

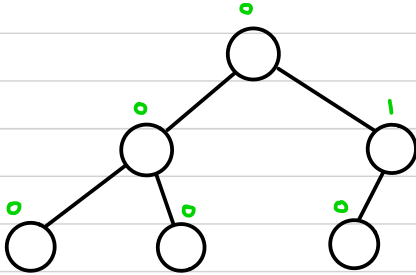


AVL TREES

No of edges

Height Balanced Binary Search Trees

Balance factor = Height of left subtree - Height of right subtree
 $\{-1, 0, 1\} \rightarrow$ Balanced Tree

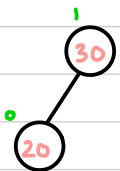


Balanced Tree

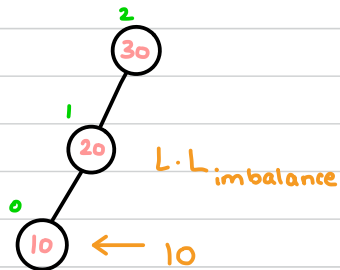
Rotations for insertion in AVL trees

- (1) LL Rotation } Single Rotation
- (2) RR Rotation }
- (3) LR Rotation } Double Rotation
- (4) RL Rotation }

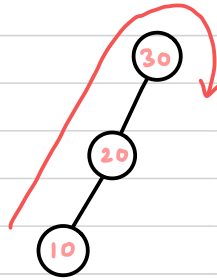
INITIAL



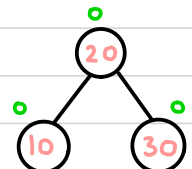
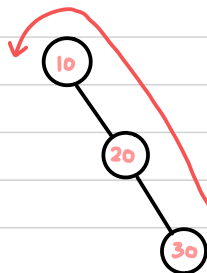
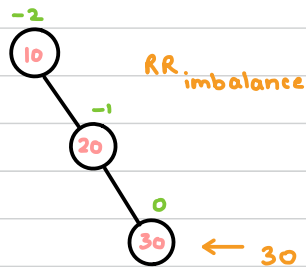
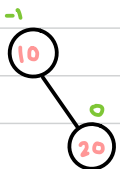
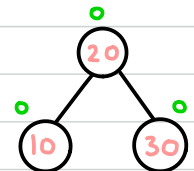
AFTER INSERTION

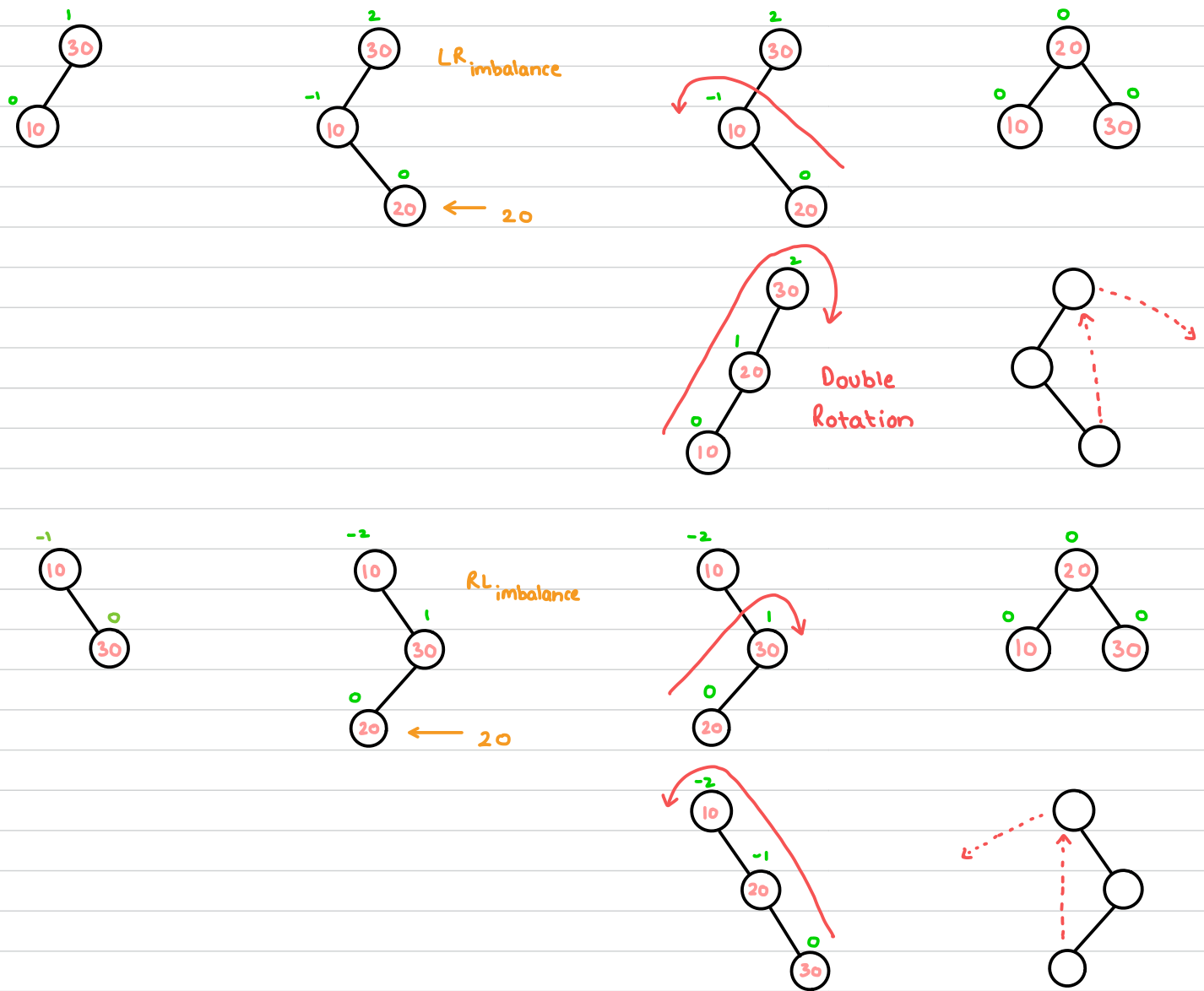


PERFORM ROTATION

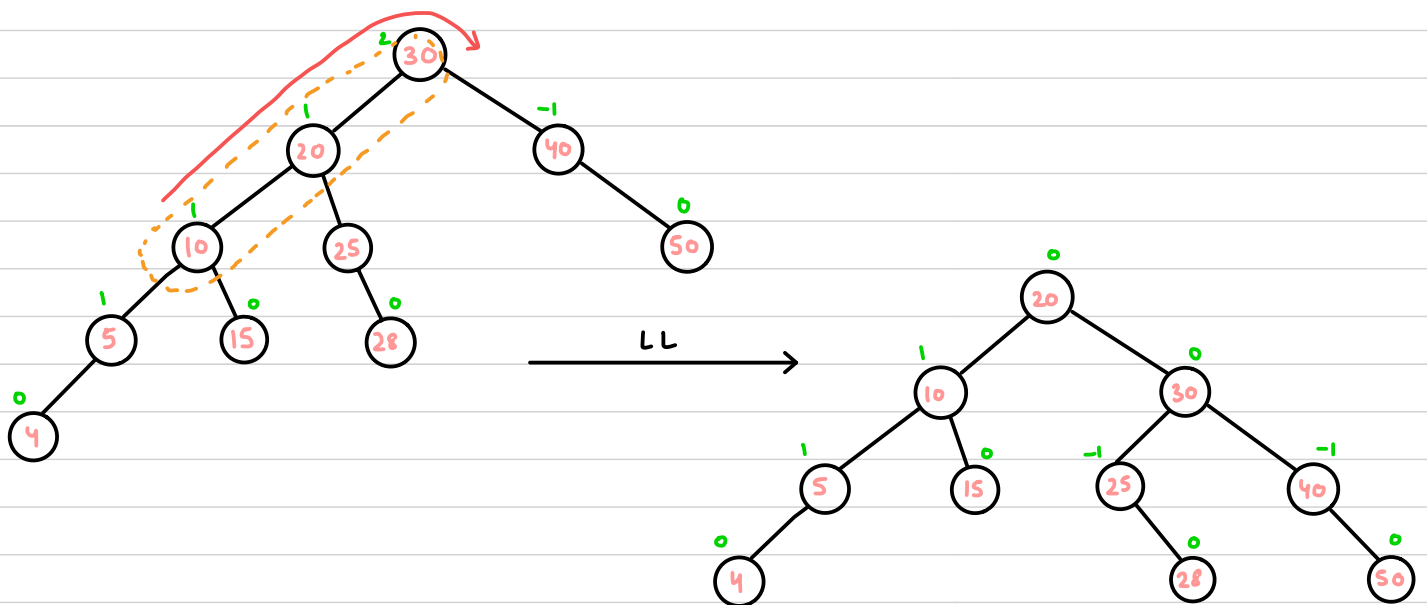


AFTER ROTATION

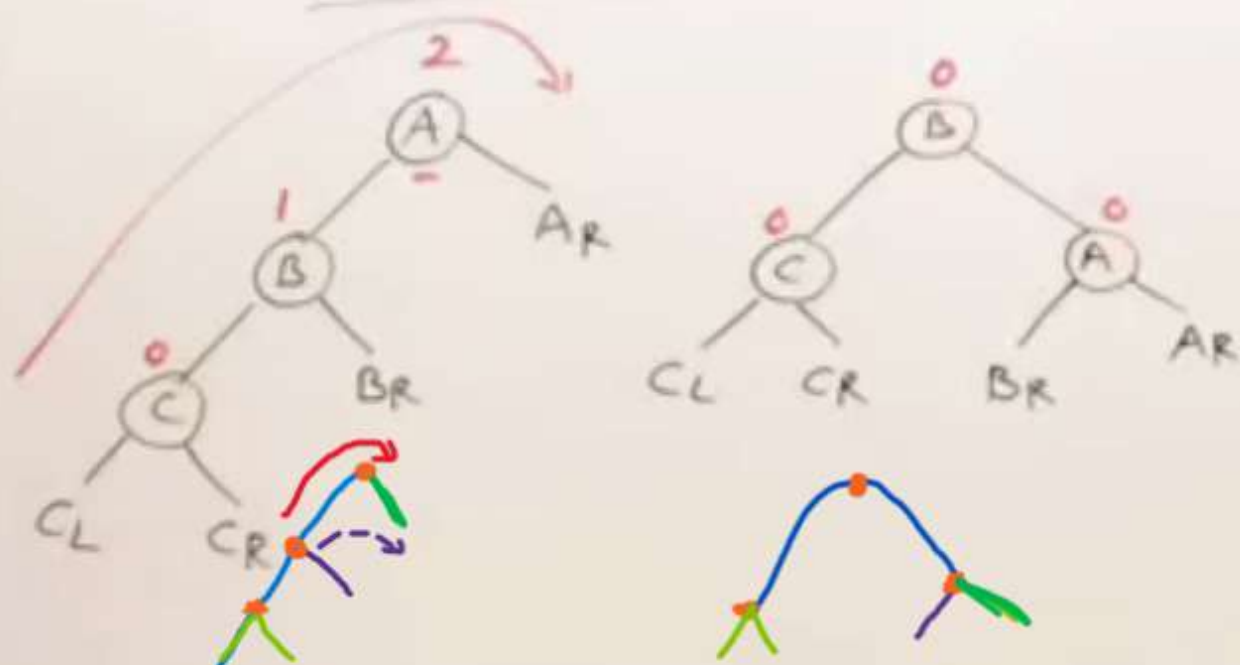




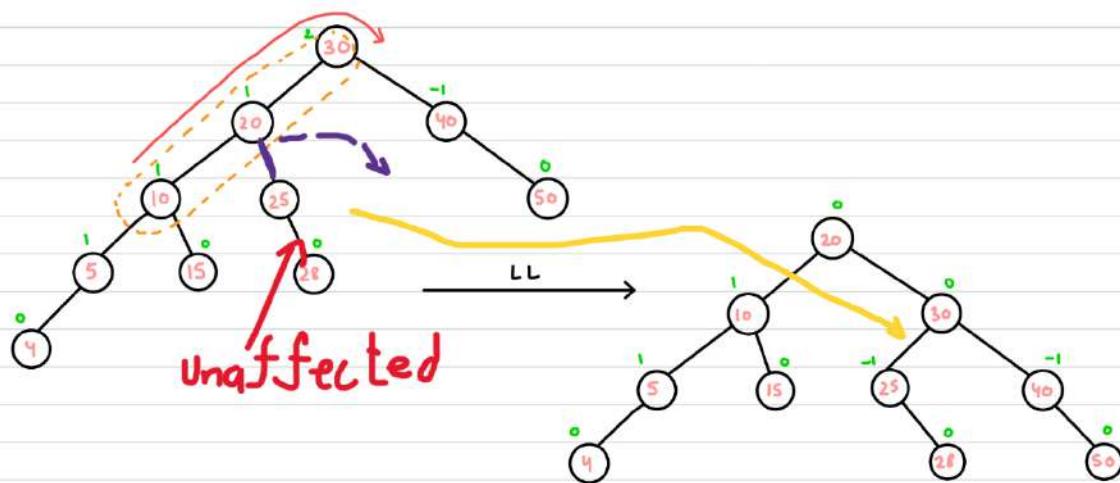
FORMULA OF ROTATION FOR INSERTION



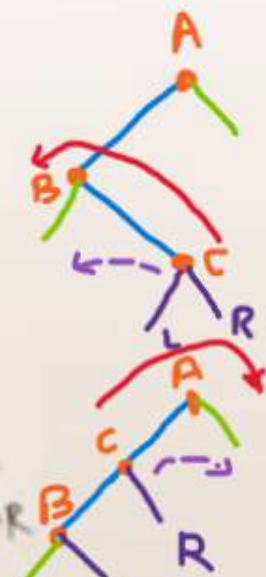
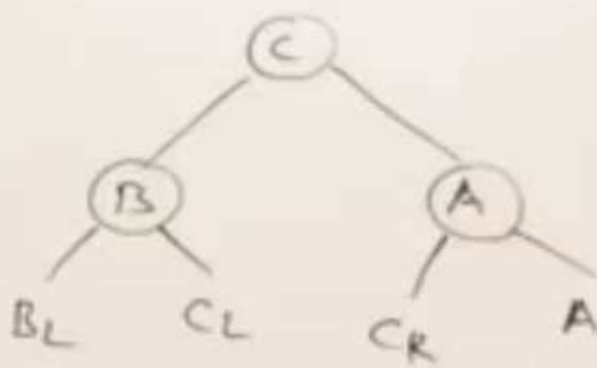
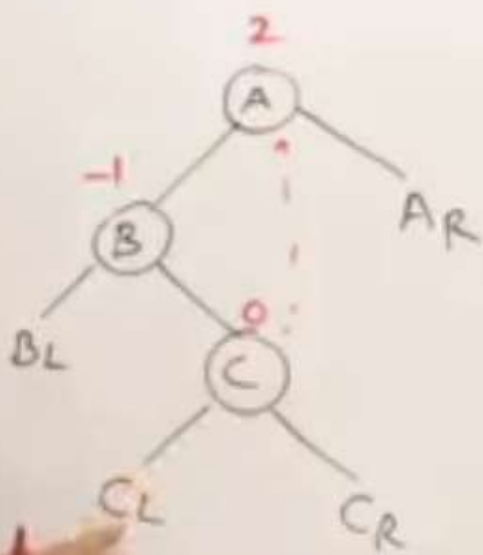
LL - Rotation



FORMULA OF ROTATION FOR INSERTION



LR - Rotation



similar to last one
now

PROGRAM FOR LL ROTATION

```

if (BalanceFactor(p) == 2 && BalanceFactor(p->lchild) == 1)
    return LLRotation(p);
else if (BalanceFactor(p) == 2 && BalanceFactor(p->rchild) == -1)
    return LRRotation(p);
else if (BalanceFactor(p) == -2 && BalanceFactor(p->lchild) == -1)
    return RRRotation(p);
else if (BalanceFactor(p) == -2 && BalanceFactor(p->rchild) == 1)
    return RLRotation(p);

return p;
}

```

```

int NodeHeight(struct Node *p)
{
    int hl, hr; // height of left subtree (HL), height of right subtree (HR)
    hl = p && p->lchild ? p->lchild->height : 0;
    hr = p && p->rchild ? p->rchild->height : 0;
    // NOT NULL
    return hl > hr ? hl + 1 : hr + 1;
}

```

```

int BalanceFactor(struct Node *p)
{
    int hl, hr;
    hl = p && p->lchild ? p->lchild->height : 0;
    hr = p && p->rchild ? p->rchild->height : 0;

    return hl - hr;
}

```

```

struct Node * LLRotation(struct Node *p)
{

```

```

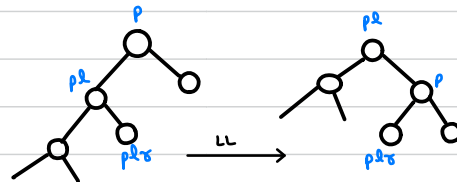
    struct Node *pl = p->lchild;
    struct Node *plr = pl->rchild;

```

```

    pl->rchild = p;
    p->lchild = pl;
    p->height = NodeHeight(p);
    pl->height = NodeHeight(pl);

```




```

if (root == p) // If rotation was performed on
               root node, root needs to be
               updated.
    root = pl;

```

```

return pl;

```

```

}

Struct Node* LRRotation(struct Node* p)
{

```

```

    Struct Node *pl = p->lchild;
    Struct Node *plr = pl->rchild;

```

```

    pl->rchild = plr->lchild;
    p->lchild = plr->rchild;

```

```

    plr->lchild = pl;
    plr->rchild = p;

```

```

    pl->height = NodeHeight(pl);
    p->height = NodeHeight(p);
    plr->height = NodeHeight(plr);

```

```

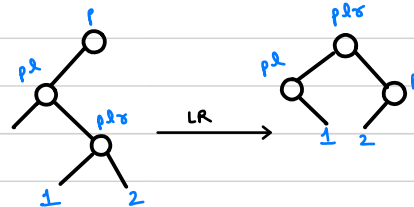
    if (root == p)
        root = plr;
    return plr; // New root;

```

```

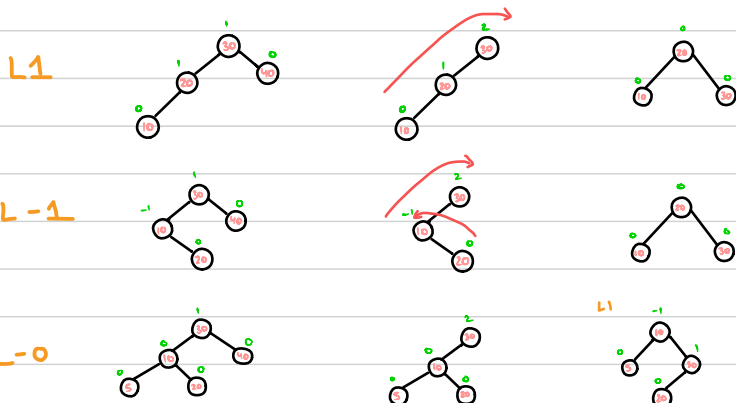
}

```



DELETION FROM AVL TREES WITH ROTATION

- | | |
|------------------|------------------|
| 1. L 1 Rotation | 4. R 1 Rotation |
| 2. L -1 Rotation | 5. R -1 Rotation |
| 3. L 0 Rotation | 6. R 0 Rotation |



// Other three will be mirror images

HEIGHT VS NODES OF AVL TREES

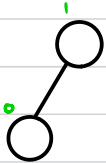
If height is given:

- Max nodes $n = 2^h - 1$ // Not $2^{h+1} - 1$ because height is starting from 1.
- Min nodes $n =$ Look in table

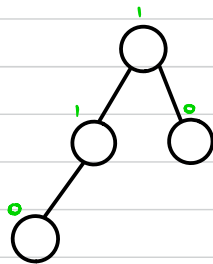
$h=1$
 $n=1$



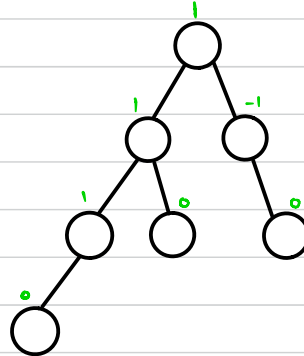
$h=2$
 $n=2$



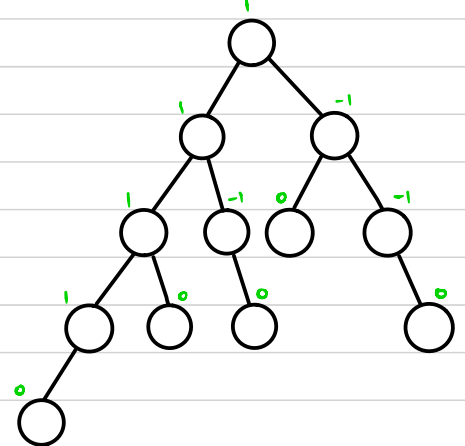
$h=3$
 $n=4$



$h=4$
 $n=7$



$h=5$
 $n=12$



h	1	2	3	4	5	6	7
n	1	2	4	7	12	20	33

$\rightarrow + \quad +1 \quad \rightarrow$

$$N(h) = \begin{cases} 0 & 0 \\ 1 & 1 \\ N(h-2) + N(h-1) + 1 & \text{otherwise} \end{cases}$$

// formula same as Fibonacci series

\rightarrow balanced series

If 'N' Nodes are given find:

- Min height = $\log_2(n+1)$
- Max height = Look in table

Foreg for 13 nodes $h=5$ // Look from node towards height
19 nodes $h=5$