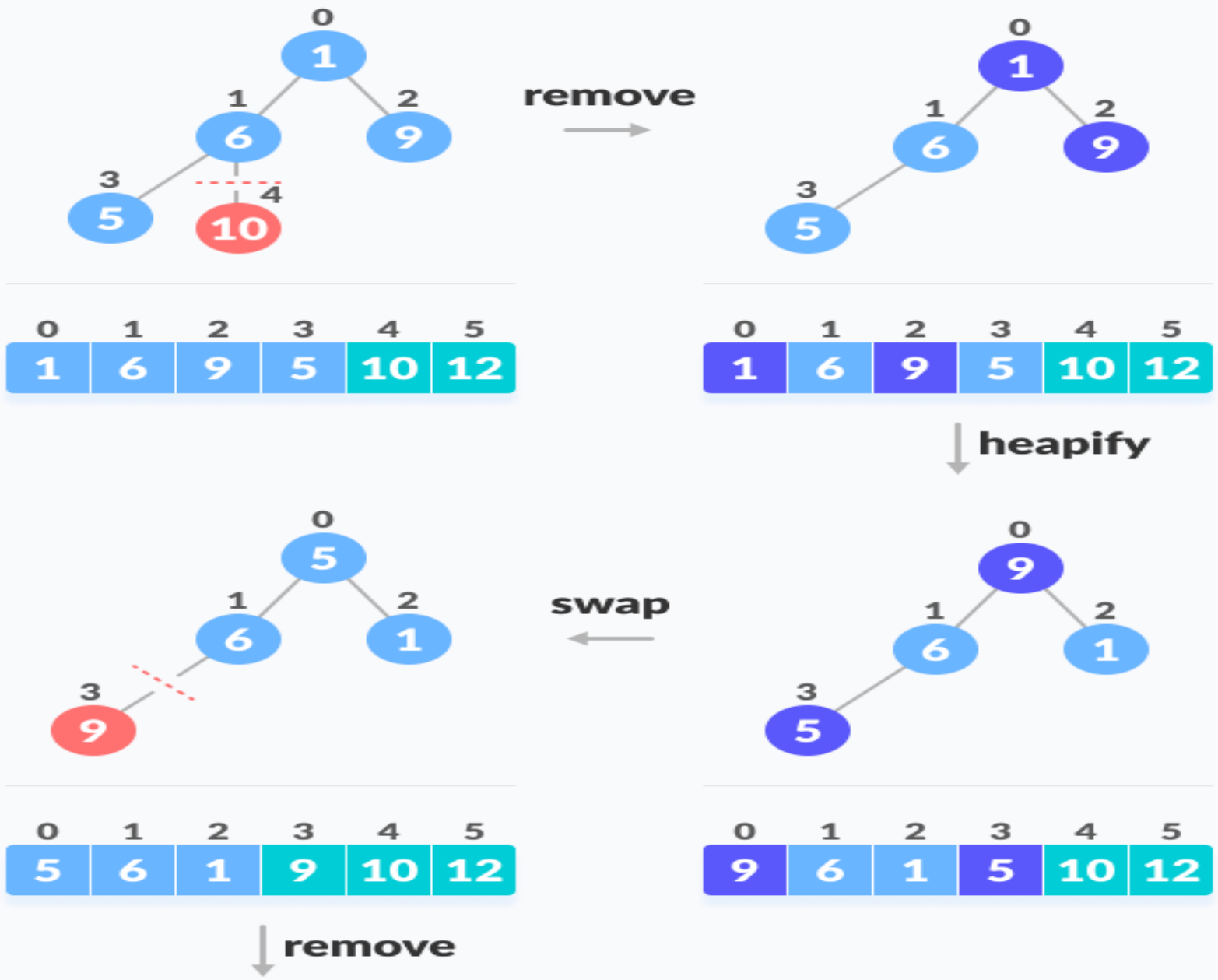
i = 0 → **heapify(arr, 6, 0)**



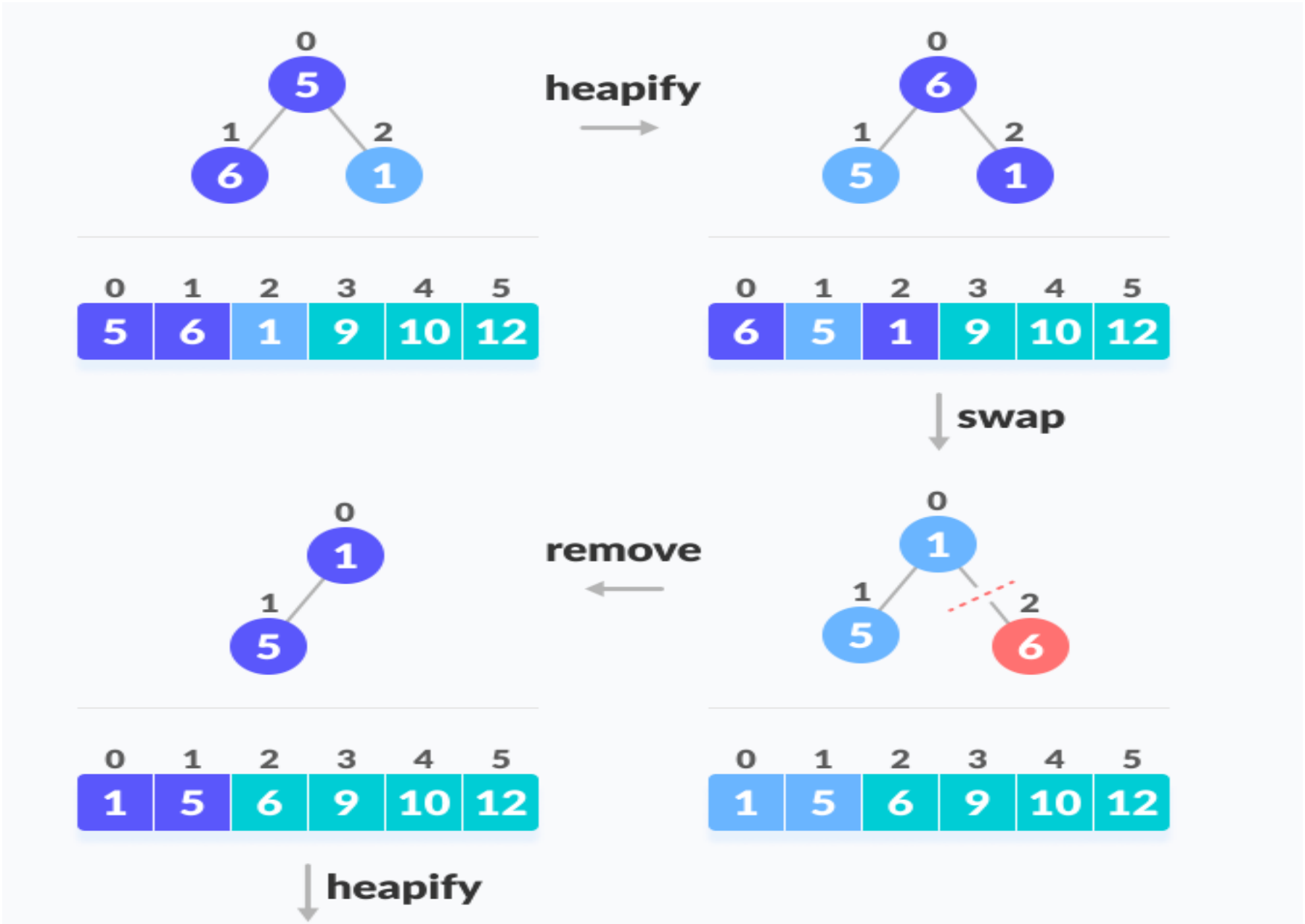
Heap Sort Example



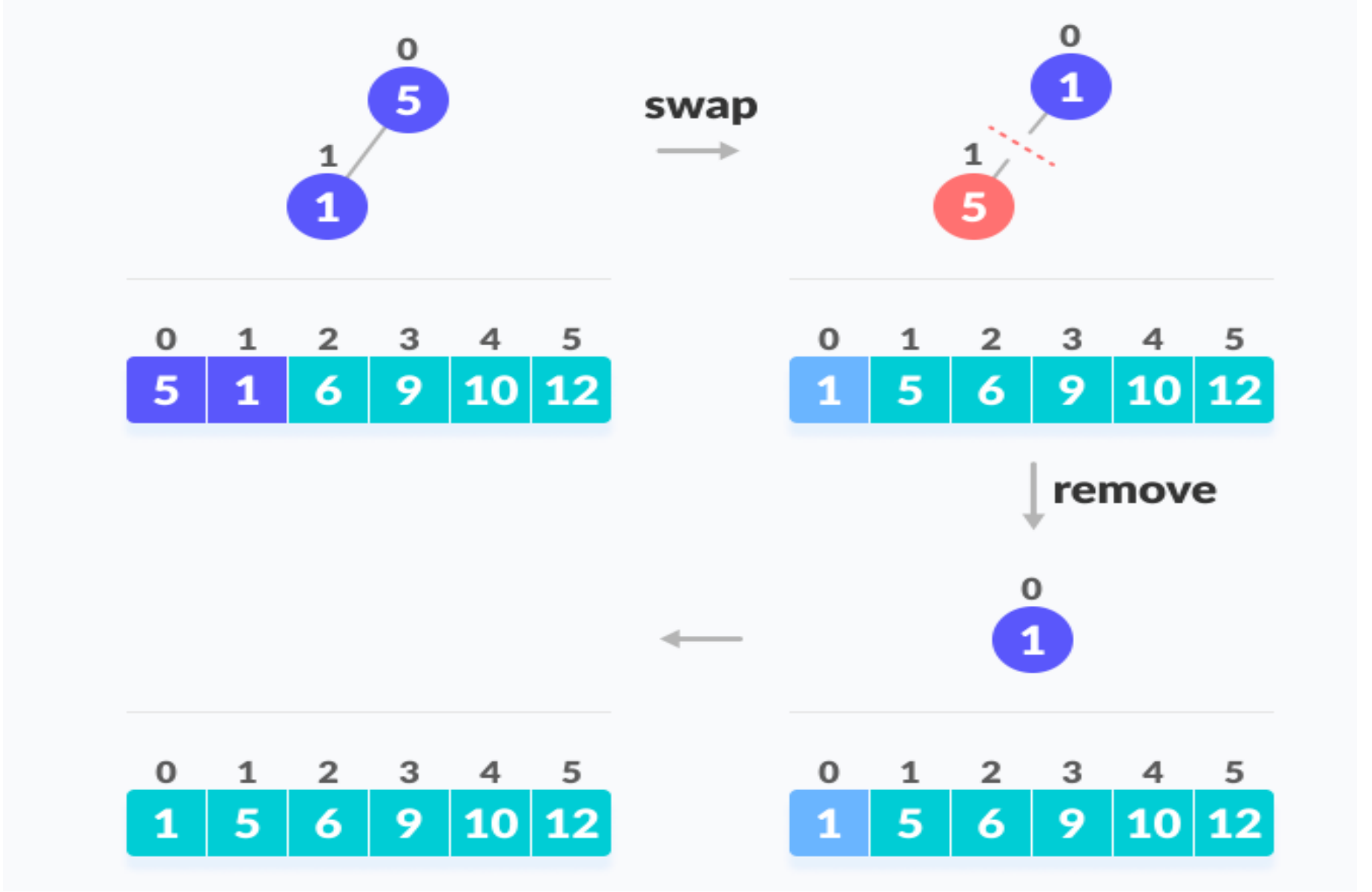
Heap Sort Example



Heap Sort Example



Heap Sort Example



Algorithm for Heap Sort

```
Algorithm heapSort(arr[], n)
{
                                        // Build heap (rearrange array)
   for i = n / 2 – 1 to 0 step -1
         heapify(arr, n, i);

                                      // One by one extract an element from heap
   for i = n – 1 to 1 step -1
   {
         swap(arr[0], arr[i]);          // Move current root to end
         heapify(arr, i, 0);          // call max heapify on the reduced heap
   }
}
```

```
Algorithm heapify(arr[],x, i)
{
   largest = i;                                  // Initialize largest as root
   left = 2 * i + 1;                             // left = 2*i + 1
   right = 2 * i + 2;                            // right = 2*i + 2

  if (left < x && arr[left] > arr[largest])      // If left child is larger than root
        largest = left;

  if (right < x && arr[right] > arr[largest])    // If right child is larger than largest so far
       largest = right;

  if (largest != i)                              // If largest is not root
     {
        swap (arr[i], arr[largest]);
        heapify (arr, n, largest);               // Recursively heapify the affected sub-tree
     }
}
```

Time complexity of Heap Sort

Heap Sort Complexity

Time Complexity	
Best	O(nlog n)
Worst	O(nlog n)
Average	O(nlog n)
Space Complexity	O(1)

Exercise on Heap Sort

- Sort the following elements using Heap sort:

  1, 6, 3, 2, 5, 8, 9, 12, 4, 10

8. Write the heap sort algorithm to sort a set of integers. **15M**

8. a. Illustrate deletion of any two elements in a max heap with atleast 10 elements. **9M**

b. Compare the performance of the following sorting methods:
i) Quick ii) Merge iii) Heap **6M**

b. Discuss about max heap and min heap, with an example. **9M**

b. How do we delete an element from a Max Heap? **7M**

a. Write a procedure for Heap sort with an example. **8M**

b. How do you insert an element into maxheap? **8M**

b. Write a C program to insertion an element into Max Heap. **5M**

b. What is Priority Queue? Explain with a suitable example implementation of maximum priority queue using max heap. **8M**

b. Define max heap. Show the step-by-step construction of a max heap resulting from the insertion of the following sequence of keys: 6, 5, 3, 1, 8, 7, 2 and 4. **7M**