

COMP2013

Data Structures and Algorithms

Lecture 8

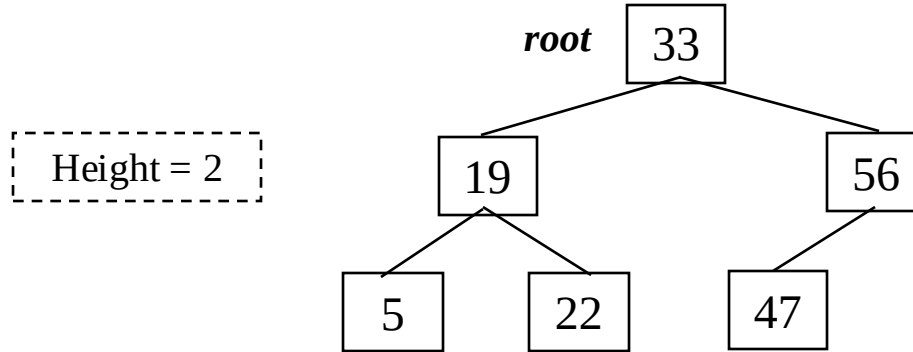
Binary Search Tree II

Dr. Jieming Shi

Binary Search Tree (BST)



- **Binary search tree** is a binary tree with this property:
For any node x in the tree,
for any node a in x 's *left subtree*, $a.key \leq x.key$, **and**
for any node b in x 's *right subtree*, $x.key \leq b.key$





<i>Operation</i>	<i>Complexity</i>	<i>Meaning</i>
Search	$O(h)$	Search a node with a key
Minimum	$O(h)$	Find the minimum node
Maximum	$O(h)$	Find the maximum node
Predecessor	$O(h)$	Find the predecessor node of a node
Successor	$O(h)$	Find the successor node of a node
Insert	$O(h)$	Insert a key
Delete	$O(h)$	Delete a key

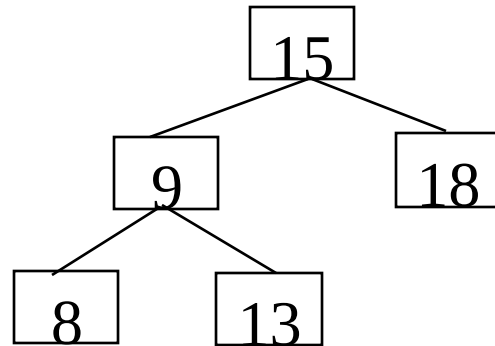
Tree height: h , but h can be larger than $\log_2 n$



- Find a node with key k
 - Return NIL if there is no such node
- Example: Search ($T.root$, 13)
 - Visit the node “15”, go left
 - Visit the node “9”, go right
 - Visit the node “13”, key found!

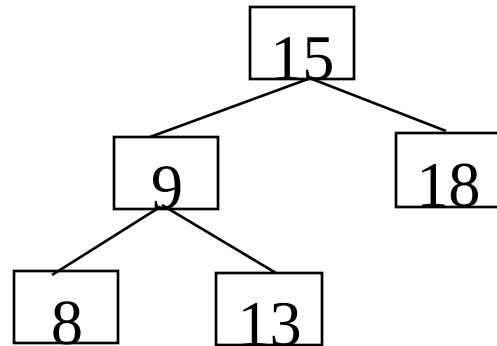
Search (x , k)

1. if $x = \text{NIL}$ or $k = x.key$
2. return x
3. if $k < x.key$
4. Search($x.left$, k)
5. else
6. Search($x.right$, k)





- Time: $O(h)$, where h is the tree height
 - Why?
 - Each call goes down at most 1 level,
 - there are at most h levels
- Search (x, k)
1. if $x = \text{NIL}$ or $k = x.\text{key}$
 2. return x
 3. if $k < x.\text{key}$
 4. Search($x.\text{left}, k$)
 5. else
 6. Search($x.\text{right}, k$)

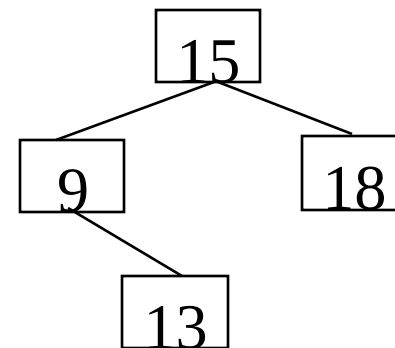
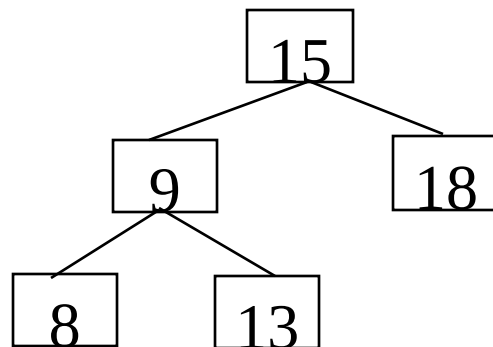




- In a binary search tree, which node contains the smallest value?
 - Leftmost
- $\text{Minimum}(T.\text{root})$
 - Time: $O(h)$
- What about $\text{Maximum}(T.\text{root})$?

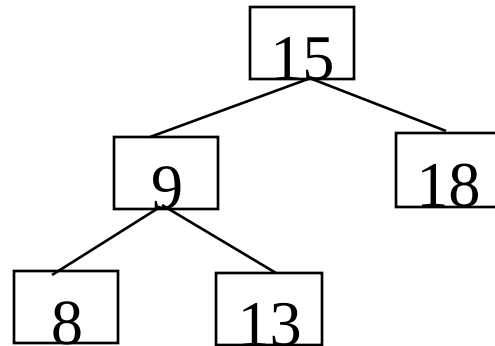
Minimum (x)

1. while $x.\text{left} \neq \text{NIL}$
2. $x \leftarrow x.\text{left}$
3. return x





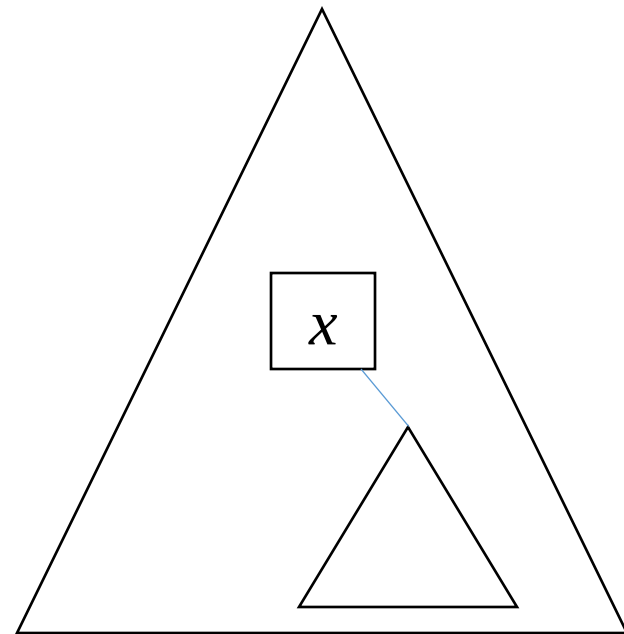
- $\text{Successor}(S, x)$: given x pointing to an element in a set S , return a pointer to the next larger element in S , or NIL if x is maximum
- What is the successor of 9?
- What is the successor of 13?



Successor: Find the successor of x in a BST



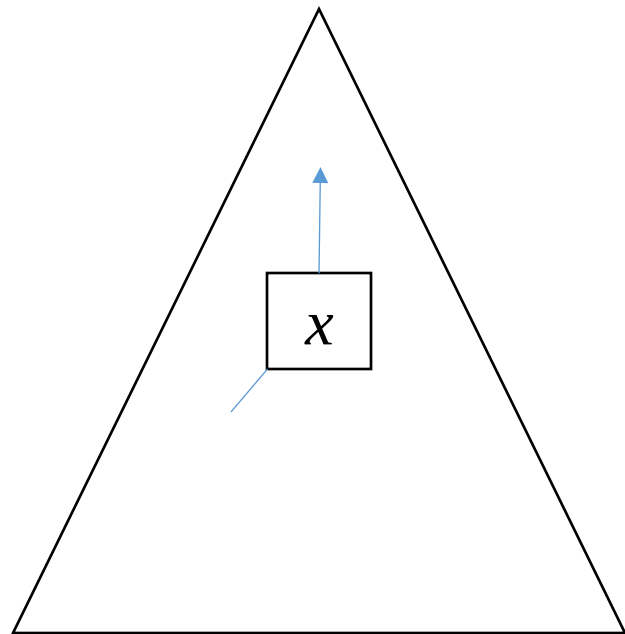
- Case 1: If the right subtree of x is nonempty,
i.e., $x.right \neq \text{NIL}$
 - The successor of x is ?
 - The minimum value in the right subtree of x
 - *i.e.*, the leftmost node in the right subtree
 - The successor cannot be in the left subtree of x
 - The successor cannot be in levels above x



Successor: Find the successor of x in a BST



- Case 2: If the right subtree of x is empty,
i.e., $x.right = \text{NIL}$
 - Where is the successor of x ?
 - Can it be in the left subtree of x ?
 - No
 - You can only go up from x to its parent, grandparent, **ancestors**, to find the successor
 - If x is in the right subtree of an ancestor, x is larger than the ancestor
 - If x is in the left subtree of an ancestor, x is **smaller** than the ancestor
 - **The lowest ancestor with x in its left subtree**

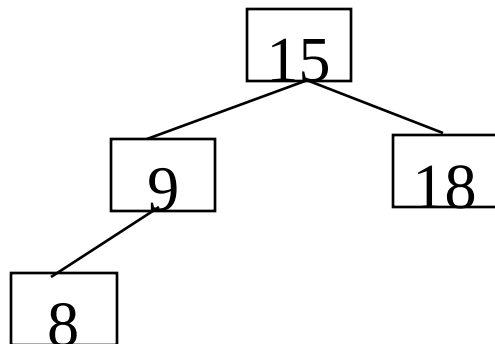


Successor: Find the successor of x in a BST

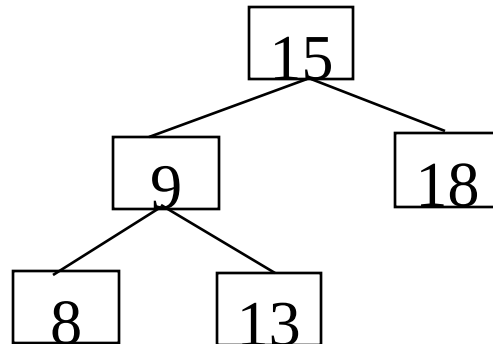


- Case 2: If the right subtree of x is empty,
i.e., $x.right = \text{NIL}$
 - **The lowest ancestor with x in its left subtree**

What is the successor of 9?



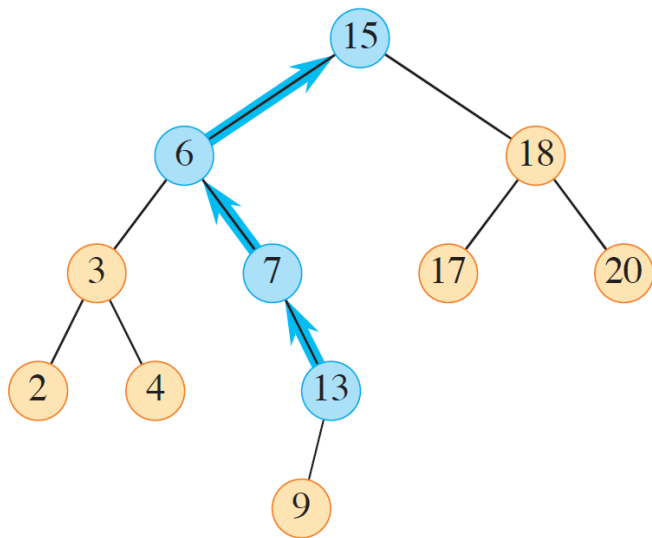
What is the successor of 13?



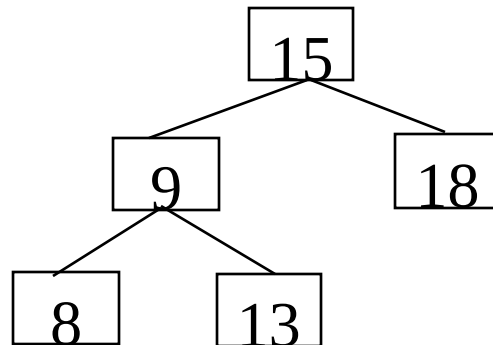
Successor: Find the successor of x in a BST



- Case 2: If the right subtree of x is empty,
i.e., $x.right = \text{NIL}$
 - Travel up from x along the parent pointer until you first see a node which is a left child of its parent, then *return the parent*



What is the successor of 13?

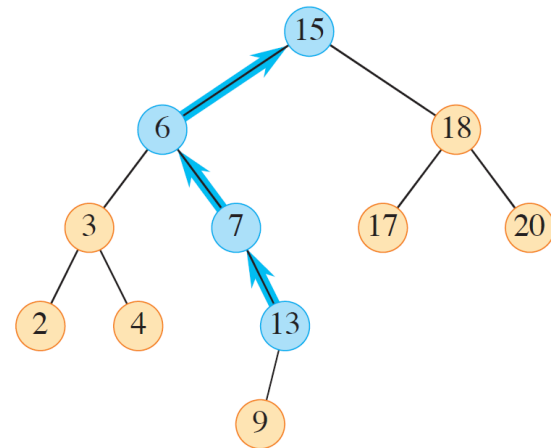
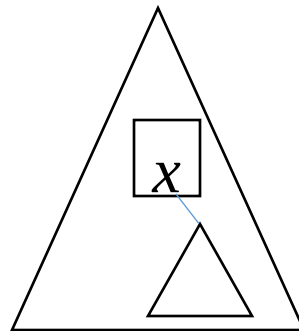


Successor: Find the successor of x in a BST



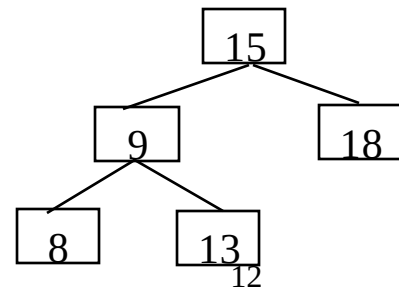
Successor (x)

1. if $x.right \neq \text{NIL}$
2. return Minimum ($x.right$)
3. while $x.p \neq \text{NIL}$ and $x = (x.p).right$
4. $x \leftarrow x.p$
5. return $x.p$



What is the successor of 13?

If the current x is the right child of $x.p$, keep going up
Stop going up when x is the left child of $x.p$
Return $x.p$



Successor: Find the successor of x in a BST



Successor (x)

1. if $x.right \neq \text{NIL}$
2. return Minimum ($x.right$)
3. while $x.p \neq \text{NIL}$ and $x = (x.p).right$
4. $x \leftarrow x.p$
5. return $x.p$

BST allows you to find the successor of a node without comparing keys.

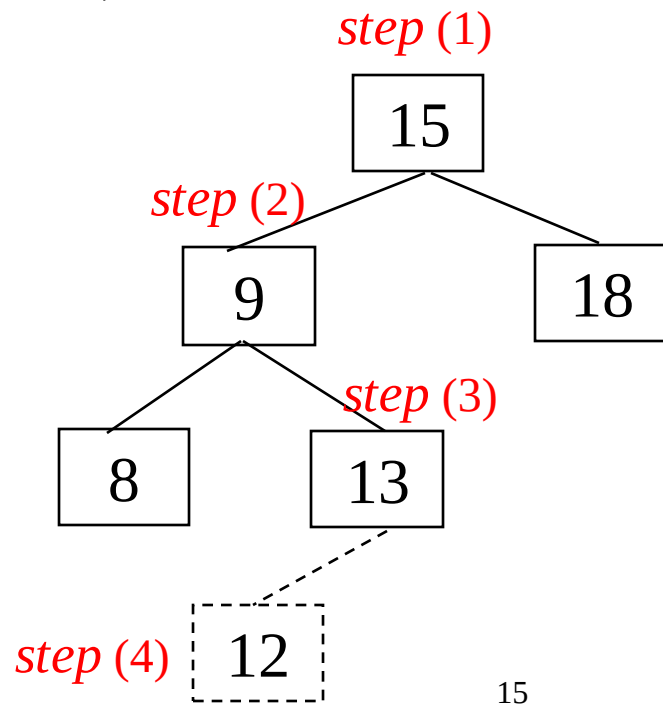


- Maximum (x)
 - Algorithm similar to Minimum (x)
 - Time: $O(h)$
- Predecessor (x)
 - Algorithm similar to Successor (x)
 - Time: $O(h)$

Insertion



- Insert a node z with $z.key$ into the BST
- Search the right position to insert z as a *leaf* in the BST, so that BST property is maintained
- Example:
 - $z.key \leftarrow 12$; $z.left \leftarrow \text{NIL}$; $z.right \leftarrow \text{NIL}$
 - Insert (T, z)



Insertion



Insert (T, z)

1. $y \leftarrow \text{NIL}; \quad x \leftarrow T.\text{root}$

2. while $x \neq \text{NIL}$

3. $y \leftarrow x$

4. if $z.\text{key} < x.\text{key}$

5. $x \leftarrow x.\text{left}$

6. else

7. $x \leftarrow x.\text{right}$

8. $z.p \leftarrow y$

9. if $y = \text{NIL}$

10. $T.\text{root} \leftarrow z$

11. elseif $z.\text{key} < y.\text{key}$

12. $y.\text{left} \leftarrow z$

13. else

14. $y.\text{right} \leftarrow z$

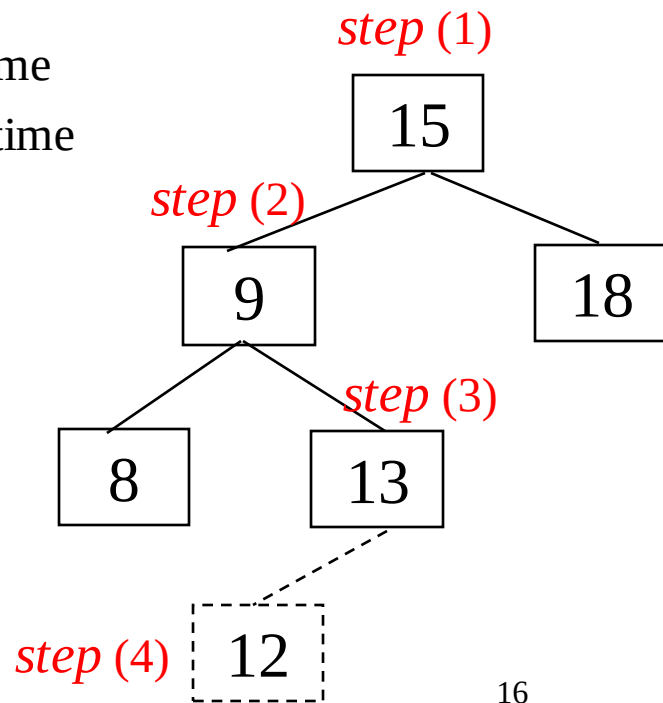
◆ Time complexity

◆ Line 1: $O(1)$ time

◆ Lines 2-7: $O(h)$ time

◆ Lines 8-14: $O(1)$ time

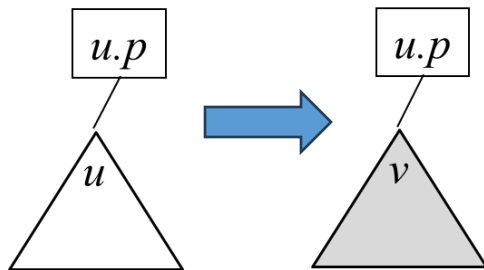
◆ Total: $O(h)$ time



Transplant, a common operation in BST Deletion



Replace the subtree *rooted* at u
by the subtree *rooted* at v



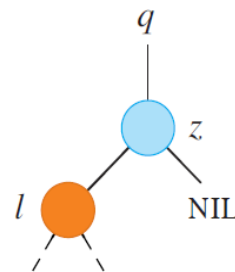
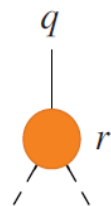
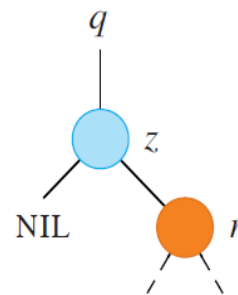
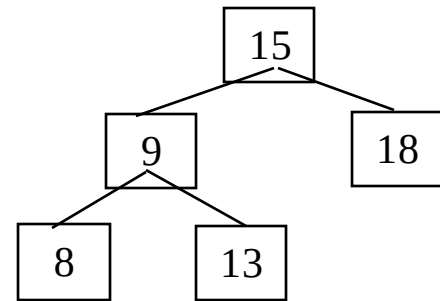
Transplant (T, u, v)

1. if $u.p = \text{NIL}$
2. $T.\text{root} \leftarrow v$
3. elseif $u = u.p.\text{left}$
4. $u.p.\text{left} \leftarrow v$
5. else
6. $u.p.\text{right} \leftarrow v$
7. if $v \neq \text{NIL}$
8. $v.p \leftarrow u.p$

Deletion: Delete node z from a BST



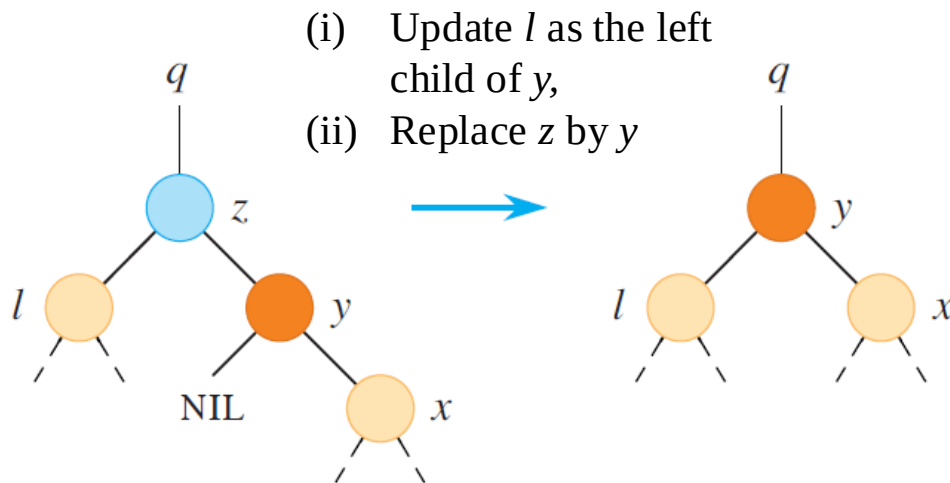
- Case 0: z has no child, i.e., z is a leaf
 - E.g., 13
 - (Included by Case 1 when r is NIL)
- Case 1, z has no left child: replace z by its right child r
 - $\text{Transplant}(T, z, z.\text{right})$
- Case 2, z has no right child: replace z by its left child l
 - $\text{Transplant}(T, z, z.\text{left})$



Deletion: Delete node z from a BST



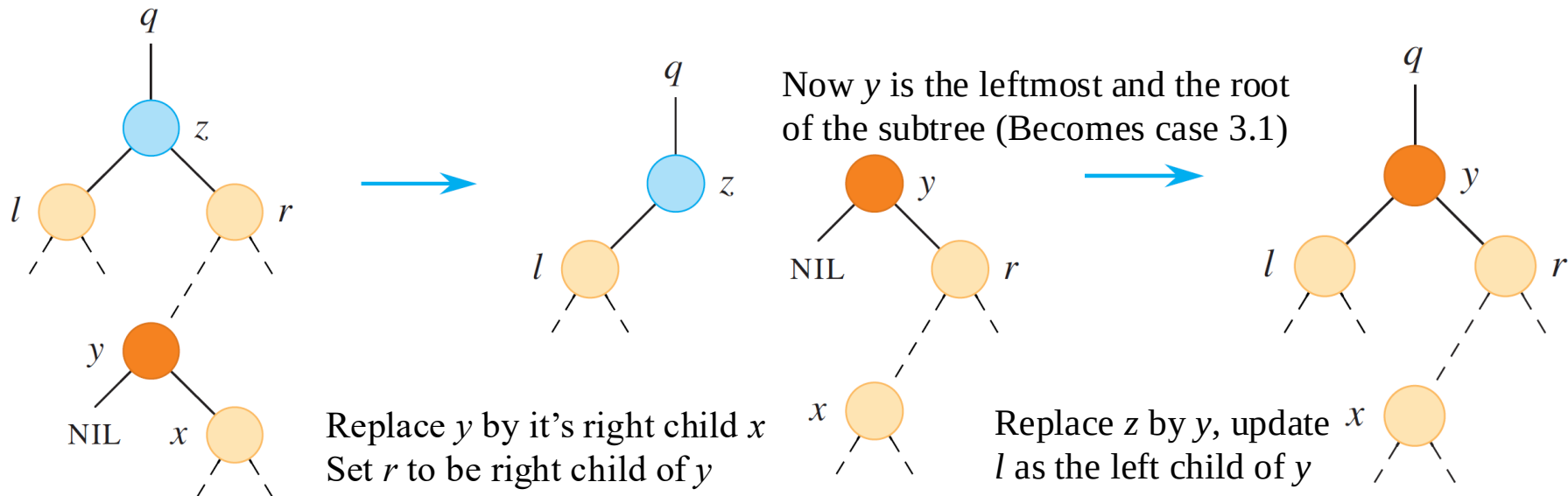
- Case 3: z has two children
 - Find z 's successor y , which must be in z 's right subtree and has no left child (y is the leftmost node in the right subtree (y 's left child must be NIL))
 - Case 3.1: If y is the right child of z
 - $l < z < y < x$



Deletion: Delete node z from a BST



- Case 3: z has two children
 - Find z 's successor y , which must be in z 's right subtree and has no left child (y is the leftmost node in the right subtree (y 's left child must be NIL))
 - Case 3.2: y is in the right subtree, but not right child of z
 - $l < z < y < x < r$



Deletion: Delete node z from a BST



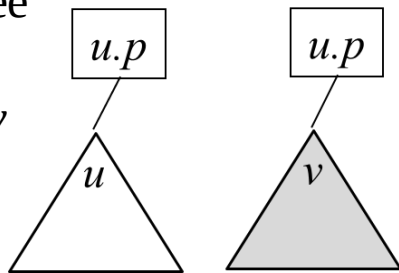
Transplant (T, u, v)

1. if $u.p = \text{NIL}$
2. $T.\text{root} \leftarrow v$
3. elseif $u = (u.p).\text{left}$
4. $(u.p).\text{left} \leftarrow v$
5. else
6. $(u.p).\text{right} \leftarrow v$
7. if $v \neq \text{NIL}$
8. $v.p \leftarrow u.p$

Delete (T, z)

1. if $z.\text{left} = \text{NIL}$
2. Transplant($T, z, z.\text{right}$)
3. elseif $z.\text{right} = \text{NIL}$
4. Transplant($T, z, z.\text{left}$)
5. else
6. $y \leftarrow \text{Minimum}(z.\text{right})$
7. If $y \neq z.\text{right}$ //case 3.2
8. Transplant($T, y, y.\text{right}$)
9. $y.\text{right} \leftarrow z.\text{right}$
10. $y.\text{right}.p \leftarrow y$
11. Transplant(T, z, y) //common operations of case 3.1, 3.2
12. $y.\text{left} \leftarrow z.\text{left}$
13. $y.\text{left}.p \leftarrow y$

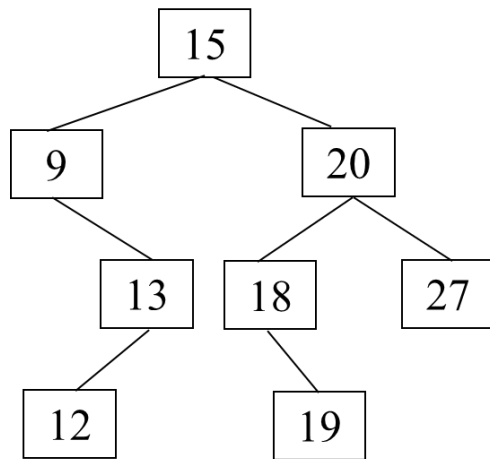
Replace the subtree
rooted at u by the
subtree rooted at v



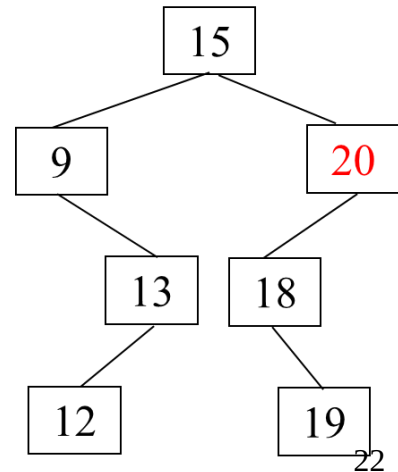
Deletion



- Example: Delete (T , node₂₇)
 - Find the parent node of “27”
 - Set its right child to “NIL”,
i.e., delete the node “27”



before deletion

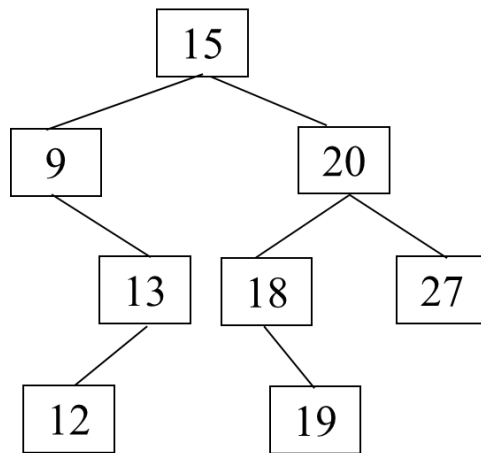


after deletion

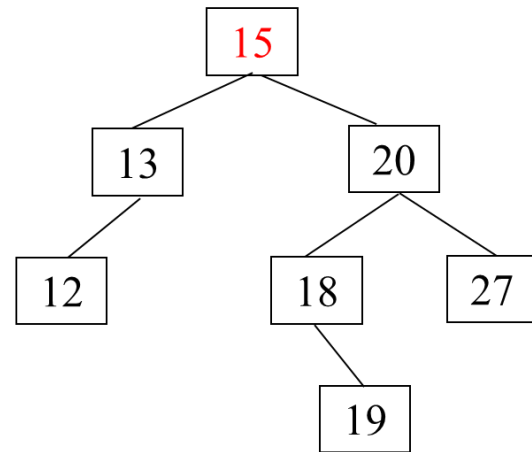
Deletion



- Example: Delete ($T, node_9$)
 - Node_9 has no left child
 - Transplant 9 by its right subtree



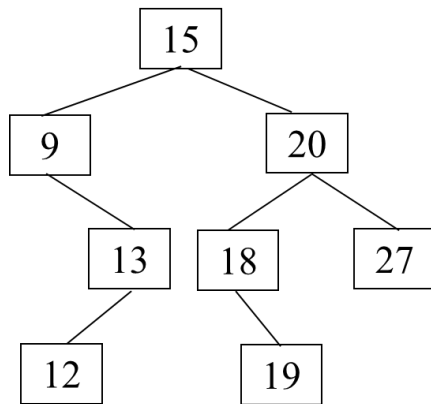
before deletion



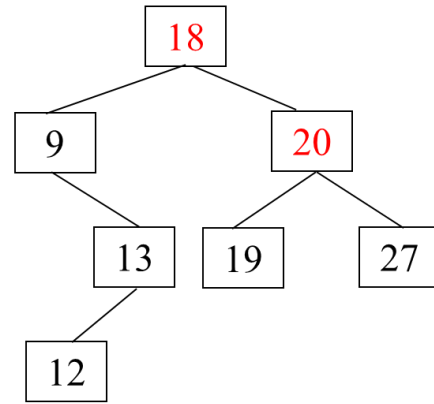
after deletion



- Example: Delete ($T, node_{15}$)
 - Case 3.2 in previous slides



before deletion



after deletion



Delete (T, z)

1. if $z.left = \text{NIL}$
2. Transplant($T, z, z.right$)
3. elseif $z.right = \text{NIL}$
4. Transplant($T, z, z.left$)
5. else
6. $y \leftarrow \text{Minimum}(z.right)$
7. If $y \neq z.right$ //case 3.2
8. Transplant($T, y, y.right$)
9. $y.right \leftarrow z.right$
10. $y.right.p \leftarrow y$
11. Transplant(T, z, y) //common operations of case 3.1, 3.2
12. $y.left \leftarrow z.left$
13. $y.left.p \leftarrow y$

◆ Transplant: $O(1)$ time

◆ Delete

- ◆ Lines 1-4: $O(1)$ time
- ◆ Line 6: $O(h)$ time
- ◆ Line 7-10: $O(1)$ time
- ◆ Line 11-13: $O(1)$ time
- ◆ Total: $O(h)$ time



- Binary Tree
- Binary Tree Walk
- Binary Search Tree (BST)
- BST Operations

Please read Chapter 12 Binary Search Trees in the textbook

Quiz instructions



- Write down your name and student ID **clearly**
- 5 questions.
- 50 minutes.
- **Write your answer directly on the quiz paper.**
- 2 empty drafting papers (do not write answer on this)
- We will get drafting papers and quiz papers back.
- Do not distribute the questions.
- Put your student ID card on the desk.
- Sit apart from each other