Binary Search Trees

1. **Binary trees.** Each node has 2 children which can be empty. It has also a parent which also can be empty.

A node without any children (both empty) is leaf
Other nodes are internal
The node without any parent is root.

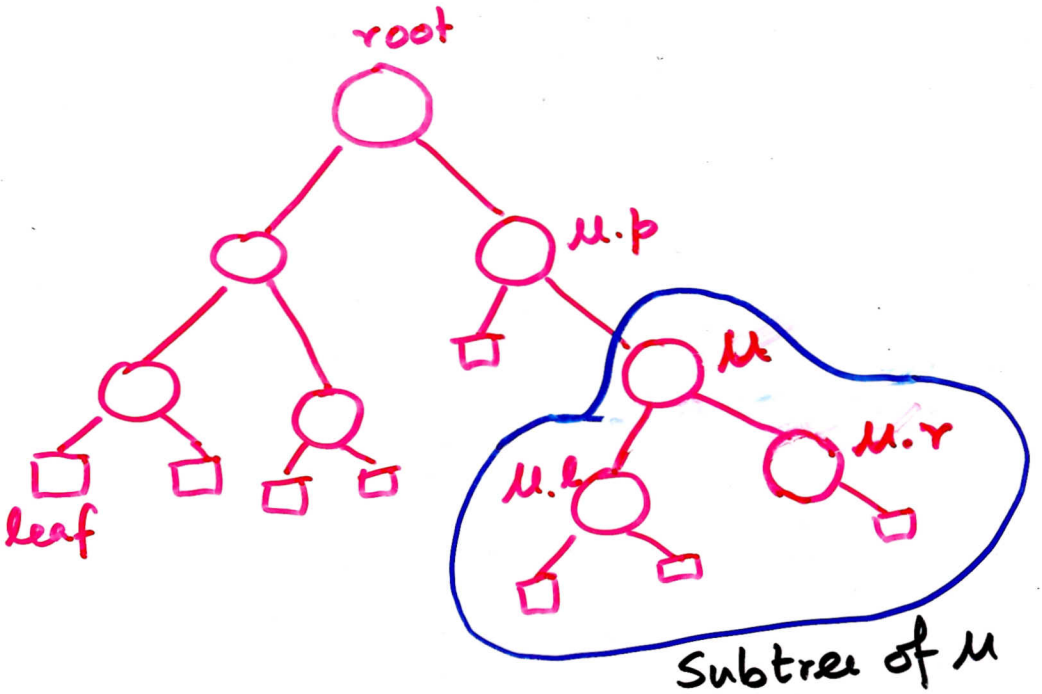
A pascal type representation:

```
type node = record p, l, r : ↑node
                   key : integer
            end

tree = ↑node
```

2. **More terminology.** y is a descendant of u if it is a child of u or a descendant of a child of u.

If v is descendant of u, then u is an ancestor of v

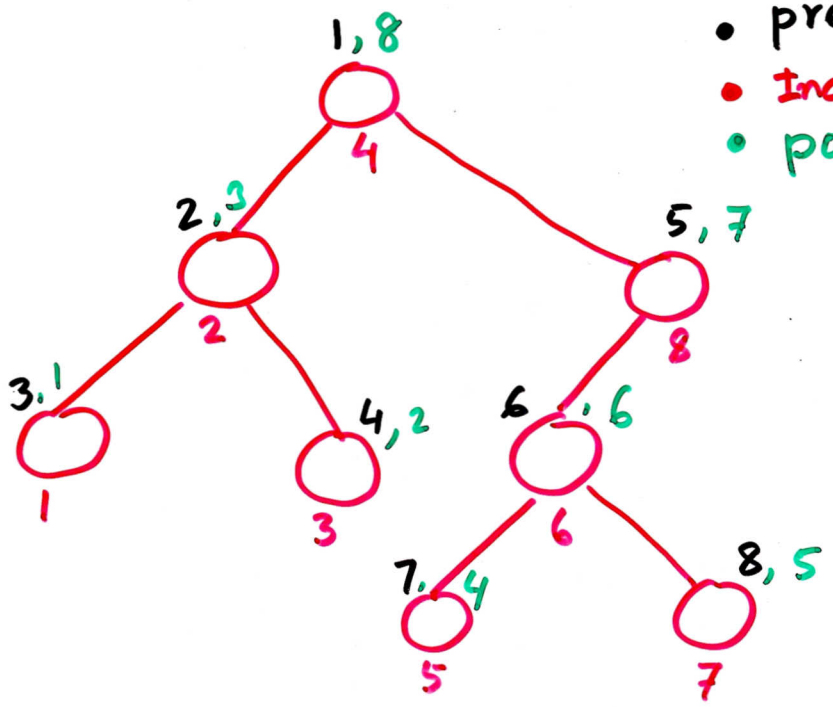
depth of u is # edges from root to u
height of u is max. # edges from u to a leaf.
subtree of u consists of u and its descendants.



3. **Traversals.** Three traversals are inorder, preorder and postorder.

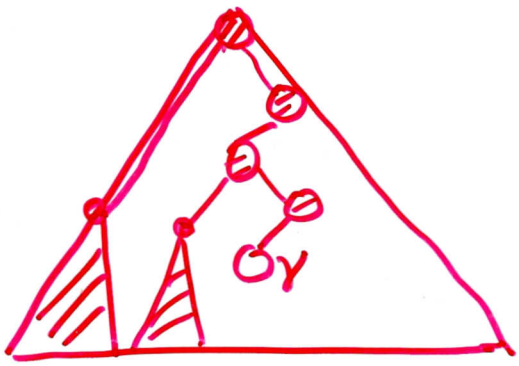
```
Procedure Inorder (u)
  if u ≠ nil then
                Inorder(u.l);
                  print (u.key);
                Inorder(u.r);
  endif

Procedure Preorder (u)
    if u ≠ nil then
                    print (u.key);
                    preorder (u.l);
                    preorder (u.r);
    endif
```

Postorder: first recurse left, then right, then print

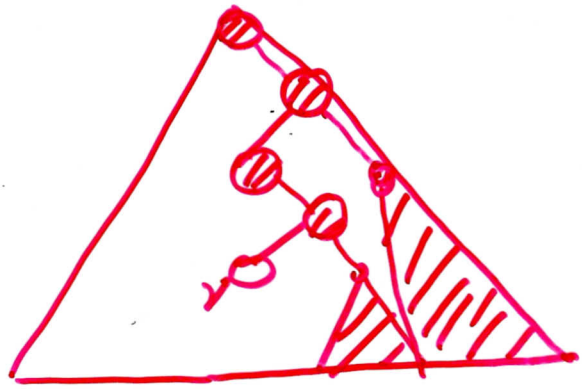


- preorder traversal
- Inorder traversal
- postorder traversal

property: u is ancestor of v iff
$pre(u) < pre(v)$ and
$post(u) > post(v)$



pre() < pre(v)

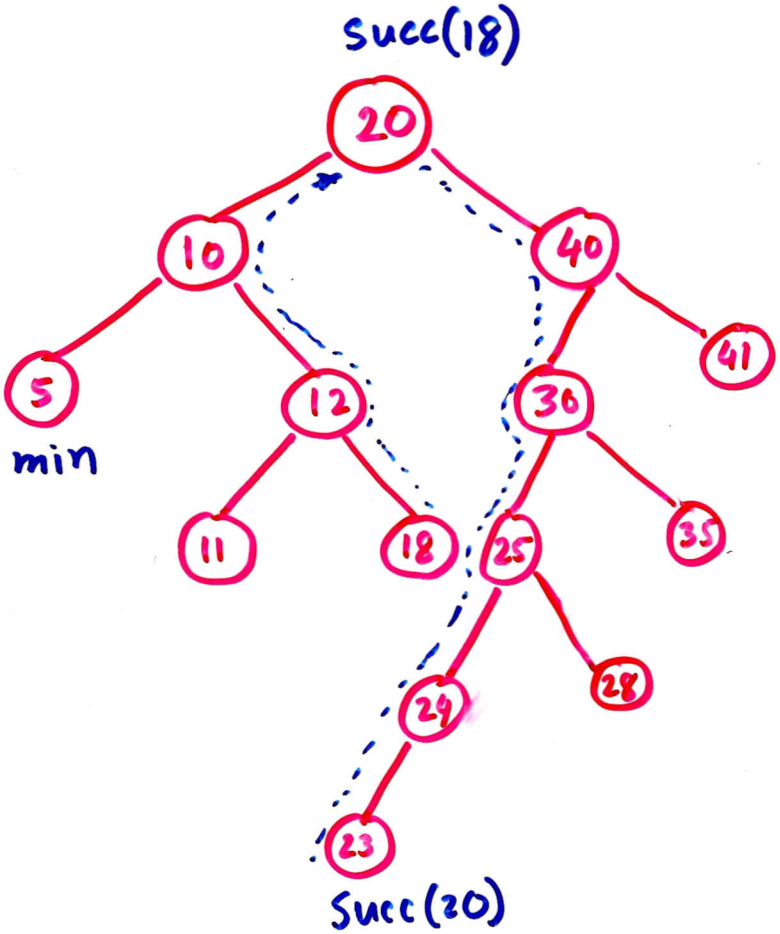


post() > post(v)

A binary tree is a binary search tree if the keys printed inorder is sorted.

4. Searching.

```
function Search(x; u)
  if u = nil or x = u.key then return u
  else
      if x < u.key then return Search(x, u.l)
          else return Search(x, u.r)
      endif
  endif
```



SEARCH.
Min.
SUCCESSOR
Insertion

All in O(h)

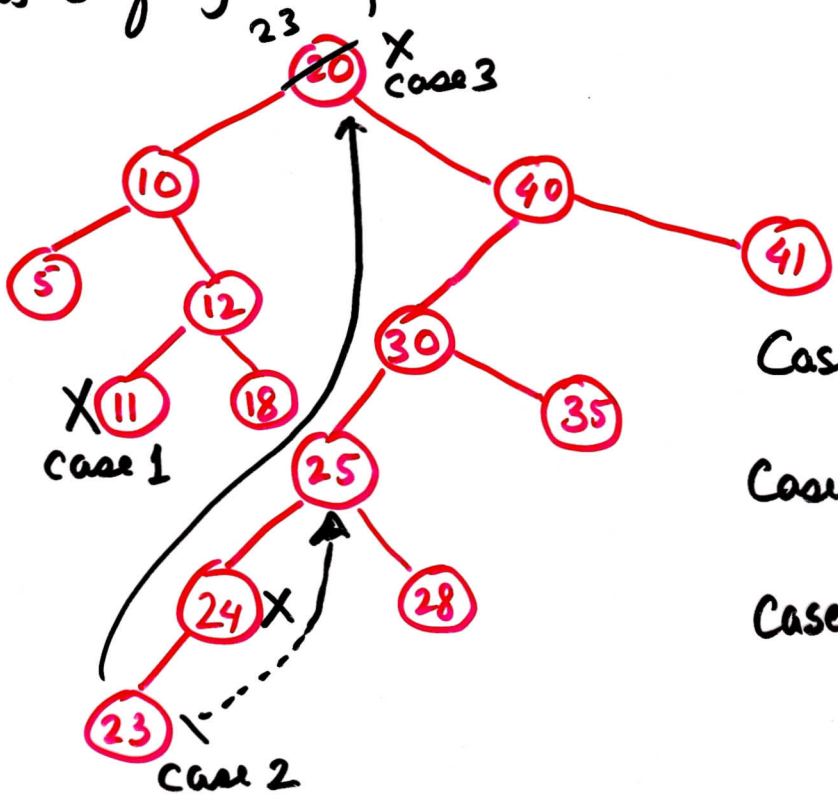
Write Successor and Min functions

5. Insertion and Deletion

```
procedure Insert(P, r)
    x := nil; u := P;
    while u ≠ nil do x := u;
        if r.key < u.key then u := u.l
        else u := u.r
        endif
    endwhile

    r.p := x;
    if x = nil then P := r
    else if r.key < x.key then
        x.l := r else x.r := r
    endif
    endif
```

O(h)

Deletion is slighty complicated:



Case 1. node has no children

Case 2. node has one child

Case 3: node has two children

a. replace key by successor key

b. delete successor