

# $(2, 4)$ Trees

# Contents:-

---

## **(2, 4) Trees**

---

### **Multi-way search tree**

---

### **Multi-way searching**

---

### **Insertion in a (2,4) tree**

---

### **Deletion in a (2,4) tree**

---

### **Time Complexity and Analysis**

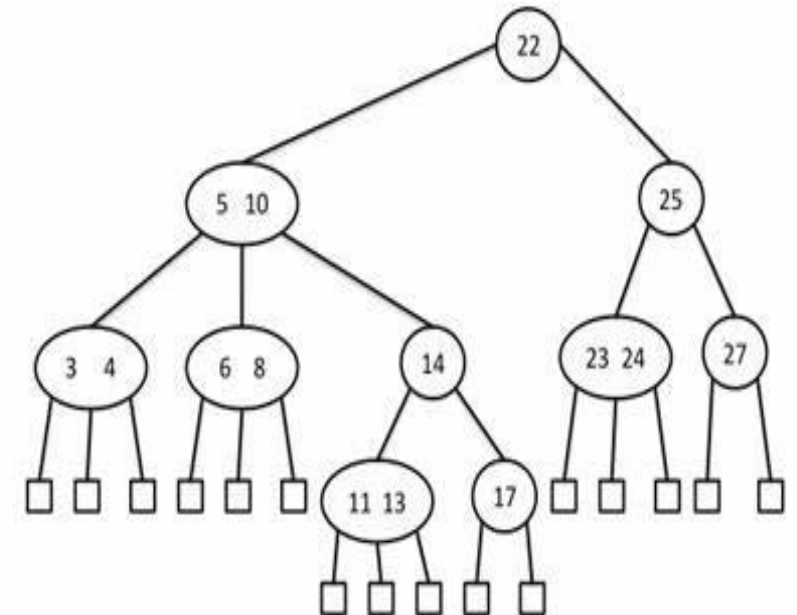
## (2, 4) Trees

- They are search trees, but they are not binary search trees
- These 2-4 trees are also called 2-3-4 trees.
- Here 2-3-4 refers to the number of children a node can have a node can have either 2, 3 or 4 children.
- Such trees in which a node can have many children but satisfy a certain kind of search properties are called multi-way search trees.

# Multi-way search tree

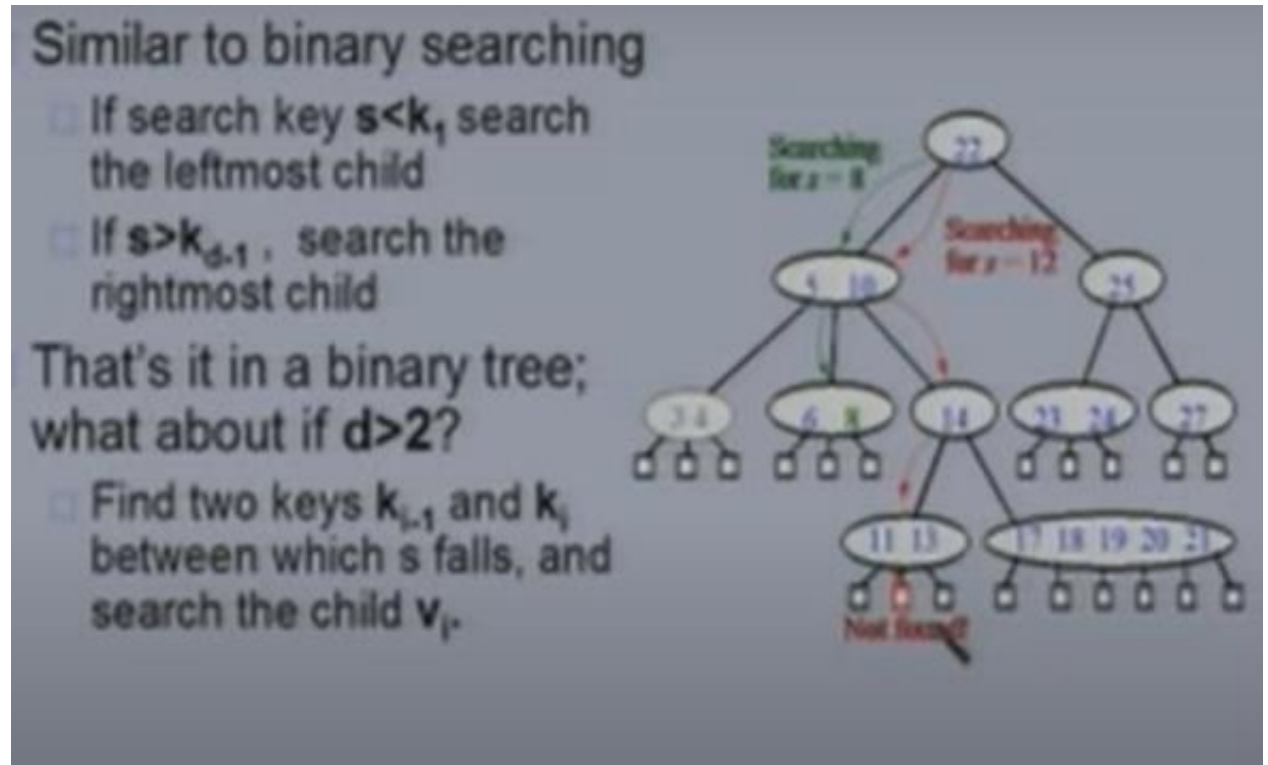
- Each internal node of a multi-way search tree has at least two children. It will have at least two which means it could have more than two children.
- Each node of a tree also stores a collection of items of the form (key, element).
- In the binary search tree there is only one such pair in each node and in a multi-way search tree there could be more than one.
- In particular there could be  $d-1$  such pairs or items, where  $d$  is the number of children that particular node has.

## Multi-way Search Tree



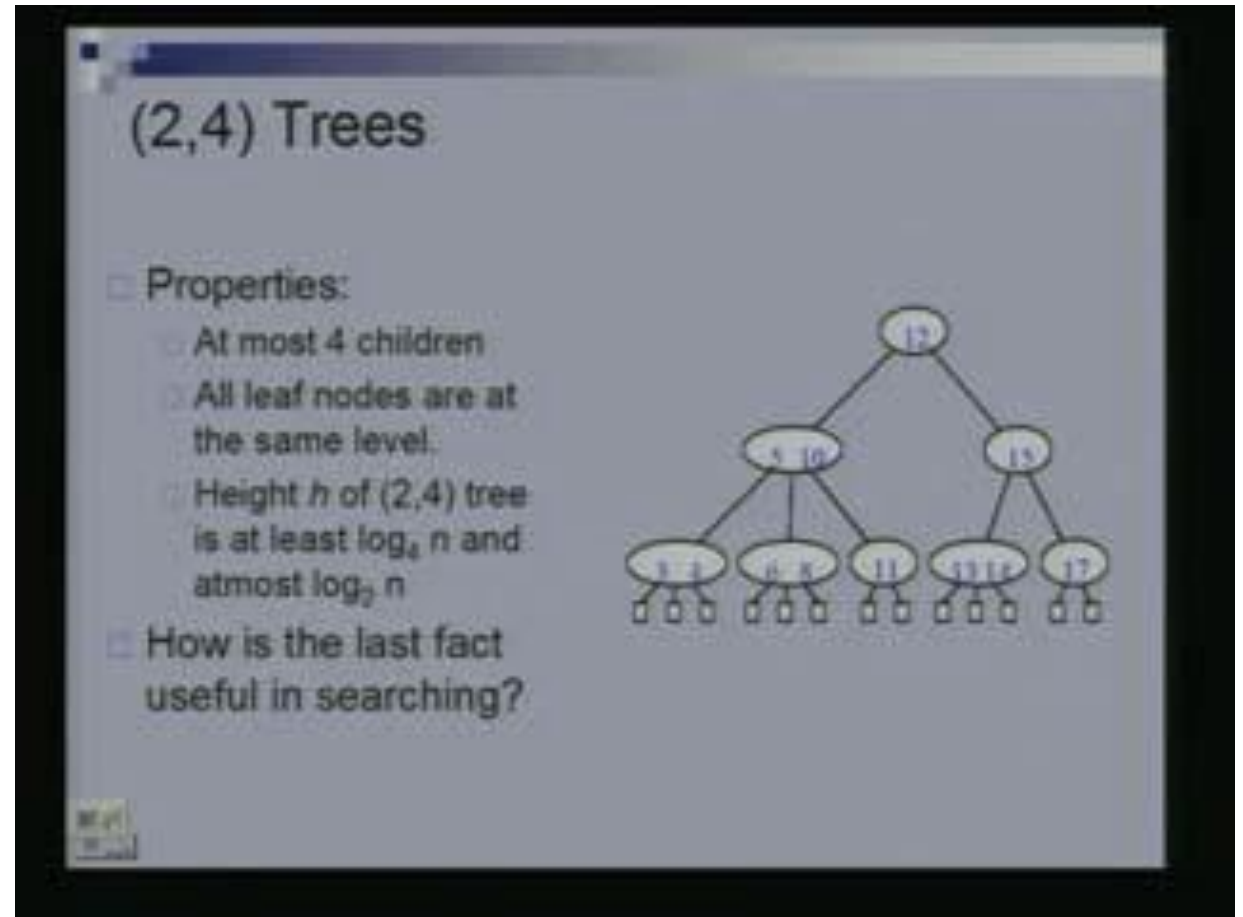
# Multi-way searching

- So, suppose we are searching for 8. You come down here, compare 8 with 22 so 8 is less. So you go here, now you will have to find, so 8 is not less than 5 and 8 is not more than 10. But 8 lies between 5 and 10. So you will follow this and then you will find that 8 is sitting here. So it's a successful search.



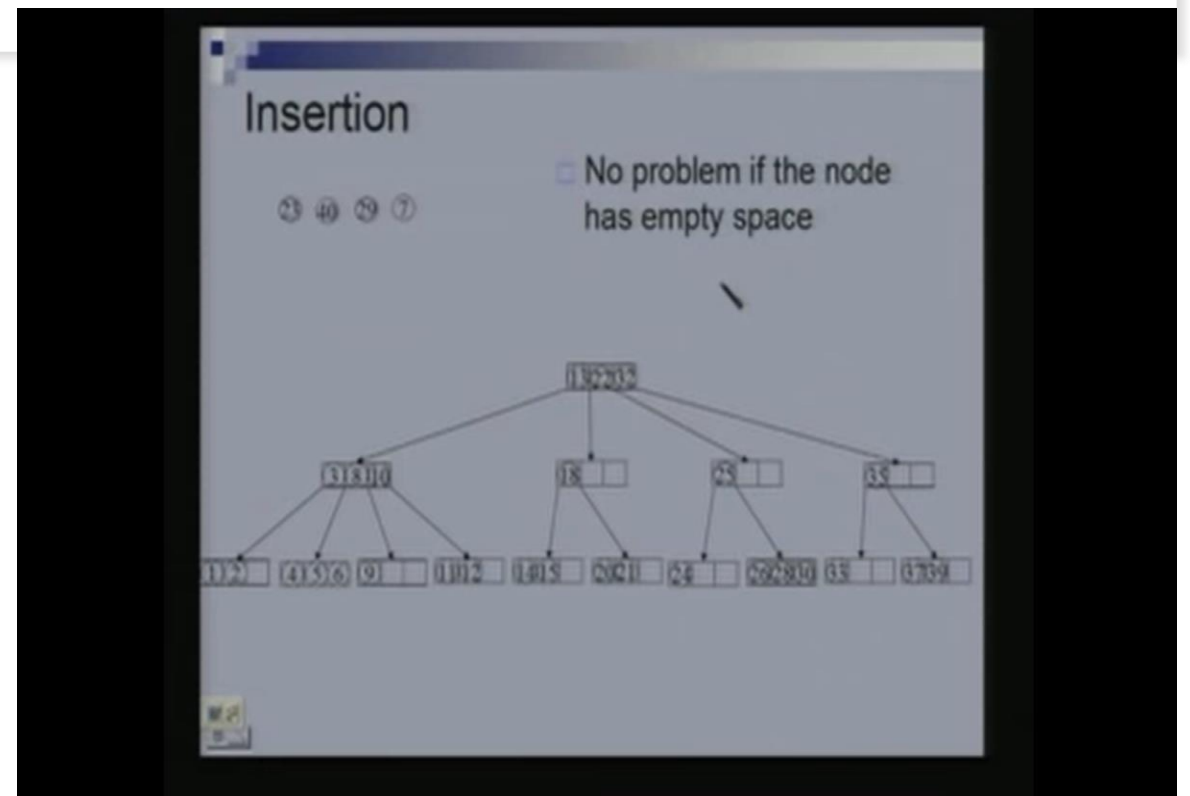
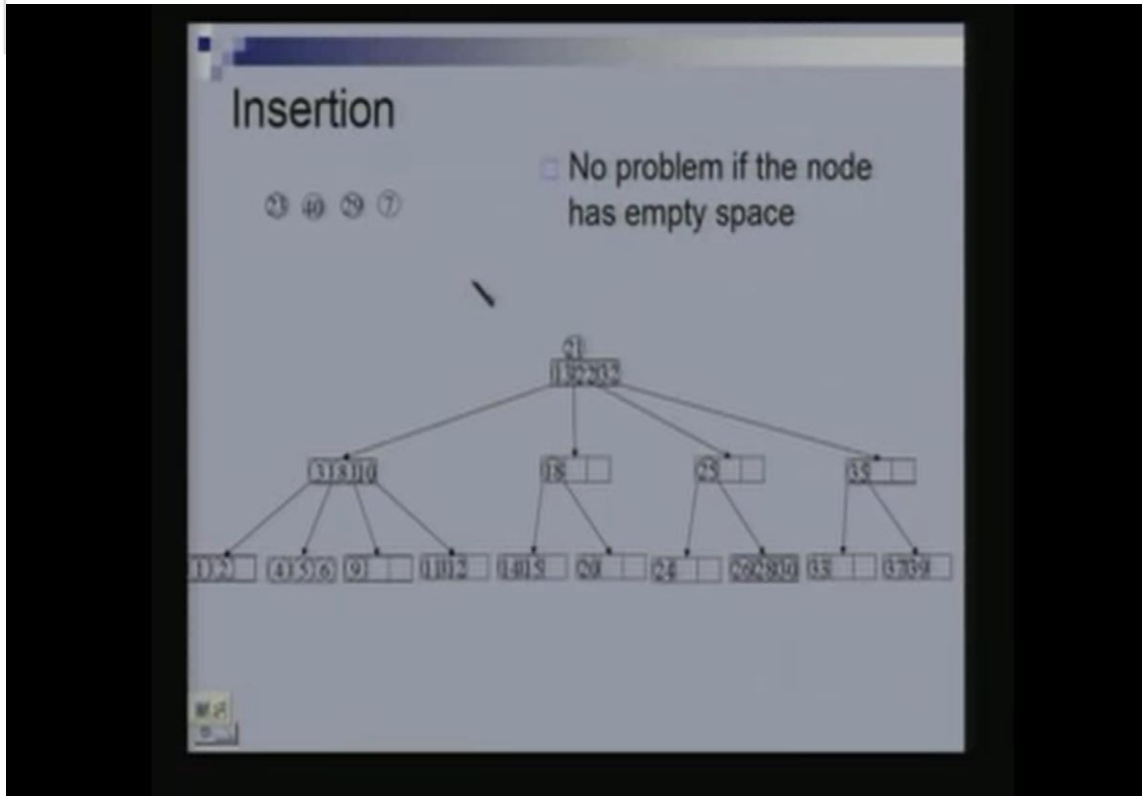
# (2, 4) Trees

- (2, 4) Tree is same as multiway search tree with 2 additional properties.
- if we do the same analysis as for complete binary tree you will find that the height of this tree is  $\log_4 n$ . So height of the 2-4 tree on  $n$  nodes always lies between these two quantities.
- So how much time does it take for me to search in a 2-4 tree? Height of the tree.
- So the time is 3 comparisons with in a node, times  $\log n$  because that is  $\log n$  is a number of node I would be visiting  $O(\log n)$



# Insertion in a (2,4) tree

- Case 1:-



## Case 2:-

### Insertion(2)

⑦

- Nodes get split if there is insufficient space.



### Insertion(2)

⑦

- Nodes get split if there is insufficient space.





## Insertion(2)

⑦

- Nodes get split if there is insufficient space.



## Insertion(3)

⑦

- One key is promoted to parent and inserted in there



# Case 3 the parent is also full

## Insertion(3)

- One key is promoted to parent and inserted in there



## Insertion(3)

- One key is promoted to parent and inserted in there



# Insertion Continue

## Insertion(4)

- ❑ If parent node does not have sufficient space then it is split.
- ❑ In this manner splits can cascade.



## Insertion(4)

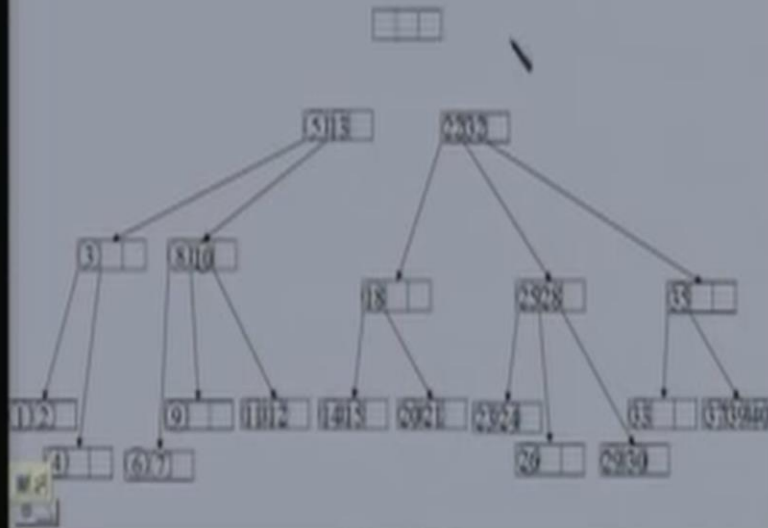
- ❑ If parent node does not have sufficient space then it is split.
- ❑ In this manner splits can cascade.



# Insertion Continue

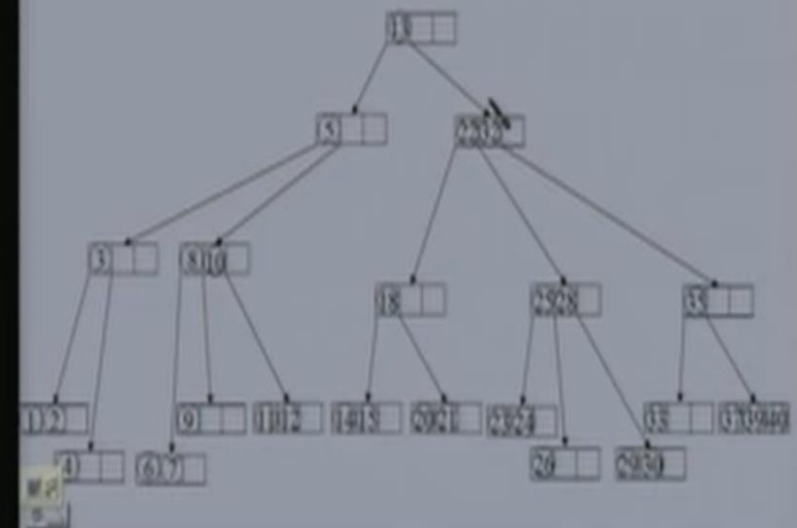
## Insertion(5)

- Eventually we may have to create a new root.
- This increases the height of the tree

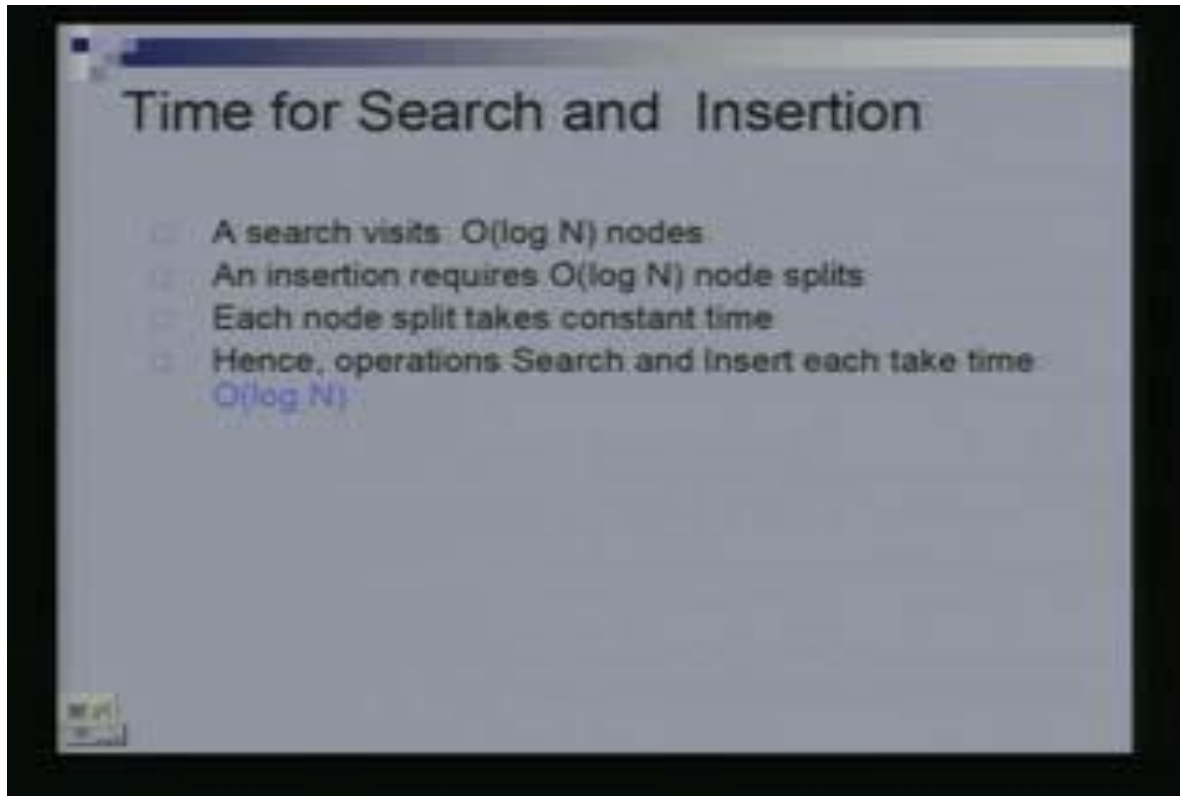


## Insertion(5)

- Eventually we may have to create a new root.
- This increases the height of the tree



# Time complexity for (2,4 tree)

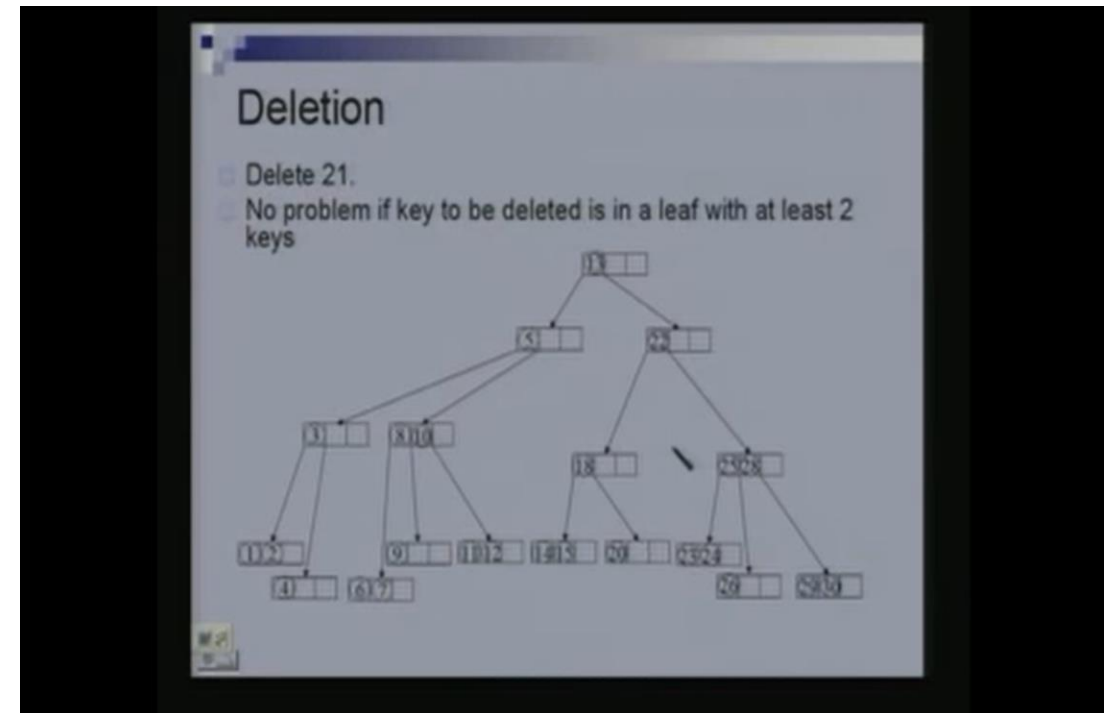
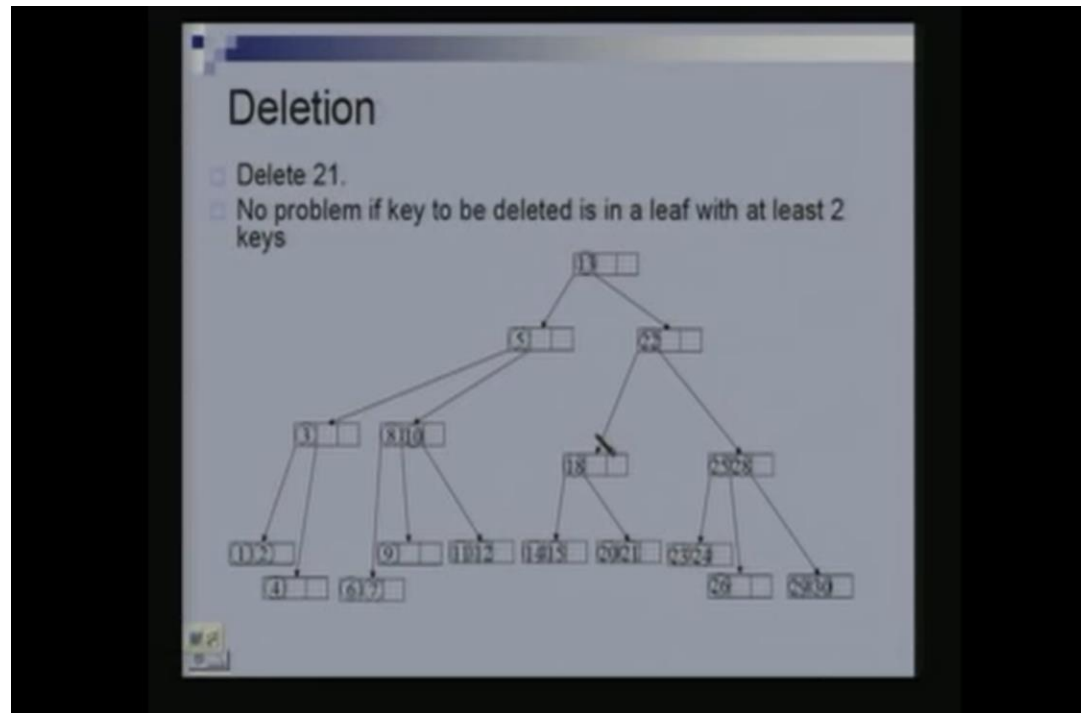


Time for Search and Insertion

- A search visits  $O(\log N)$  nodes
- An insertion requires  $O(\log N)$  node splits
- Each node split takes constant time
- Hence, operations Search and Insert each take time  $O(\log N)$

# Deletion:-

- Case 1 :-
- Leaf node with at least 2 keys.



# Deletion

- Case 2:- internal node

## Deletion(2)

- If key to be deleted is in an internal node then we swap it with its predecessor (which is in a leaf) and then delete it.
- Delete 25



## Deletion(2)

- If key to be deleted is in an internal node then we swap it with its predecessor (which is in a leaf) and then delete it.
- Delete 25



# Deletion

- Case 3:- Nodes become empty

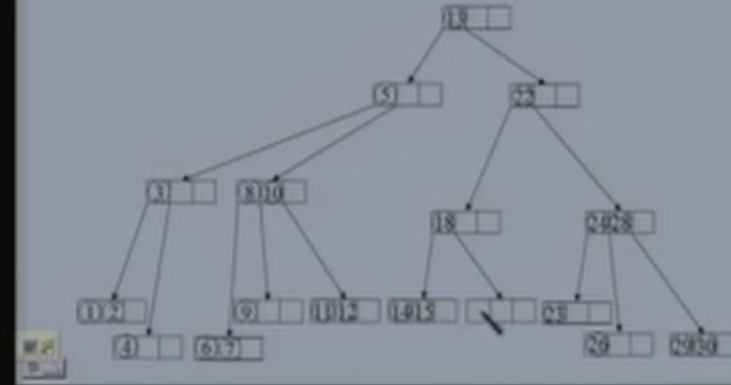
## Deletion(3)

- If after deleting a key a node becomes empty then we borrow a key from its sibling.
- Delete 20



## Deletion(3)

- If after deleting a key a node becomes empty then we borrow a key from its sibling.
- Delete 20

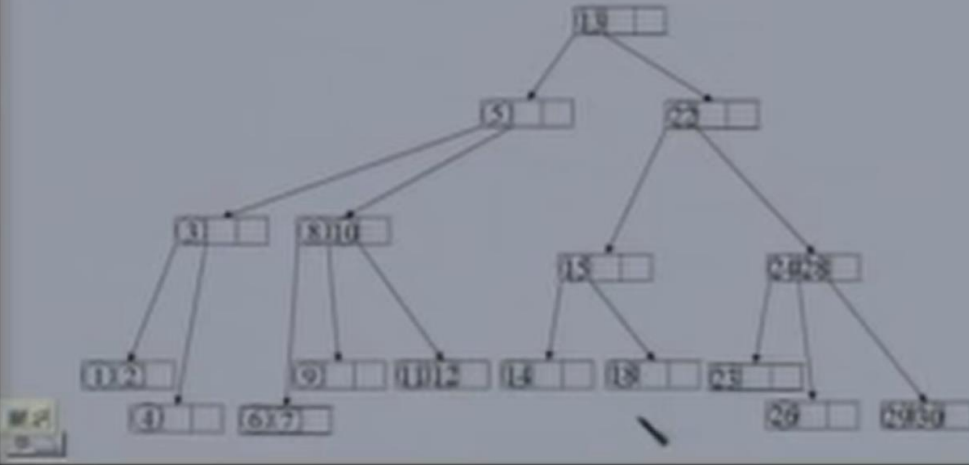




# Deletion

## Deletion(3)

- If after deleting a key a node becomes empty then we borrow a key from its sibling.
- Delete 20



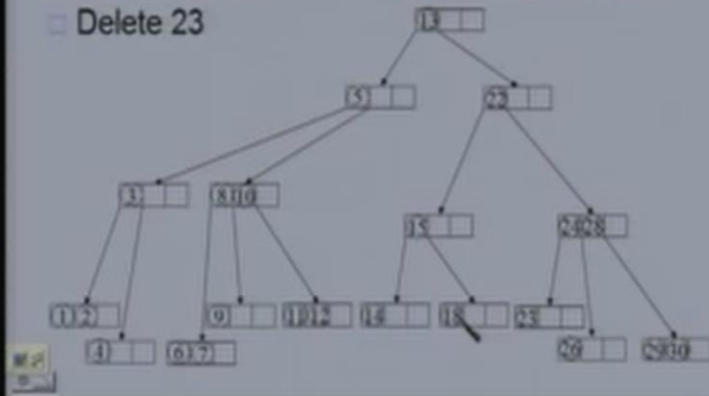
- Here we are borrowing a key from sibling and we cannot borrow 15 as it will not follow search property so we will do rotation as in avl tree so 15 goes up and 18 goes down.

# Deletion

- Case 4:- if we cannot borrow from sibling

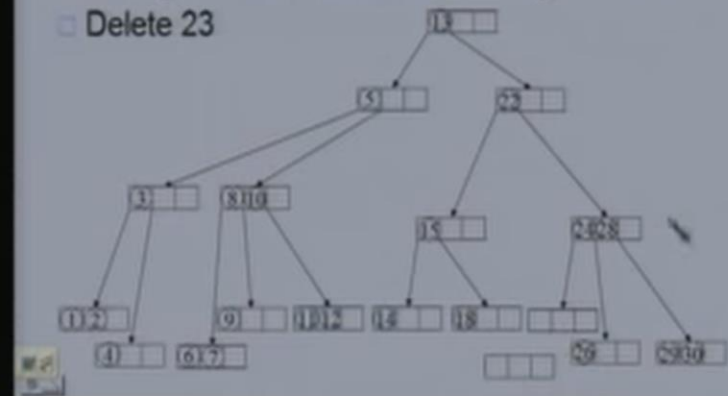
## Deletion(4)

- If sibling has only one key then we merge with it.
- The key in the parent node separating these two siblings moves down into the merged node.
- Delete 23



## Deletion(4)

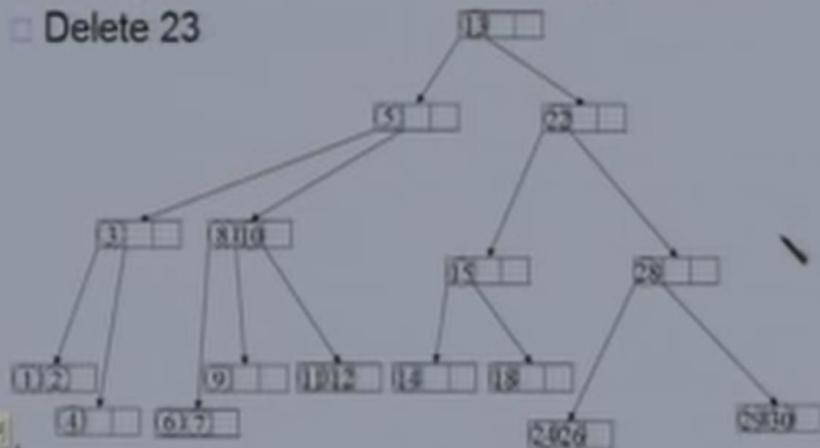
- If sibling has only one key then we merge with it.
- The key in the parent node separating these two siblings moves down into the merged node.
- Delete 23



# Deletion

## Deletion(4)

- If sibling has only one key then we merge with it.
- The key in the parent node separating these two siblings moves down into the merged node.
- Delete 23



- So here one key from parent comes down to the new node so there is 24 and 26 in the new node and the parent has one key with 2 children node.

## Conclusion:-

### (2,4) Conclusion

- The height of a (2,4) tree is  $O(\log n)$ .
- Split, transfer, and merge each take  $O(1)$ .
- Search, insertion and deletion each take  $O(\log n)$ .

- So, we studied this complex data structure because they are very connected to red black trees which are very useful.