Data Structures Lab

Session 9

Course: Data Structures (CS2001) **Semester:** Spring 2024

Instructor: Bushra Sattar **SLA: M Anus**

Note:

- Maintain discipline during the lab.
- Listen and follow the instructions as they are given.
- Just raise hand if you have any problem.
- Completing all tasks of each lab is compulsory.
- Get your lab checked at the end of the session.

Binary Tree and Binary Heap Data Structure Heap Sort

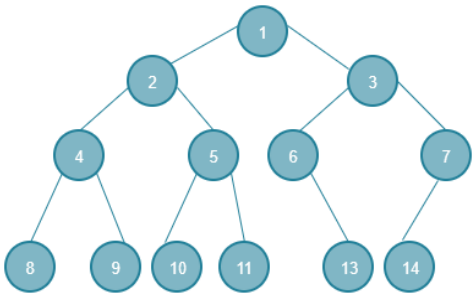
What is a Tree?

A tree is non-linear and has a hierarchical data structure consisting of a collection of nodes such that each node of the tree stores a value and a list of references to other nodes (the "children").

Binary Tree

A binary tree is a tree data structure in which each parent node can have at most two children. Each node of a binary tree consists of three items:

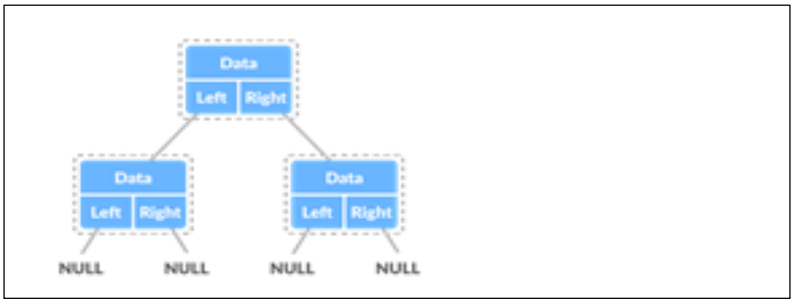
data item
address of left child
address of right child



Binary Tree Representation

A node of a binary tree is represented by a structure containing a data part and two pointers to other structures of the same type.

```
struct node
{
 int data;
 struct node *left;
 struct node *right;
};
```



Complete Binary Tree

A complete binary tree is a special type of binary tree where all the levels of the tree are filled completely except the lowest level nodes which are filled from as left as possible.

Properties of Complete Binary Tree:

- A complete binary tree is said to be a proper binary tree where all leaves have the same depth.
- In a complete binary tree number of nodes at depth d is 2d.
- In a complete binary tree with n nodes height of the tree is log(n+1).
- All the levels except the last level are completely full.

Heap

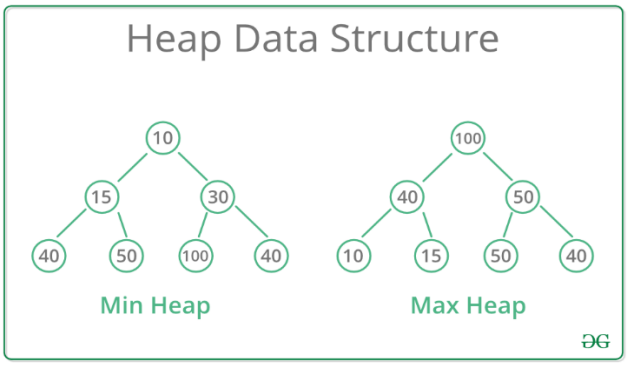
A Heap is a special Tree-based data structure in which the tree is a complete binary tree.

Binary Heap

A Binary Heap is a complete binary tree i.e. All levels of the tree are completely filled except the last level. Binary heaps are either min heap or max heap.

In the min-heap, the value at the root node must be smaller than its child or we can say minimum value among the tree.

In the max heap, the value at the root node must be the greatest node among the whole tree. A binary heap follows a heap ordering property.



Working of Heap Sort

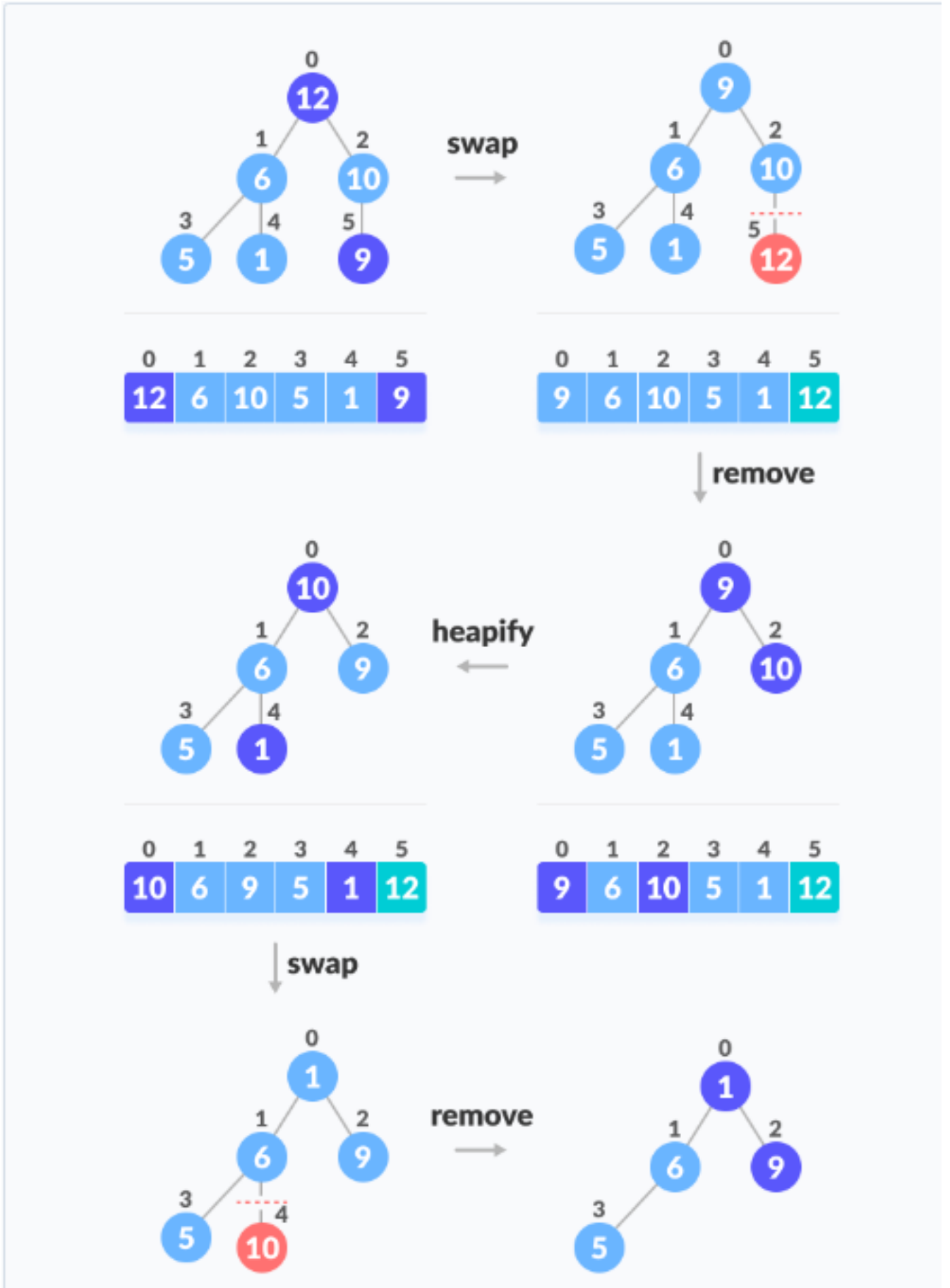
Since the tree satisfies Max-Heap property, then the largest item is stored at the root node.

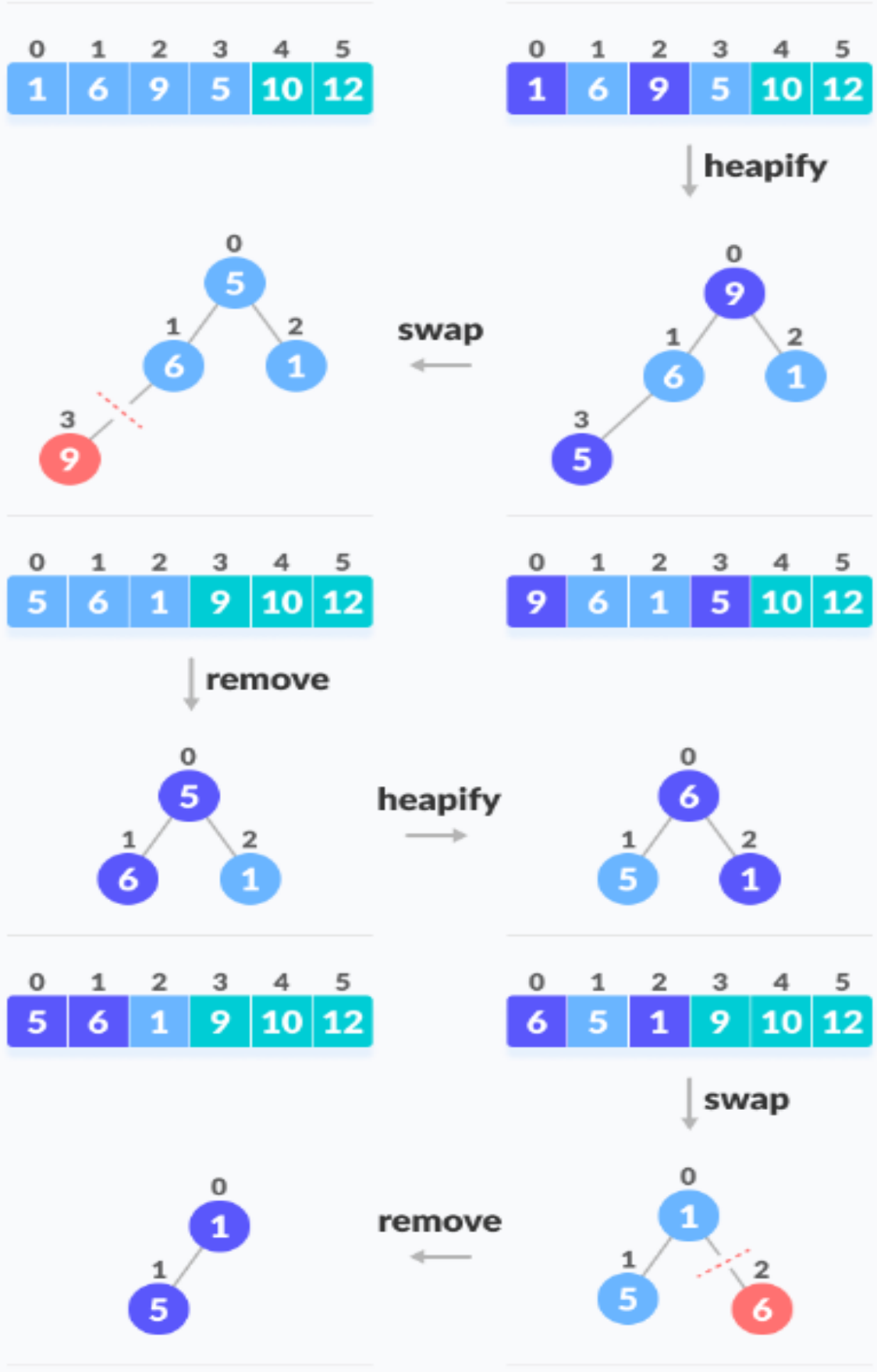
Swap: Remove the root element and put at the end of the array (nth position) Put the last item of the tree (heap) at the vacant place.

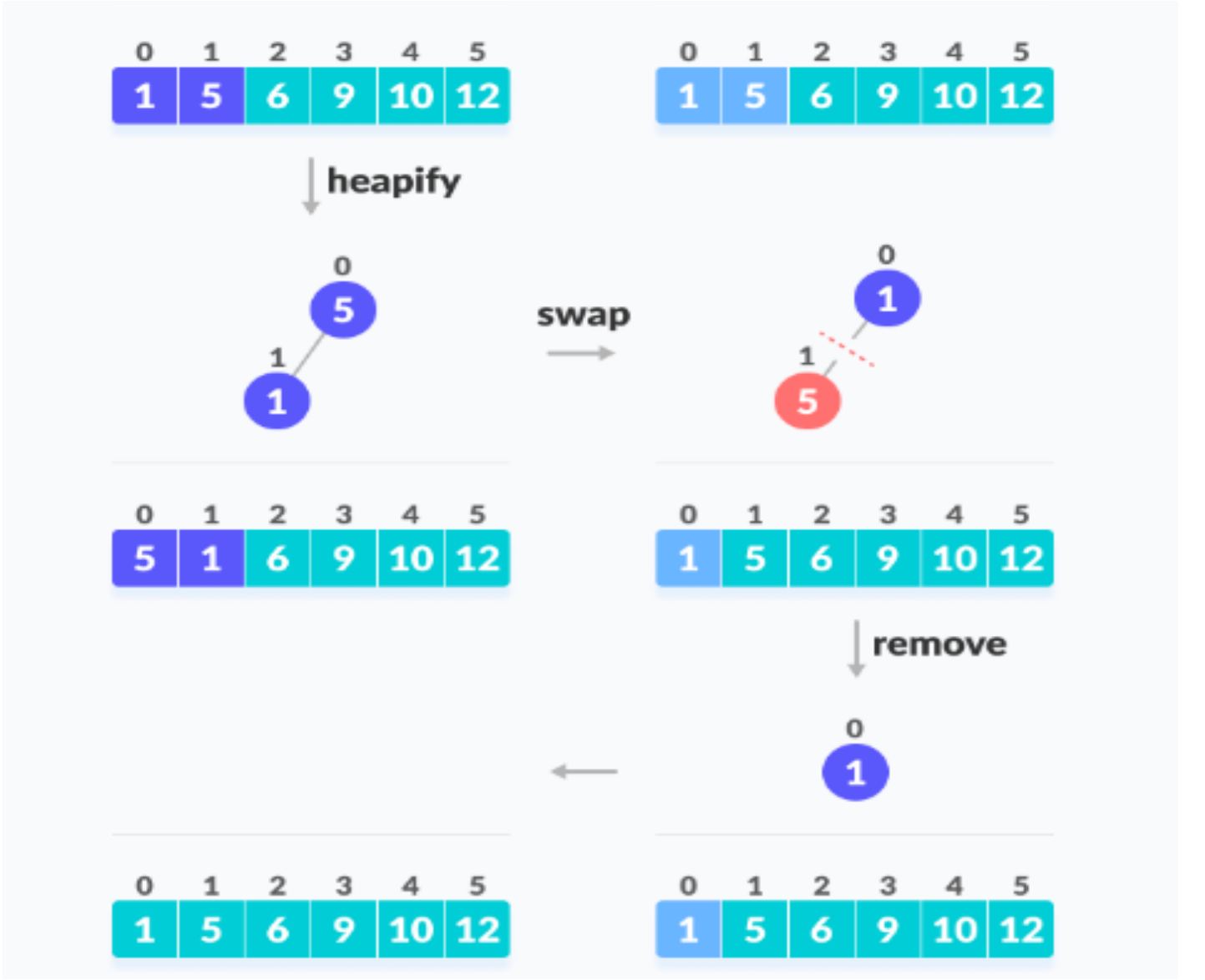
Remove: Reduce the size of the heap by 1.

Heapify: Heapify the root element again so that we have the highest element at root.

The process is repeated until all the items of the list are sorted.


swap
remove
heapify
swap
remove


0 1 2 3 4 5
1 6 9 5 10 12
0 1 2 3 4 5
1 6 9 5 10 12
heapify
0
5
1
6
2
1
3
9
swap
0
9
1
6
2
1
3
5
0 1 2 3 4 5
5 6 1 9 10 12
0 1 2 3 4 5
9 6 1 5 10 12
remove
0
5
1
6
2
1
heapify
0
6
1
5
2
1
0 1 2 3 4 5
5 6 1 9 10 12
0 1 2 3 4 5
6 5 1 9 10 12
swap
0
1
1
5
remove
0
1
1
5
2
6

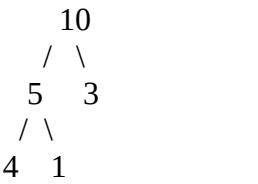

0 1 2 3 4 5
1 5 6 9 10 12
heapify
0
5
1
1
swap
0
1
1
5
0 1 2 3 4 5
5 1 6 9 10 12
0 1 2 3 4 5
1 5 6 9 10 12
remove
0
1
0 1 2 3 4 5
1 5 6 9 10 12
0 1 2 3 4 5
1 5 6 9 10 12

Lab Task

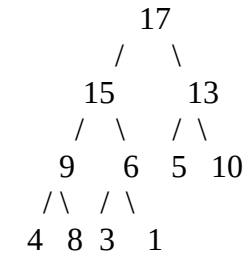
Note: In order to implement a binary tree, use array implementation only. The first index will be considered a root node in the array. The left child will be placed at 2*i+1 and the right child at 2*i+2. The parent node can be found at (i-1)/2.

Task 01: Create a class of complete binary tree which must contain the following methods:
- Insert a node
- Delete a node in such a way that it must not violate the complete binary tree property
- Count the number of parent nodes
- Count the number of leaf nodes
- Count the number of even nodes

Task 02: Given an array of N elements. The task is to build a Binary Heap from the given array. The heap can be either Max Heap or Min Heap.
Examples:
Input: arr[] = {4, 10, 3, 5, 1}
Output: Corresponding Max-Heap:



Input: arr[] = {1, 3, 5, 4, 6, 13, 10, 9, 8, 15, 17}
Output: Corresponding Max-Heap:

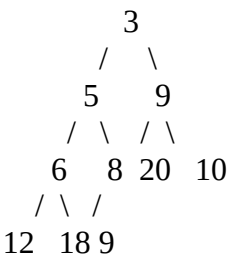


Note:
Root is at index 0 in array.
Left child of i-th node is at (2*i + 1)th index.
Right child of i-th node is at (2*i + 2)th index.
Parent of i-th node is at (i-1)/2 index.

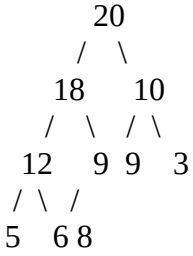
Task 03: Given a Binary Heap and an element present in the given Heap. The task is to delete an element from this Heap.

Task 04: Given an array representation of min Heap, convert it to max Heap.
Examples:

Input: arr[] = {3, 5, 9, 6, 8, 20, 10, 12, 18, 9}



Output: arr[] = {20, 18, 10, 12, 9, 9, 3, 5, 6, 8}



Input: arr[] = {3, 4, 8, 11, 13}
Output: arr[] = {13, 11, 8, 4, 3}

Task 05: Given below is a level order traversal of a complete binary tree, your task is to check if the given tree is a heap or not. Also, identify the type of heap such as min heap or max heap.
0, 1, 2, 9, 6, 3, 5, 8,

Task 06: Given an array of elements, sort the array in decreasing order using min heap.

Examples:

Input : arr[] = {5, 3, 10, 1}
Output : arr[] = {10, 5, 3, 1}

Input : arr[] = {1, 50, 100, 25}
Output : arr[] = {100, 50, 25, 1}

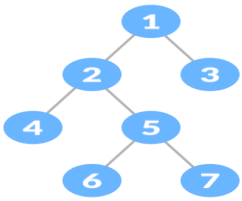
Task 07: Here is a family tree where each node represents a family member and their corresponding wealth. In this scenario, we have a complete binary tree structured as follows:

- The root node contains $50,000.
- The left child of the root node contains $25,000, and the right child contains $30,000.
- The left child of the left child contains $10,000, and its right child contains $15,000. Similarly, the left child of the right child contains $20,000, and its right child contains $10,000.
- The bottom level consists of nodes with $5,000 each.

Types of Binary Tree

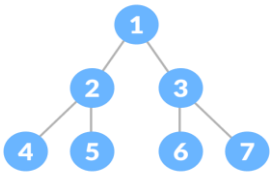
1. Full Binary Tree

A full Binary tree is a special type of binary tree in which every parent node/internal node has either two or no children.



2. Perfect Binary Tree

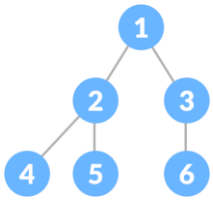
A perfect binary tree is a type of binary tree in which every internal node has exactly two child nodes and all the leaf nodes are at the same level.



3. Complete Binary Tree

A complete binary tree is just like a full binary tree, but with two major differences

Every level must be completely filled
All the leaf elements must lean towards the left.
The last leaf element might not have a right sibling i.e. a complete binary tree doesn't have to be a full binary tree.



How a Complete Binary Tree is Created?

1. Select the first element of the list to be the root node. (no. of elements on level-I: 1)
2. Put the second element as a left child of the root node and the third element as the right child. (no. of elements on level-II: 2)
3. Put the next two elements as children of the left node of the second level. Again, put the next two elements as children of the right node of the second level (no. of elements on level-III: 4) elements).
4. Keep repeating until you reach the last element.

Algorithm to insert an element in the max heap.

```
insertHeap(A, n, value)
{
n=n+1; // n is incremented to insert the new element
A[n]=value; // assign new value at the nth position
i = n; // assign the value of n to i
// loop will be executed until i becomes 1.
while(i>1)
{
parent= floor value of i/2; // Calculating the floor value of i/2
// Condition to check whether the value of parent is less than the given node or not
if(A[parent]<A[i])
{
swap(A[parent], A[i]);
i = parent;
}
else
{
return;
}
}
}
```

Deletion in Heap:

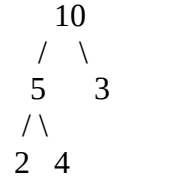
The standard deletion operation on Heap is to delete the element present at the root node of the Heap. That is if it is a Max Heap, the standard deletion operation will delete the maximum element and if it is a Min heap, it will delete the minimum element.

Process of Deletion:

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.

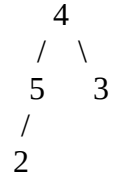
- Replace the root or element to be deleted by the last element.
- Delete the last element from the Heap.
- Since, the last element is now placed at the position of the root node. So, it may not follow the heap property. Therefore, heapify the last node placed at the position of root.

Suppose the Heap is a Max-Heap as:



The element to be deleted is root, i.e. 10.

Process:
The last element is 4.
Step 1: Replace the last element with root, and delete it.



Step 2: Heapify root.
Final Heap: