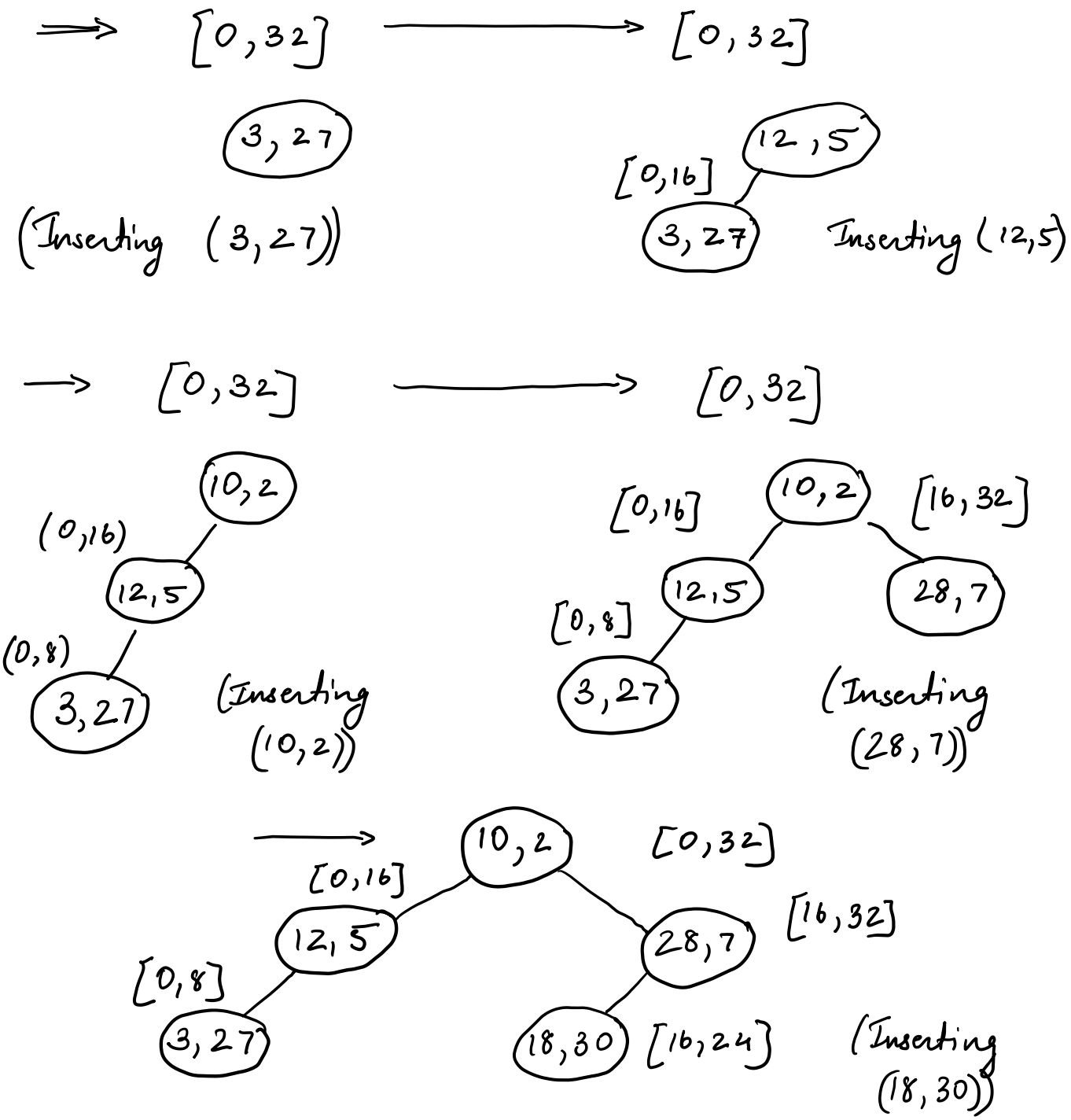
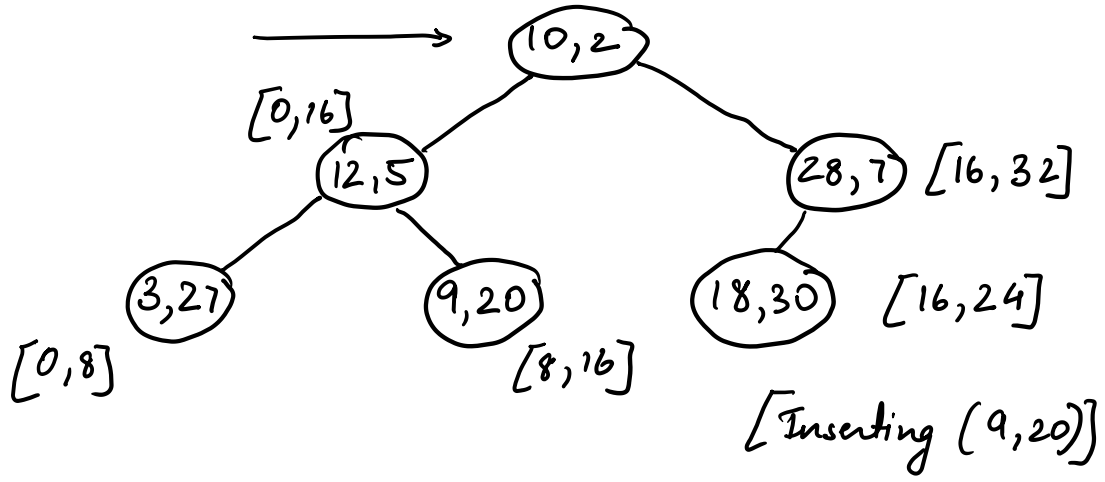
ANOL KURIAN VADAKKEPARAMPIL
5626-8544

DS : ASSIGNMENT 2.

Q1. Insert operations into initially empty RPST in sequence with keys

(3,27) , (12,5) , (10,2) , (28,7) , (18,30) , (9,20)

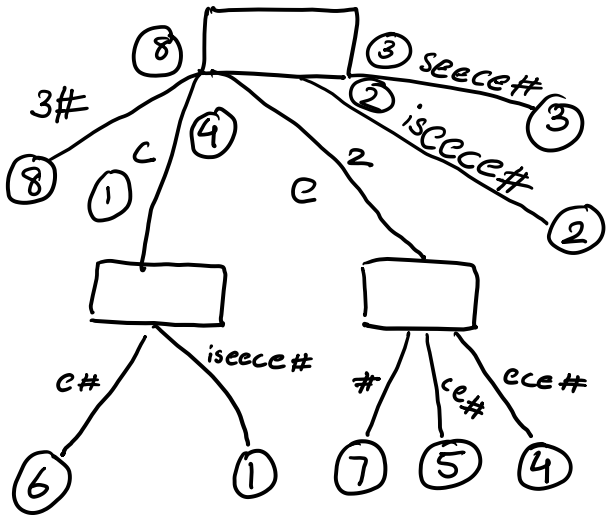




Q2. string ciseece#

c i s e e c e #
1 2 3 4 5 6 7 8

ciseece#	1
iseece#	2
seece#	3
eece#	4
ece#	5
ce#	6
e#	7
#	8



- There are total 8 keys
- The element nodes store index values of starting points of the suffixes they represent
- The uncircled digits on the edges represent branching nodes (3 for this case)

Q.3 To insert an interval with integer endpoints $[0, k]$

- First check if the root is empty or not. If the root is empty we assign the interval to the root

- If the root is not empty then we begin given the root & check at each node if the range of interval to be inserted is contained in the range of interval of the node

- If the interval is contained in the range insert at that node & we stop traversing farther down that specific path of the tree & return to the parent

- If the interval range is not within the interval range of the node, we check if there is an overlap between the two ranges

- If there is an overlap, we go in the direction of the left subtree & repeat from step 3

- We repeat the same process from step 2 for the right subtree

To insert into the tree we use the following algorithm

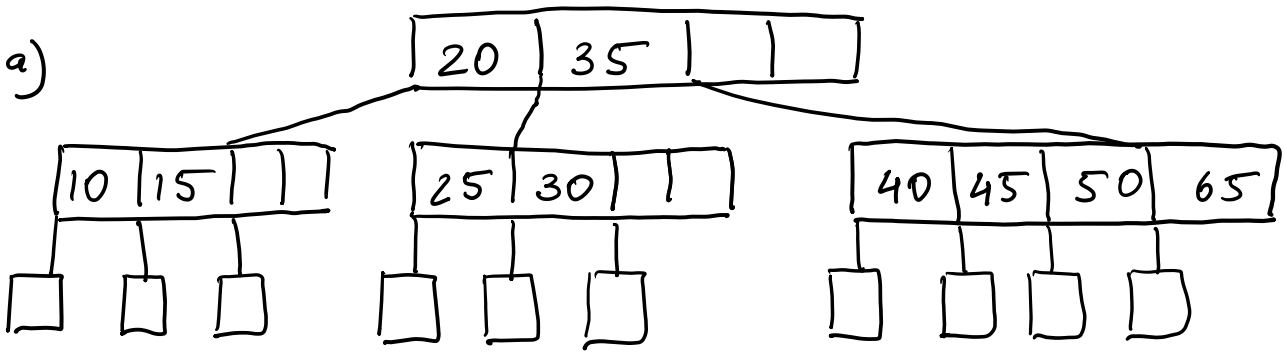
```
insert (s,e,r) {

    //s ⟶ start of interval
    //e ⟶ end of interval
    //r ⟶ root of subtree

if (s <= s(r) && e(r) <= e)

    add [s,e] to r ;))

        interval spans node range

else {
    if [s < s(r) + e(r)) (2)

        insert (s,e, r . left child) ;
    if (e > ( s(r) + e(r) /2)

        insert (s,e, r . right child) ;
```

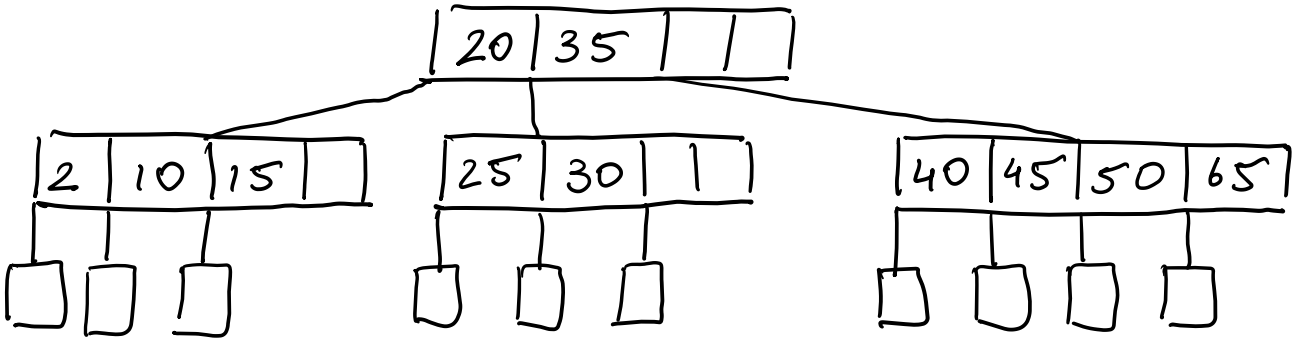
The time complexity of the insert algorithm is $O(\log n)$

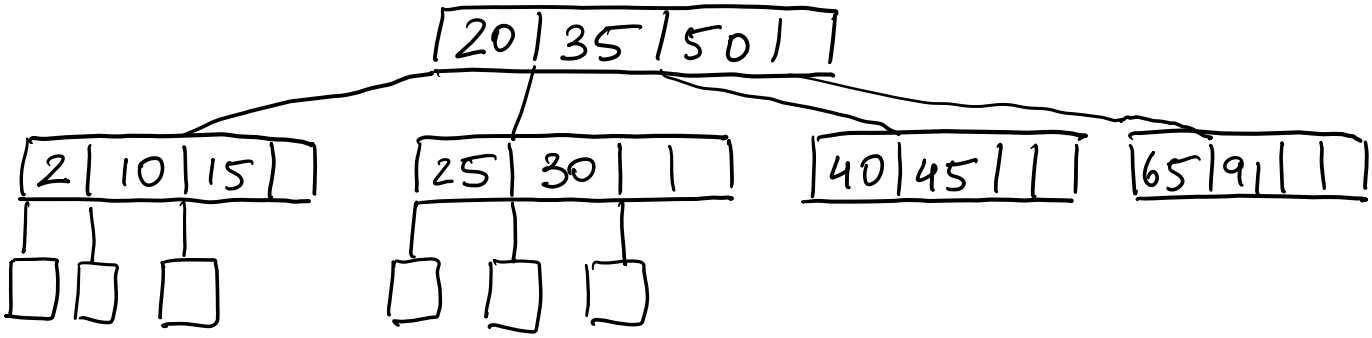
- The worst case occurs in a segment tree what the leftmost inserted interval node, L is the right child of its parent & the rightmost inserted node R is the left child of its parents.

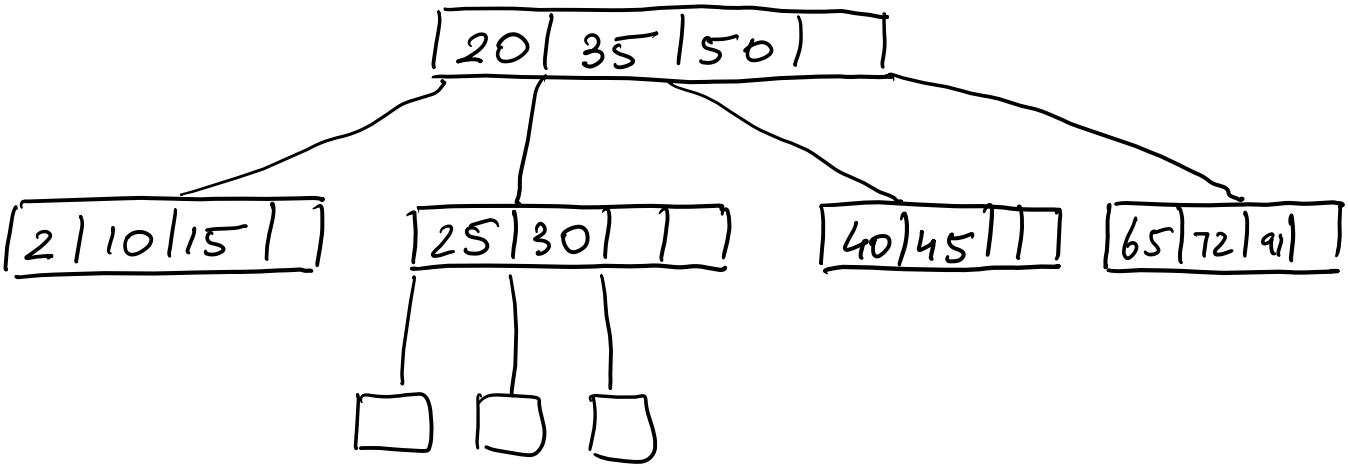
- In this case both L, R & all ancestors of L, R & the other children of each of these ancestors, reached, since the number of ancestor is $O(\log n)$ visited from root is each order the complexity for insert operation

Q4. a)




20 35
2 10 15
25 30
40 45 50 65


20 35 50
2 10 15
25 30
40 45
65 91


20 35 50
2 10 15
25 30
40 45
65 72 91

b)

Assumption : we have enough memory to hold all h nodes accessed on the way down

$h = 2$ (height of tree)

Disk access needed for each insertion

insert	65	: 3	(2 read, 1 write)
insert	2	: 3	(2 read, 1 write)
insert	91	: 5	(2 read, 3 write)
insert	72	: 3	(read, 1 write)