COL106
Data Structures and Algorithms

Subodh Sharma and Rahul Garg

Announcements

Announcements

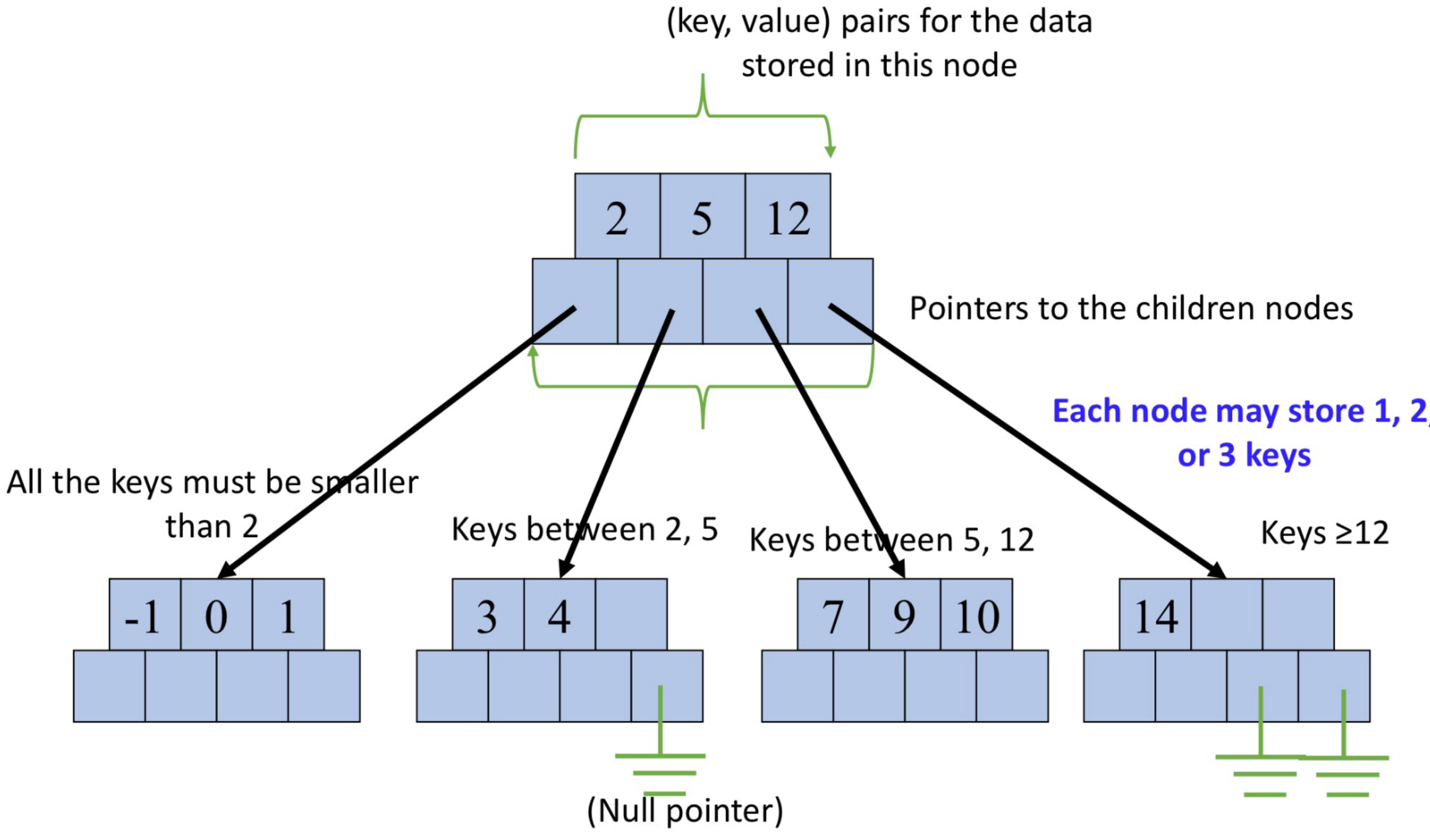
- Lab examination postponed to after the mid-semester examination due to logistical issues
- Minor syllabus
  - Everything covered till today, plus
  - 2-4 Trees, a-b Trees and Red-Black

2-4 Trees Removal

About 2-4 Trees

- Search trees
- Not binary
- Called 2-4 trees or 2-3-4 trees
- Balanced
- Each leaf node has the same depth

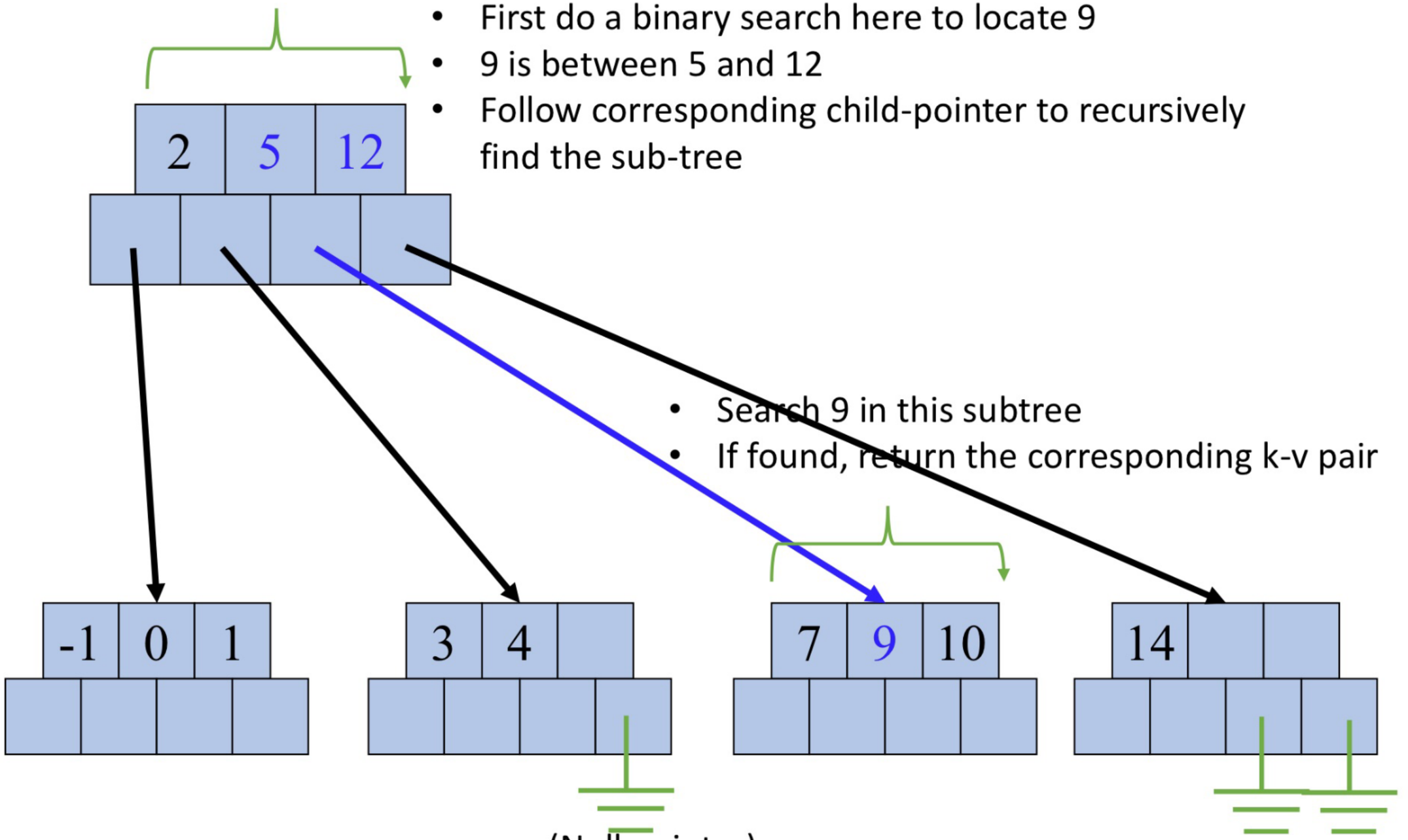
2-4 Tree Node Structure



Find in 2-4 Trees: Similar to Binary Search

```
find(k, T)
    if k found in T return found
    if T is a leaf, return not-found
    if k < T->key[0]
       return find(k, T->child[0])
  if k > T->key[last]
    return find(k, T->child[last+1])
  find i such that:
    T->key[i] < k < T->key[i+1]
    return find(k, T->child[i])
```

How to Find a Key? Find 9



- First do a binary search here to locate 9
- 9 is between 5 and 12
- Follow corresponding child-pointer to recursively find the sub-tree

- Search 9 in this subtree
- If found, return the corresponding k-v pair

Height of 2-4 Trees

- Consider a 2-4 tree of height h
- All the leaves are at depth h
- Each internal node has 2 or 3 or 4 children

$$\mathbf{2^{h+1}-1 \leq N(h) \leq (4^{h+1}-1)/3}$$

$$\mathbf{\tfrac{1}{2}\log_2(3N+1) \leq h \leq \log_2(N+1)}$$

h = O(log(N))

Find runtime is O(log(N))

Insertion Summary

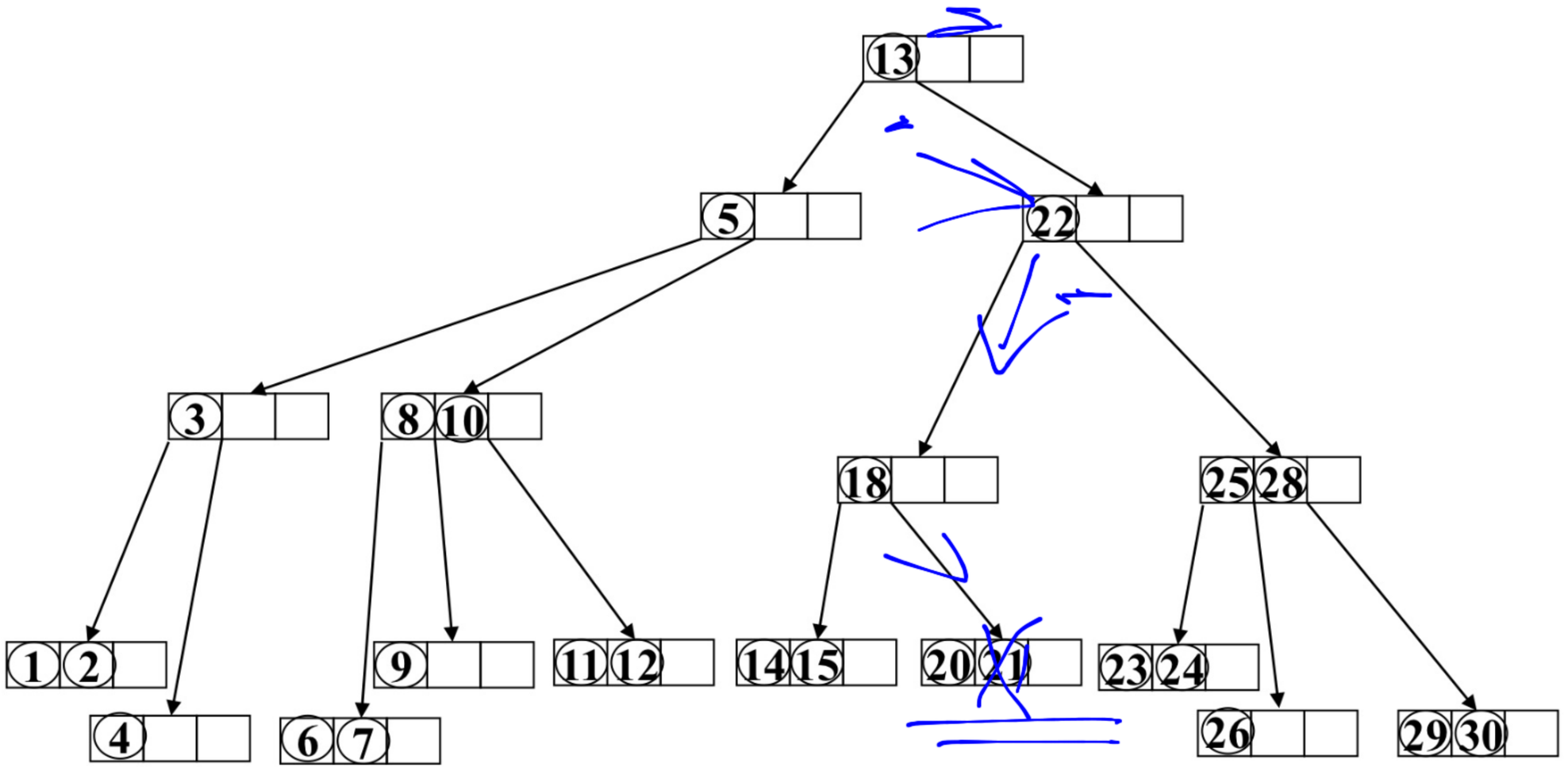
- Find and travel to the correct node in leaf
- If leaf node has space insert there. End.
- If leaf is full, split it and promote the median key to its parent
  - Apply the insert recursively to the parent
  - If parent has space, insert there else split parent and insert median key to parent’s parent and so on
  - Height only increases by splitting the root
  - All leaf nodes remain at the same depth

2-4 Trees

Removal/Deletion

Deletion

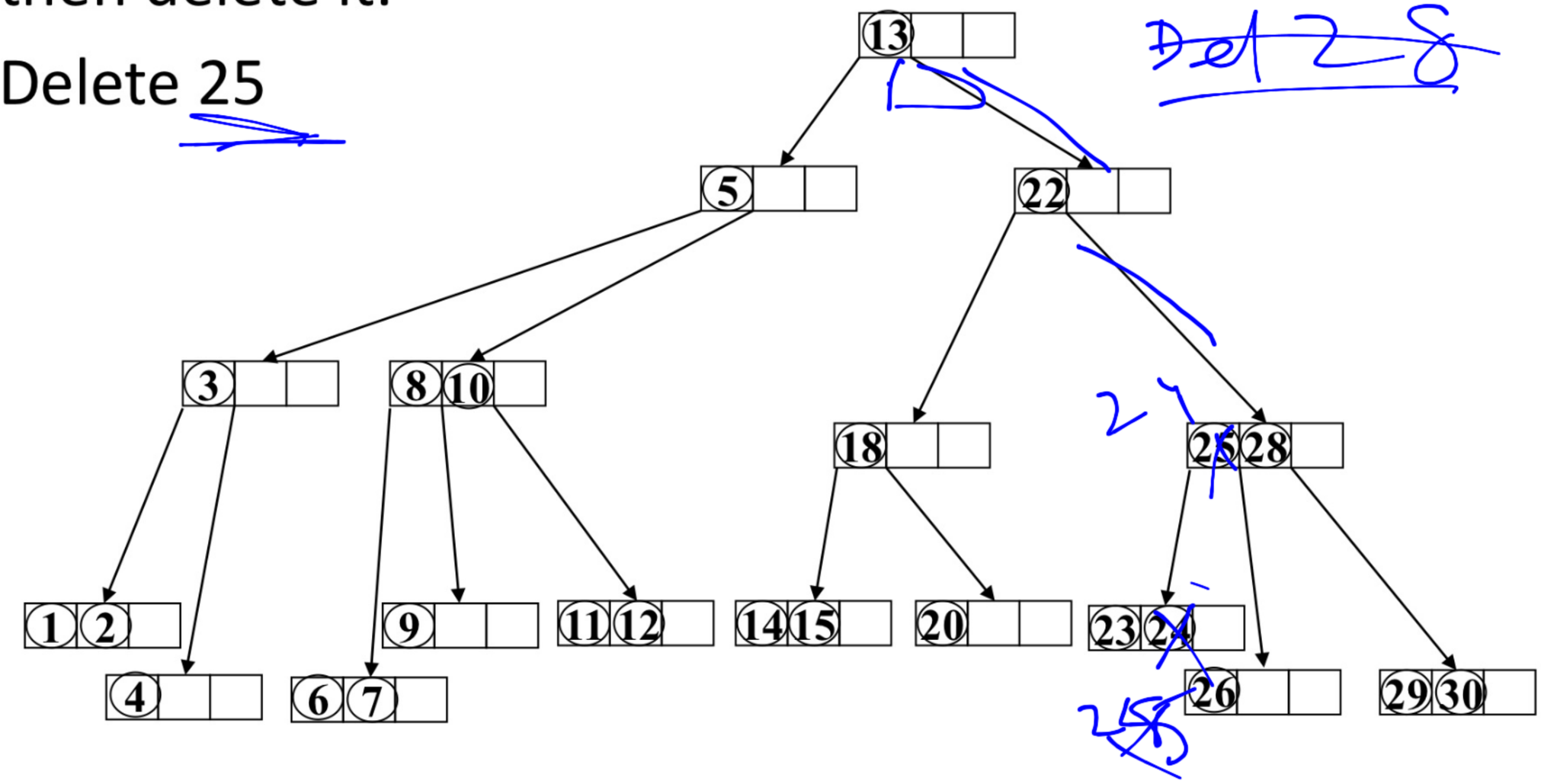
- Delete 21.
- No problem if key to be deleted is in a leaf with at least 2 keys



Slide adapted from Naveen Garg, IIT Delhi

Deletion

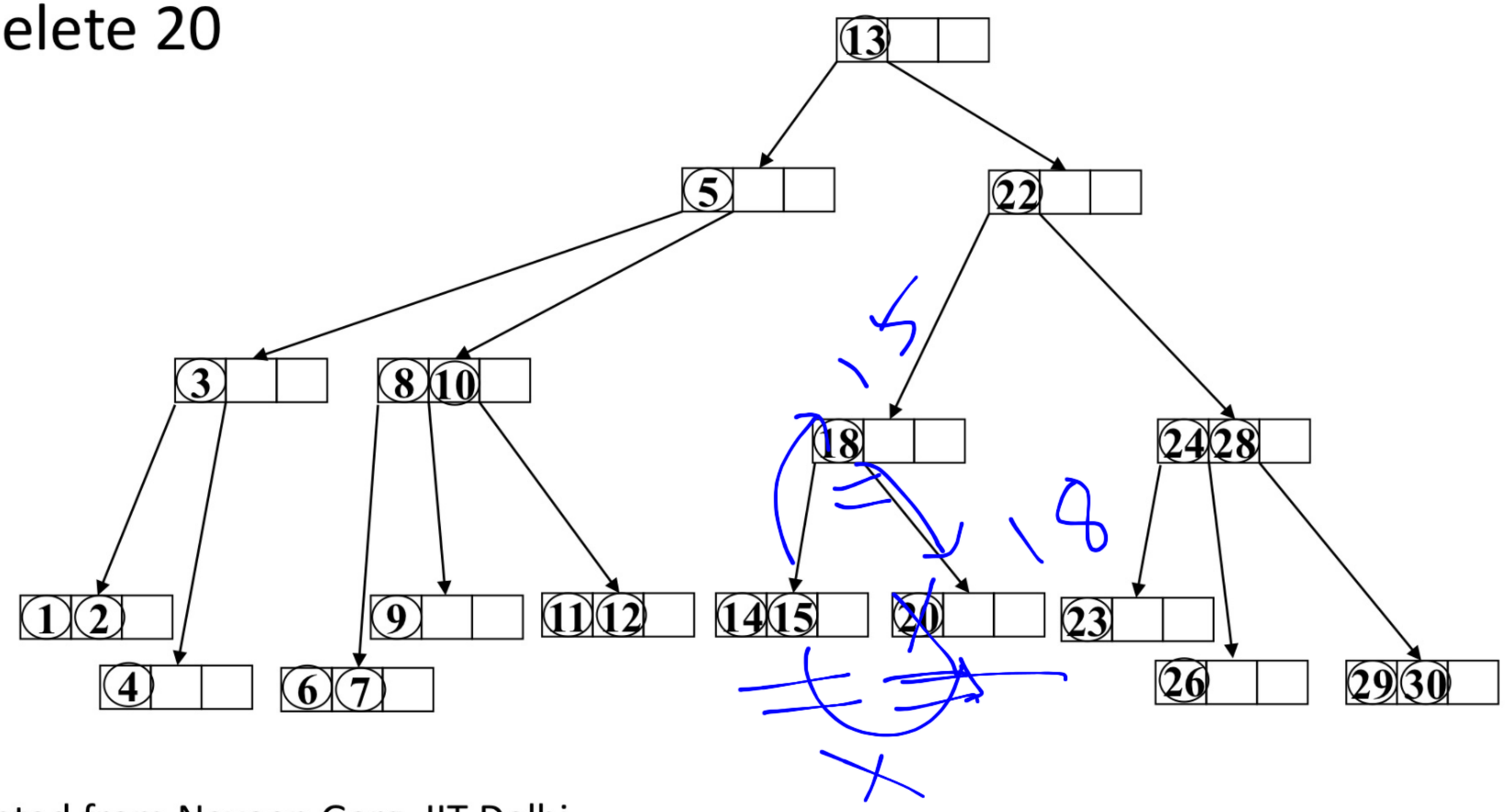
- If key to be deleted is in an internal node then we swap it with its predecessor (which is in a leaf) and then delete it.
- Delete 25



Slide adapted from Naveen Garg, IIT Delhi

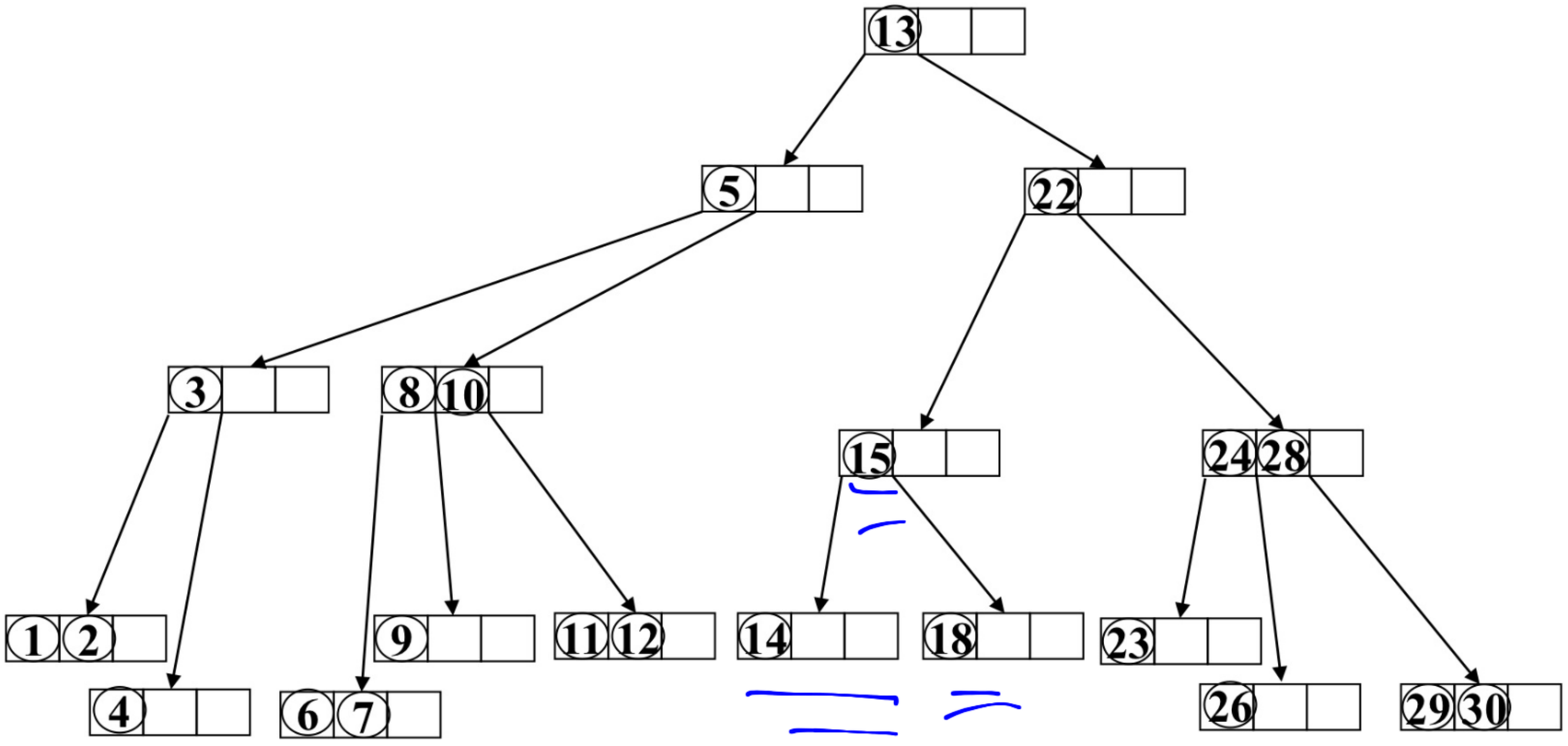
Deletion

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



Slide adapted from Naveen Garg, IIT Delhi

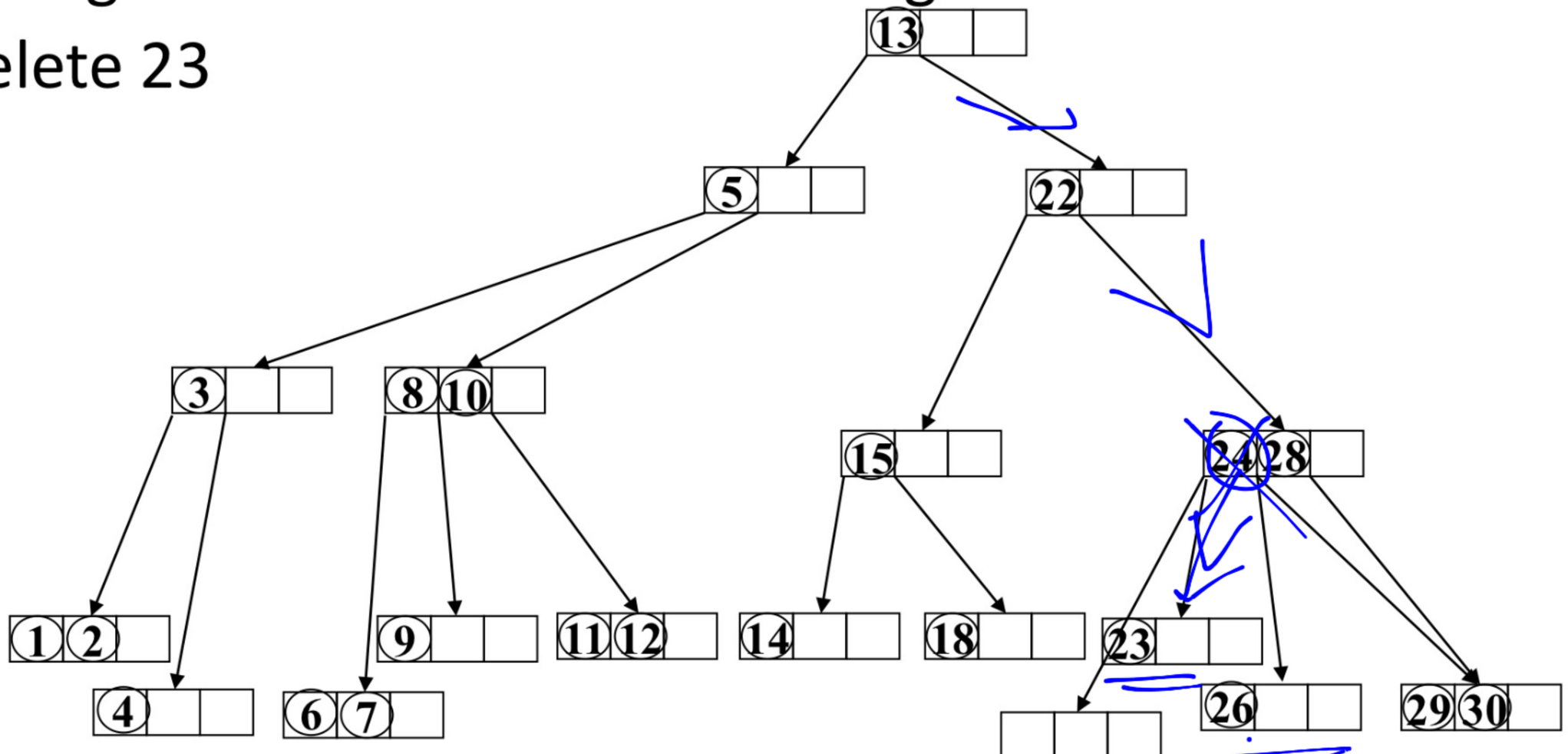
Deletion



Slide adapted from Naveen Garg, IIT Delhi

Deletion

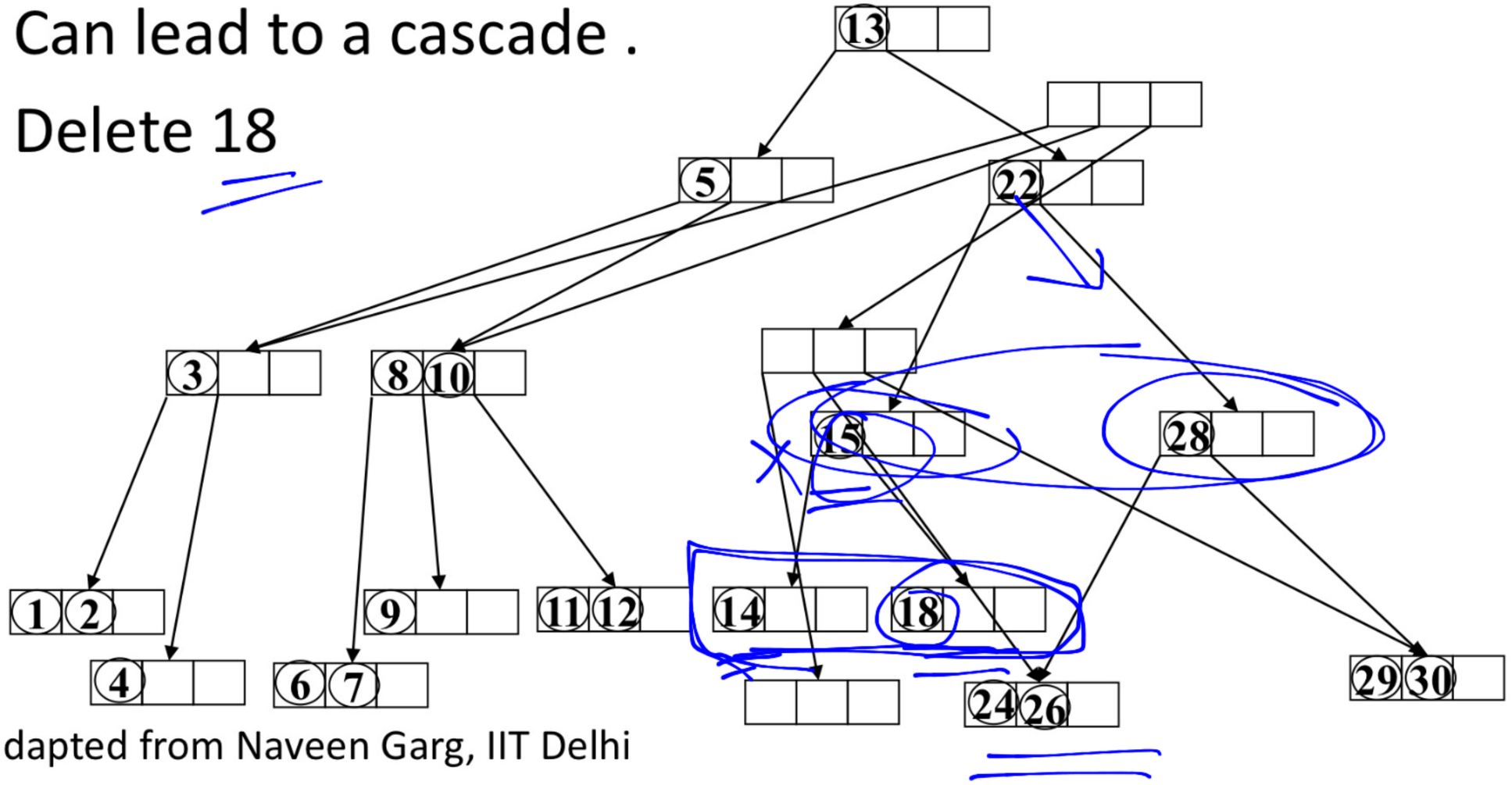
- If sibling has only one key then we merge with it.
- The key in the parent node separating these two siblings moves down into the merged node.
- Delete 23



Slide adapted from Naveen Garg, IIT Delhi

Delete

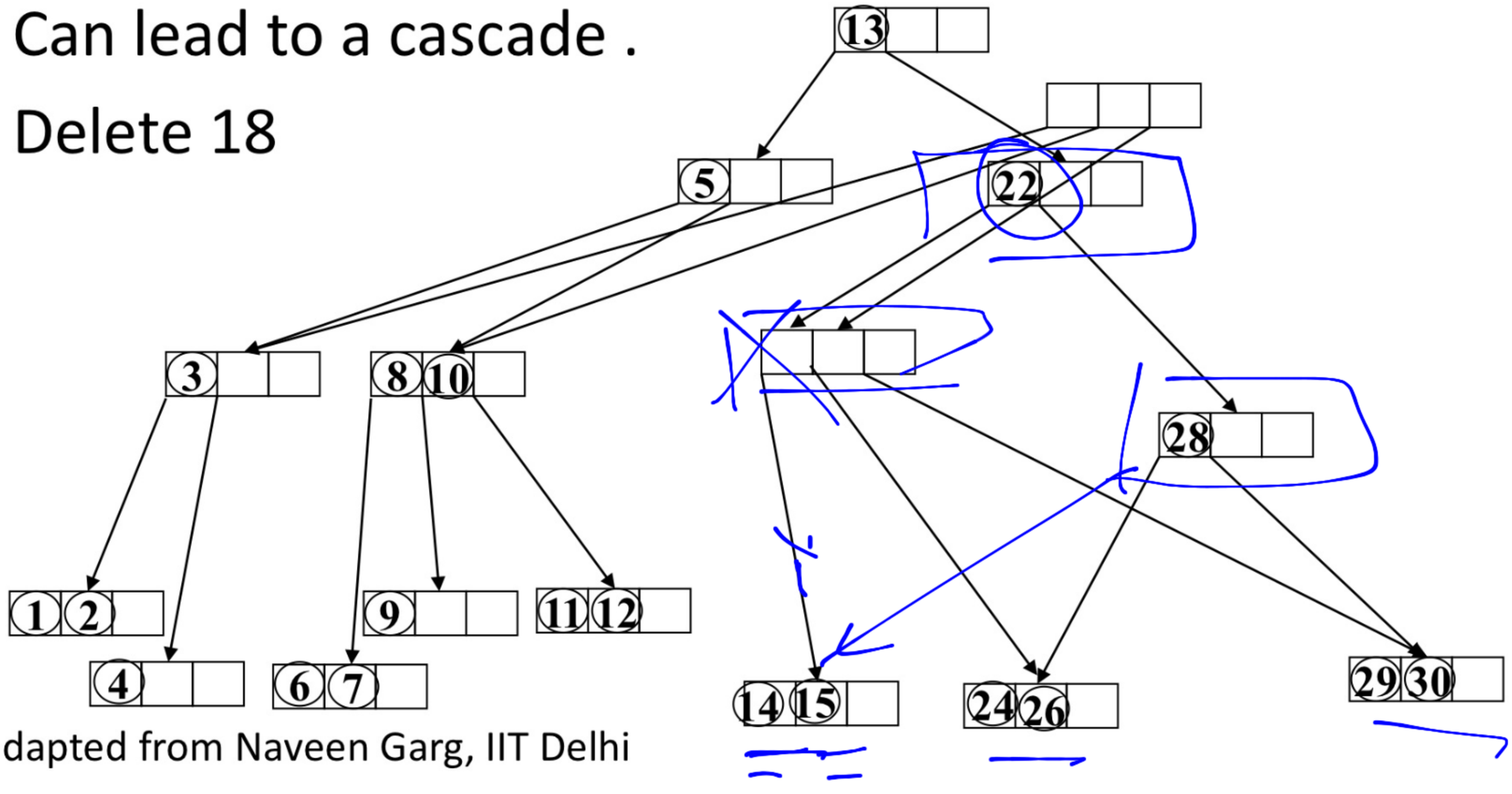
- Moving a key down from the parent corresponds to deletion in the parent node.
- The procedure is the same as for a leaf node.
- Can lead to a cascade .
- Delete 18



Slide adapted from Naveen Garg, IIT Delhi

Delete

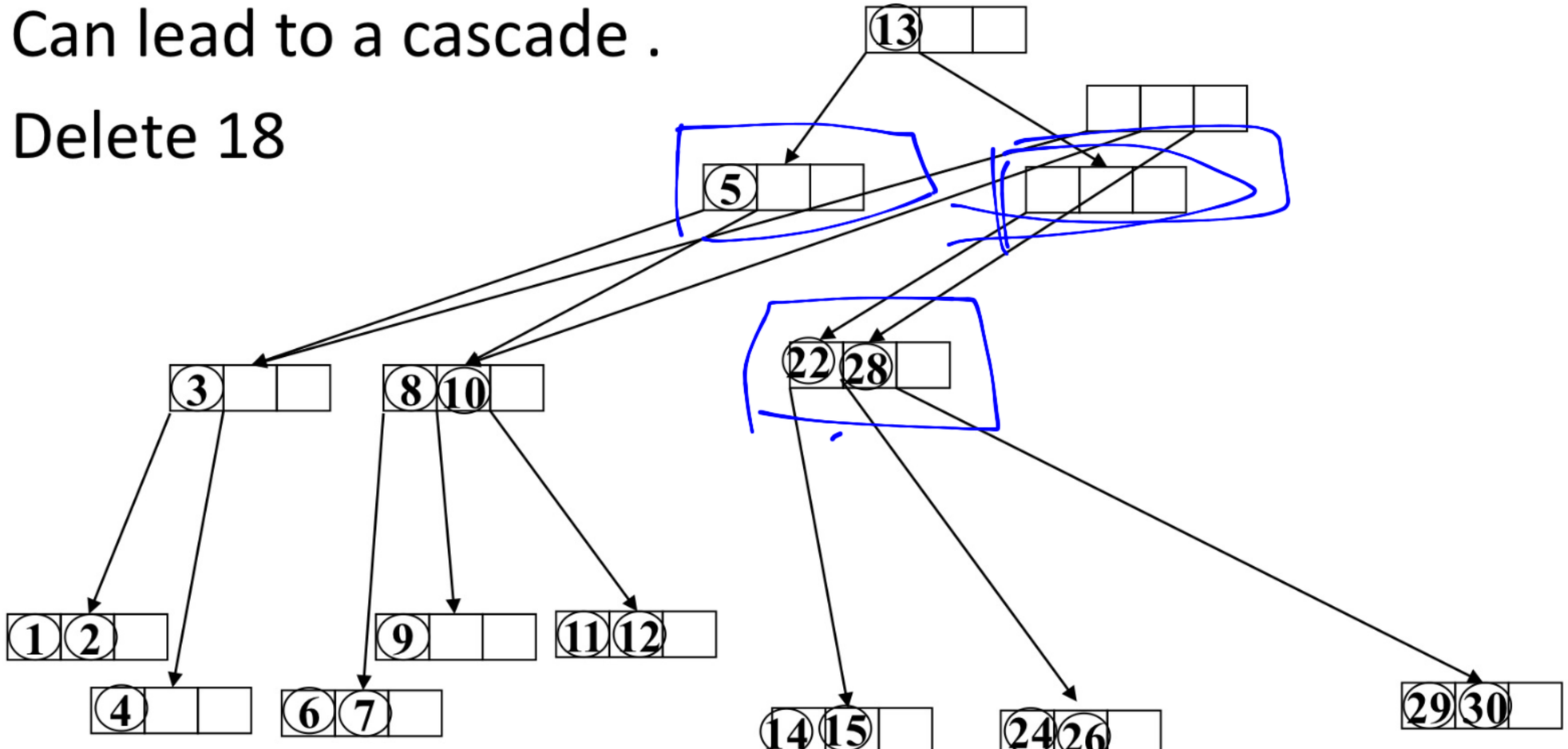
- Moving a key down from the parent corresponds to deletion in the parent node.
- The procedure is the same as for a leaf node.
- Can lead to a cascade .
- Delete 18



Slide adapted from Naveen Garg, IIT Delhi

Delete

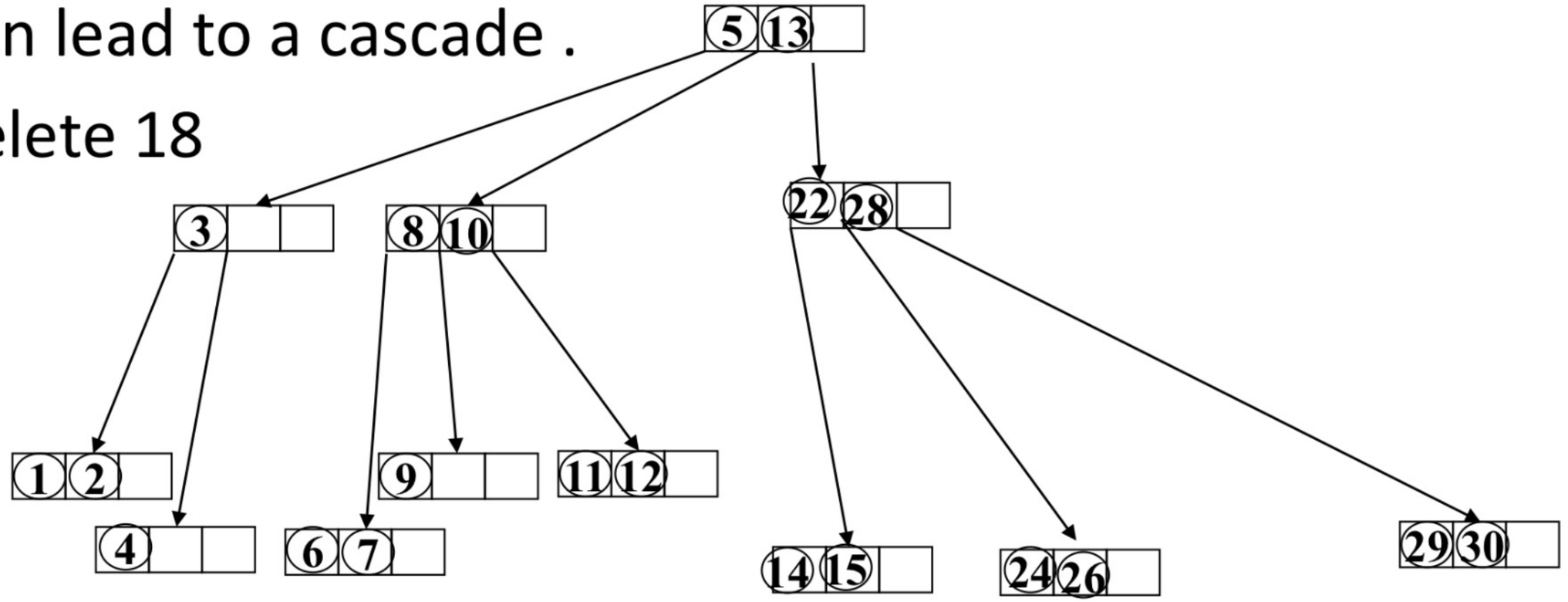
- Moving a key down from the parent corresponds to deletion in the parent node.
- The procedure is the same as for a leaf node.
- Can lead to a cascade .
- Delete 18



Slide adapted from Naveen Garg, IIT Delhi

Delete

- Moving a key down from the parent corresponds to deletion in the parent node.
- The procedure is the same as for a leaf node.
- Can lead to a cascade .
- Delete 18



Slide adapted from Naveen Garg, IIT Delhi

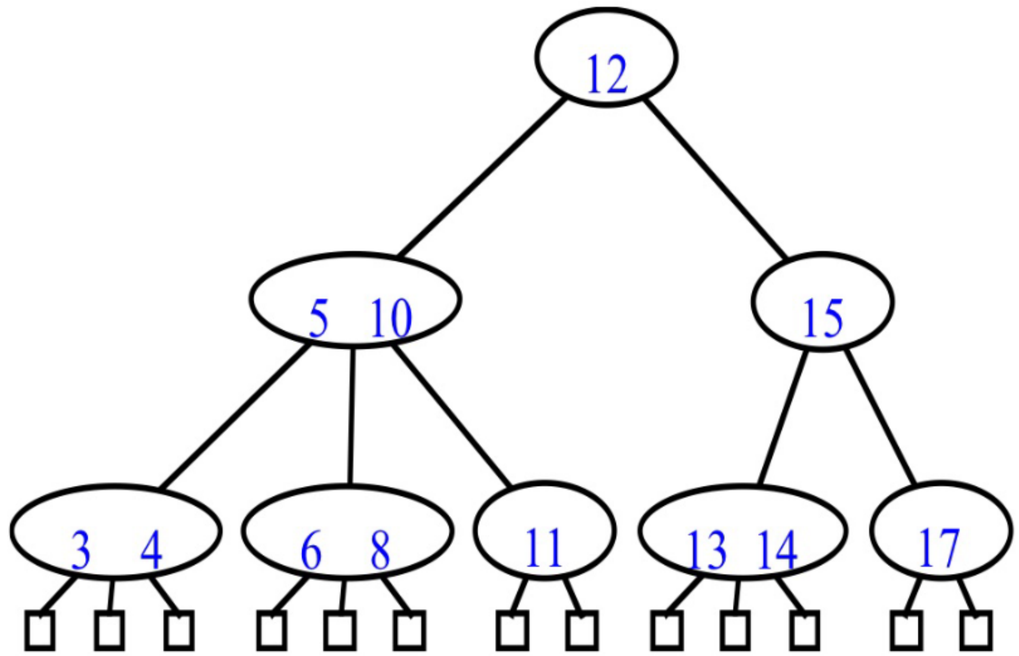
(2,4) Trees Summary

- The height of a (2,4) tree is $O(\log n)$
- Split, transfer, and merge operations take $O(1)$ time
- Search, insertion and remove take $O(\log n)$ time
- Indexing and sorting – Fundamental operations
- AVL Trees (height balanced trees) give O(log(N)) search, insert and remove
- (2, 4) trees give another way to balance using multiple children
- They take O(log(N)) time for search, insert and remove
- Also related to Red-Black trees that we will cover next

Generalization of 2-4 Trees: a-b Trees

(a,b) Trees

- A multiway search tree.
- Each node has at least *a* and at most *b* children.
- Root can have less than *a* children but it has at least 2 children.
- All leaf nodes are at the same level.
- Height *h* of (a,b) tree is at least $\log_b n$ and at most $\log_a n$.

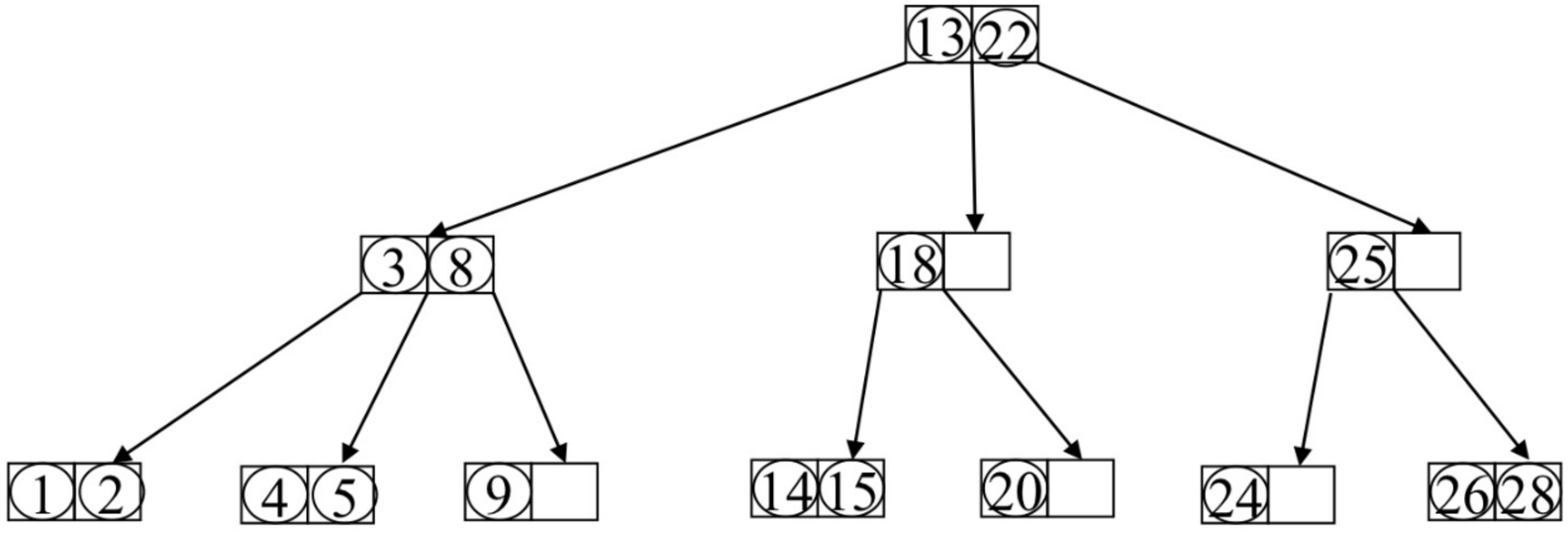


Slide adapted from Naveen Garg, IIT Delhi

Insertion

21 23 29 7

- No problem if the node has empty space

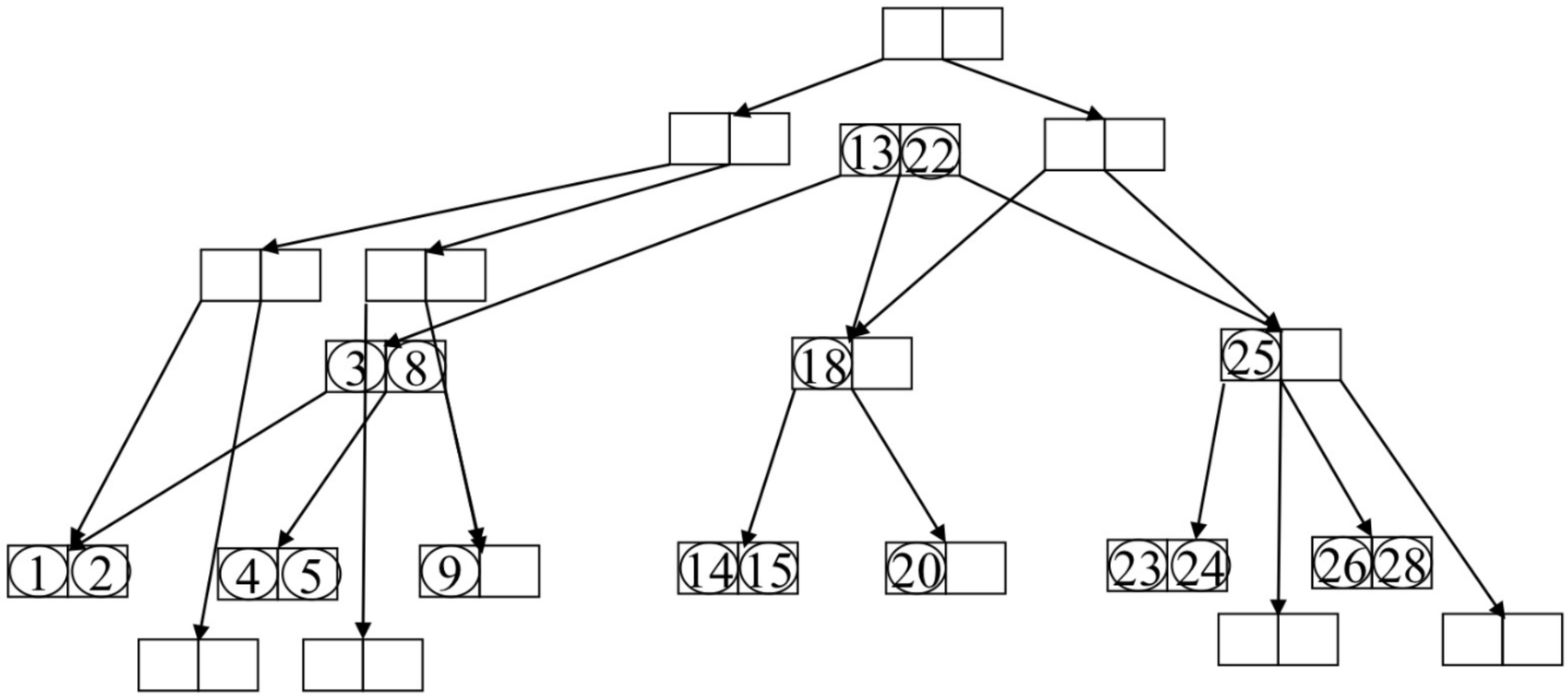


Slide adapted from Naveen Garg, IIT Delhi

Insertion(2)

- Nodes get split if there is insufficient space.
- The median key is promoted to the parent node and inserted there

29 7



Slide adapted from Naveen Garg, IIT Delhi

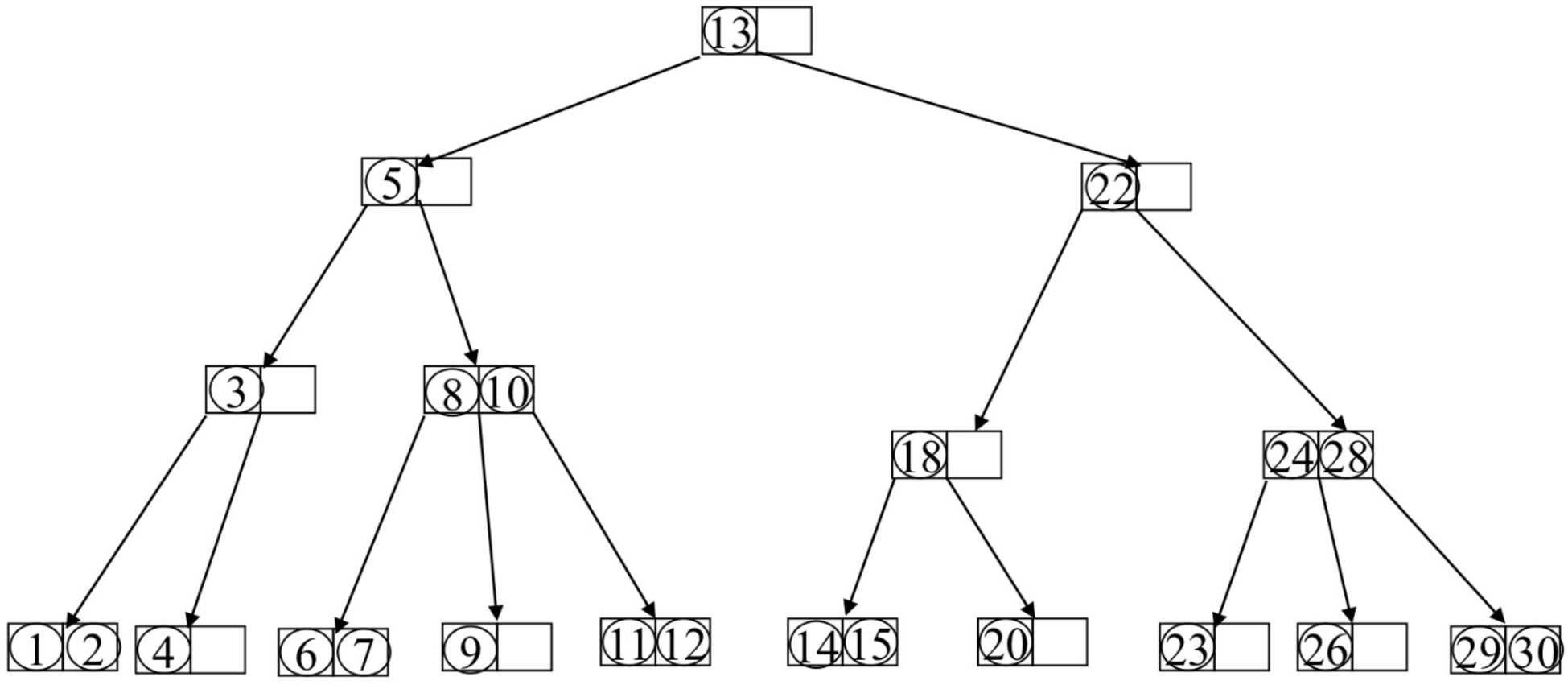
Insertion(3)

- A node is split when it has exactly *b* keys.
- One of these is promoted to the parent and the remaining are split between two nodes.
- Thus one node gets $\lceil \frac{b-1}{2} \rceil$ and the other $\lfloor \frac{b-1}{2} \rfloor$ keys.
- This implies that a-1 >= $\lfloor \frac{b-1}{2} \rfloor$

Slide adapted from Naveen Garg, IIT Delhi

Deletion (2, 4 Trees)

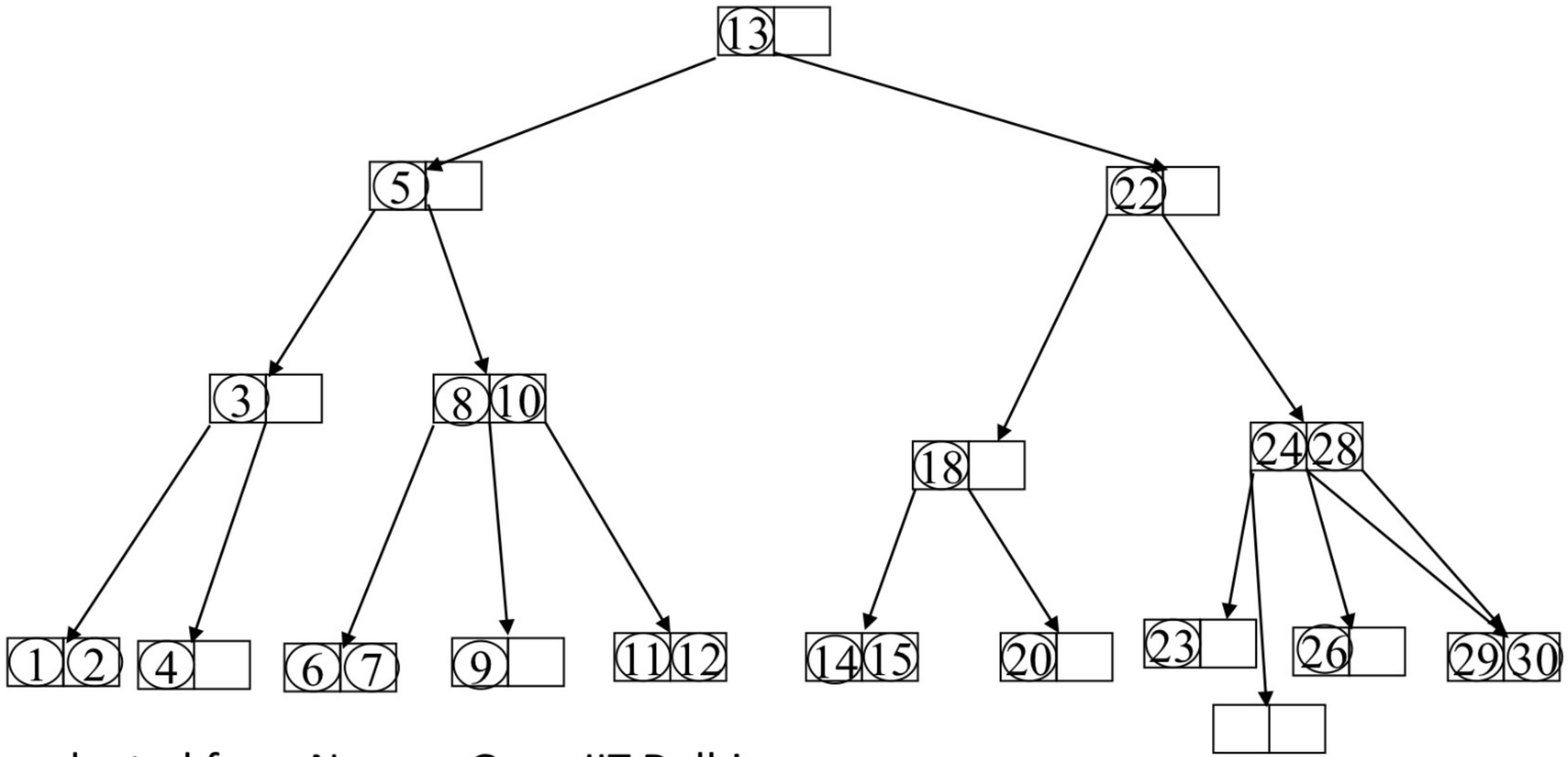
- In 2,4 trees, if after deleting a key a node becomes empty then we borrow a key from its sibling.
- Delete 20



Slide adapted from Naveen Garg, IIT Delhi

Deletion (2,4) Trees

- 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



Slide adapted from Naveen Garg, IIT Delhi

Deletion: (a, b) Trees

- In an (a,b) tree we will borrow from sibling if node has a-2 keys
- We will merge a node with its sibling if the node has *a*-2 keys and its sibling has *a*-1 keys.
- Thus the merged node has 2(*a*-1) keys.
- This implies that 2(*a*-1) <= *b*-1 which is equivalent to *a*-1 <= $\lfloor \frac{b-1}{2} \rfloor$
- Earlier too we argued that *a*-1 <= $\lfloor \frac{b-1}{2} \rfloor$
- This implies *b* >= 2*a*-1
- For a=2, *b* >= 3.
- Can also have 2-3 Trees – Homework

Slide adapted from Naveen Garg, IIT Delhi

Conclusion

- (a, b) Trees are a generalization of 2-4 trees
- The height of a (a,b) tree is $O(\log n)$.
- $b >= 2a-1$.
- For insertion and deletion we take time proportional to the height.

Slide adapted from Naveen Garg, IIT Delhi

Thank You

Find

```
Find(k, T)
    if k found in T return found
    if T is a leaf, return not-found
    if k < T->key[0]
        return find(k, T->child[0])
     if k < T->key[last]
        return find(k, T->child[last+1])
     find i such that:
        T->key[i] < k < T->key[i+1]
        return(k, T->child[i])
```