Circular Queue:

```
# define N5
    int queue[N];
    int Front = -1;
    int Rear = -1;
    void enqueue (int x)
  {
  if(Front == -1 && Rear == -1)
  {
   Front = Rear = 0;
   queue[Rear] = x;
  }
 elseif ((Rear+1) % N == Front)
   Printf("Queue is full");
 else {
    Rear = (Rear +) % N;
    queue [Rear] = x;
 } }
  void dequeue ()
 {
  if (Front == -1 && Rear == -1)
  Printf("Queue is empty");
  else if (Front == rear)
 {
  Printf("%d", deleted element", queue[Front])
    Front = Rear = -1;
  }
   else {
Printf("%d deleted element is", queue[Front]);
```

```
    Front = (Front + 1) % N;
}
  Void display()
        int i = 0;
{
if( Front == -1 && Rear == -1)
 Printf("Queue is empty");
 while (i != Rear)
{
  Printf("%d", queue[i]);
    i = (i+1) % N;
 Printf ("%d", queue[Rear]);
}
  Void main()
{
    enqueue (10);
    enqueue (20);
    enqueue (5);
    enqueue (6);
    display ();
    dequeue ();
   dequeue ();
    enqueue (10);
    enqueue (-1);
}
```

Application of circular queue

i) It is used in page replacement algorithm
ii) CPU Sheduling
   Round Robin Algorithm
iii) Holding Computers System supply a holding one for maintaining Communication b/w two process or program

Dequeue

- Input restricted
- Output restricted

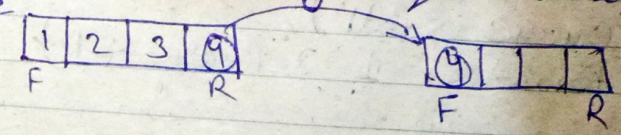
Threads => When large process divided into small-small parts they are called threads.

Application of Dequeue

- UNDO
- REDO

Multiprocessing

In multiprocessing deque is used



Deque using circular Queue:-

```
# define N5
int dequeue [N];
int Front = -1;
int rear = -1;
Void enqueue front ()
{
if (Front == -1 && Rear == -1)
Printf("Queue is empty");
   Front=Rear=0;
dequeue [Front] = x;
else if (Front == 0)
{
  Front = N-1;
  dequeue [Front] = x;
else if {
        Front --;
dequeue [Front] = x;
}
else
  Pf("Queue is full");
}

Void enqueue rear (int x)
{ if (Front == 0 && Rear = N-1) || (Front = Rear)
  Printf("Queue is full");
else if (Front == -1 && Rear == -1)
{
  Front = Rear = 0;
dequeue [Rear] = x;
}else if (Rear == N-1)
{
  Rear = 0;
  dequeue [Rear] = x;
  else
  { R++;
  dequeue [Rear] = x;
}}

  void display ()
{ int i = Front;
  while (i != Rear)
{
  Printf("%d", dequeue[i]);
    i = (i+1) % N;
}
  Printf("%d", dequeue [Rear]);
}

  Void dequeue front ()
{ if (Front == -1 && Rear == -1)
  Printf("Empty");
  else if (Front == Rear)
{
  Printf("%d", deleted queue[Front]);
    Front = Rear = -1;
}
  else if (Front == N-1)
{
  Printf("%d", dequeue [Front]);
    Front = 0;
```

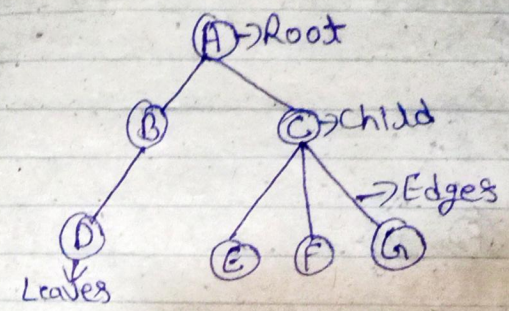
```
else
  Printf("%d", dequeue[Front]);
     Front ++;
}

  void dequeue rear ()
{ if (Front==-1 && Rear==-1)
  Printf("empty");
  else if (Front==Rear)
{
  Printf("%d", dequeue[Rear]);
     Front=Rear=-1;
else if (Rear=0)
{
  Rear=N-1;
}
  else {
  Printf("%d", dequeue[Front]);
     Rear--;
}

  void main()
{
   enqueue Front (10);
   enqueue front (20);
   enqueue Rear (30);
   enqueue Rear (40);
   enqueue front (15);
   enqueue front (35);
   display();
   dequeue front ();
}
```

Unit ⇒ 4 Tree

It is non-linear data structure



General tree

Tree is also called as Hierichal data structure.

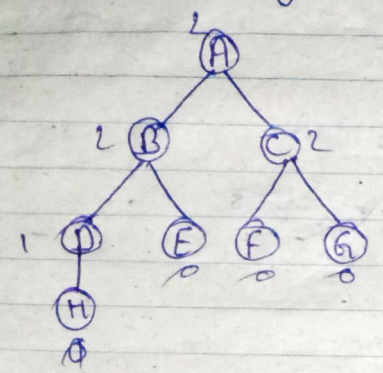
Tree termonology-

Path = A → C — f

Important term

i) Path
ii) Root
iii) Leaves
iv) Child
v) Parent
vi) Edge
vii) Level
viii) Sub-tree
ix) Traversing (To visit data atleast one time)
x) Keys (Data)

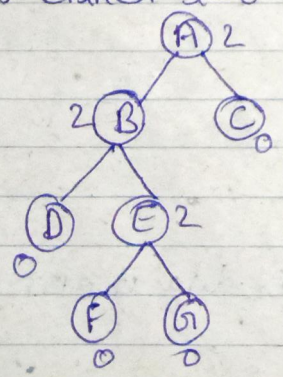
Binary tree ⇒ Binary tree is known a when degree of all node <=2



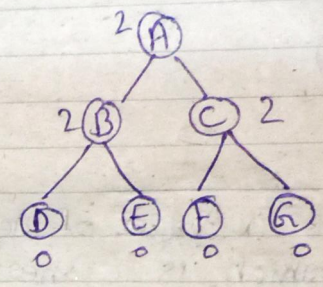
Degree of tree ⇒ Degree of tree is calculated as the highest number of nodes in a tree. Ex ⇒ in the above tree the degree of tree is =2.

Types of Binary tree-

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

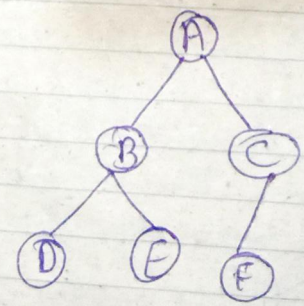


ii) Perfect Binary tree ⇒ A perfect binary tree is a type of binary tree in which every internal node has exactly two children and all the leaves node are not same level.

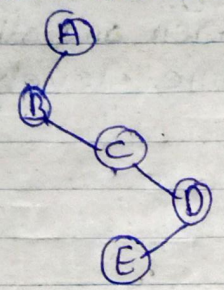


iii) Complete Binary tree ⇒ It is just like a full binary tree but with two major differences.

- Every level must be complete filled
- All leaves element must lean towards the left
- The last leave element might not have right sibling.

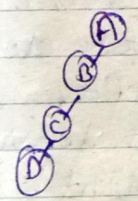


iv) Degenerate or Pathological Binary tree ⇒ It is tree having single child either have left or right.

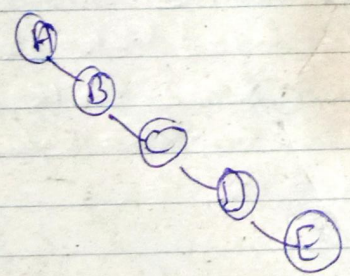


v) Skewed ⇒ It is a type of degenerate binary tree in which is either dominated by left node or right node. They are of two types

- Left Skewed BT



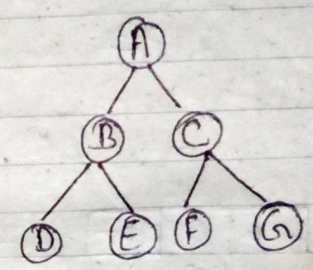
- Right Skewed BT



$\lceil \rceil \Rightarrow (ceil)$

Representation of binary tree

There are two type representation
i) Array
ii) LinkList



i) Array representation ⇒

i^{th} index

Left child $= [(i*2)+1]$

Right child $= [(i*2)+2]$

Parent $= \left\lfloor \frac{(i-1)}{2} \right\rfloor$

Case 1

A	B	C	D	E	F	G
0	1	2	3	4	5	6

Left child $= (2*i)$

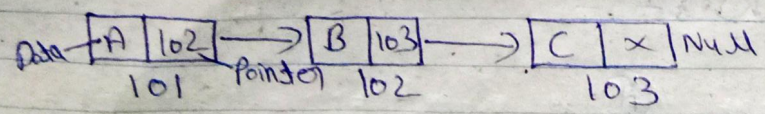
Right Child $= [(2*1)+1]$

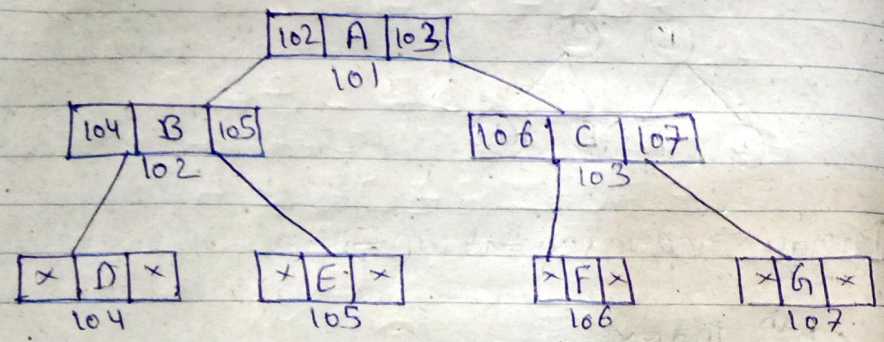
Parent $= \left[\frac{i}{2} \right]$

Case 2

N	A	B	C	D	E	F	G
0	1	2	3	4	5	6	7

ii) LinkList ⇒ It is a kind of list to link with each other fields are divided into parts Data, pointer.

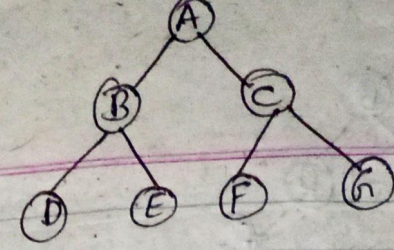




Array representation is a static representation where as LinkList is a Dynamic representation.

Traversal of Binary Tree

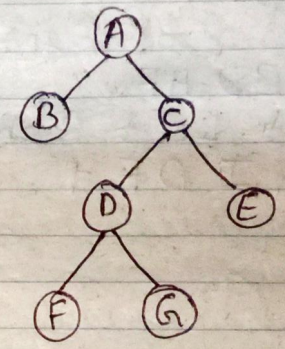
i) Post order ⇒ Left Right Root
ii) In order ⇒ Left Root Right
iii) Pre order ⇒ Root Left Right



i) Post order ⇒ D E B F G C A

ii) In order ⇒ D B E A F C G

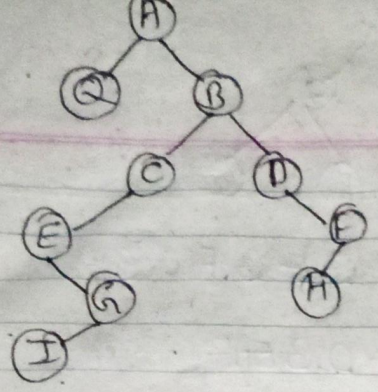
iii) Pre order ⇒ A B D E C F G



i) Post order ⇒ B F G D E C A

ii) In order ⇒ B A F D G C E

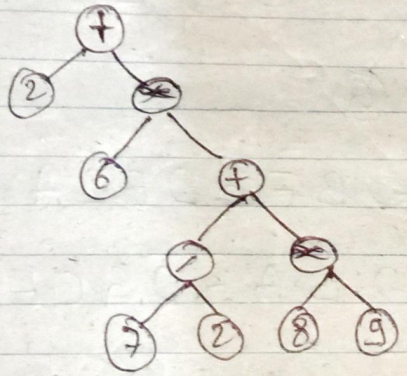
iii) Pre order ⇒ A B C D F G E



i) Post order ⇒ Left Right Root
Q I G E C H F D B A

ii) In order ⇒ Left Root Right
Q A E I G C B D H F

iii) Pre order ⇒ Root Left Right
A Q B C E G I D F H



i) Post order ⇒ 2 6 7 2 / 8 9 * + * +

ii) In order ⇒ 2 + 6 * 7 / 2 + 8 * 9

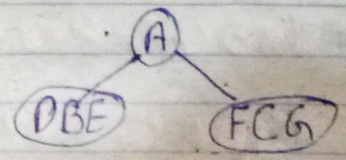
iii) Pre order ⇒ + 2 * 6 + / 7 2 * 8 9

Counting binary Tree

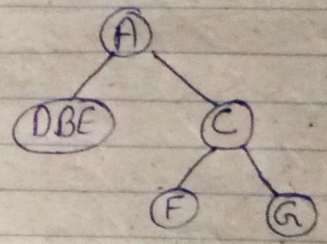
i)

Post ⇒ D E B F G C A
In ⇒ D B E A F C G

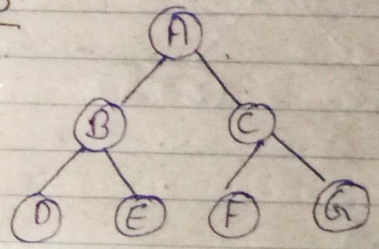
Step-1



Step-2



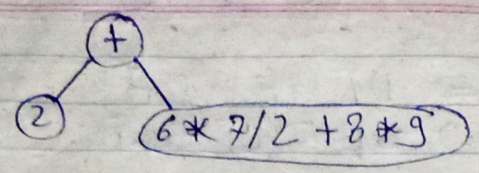
Step-3



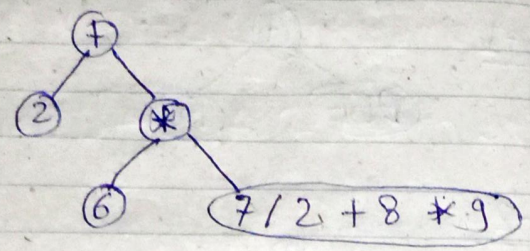
ii)

Post ⇒ 2 6 7 2 / 8 9 * + * +
In order ⇒ 2 + 6 * 7 / 2 + 8 * 9
or
Pre order ⇒ + 2 * 6 + / 7 2 * 8 9
In order ⇒ 2 + 6 * 7 / 2 + 8 * 9

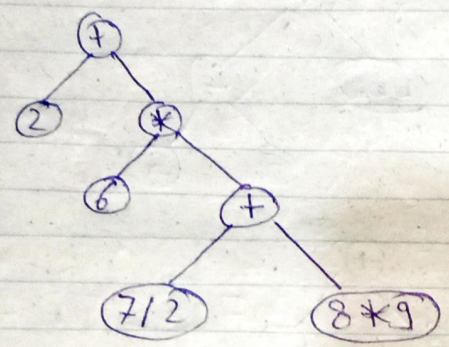
Step-1



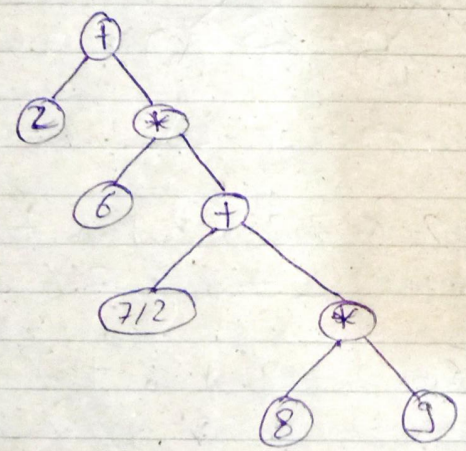
Step-2



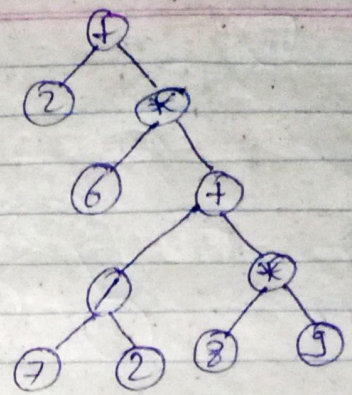
Step-3



Step-4



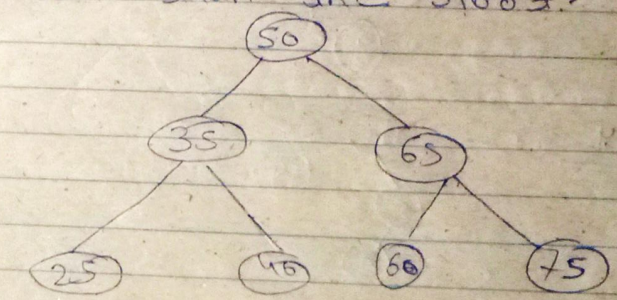
Step-5



Binary Search Tree (BST)

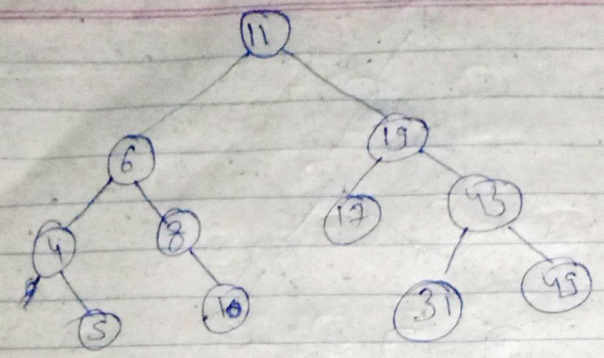
In BST every data are store in short method.

In BST tree he arrange data in ordered sequence such that the left sub-tree value is always less than the root and right sub-tree value is always greater than the root.

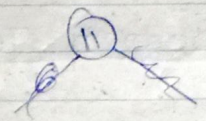


Construct a Binary Search tree are from the following.

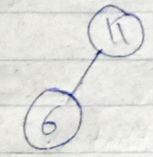
11, 6, 8, 19, 4, 10, 5, 17, 43, 49, 31



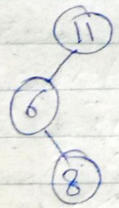
Step-1



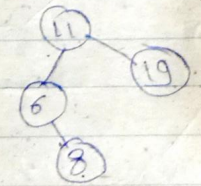
Step-2



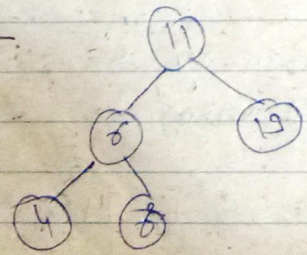
Step-3



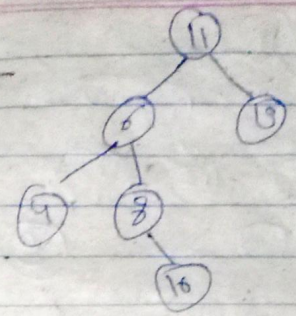
Step-4



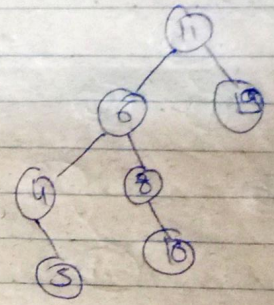
Step-5



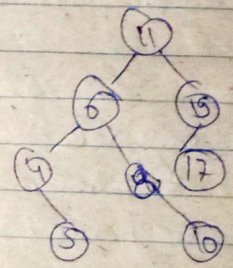
Step-6



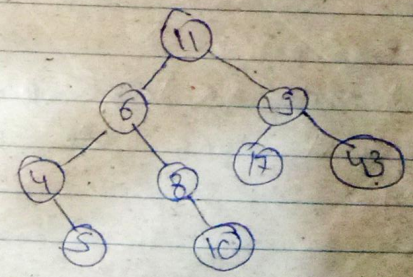
Step-7



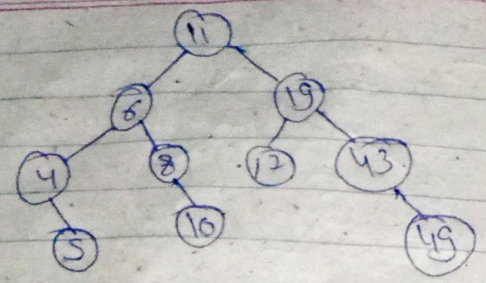
Step-8



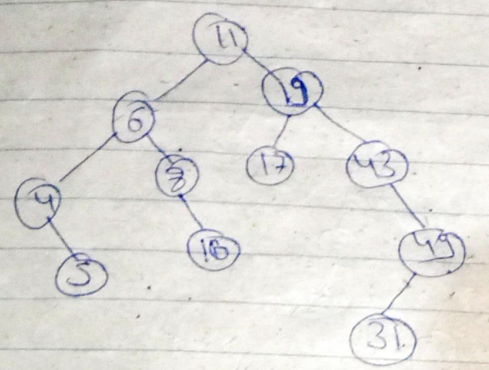
Step-9



Step 11-



Step 12-



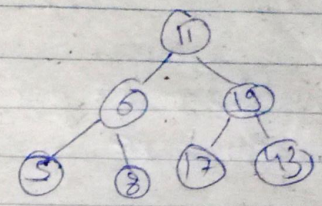
- Node having 0 degree.
- Node having 2 degree.
- Node having 1 degree.

0 degree -

If last node or leaf node is deleted nothing special done it can be deleted directly.

1 degree -

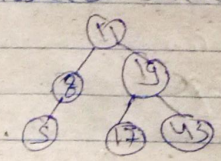
Substitute the child node with the deleted node.

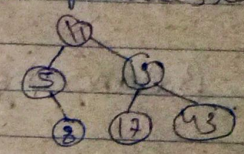


2 degree -

we have two condition -

- Inorder successor
- Inorder predecessor

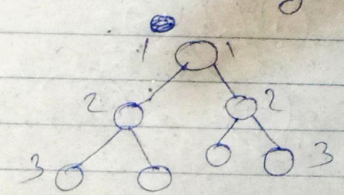




Q. What is AVL tree? How to balance AVL Tree:

AVL Tree (Anderson Velski Landis)

It is also known as Height Balance Tree. it is special type of BST. who have three balance factor or hieght balance factor they are {-1, 0, 1}



(3-3=0)

0

when balance factor is 0 then Hieght of left sub-tree and

1

Height of left sub-tree is greater than height of right sub-tree

-1

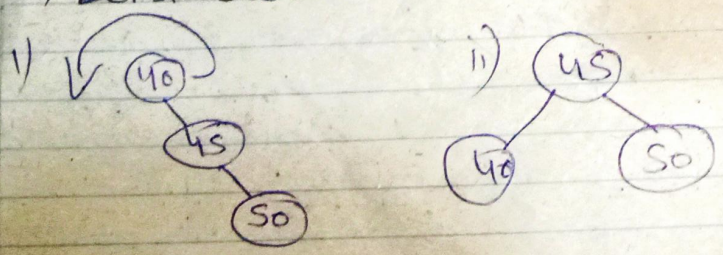
Height of left sub tree is less than height of right sub-tree

Balanceing of AVL tree

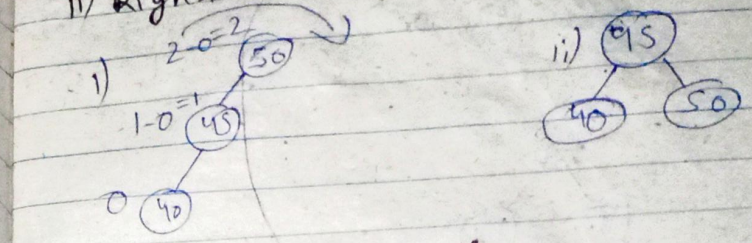
- There may be certain condition-

when the hieght of AVL tree is not balance then we have to balance it. for this purpose we will use certain kind of rotation.

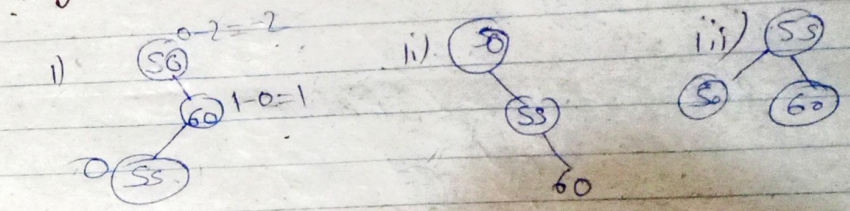
i) Left rotation-



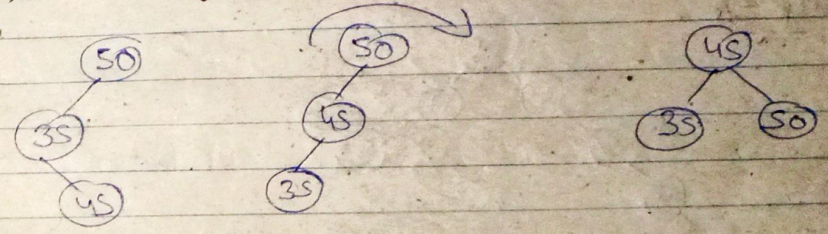
ii) Right rotation-



iii) Right left rotation-



iv) Left Right rotaion-



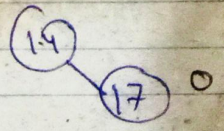
Q- Construct of AVL Tree following

14, 17, 11, 7, 53, 4, 13, 12, 8, 60, 19, 16, 20

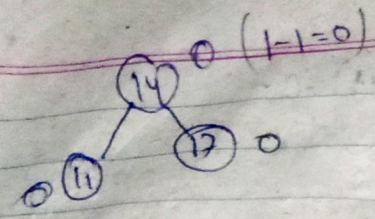
Step-1

0-1=-1

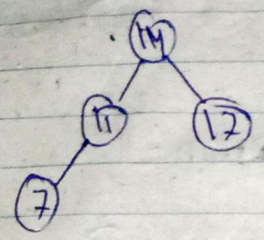
Step-2



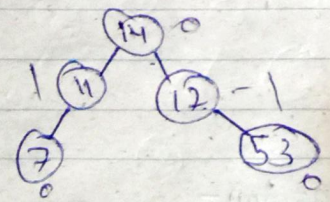
Step-3



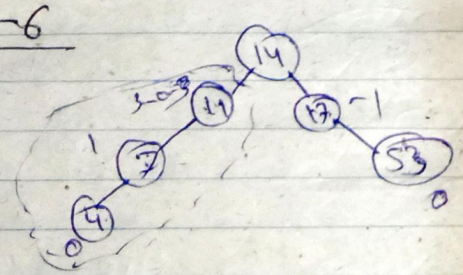
Step-4



Step-5

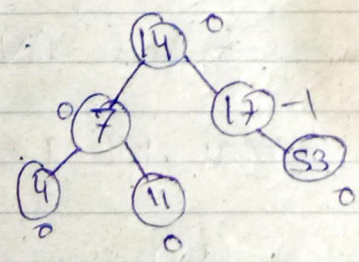


Step-6

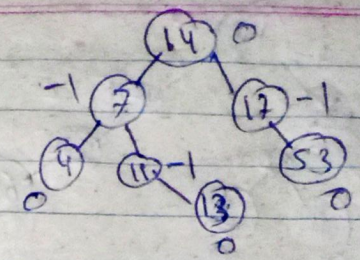


unBalance

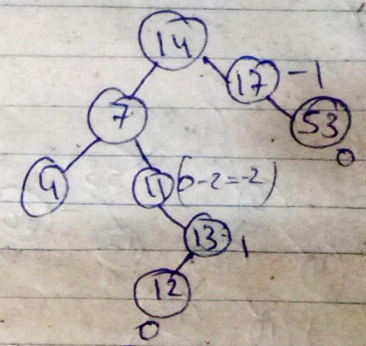
Step-7



Step-8

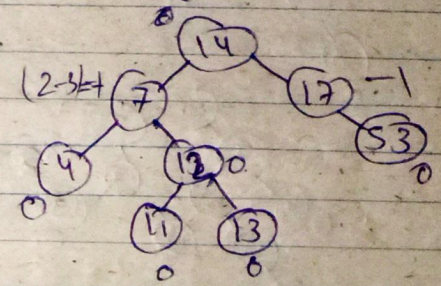


Step-9

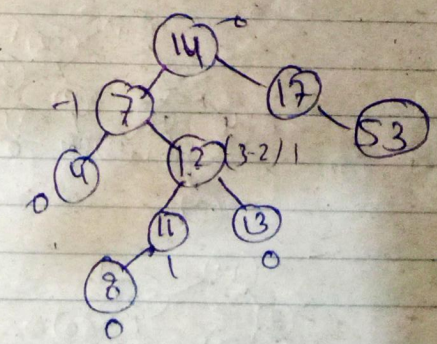


unBalance

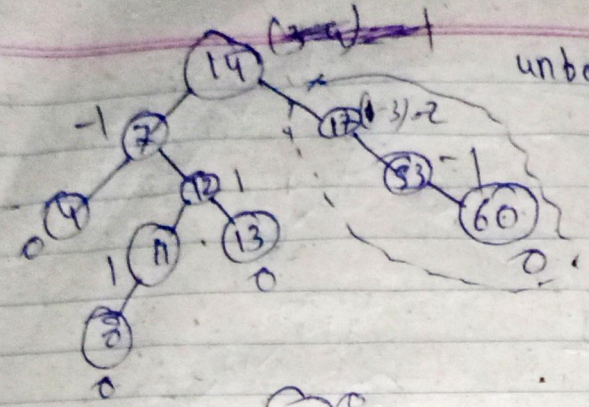
Step-10



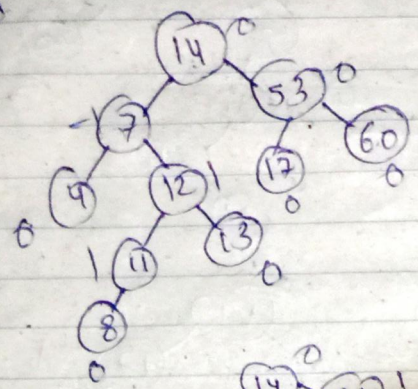
Step-11



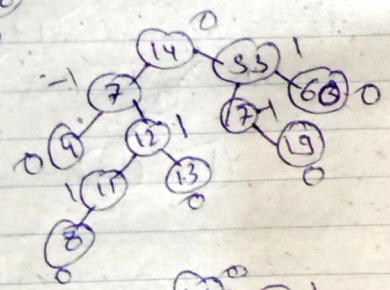
Step-12



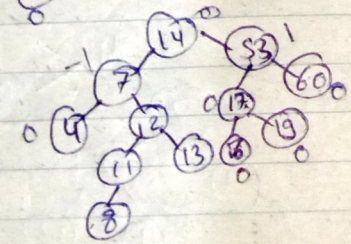
Step-13



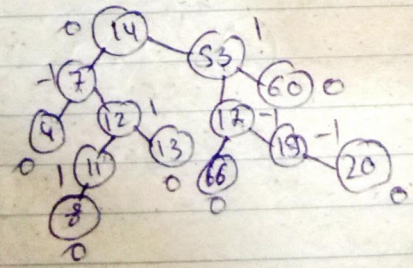
Step-14



Step-15



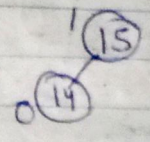
Step-16



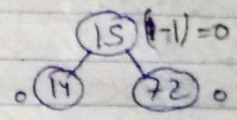
Q Construct ~~the~~ of AVL tree following-
15, 14, 72, 63, 12, 8, 5, 9, 33, 55, 7, 6,
11

Step-1

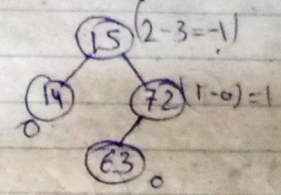
Step-2



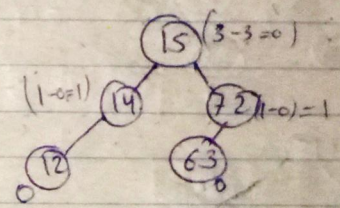
Step-3



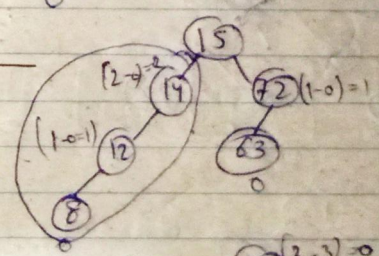
Step-4



Step-5

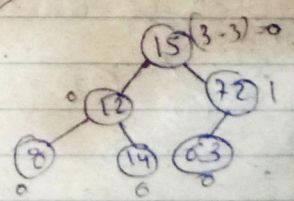


Step-6

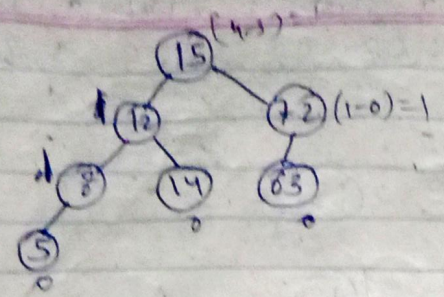


Unbalance (Now balance in next step)

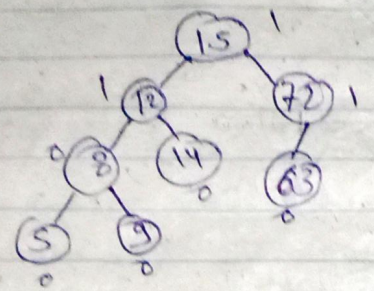
Step-7



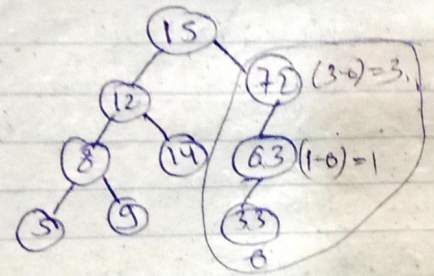
Step-8



Step-9

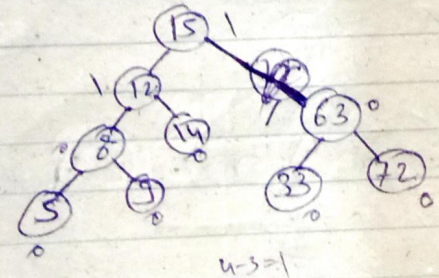


Step-10

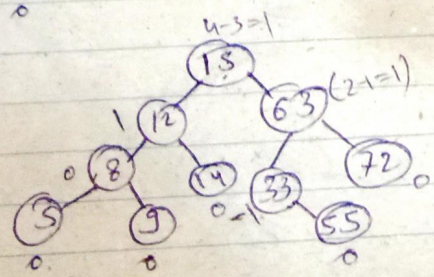


UnBalance

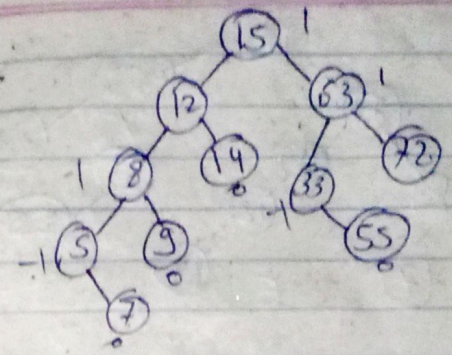
Step-11



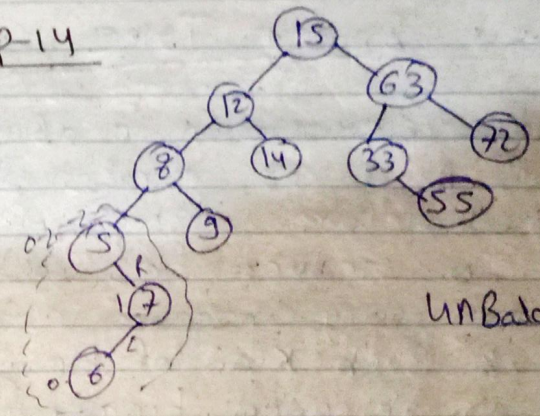
Step-12



Step-13

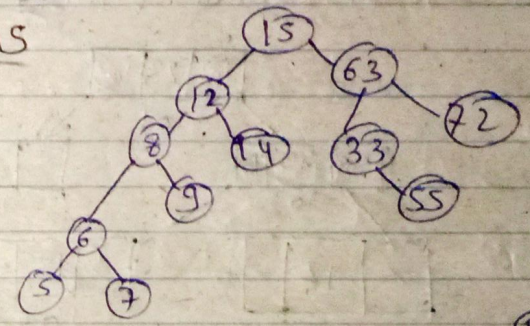


Step-14

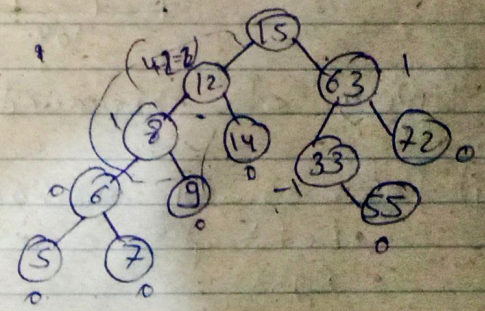


UnBalance

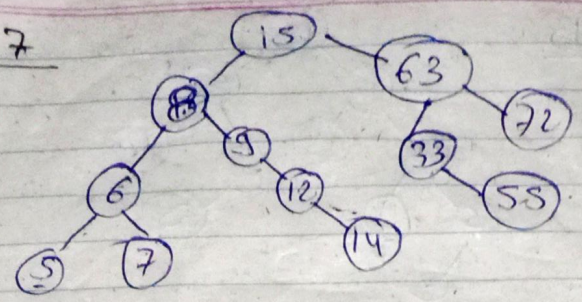
Step-15



Step-16

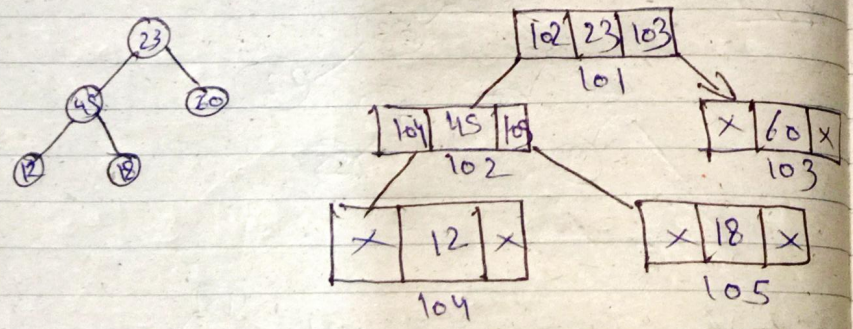


Step-17



Threaded Binary Tree

~~The~~ In Linklist representation in a binary tree the no. of Null is greater the no. of links, Therefore wastage of memory. To reduce this wastage we are intruducing threades in the linklist. Thus such kind of binary tree is called as threaded binary tree

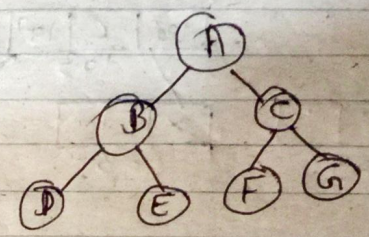


Types of threaded binary tree.

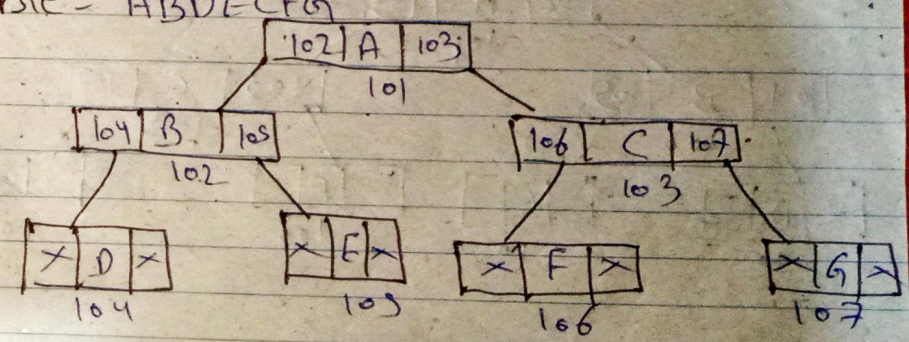
i) Post order threaded BT
- Left post order threaded Binary tree
- Right " " " " "
- Two way postorder threaded Binary tree

ii) In order Threaded BT
- Left inorder threaded binary tree
- Right " " " "
- Two way

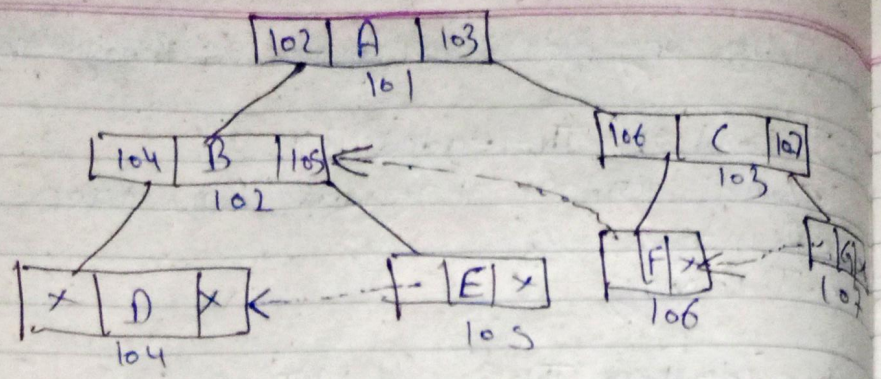
iii) Pre-order Threaded BT
- Left pre-order threaded BT
- Right " " "
- Two way



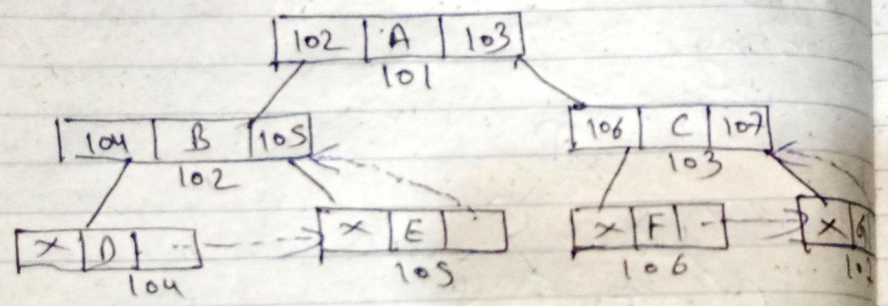
Post - DEBFCGA
In - DBEAFCG
Pre - ABDECFG



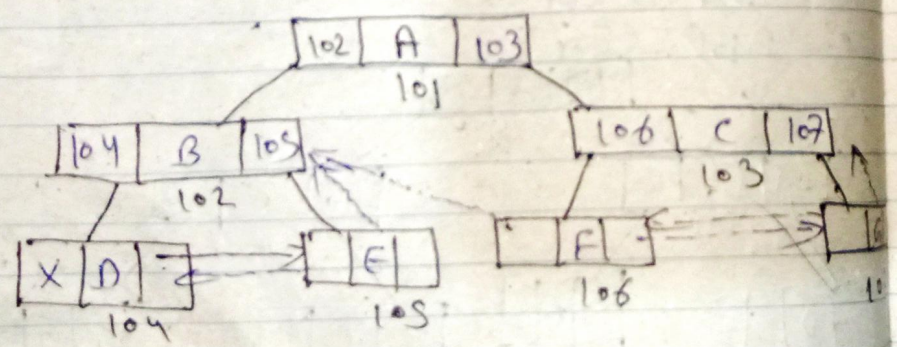
i) a Left Postorder Threaded BT -



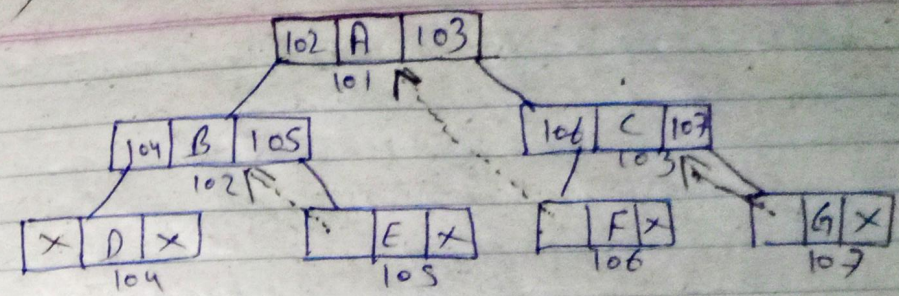
Right postorder threaded BT



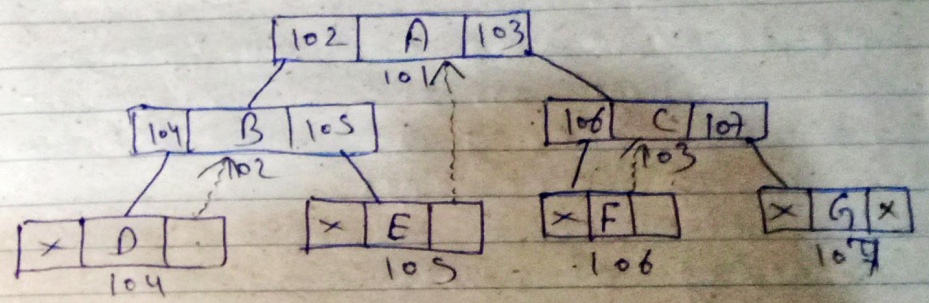
Two-way postorder threaded Binary tree



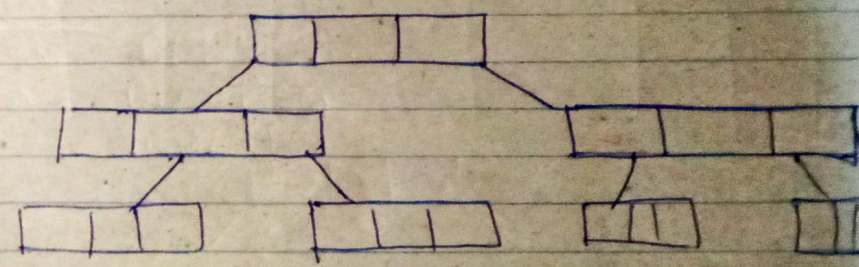
ii) a) Left in-order threaded BT



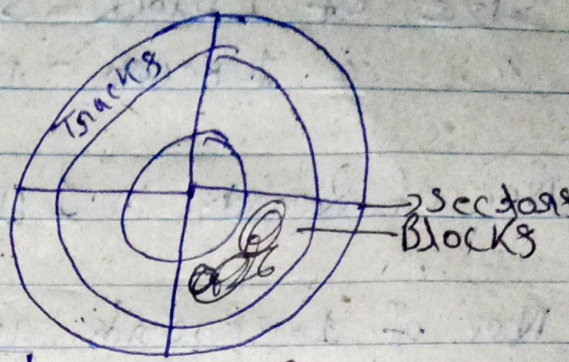
b) Right in-order threaded BT



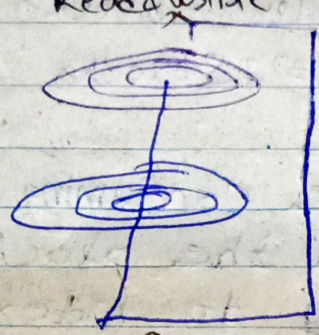
c) Two way in-order threaded BT

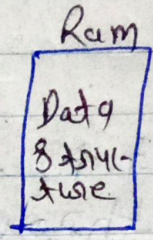


B. Tree



से० फ़रो
(आगे-पीछे) Trackchange करना





Any Data address are required there

- Tracks no.
- Sectors no.
- Blocks no.
- offset

128byte = 10byte 50byte 10byte 8byte 50byte

EID	Name	Dept	Section	Address
1	AJay	CS	A	Satna
2	vijay	IT	B	Rewa
3	Rohit	MBA	C	Jabalpur
4	mohit	M.Com	D	//
5	Ram	BCA	A	satna
6	Shyam	Ag	E	Satna
⋮	⋮	⋮	⋮	⋮
100	Records			

Record

Size of 1 Block = 512 bytes
" " 1 Record = 128 bytes

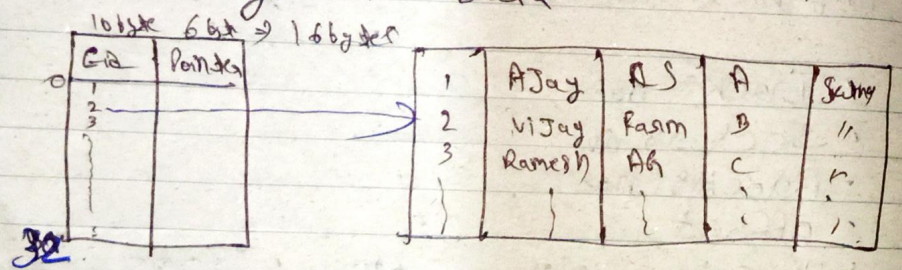
No. of Records in 1 block = $\frac{512}{128}$ = 4

No. of 100 Records no. of blocks
= $\frac{100}{4}$ ⇒ 25 blocks

Since it is a time consuming process to go through all the block so, we will use indexing to index the record for faster accessing.

Address of a record is called recordpointer

Indexing of Data



No. of entry blocks = $\frac{512}{16}$ ⇒ 32

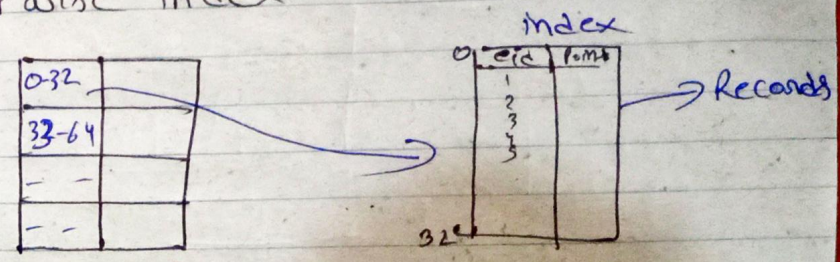
1. for 100 Records No. of blocks = $\frac{100}{32}$ ⇒ 4

In the above ex to access one record we need 2 blocks one is index and second is record.

If there are thousand records then no. of index will also increase and again time complexcity will also increase so to solve this problem so we will make another index and that index is called "Sparse index".

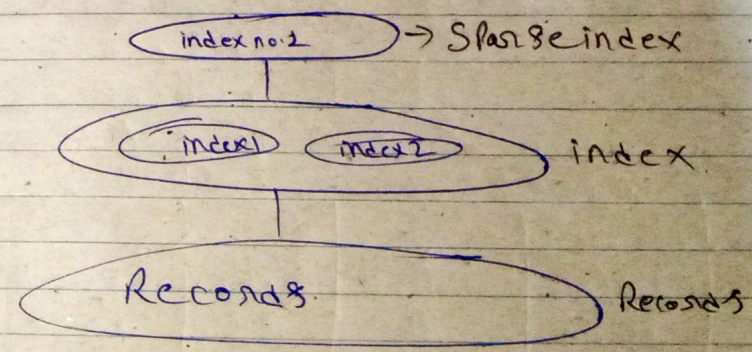
Multilevel indexing ⇒

Sparse index



This index is also called multilevel index.

M-way search tree



~~Address of a record is c~~

BST is a type of M-way Search tree who has two child