efficiency is log n

AVL tree (Adelson / Velski / Landis) (Balanced BST)

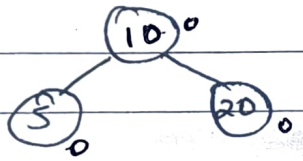
→ In a binary search tree to perform any operation the worst case efficiency (time taken) is order of n. O(n).

→ An AVL tree is a BST wherein in worst case to perform any operation we take O(log n) time.

AVL tree is a BST wherein the balance factor of each node is either 0, +1 or −1.

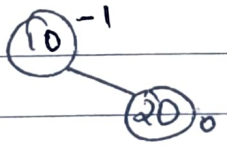
Balance factor = ht. of left sub tree − ht. of right sub tree

eg:



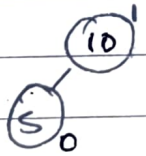
(5) Left subtree = 0
Right " = 0
Bal factor = 0

eg

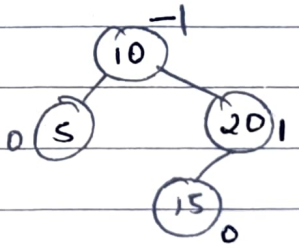


20
10 left = 1 right = 1
Bal = 1 − 1 = 0

eg



eg



∴ AVL.

→ If a BST is not an AVL tree, to convert a BST to an AVL tree we perform an operation called rotation.

→ There are 2 types of rotations:

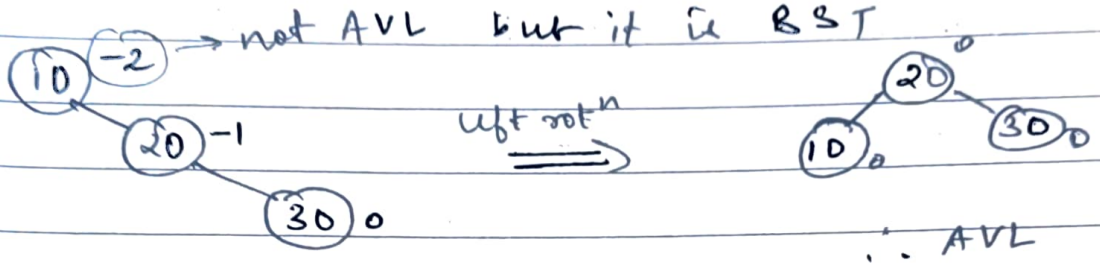
1. Single rotation
2. Double rotation

(1) Single rotation

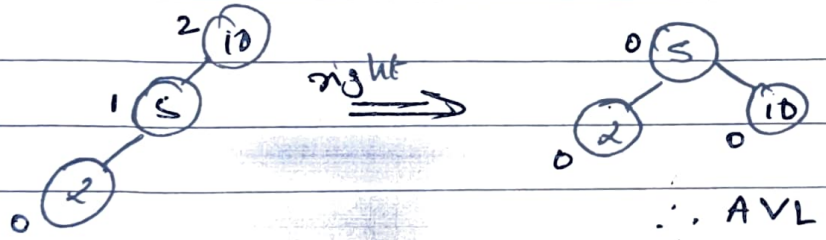
In a single rotation, there are 2 types:

1. Left rotation
2. Right rotation

Left rotation



Right rotation

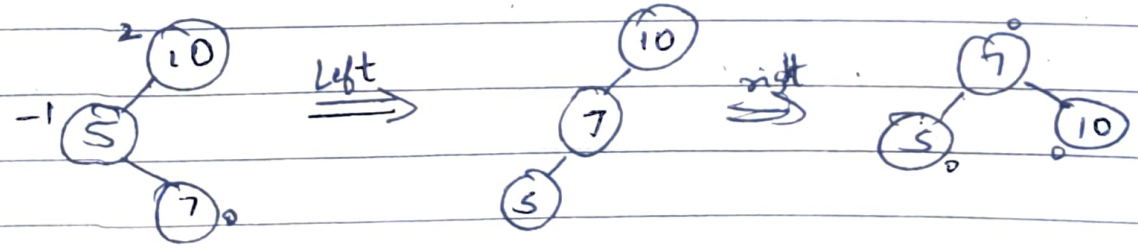


(2) Double Rotation

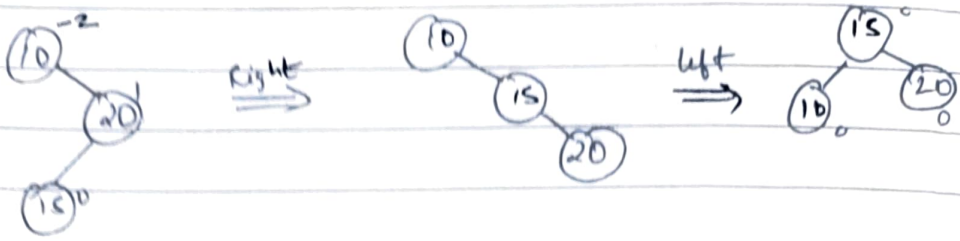
There are 2 types:

1. Left - right rotation (LR)
2. Right - left rotn (RL)

LR

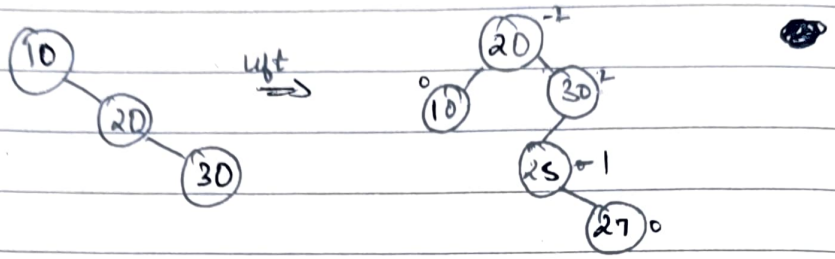


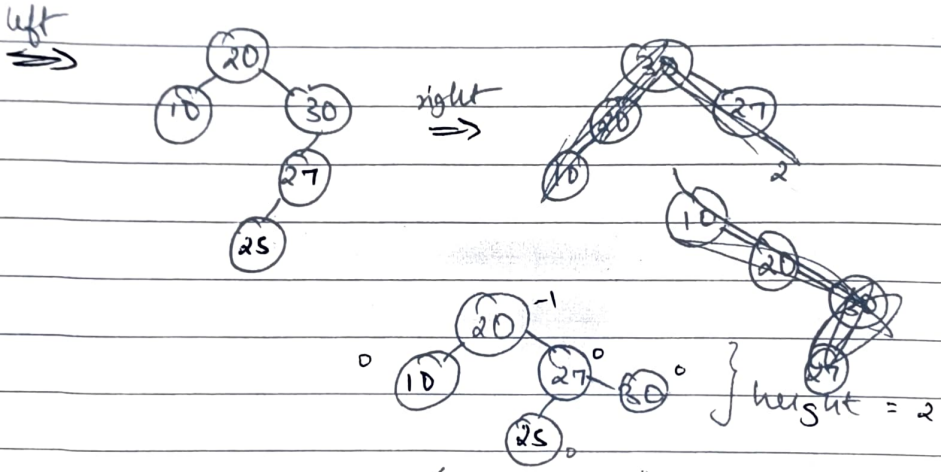
RL

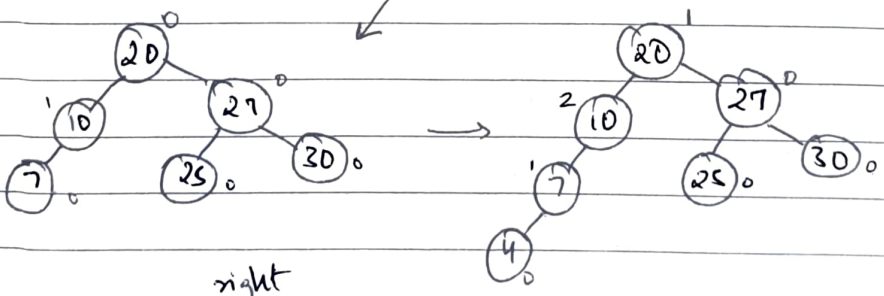


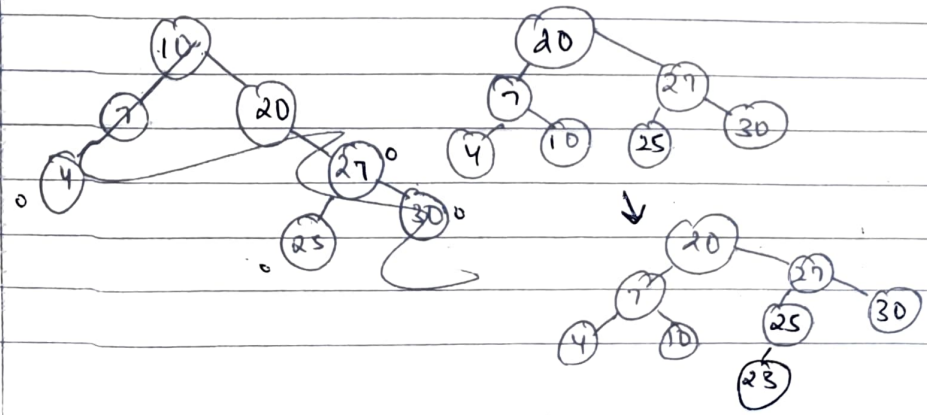
8. Construct an AVL tree for the elements

10 20 30 25 27 7 4 23

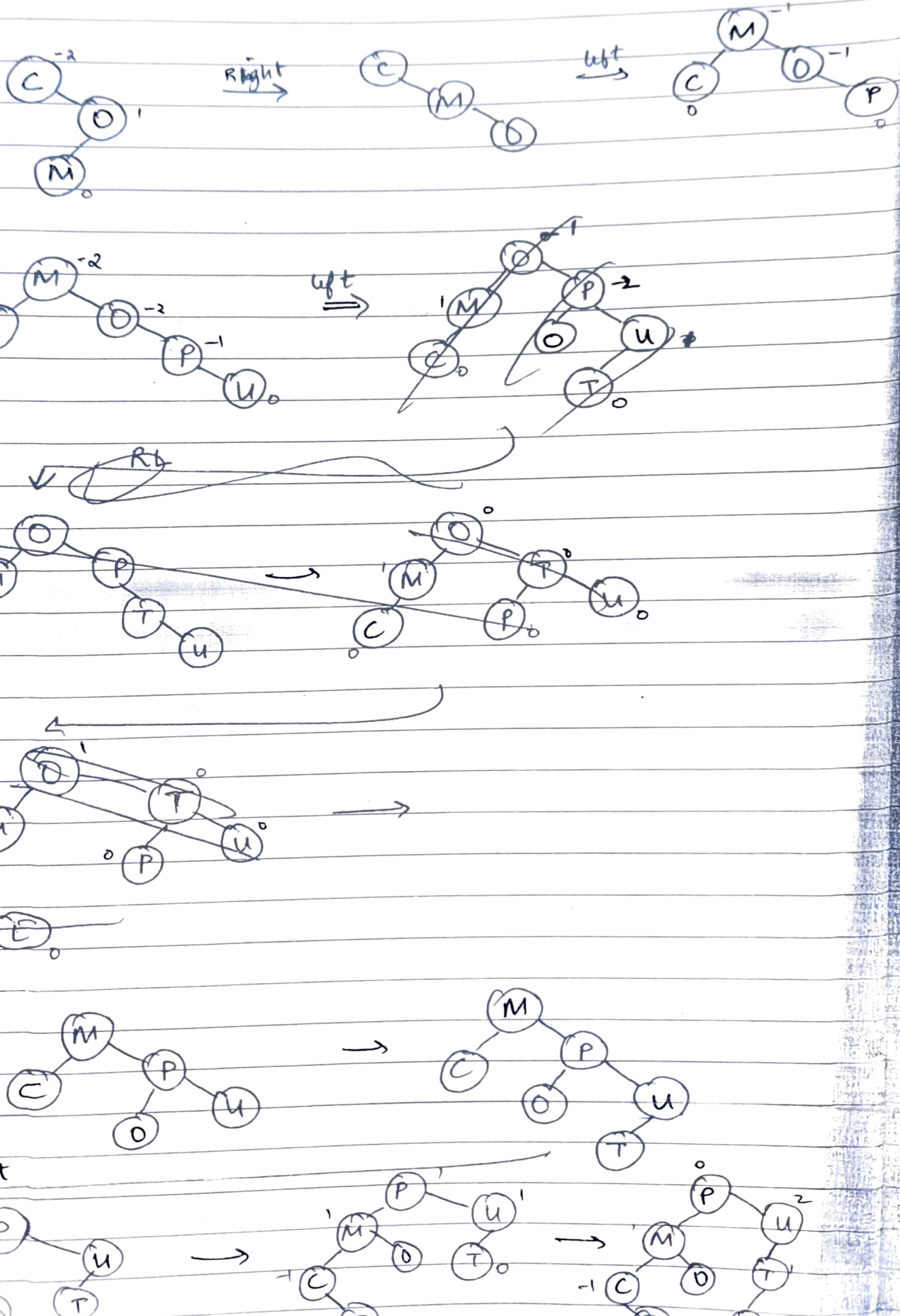




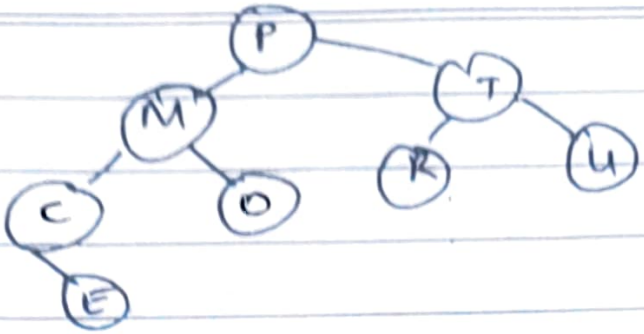




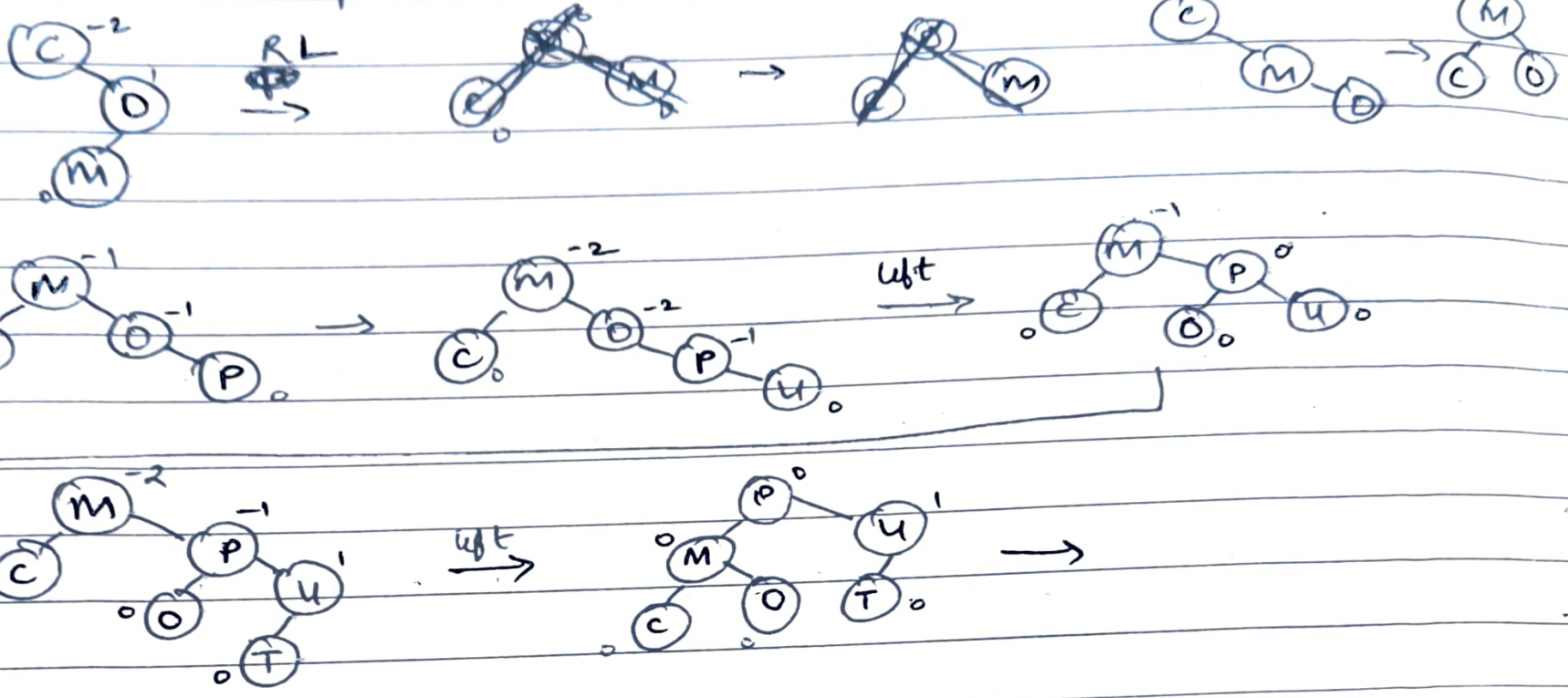
Q Create an AVL tree for characters of word COMPUTER



right →



8 COMPUTING



Splay tree

Splay tree is a self organizing BST with a property that recently accessed elements will be the root node.

Splay tree was invented by Daniel Sleator and Robert Tarjan.

Applications of splay tree are:

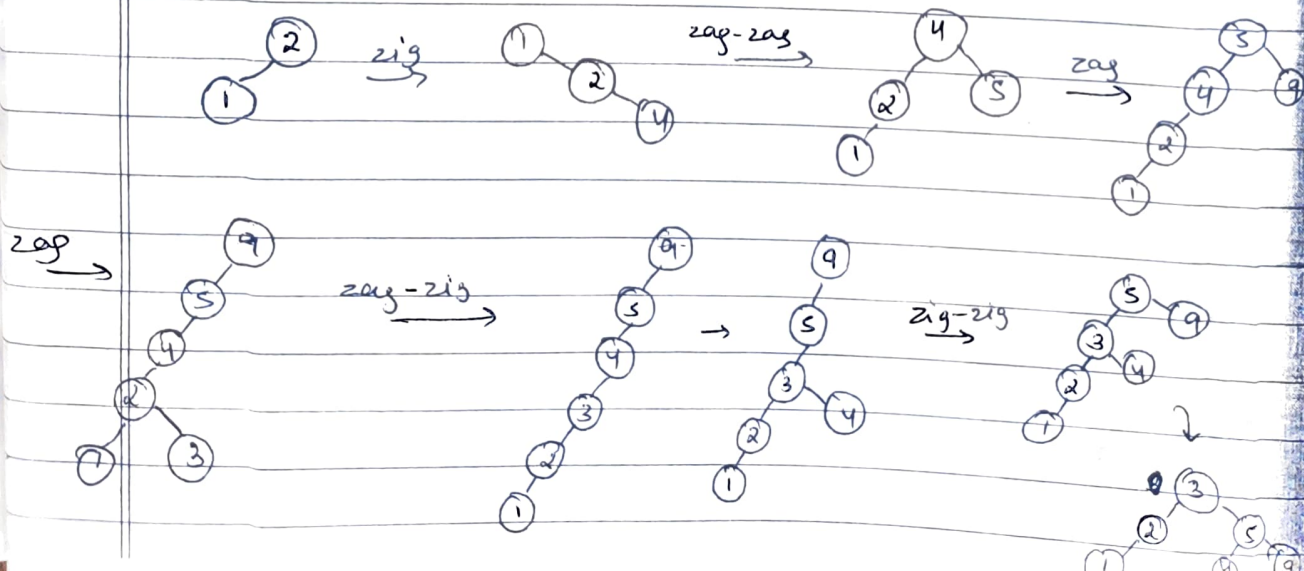
1. Cache memory
2. Virtual memory mgmt
3. Routing tables

To construct a splay we do the following rotations

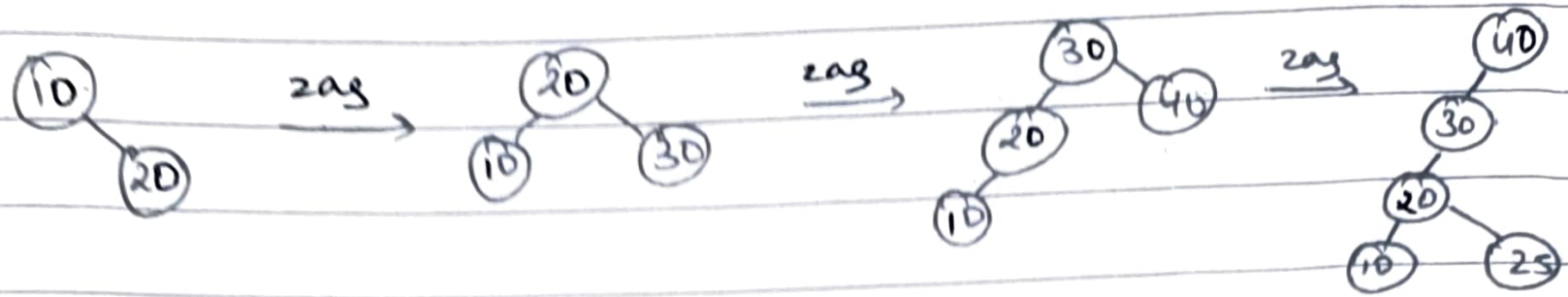
1. Zig (right rotn)
2. Zag (left)
3. Zig - Zag
4. Zag - Zig
5. Zig - Zig
6. Zag - Zag

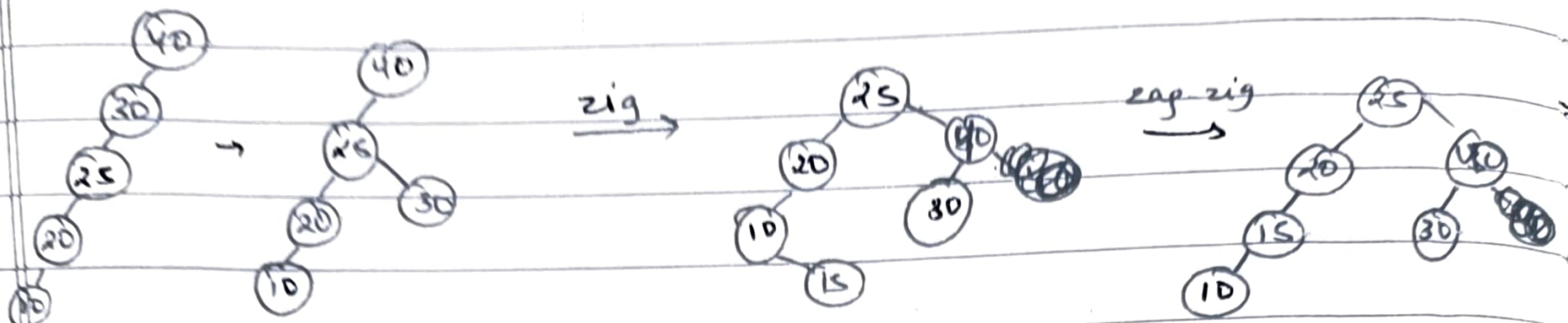
Q Construct a splay for the given elements:

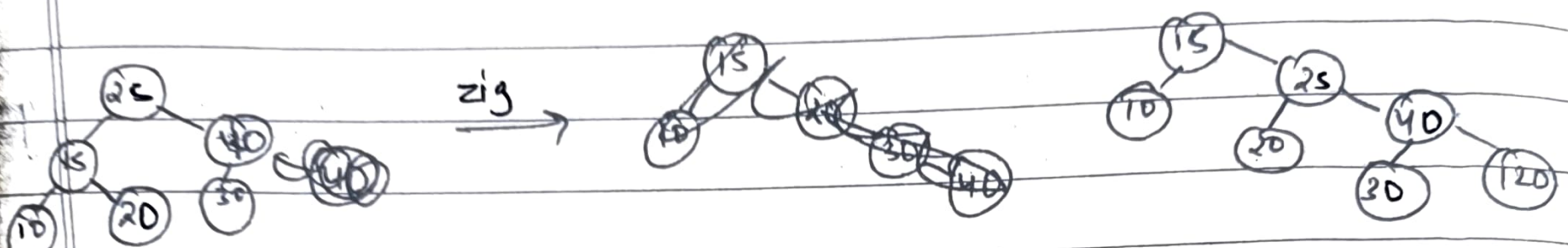
2 1 4 5 9 3



10 20 30 40 25 15 120







taken from word retrieval

TRIE - data structure to implement dictionary

The most popular data structure used to represent a set of strings.

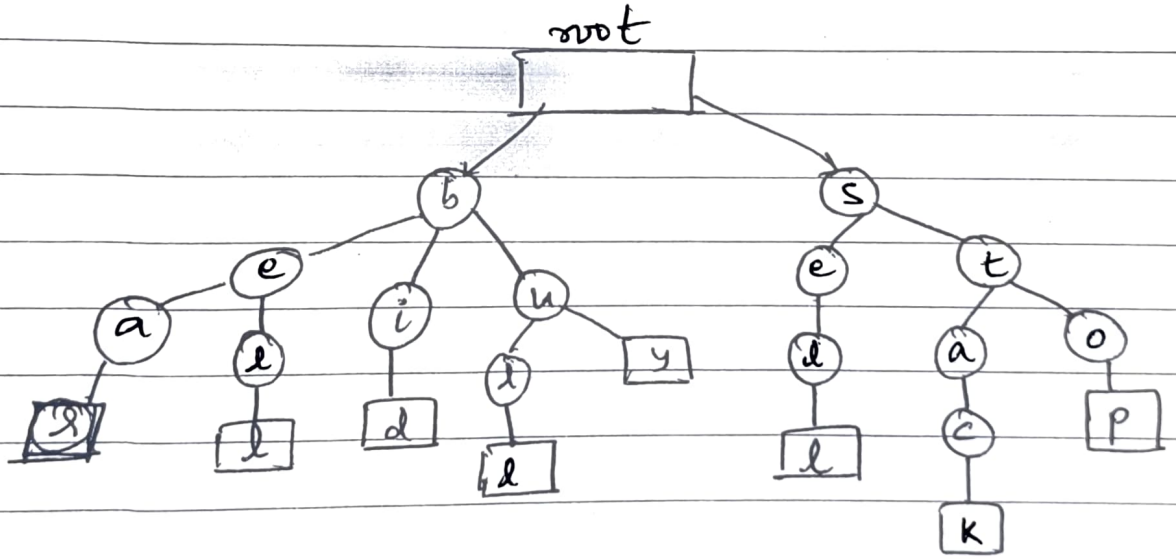
There are 2 types of TRIEs:

1. Standard TRIE
2. Compressed TRIE

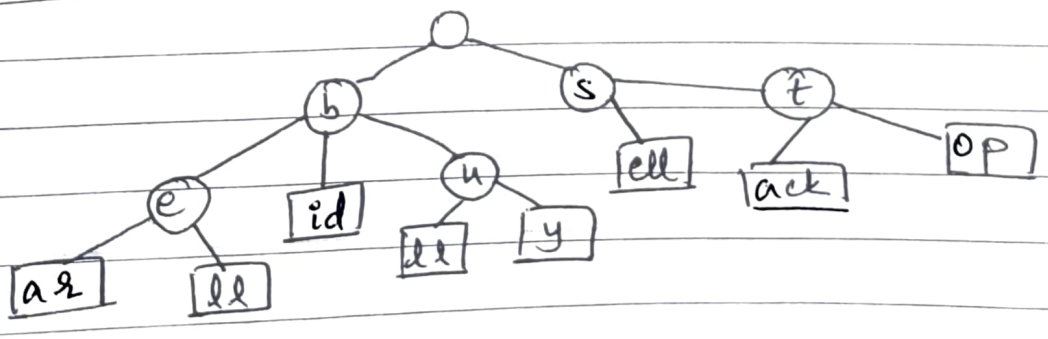
TRIE is a multiway search tree.

eg: TRIE for the words bear, bell, bid, bull, buy, sell, stack, stop.

Standard TRIE.



Compressed trie



Hashing

Hashing is a process of mapping huge amount of data into a smaller (table) with the help of a hash function.
The table which stores the data is called hash table.

To convert the data into a hash value we use the hash function. If h is the hash function and x is the data then the hash function can be denoted as $h(x)$.
A simple hash function can be represented as

$h(x) = x \bmod 10$ where 10 is the size of the hash table.

54 63 25 78 11

$54 \bmod 10 = 4$
$63 \bmod 10 = 3$

0	
1	11
2	
3	63
4	54
5	25
6	
7	
8	78
9	

If x_1 and x_2 are the data to be stored and if $h(x_1) = h(x_2)$
↓
"collision"

eg: $h(x) = x \bmod 10$

$x_1 = 77$ $\quad h(x_1) = 7$

$x_2 = 27$ $\quad h(x_2) = 7$

Collision can be avoided using 2 techniques:

1. open addressing
2. separate chaining.

Open addressing

(1) Linear probing (2) Quadratic probing (3) Double hashing

~~$h(x) = x \bmod 10$~~

(1) Linear probing

search for next free slot

$h(x) = x \bmod 10$

0	19
1	28
2	39
3	
4	
5	55
6	75
7	
8	78
9	89

78
89
19
28
39
55
75

$i = 0, 1, 2, 3, 4$

$h(2) = (x + i) \bmod \text{SIZE}$ (10)

$0 \rightarrow 19$
$1 \rightarrow 28$
$2 \rightarrow 39$
$3 \rightarrow 0$

Quadratic probing

SIZE-1

$i = 0, 1, 2 \ldots n-1$

$h_i(x) = (h(x) + i^2) \bmod 10$

$h(x) = x \bmod 10$

78
89
19
28
39

0	19
1	
2	28
3	39
4	
5	
6	
7	
8	78
9	89

~~$h_0(78) = 8 \bmod 10 = 8$~~

~~$h_1(89) = (9+1) \bmod 10 = 0$~~

19 collides

so (1) $(9+1) \bmod 10 = 0$

28 collides

(1) ~~$(8+1) \bmod 10 = 9$~~ ✗

$(8+4) \bmod 10 = 2$

39 collides

(1) $9+1 \bmod 10 = 0$ ✗

$(9+4) \bmod 10 = 3$

(3) Double hashing

We use 2 hash functions

$h(x) = x \bmod 10 + (x+5) \bmod 10$

Separate chaining - uses concept of L.L to avoid collision.

$h(x) = x \bmod 10$

78 89 19 28 39 54 63 70 44