IV - TREES

Page No.
Date

* Symbol Tables :-

- Symbol Tables is standard data structure used in language processing
- Information about various source language constructs can be stored in symbol tables.
- Each symbol have attribute associated with it.

Attribute are :-

1. Symbol Name
2. Symbol size
3. Symbol value
4. Symbol type
5. Symbol scope
6. Symbol Address

* Symbol Table operation :-

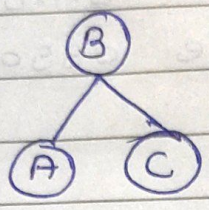
- The two main operation
  1. Insert()
  2. Delete (symbtab, ele)
  3. Create (symtab)

* Static tree table :-

- Fixed no of entry in symbol table
- No insertion of any entry
- No deletion of entry
- eg - OBST

* cost of tree =

$$\sum \left(\frac{\text{no of nodes} \times \text{no of comparison}}{\text{total no of nodes present in tree}} \right)$$



Node	No of com
A,C	2
B	1

$$\text{cost of tree} = \frac{(2 \times 2) + (1 \times 1)}{3}$$

$$= \frac{4+1}{3} = \frac{5}{3} = 1.66$$

* **Dynamic tree :-**
  - Dynamic symbol table is kind of symbol table in which entries are constantly changing.
  - Frequently ingertion and deletion can be performed
  - eg - AVL tree
  - Dynamic symbol table is used

* **Dynamic Programming :-**
  - Dynamic programming is method in which solution to prombiem is obtained by making sequence of decision.
  - The optimal solution is obtain from no of possible solution being generated

PAGE NO.
DATE:

① What is OBST is data structure ? What are Advantages of OBST ?

⇒

- The element with lesser probability should be placed away from the root, the binary search tree created with such kind of arrangement is called optimal Binary search tree (OBST). Such tree with optimal cost is called optimal binary search tree.

- As we know that in binary search tree the nodes in the left subtree have lesser value than the root node and the nodes in the right subtree have greater value then the root node

- The frequency and key-value determine the overall cost of searching a node. The cost of searching is very important factor in various application.

~~Advantages~~:

(i) ~~Randomization~~ : The cost of searching a node from OBST is reduced.

(ii) The optimal binary search is not modified once it is constructed.

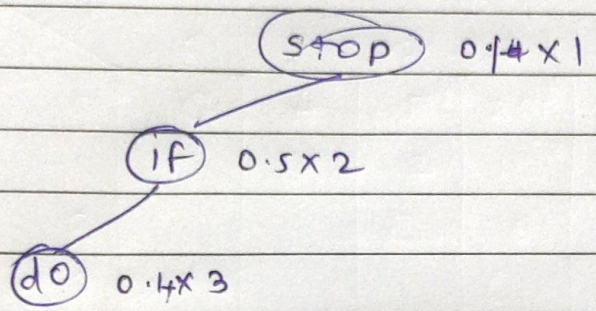
(iii) Proposed Dynamic programming gives a with little time complexity.

(iv) The most frequently accessed node is nearer to root node. Hence it can be accessible easily.

PAGE NO.
DATE:

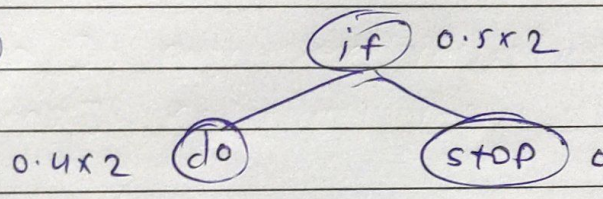
List binary search tree with 3 words (w1, w2, w3) = (do, if stop) words occurs with probabilities (p1, p2, p3) = (0.4, 0.5, 0.1) find expected access time

(1)



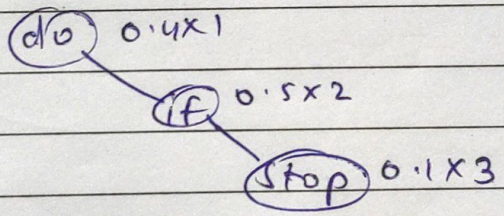
Expected no. of comparisons = 2.3

(2)



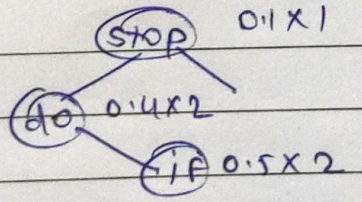
EC2 = 1.5

(3)



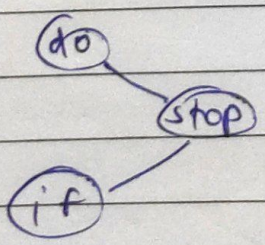
EC3 = 1.7

(4)



EC4 = 2.4

(5)



EC5 = 2.1

∴ BST is optimal with expected no. of comparison = 1.5.

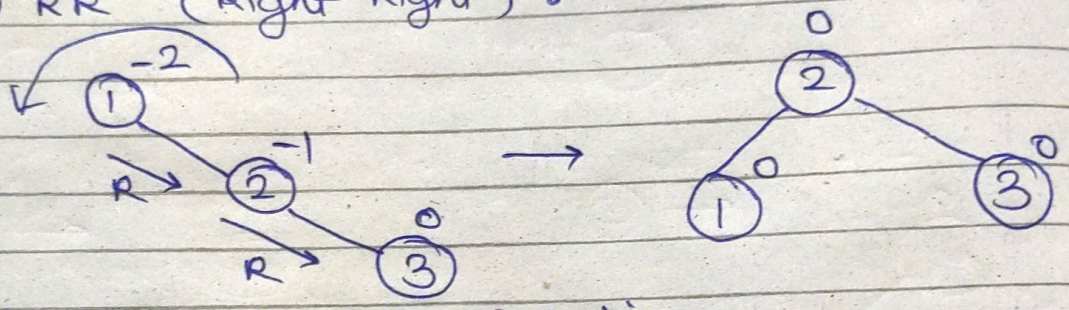
* AVL Tree - A height balanced tree.

→ Balance factor : (BF)
= left subtree height - Right subtree height

* the Balance factor for AVL is only {-1, 0, 1} is allowed.

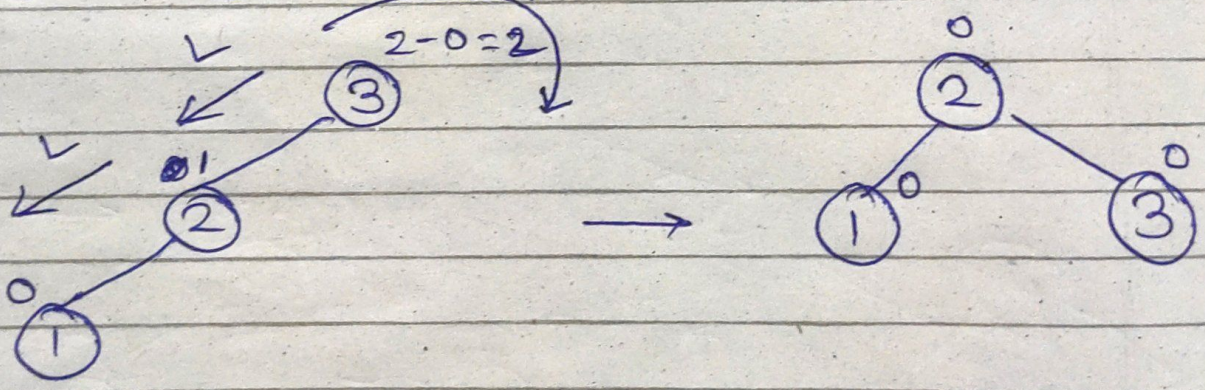
* Rotation's :-

① RR (Right Right):



- RR need left Rotation.
- The middle node is alway's Root.

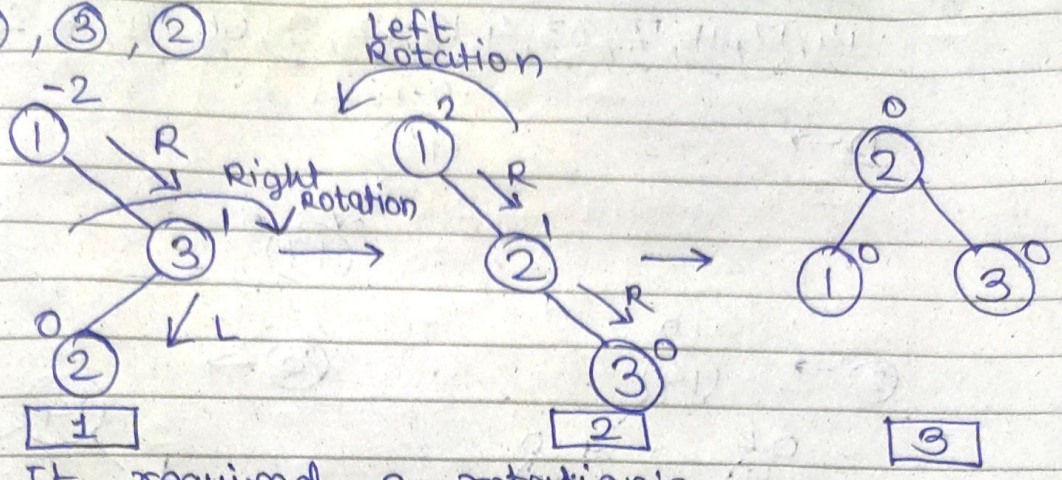
② LL (Left Left):



- The LL have Right Rotation
- Middle node is ROOT.

③ RL (Right left)

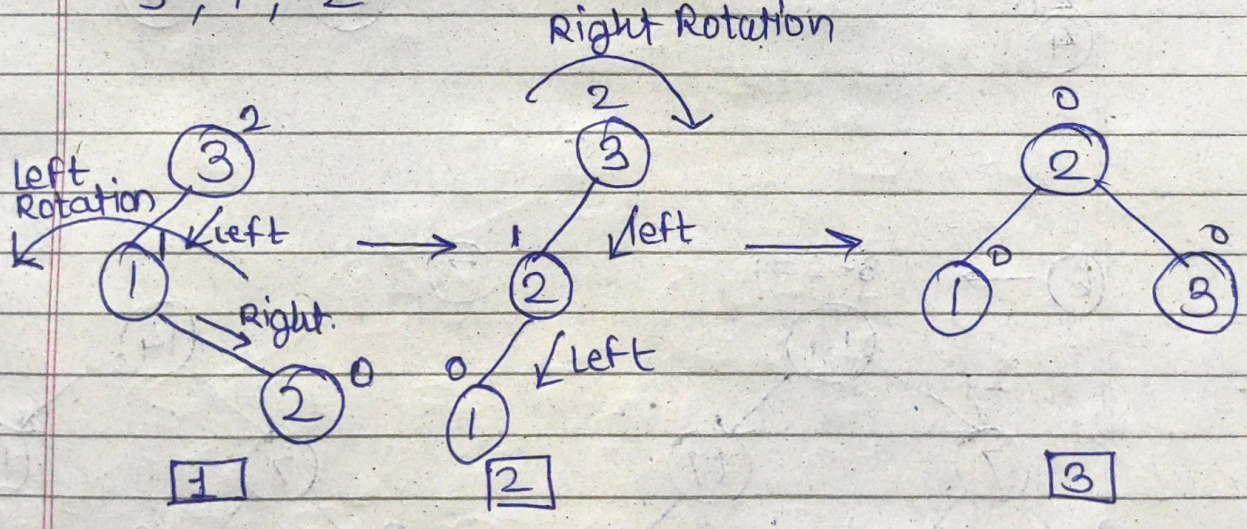
① , ③ , ②



- It required 2 rotation's
- In step 1 of RL we need to do Right Rotation on half of tree.
- In step 2 we need to do operation on RR tree like left Rotation.
- And middle node is root.

④ LR (Left Right)

3 , 1 , 2



- It required 2 rotation's
- In step 1 of LR we need to do left Rotation on half of tree
- Now our tree is LL
- In step 2 we need to do operations OR LL tree like Right rotation,
- And middle node is root.

* ~~A-A tree :→~~

* Red black Tree :→

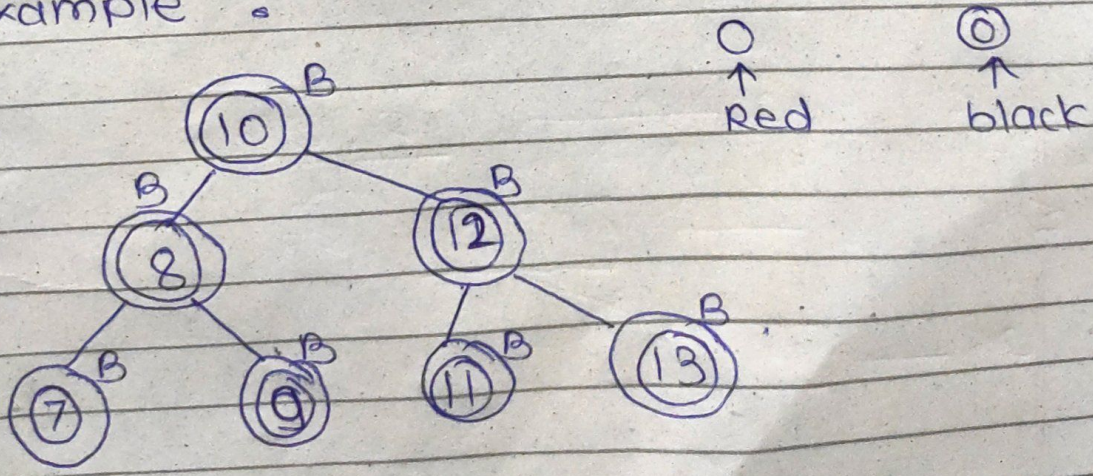
- Red-black Tree is basically a BST.
- It is alternative to the AVL tree.

→ properties of Red black Tree.

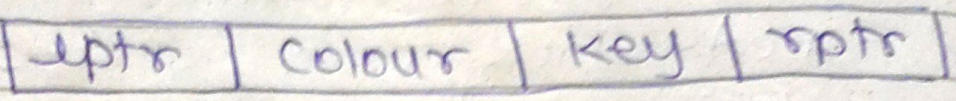
1. every node is coloured red or black
2. The Root is alway's black.
3. Every leaf which is Nill is black (Nill ≠ NULL)
   (Not leaf nod only Null Node)
4. If node is red then its children are black. (No two adjacent nodes are red)
5. Each path from Root to leaf node having same Number of black node's.
6. It is self balance BST.

- It contain 1 flag like 0 or 1 for colour. (red, black).
-

Example :



* Node Structure.



- Node contain 4 field's
  lptr → left pointer.
  rptr → Right pointer.
  key → value to be store.
  colour → this field contain colour of Node.
  e.g: Red = 1 , Black = 1

- There is no R - R parent child relation.

*

PAGE NO.
DATE:

Q. AA Tree

→

- The AA tree is created from the idea of Red black tree, but contains fewer rotations & less complex.

- Time complexity is O(log n).

* Properties of AA trees:

- Every node is colored either red or black.

- The root node is always black.

- All the external nodes are black.

- If node is red then its children must be black.

- All path from any node to a descendant leaf must contain the same number of black nodes.

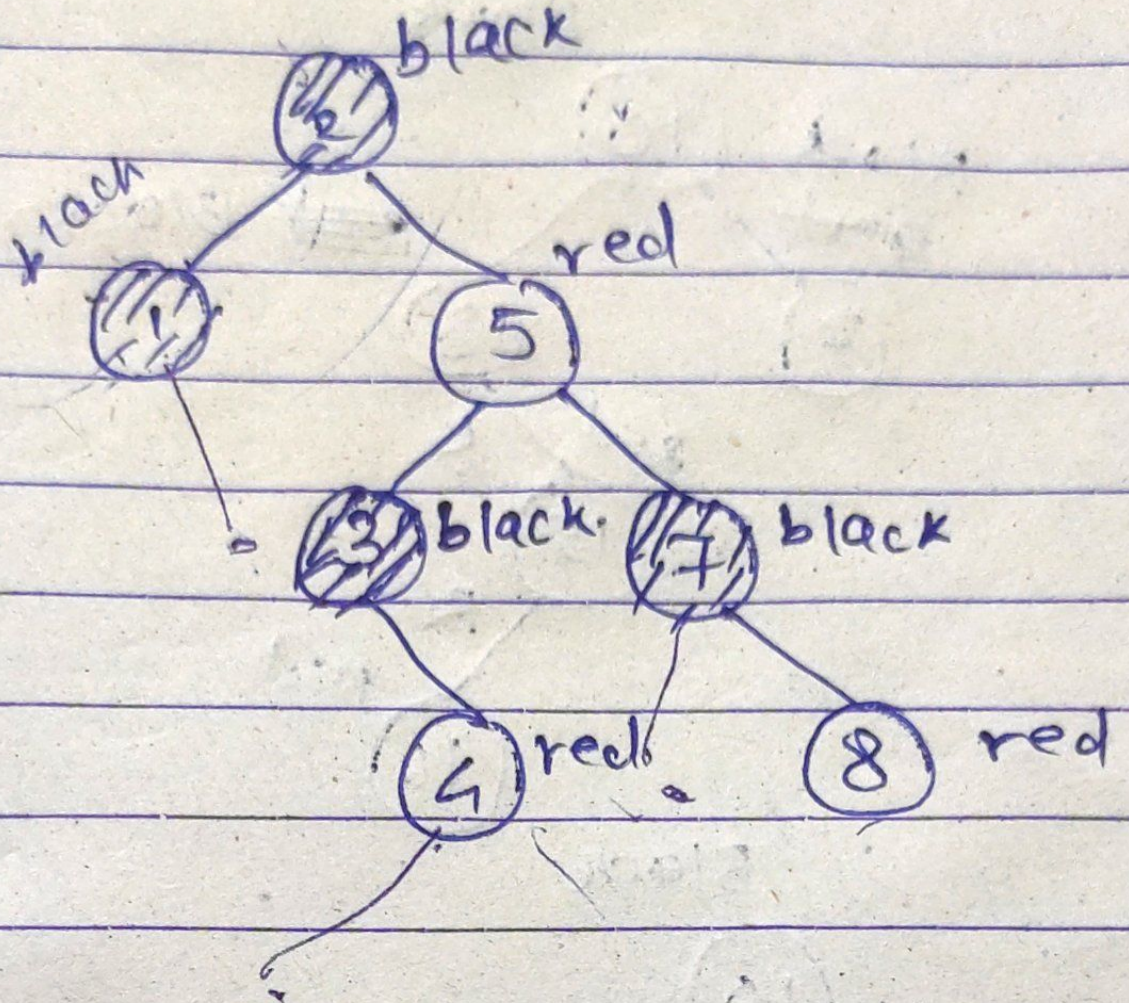
- The left child may not be red.

* Advantages:

1) It simplifies deletion by removing complex cases.

2) The AA tree eliminates half the restructing cases.

e.g.



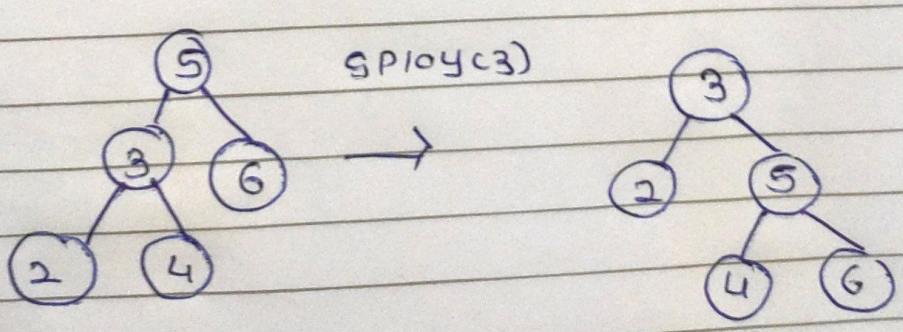
ખાસ કરી operation

* Splay Tree :-

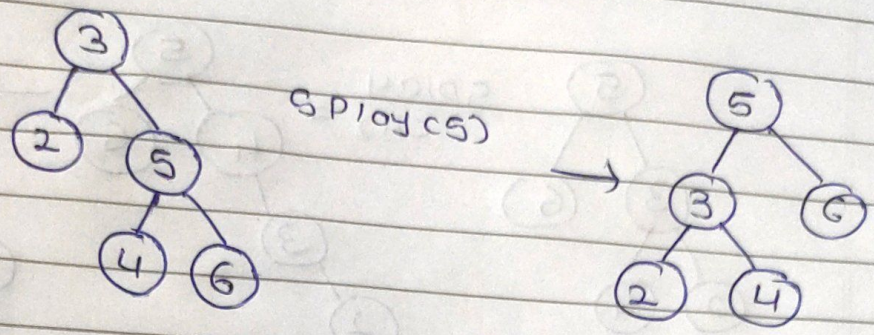
- Splay tree is another variant of binary search tree.
- The time complexity of binary search tree in average case is O(log n) and in worst case O(n).
- In splay tree splaying means rearranges all the element in tree so that splayed element is placed at root of tree.
- Because of splaying, we bring most frequenty used element closer to root of tree.
- whenever element is accessed the splaying operation is performed to move it to root.
- Time complexity of splay tree is $O(\log_2 n)$

* Rotation in splay tree :-

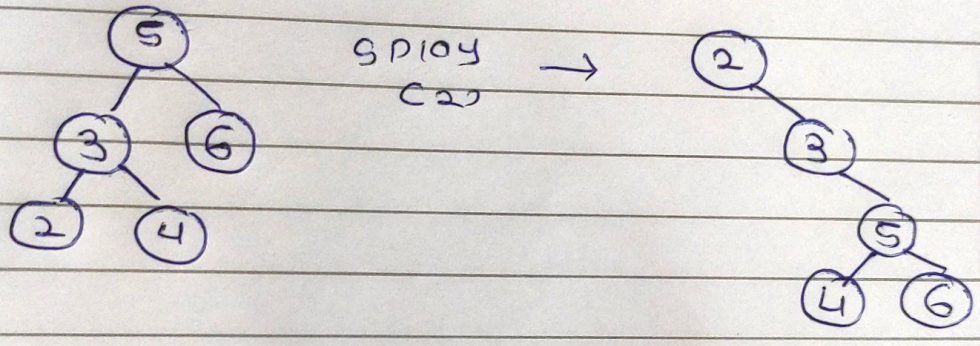
① Zig Rotation (Single Right Rotation)
- every node moves one position to right from it's current position



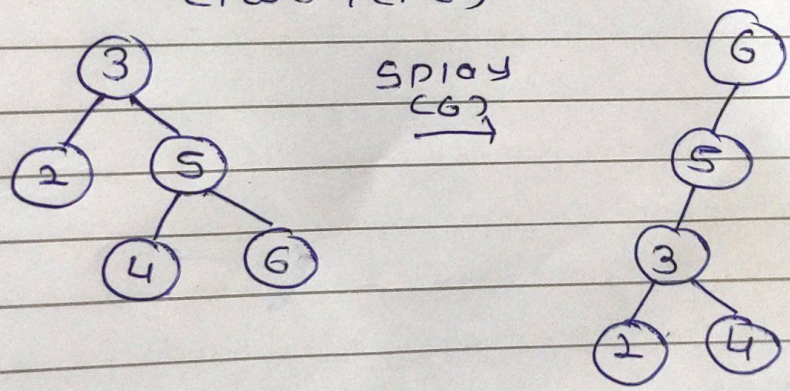
(2) Zag Rotation:-
(Single left Rotation)



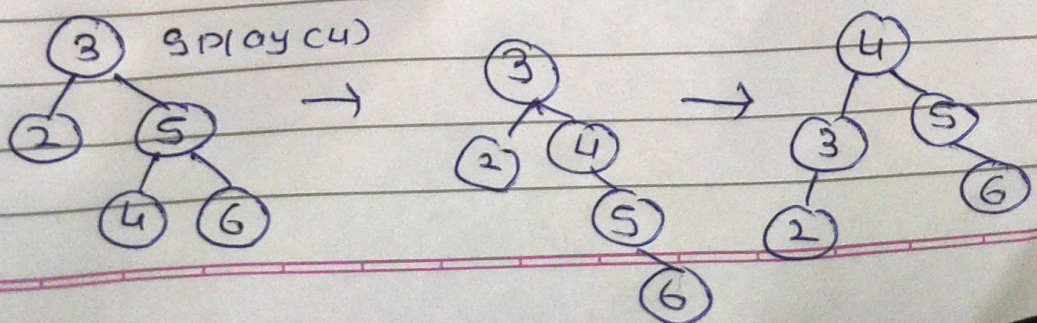
(3) Zig-Zig Rotatation
(Two right Rotation)



(4) ~~Zig~~ - Zag Rotation:-
Zag (Two left)



(5) Zig-Zag Rotation
(Right-left)



6) Zig-Zig Rotation:
(Left, Right)