UNIT-V

HASHING

The process of mapping large amount of data into a smaller table using hash function, hash value, hash table is called as hashing.

HASH FUNCTION :-

It provides the mapping between large amount of data & smaller table.

HASH VALUE :-

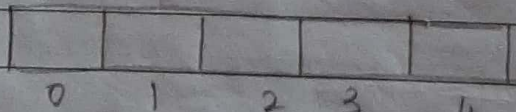
The index value returned by hash function is called hash value.

TYPES OF HASHING TECHNIQUE :-

1. Static Hashing.
2. Dynamic Hashing.

If the size of the table is fixed it is static hashing.
If the size of the table is not fixed is called dynamic hashing

IMP OVERFLOW OR COLLISION IN HASHING :-



$h(K) = K \% m$

It is linear probing or open addressing.
If the ~~hashing value~~ hashvalue is there then it is called as overflow or collision.
To overcome linear ~~has~~ probing or open addressing.

```
void search()
{
    int x, key;
    x = h(k);
printf("Enter key:");
scanf("%d", &key);
if (key == h(k))
    printf("element found in %d position", key);
else
    while (h(K) < key)
    {
        x = (h(K) + i) %
```

```
int search(int K, int hashsize)
{ int i, index;
hvalue = H(K);
for(i=0; i<hashsize; i++)
{
    index = (h_value + i) % hashsize;
    if (a[index] == K)
        return 1;
    else
        ~~return 0~~ continue;
}
```

```
return 0;
printf ("Element not found");
}
```

v.IMP BASIC HASHING TECHNIQUE:

1. direct
2. Subtraction
3. Modulo division
4. digit extraction
5. mid square
6. folding
7. Rotation.
8. Pseudo random generator.

6. FOLDING:-

Consists of two methods:-

(i) fold shift

```
  1 2 3 4 5 6 7 8 9
      4 5 6
      1 2 3
      7 8 9
    ① 3 6 8
```

discarded

→ adds address to the hash table.

(ii) fold boundary

123	456	789
	321	
	987	

① 764

8. PSEUDO RANDOM NUMBER GENERATOR:

$y = (ax + c) \% m$ (divide by prime no.)

TREES

Binary tree is a tree which has finite set of nodes i.e either empty or consists of a root & two subtrees.

They are called as left subtree & right subtree.

If tree is not empty then first node in tree is called root node.

Left subtree: It is a tree which is connected to left of root.

Right subtree: It is a tree which is connected to right of root.

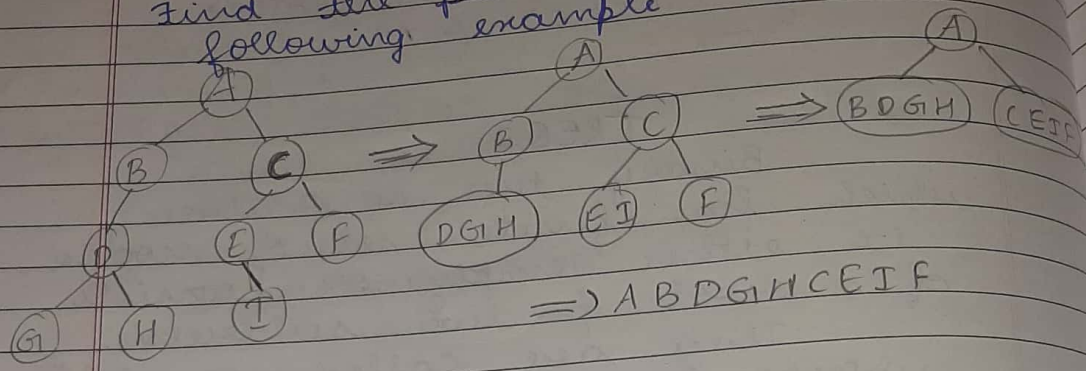
Binary tree Traversals

1] Preorder 2] Postorder 3] Inorder

Preorder

1. Process root node
2. Traverse the left subtree in preorder.
3. Traverse the right subtree in preorder.

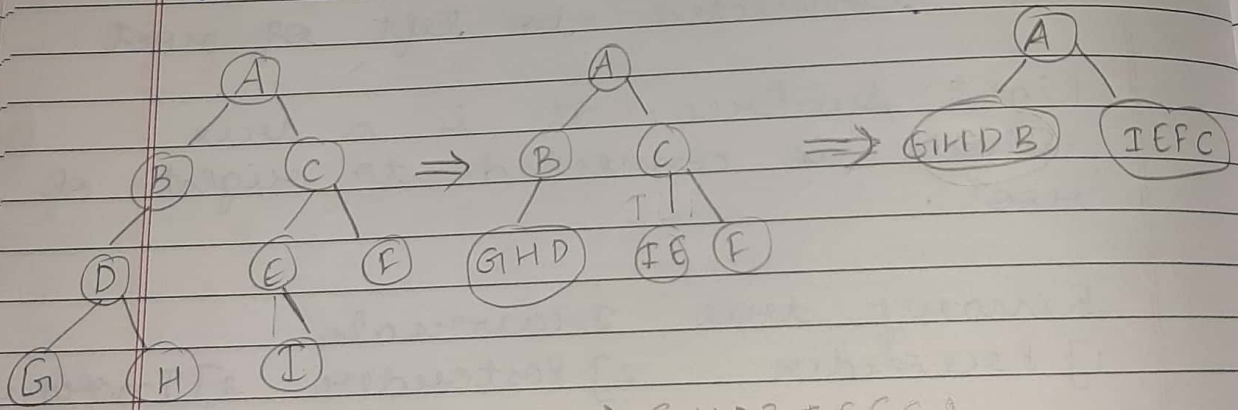
Find the preorder sequence of the following example



⇒ A B D G H C E I F

POSTORDER:

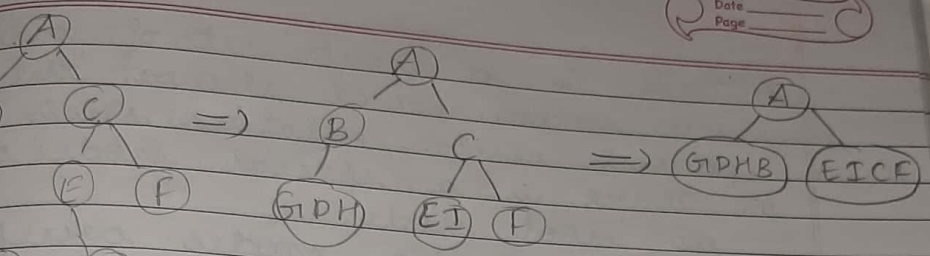
1. Traverse the left subtree in postorder
2. Traverse the right subtree in postorder
3. Process root node.



⇒ G H D B I E F C A

INORDER:

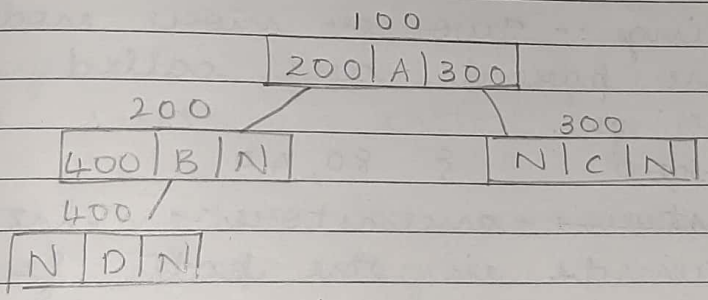
1. Traverse the left subtree in Inorder
2. Process the root node.
3. Traverse the right subtree



=> GDHBAEICF

BINARY TREE REPRESENTATION:
1] linked Representation.
2] Array Representation.

LINKED




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

ARRAY

A	B	C	D
0	1	2	3

Tree is a finite set of one or more nodes that shows the parent child relation such that
i) There is a special node called as root node
ii) The remaining nodes are partitioned into disjoint subsets called as $T_1, T_2, T_3 \ldots T_n$ where $(n \geq 0)$.
and T_1, T_2, T_3 are called as the subtrees.

child node :- Node obtain from the parent node.

Siblings :- Two or more nodes having same parent are called as siblings.
Eg:- 50, 60 & 80, 40

~~ancestors~~ :- ancenstors :- The node obtained in the path from specified node x while moving upwards towards root node are called ancenstors
Eg:- 50 & 100 are ancenstors of 70.

Descendants :- The nodes in the path below parent are descendant

left descendants :- The node that lie towards left subtree of node x are called as left descendants

Right descendants :- The node that lie towards ~~left subtree~~ Right subtree of node x are called Right descendants.

Left subtree :- All the nodes that are left descendants of node x from the left subtree of x.

Right subtree :- All the nodes that are right descendants of node x from the right subtree of x.

Degree :- The no. of subtrees of a node is called its Degree.

leaf node :- A node in a tree that has degree zero is called leaf nodes. Eg :- 70, 35, 30, 40

Internal nodes :- All the nodes except leaf nodes in a tree are called Internal nodes.

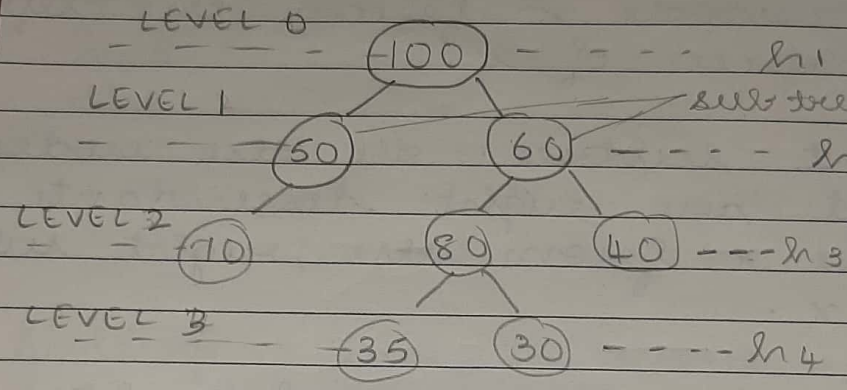
External nodes :- The null link of any link node in a tree is called external nodes

Eg :- 50, 70, 35, 30, 40

Level :- The distance of a node from root node is called level of the node.

Eg:

Height (Depth) :- The maximum level of any leaf node in a tree
Eg:- 3

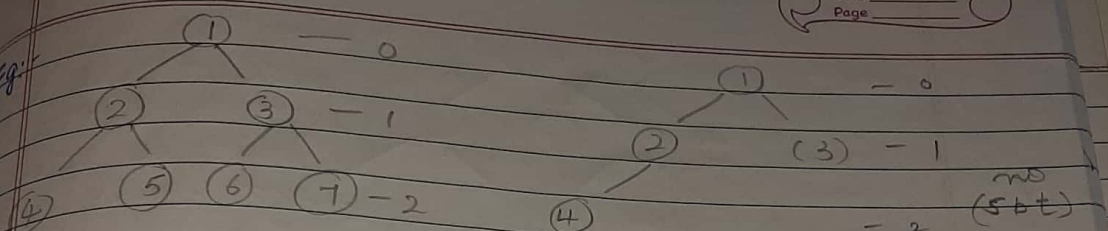


TYPES OF BINARY TREE :

1. Strickly binary tree (Full binary tree)
2. Skewed tree
3. complete
4. Expression
5. Binary search tree

1. STRICKLY BINARY TREE :
   A binary tree having 2^i nodes in any given level i is called as strickly binary tree where $(i \geq 0)$.

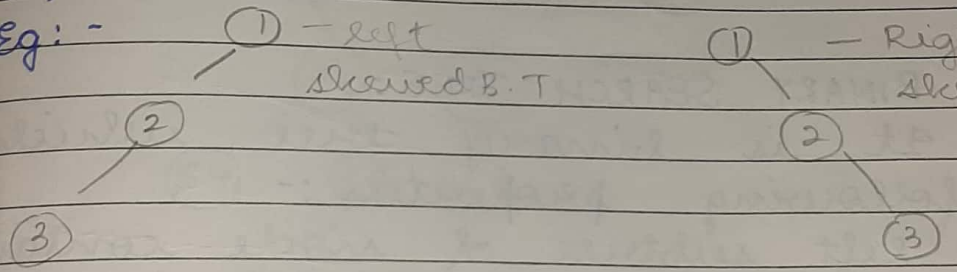
Eg:-



2. SKEWED BINARY TREE:

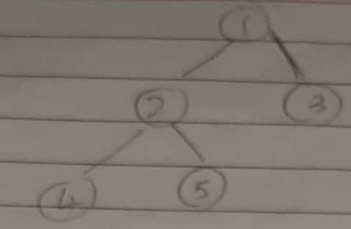
It consists of only left subtree or right subtree. A tree with only left subtree is called left skewed binary tree. & a tree with only right subtree is called as right skewed binary tree

Eg:-

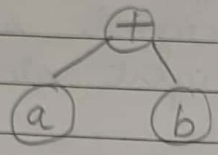


3. COMPLETE BINARY TREE:

It is a binary tree where every level except last level is completely filled. If the nodes in the last level are not completely filled then all the nodes in that level should be filled only from left to right.



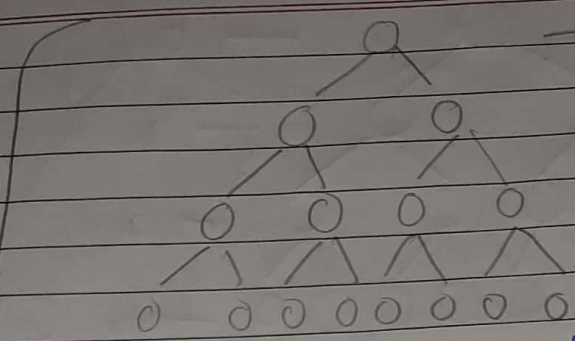
4. EXPRESSION BINARY TREE:
It is a binary tree where each internal node corresponds to operator & each leaf node corresponds to operand.



5. BINARY SEARCH TREE:
IMP It is binary tree which has following properties :-
(i) left subtree of node contains only the nodes with keys lesser than nodes key.
(ii) Right subtree of node contains only the nodes with keys greater than nodes key.

Example for constructing a binary search tree :-
(i) 100, 50, 60, 125, 69, 110, 101, 59, 68
(ii) 210, 56, 200, 220, 226, 229, 98, 75, 80

→ (i)

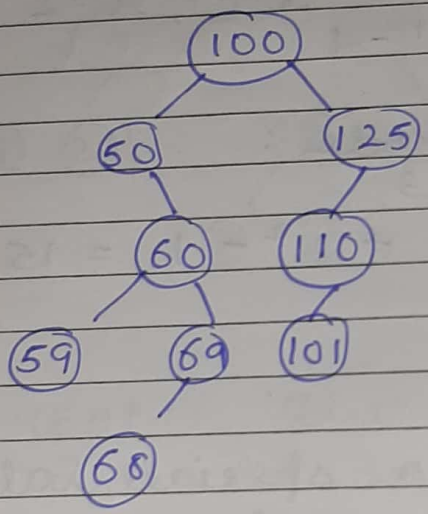


No. of nodes at level $i = 2^i$

$= 2^0 + 2^1 + 2^2 \ldots\ldots 2^i$

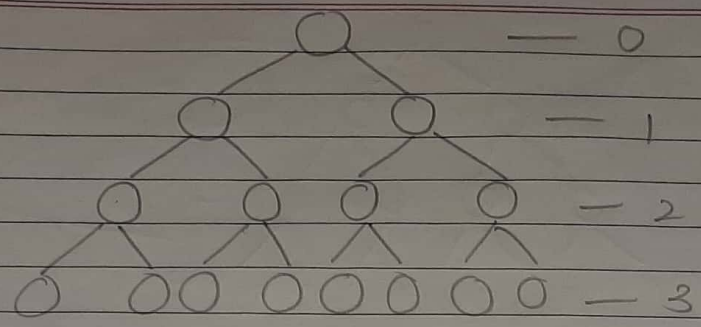
~~$S = a(r^n - 1)/(r-1)$~~

$a = 1,\ n = i+1,\ r = 2$



Properties of binary tree :-

1. Maximum nos of nodes on level i of binary tree $= 2^i, (i \geq 0)$
2. The maximum no of nodes in a binary tree of depth $k = 2^k - 1$, where $k = (i+1)$



no of nodes at level $i = 2^i$

$= 2^0 + 2^1 + 2^2 \ldots 2^i$

$\therefore S = a(r^n - 1)/(r - 1)$ ⇒ (The geometric progression of above sequence)

$a = 1,\ n = i + 1,\ r = 2$

$= 2^{i+1} - 1$

i = depth or height

$\therefore$ no of nodes:

eg:- $i = 3$

$\therefore 2^{k} - 1 = 2^4 - 1 = 15$

$k = i + 1$

HEAP:

Heap is a special balanced binary tree where root is compared with its children and arranged accordingly. It has to be a complete Binary tree.

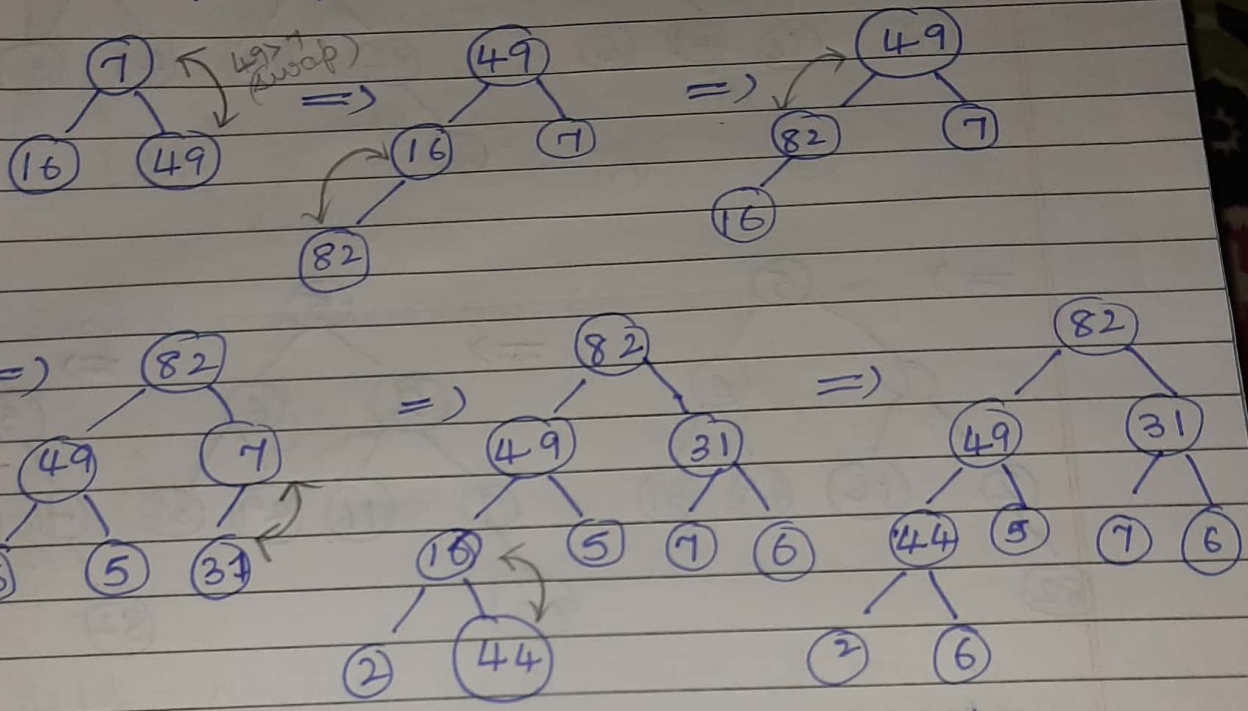
There are two types of heap:

1] Max heap 2] Min heap

MAX HEAP: The value of root node must be greater than its children.

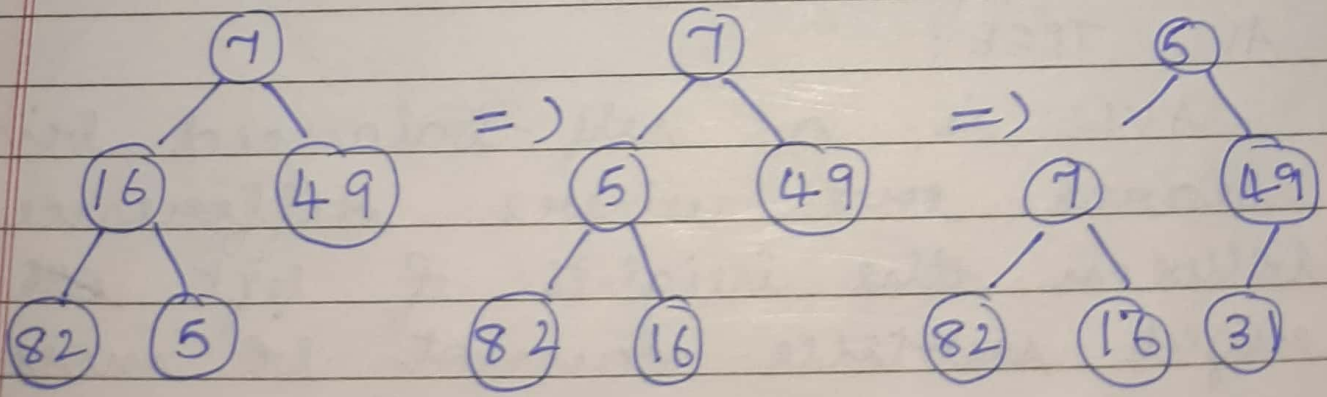
The process of exchanging the element is called heapify part.

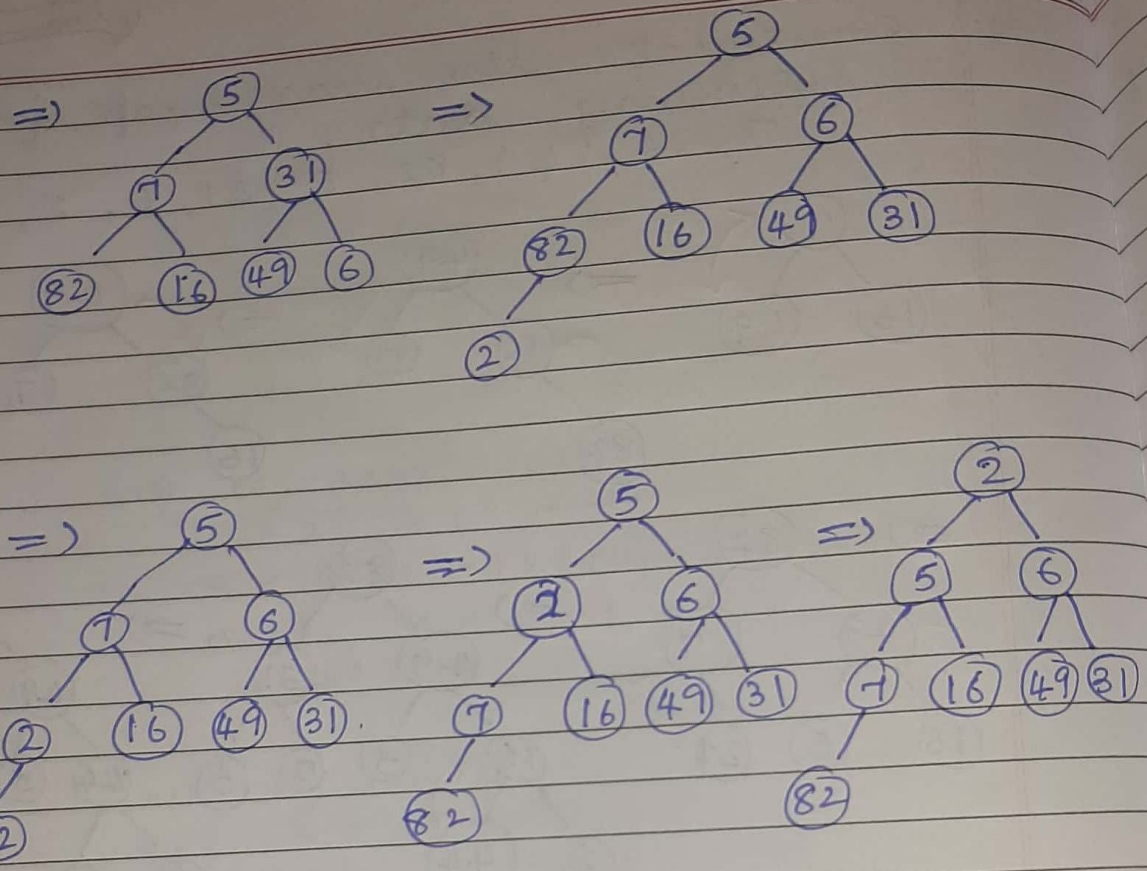
eg:- construct the max heap for following data.
7, 16, 49, 82, 5, 31, 6, 2, 44



2. MIN HEAP:- The value of root node should be less than its children.

eg:- construct the min heap for 7, 16, 49, 82, 5, 31, 6, 2, 44





APPLICATIONS OF HEAP:

1. Heap is used in heap sort
2. In implementation of priority queue.

APPLICATIONS OF BINARY SEARCH TREE:

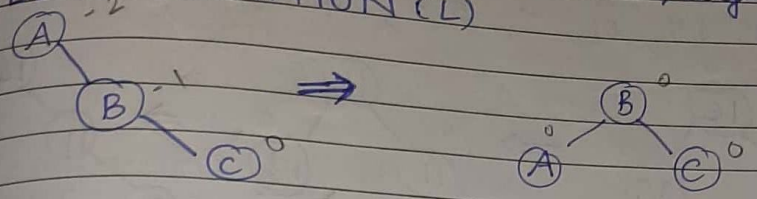
1. Used in sorting
2. To implement priority queue

AVL TREE:

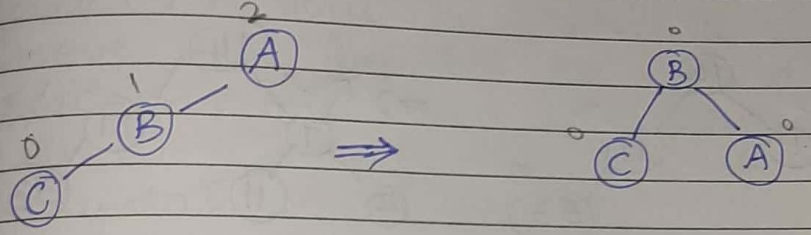
AVL is a self-balanced binary search tree where difference between the heights of left & right subtree cannot be more than one & cannot be less than -1.

Balanced factor → 1, 0, -1.
(Height of left subtree - height of right subtree)

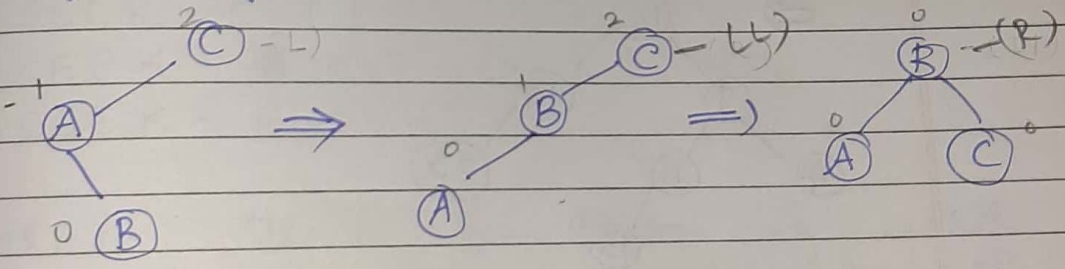
LEFT ROTATION (L)



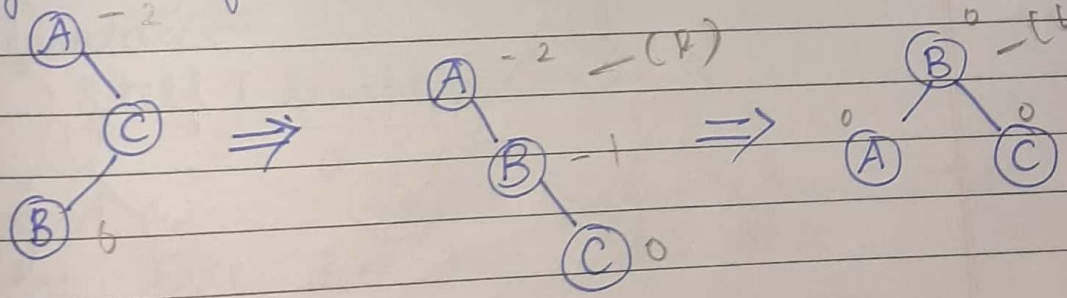
Right rotation (R):-



Left - Right (LR) rotation :



Right - left (RL) rotation :

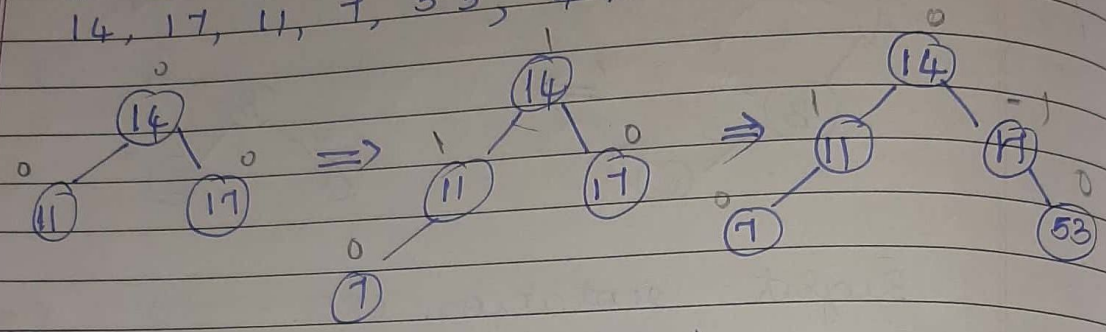


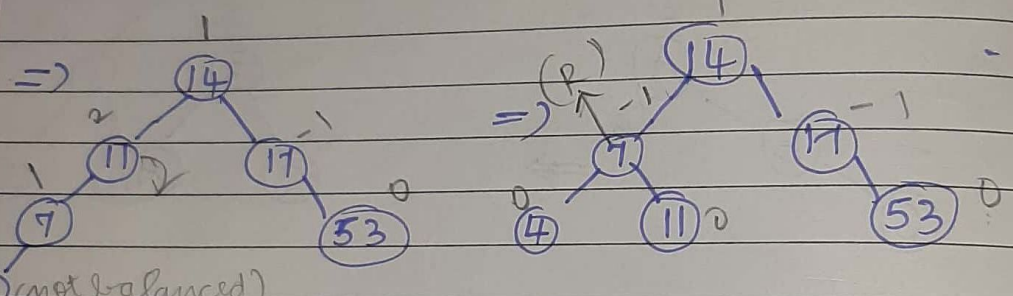
If any tree is imbalanced go from leaf node & find which rotation is suitable

Example:-

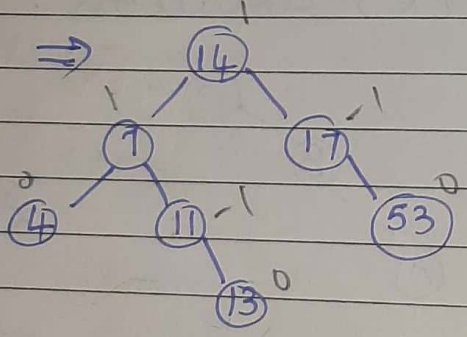
Insert these numbers into empty AVL tree

14, 17, 11, 7, 53, 4, 13

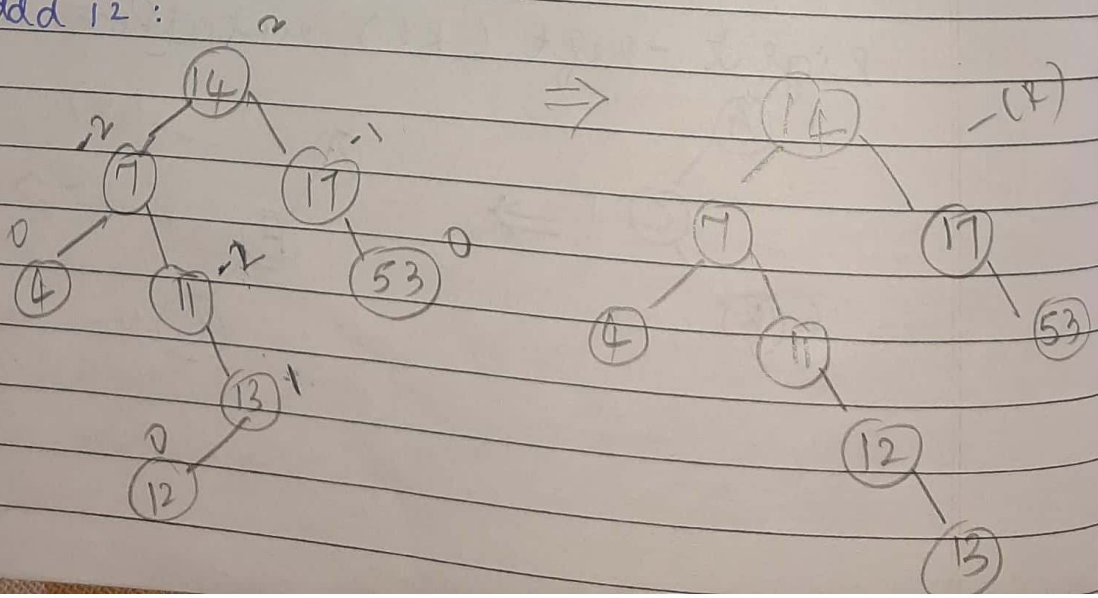


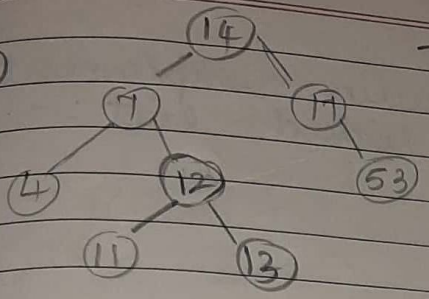


(not balanced)



add 12:





(continuation of UNIT 2:)

FILE HANDLING FUNCTIONS IN C:

1. fseek( ): This function is used to move file pointer to a different position or specified position in a file.

   Syntax:-

   fseek(file pointer, offset, origin);

   possible values for origin

   SEEK - SET:

   SEEK - CUR:

   SEEK - END:

2. ftell:- This function is used to tell the current position of the file pointer.

   Syntax:-

   ftell(filepointer);

3. Rewind( ):- This functions sets the file pointer to the beginning of the file or strring.

Syntax:- rewind (file pointer);

continuation :-
circular queue implementation using singly linked list :-

```
struct node
{
  int data;
  struct node *link;
};
struct node *front = NULL;
struct node *rear = NULL;
void insert()
{
  struct node *temp;
temp = (struct node *)malloc(sizeof
                    (struct node));
printf("Enter element:");
scanf("%d", &temp->data);
temp->link = NULL;
if(rear == NULL)
{
    rear = temp;
    front = temp;
}
else
{
    rear->link = temp;
    rear = temp;
    rear->link = front;
}
}
```

deletion :-

```
void delete ()
{
    struct node *curr;
    if (front == NULL)
    printf ("queue is empty:");
    else
    {
        curr = front;
    printf ("%d element is deleted",
                                front->data);
        front = front->link;
        rear->link = front;
        free (curr);
```

Display :-

```
    void display()
    {
    struct node * temp;
    if (front == NULL || rear == NULL)
    printf ("queue is empty");
    else
    {
        temp = front;
        do
        {
            printf ("%d" temp->data);
            temp = temp->link;
        } while (front != temp);
    }
}
```

Scanned by CamScanner

Construct AVL tree for following elements 21, 26, 30, 9, 4, 14, 28, 18, 15, 10, 2, 3