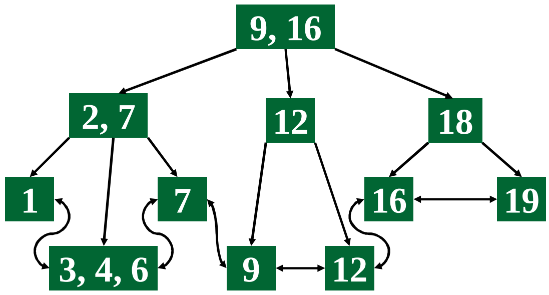
B+ Trees

- Similar to B trees, with a few slight differences
- All data is stored at the leaf nodes (*leaf pages*); all other nodes (*index pages*) only store keys
- Leaf pages are linked to each other
- Keys may be duplicated; every key to the right of a particular key is >= to that key

B+ Tree Example

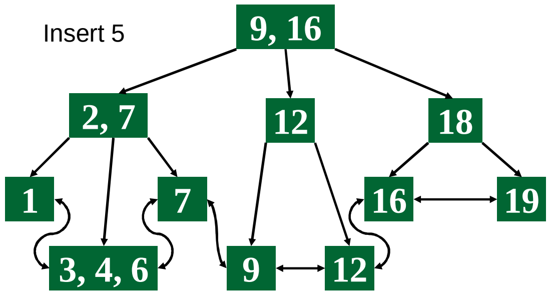


B+ Tree Insertion

- Insert at bottom level
- If leaf page overflows, split page and copy middle element to next index page
- If index page overflows, split page and move middle element to next index page

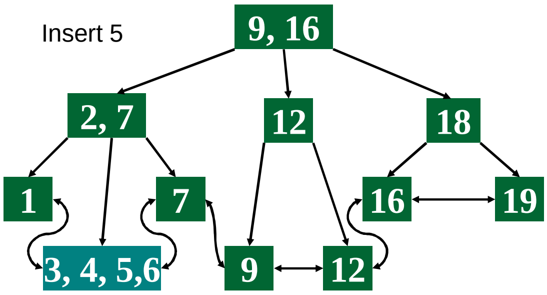
B+ Tree Insertion Example

Insert 5



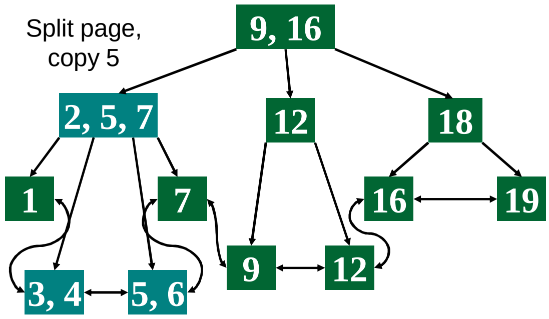
B+ Tree Insertion Example

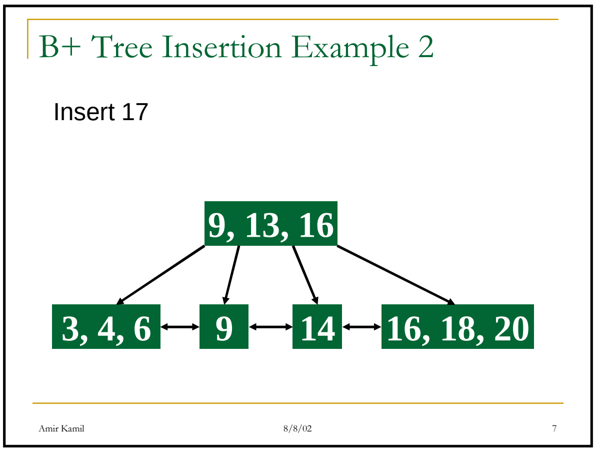
Insert 5



B+ Tree Insertion Example

Split page, copy 5





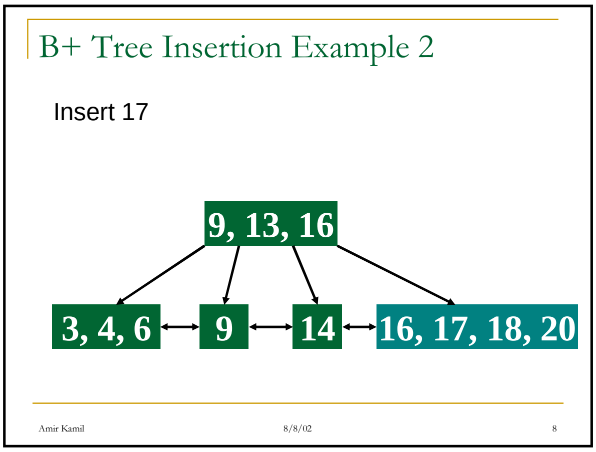
B+ Tree Insertion Example 2

Insert 17

9, 13, 16

3, 4, 6 ↔ 9 ↔ 14 ↔ 16, 18, 20

Amir Kamil 8/8/02 7



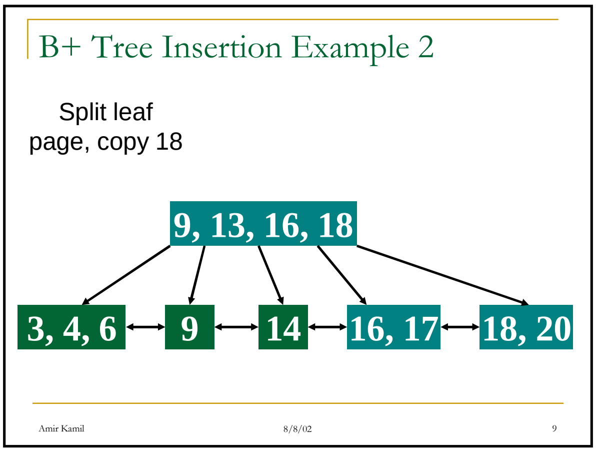
B+ Tree Insertion Example 2

Insert 17

9, 13, 16

3, 4, 6 ↔ 9 ↔ 14 ↔ 16, 17, 18, 20

Amir Kamil 8/8/02 8



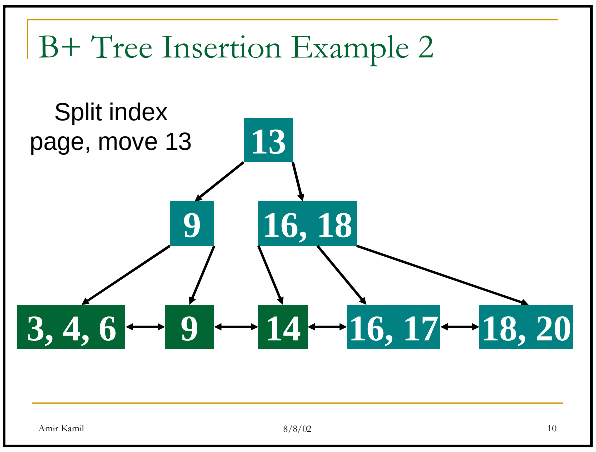
B+ Tree Insertion Example 2

Split leaf page, copy 18

9, 13, 16, 18

3, 4, 6 ↔ 9 ↔ 14 ↔ 16, 17 ↔ 18, 20

Amir Kamil 8/8/02 9



B+ Tree Insertion Example 2

Split index page, move 13

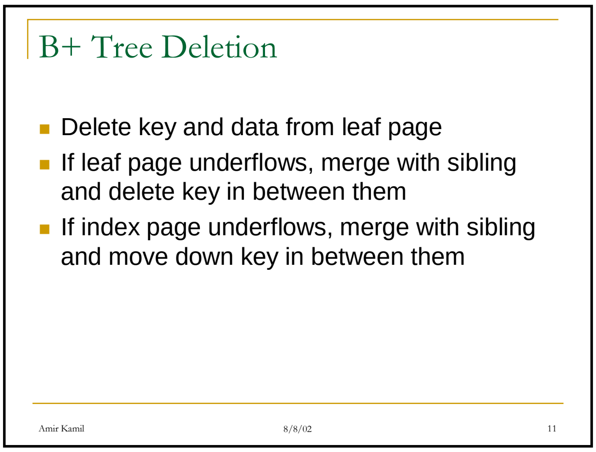
13

9

16, 18

3, 4, 6 ↔ 9 ↔ 14 ↔ 16, 17 ↔ 18, 20

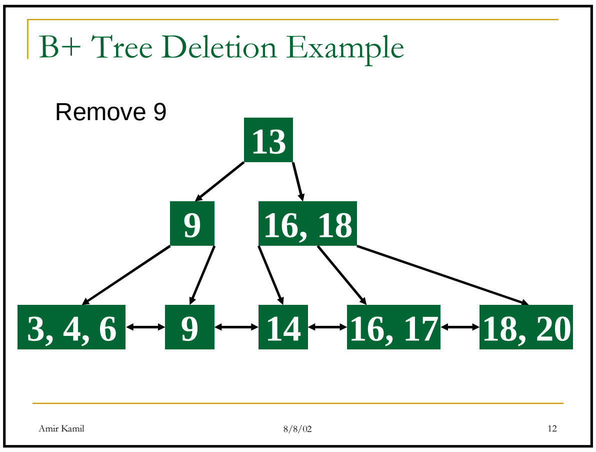
Amir Kamil 8/8/02 10



B+ Tree Deletion

- Delete key and data from leaf page
- If leaf page underflows, merge with sibling and delete key in between them
- If index page underflows, merge with sibling and move down key in between them

Amir Kamil 8/8/02 11



B+ Tree Deletion Example

Remove 9

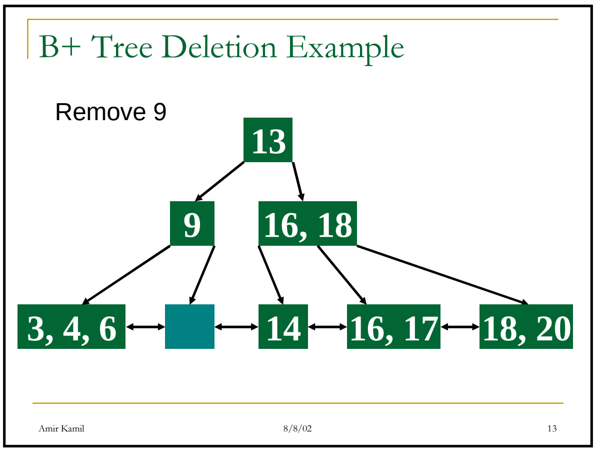
13

9

16, 18

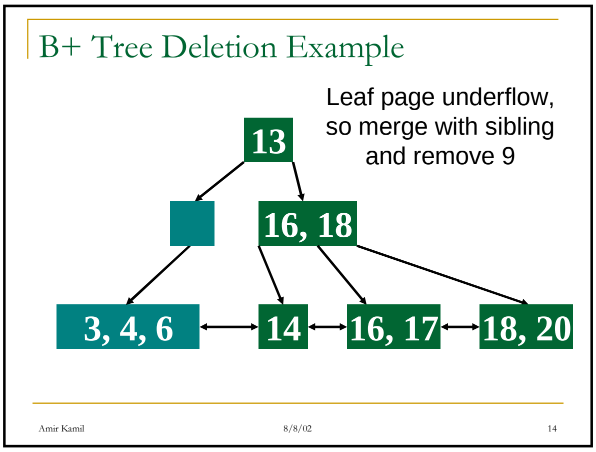
3, 4, 6 ↔ 9 ↔ 14 ↔ 16, 17 ↔ 18, 20

Amir Kamil 8/8/02 12



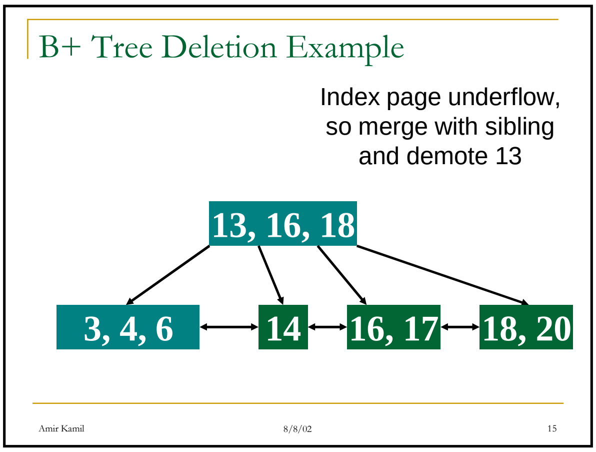
B+ Tree Deletion Example

Remove 9



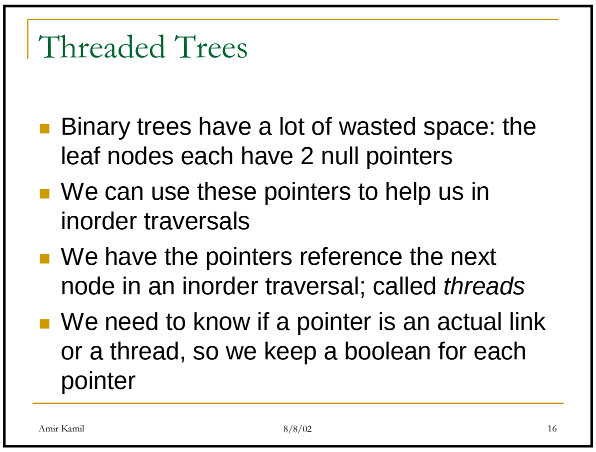
B+ Tree Deletion Example

Leaf page underflow,
so merge with sibling
and remove 9



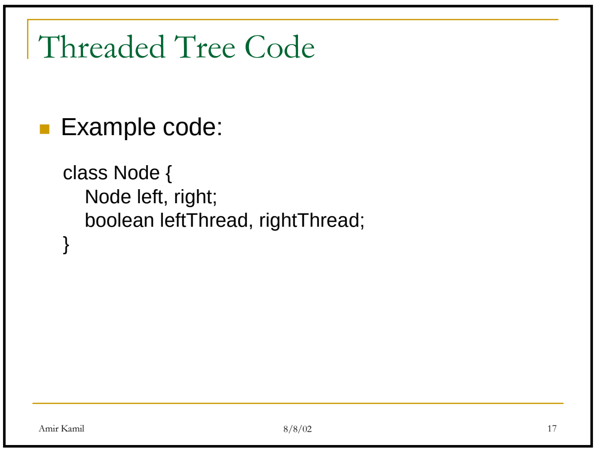
B+ Tree Deletion Example

Index page underflow,
so merge with sibling
and demote 13



Threaded Trees

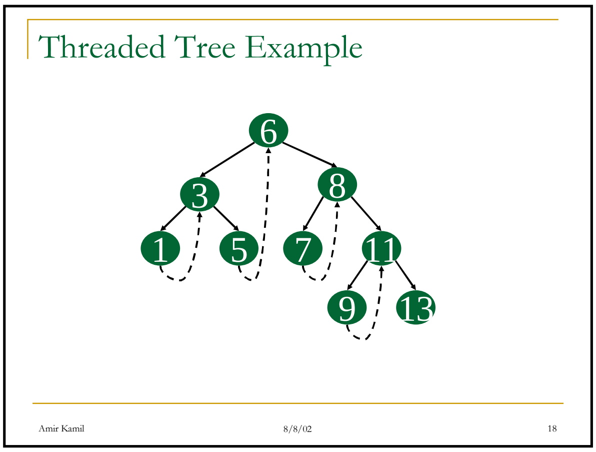
- Binary trees have a lot of wasted space: the leaf nodes each have 2 null pointers
- We can use these pointers to help us in inorder traversals
- We have the pointers reference the next node in an inorder traversal; called *threads*
- We need to know if a pointer is an actual link or a thread, so we keep a boolean for each pointer



Threaded Tree Code

- Example code:

```
class Node {
    Node left, right;
    boolean leftThread, rightThread;
}
```

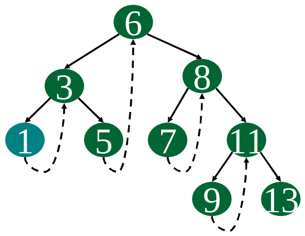


Threaded Tree Example

Threaded Tree Traversal

- We start at the leftmost node in the tree, print it, and follow its right thread
- If we follow a thread to the right, we output the node and continue to its right
- If we follow a link to the right, we go to the leftmost node, print it, and continue

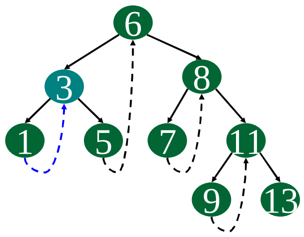
Threaded Tree Traversal



Output
1

Start at leftmost node, print it

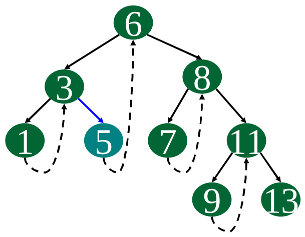
Threaded Tree Traversal



Output
1
3

Follow thread to right, print node

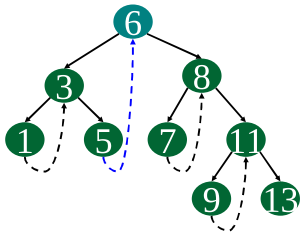
Threaded Tree Traversal



Output
1
3
5

Follow link to right, go to
leftmost node and print

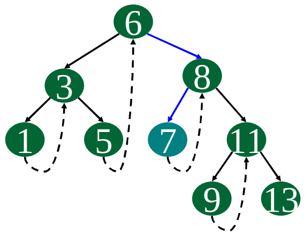
Threaded Tree Traversal



Output
1
3
5
6

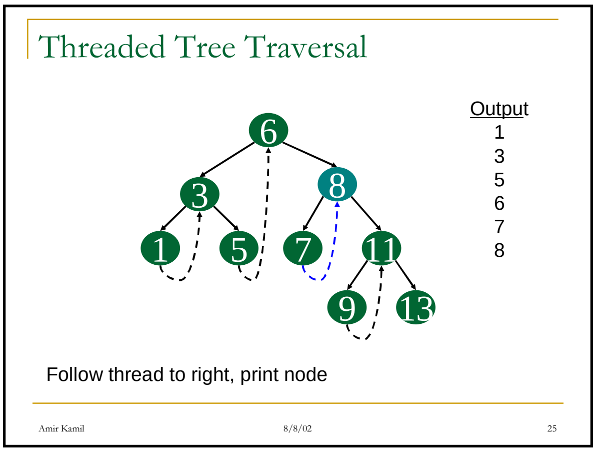
Follow thread to right, print node

Threaded Tree Traversal



Output
1
3
5
6
7

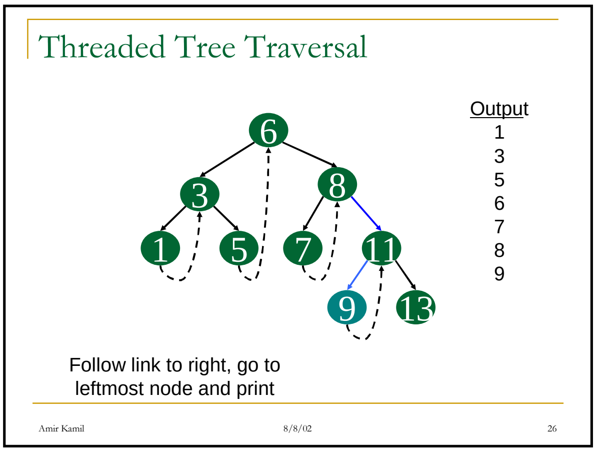
Follow link to right, go to
leftmost node and print



Threaded Tree Traversal

Output
1
3
5
6
7
8

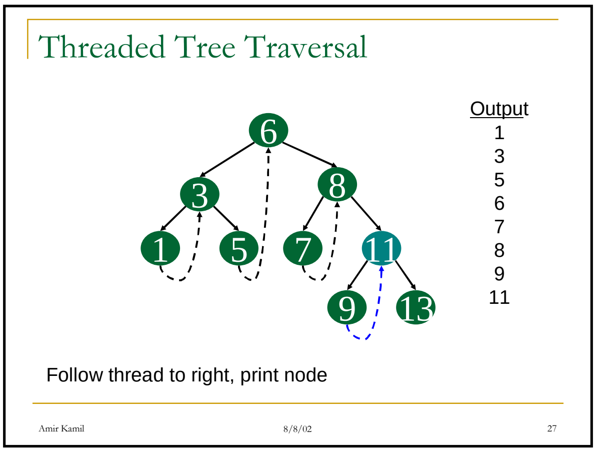
Follow thread to right, print node



Threaded Tree Traversal

Output
1
3
5
6
7
8
9

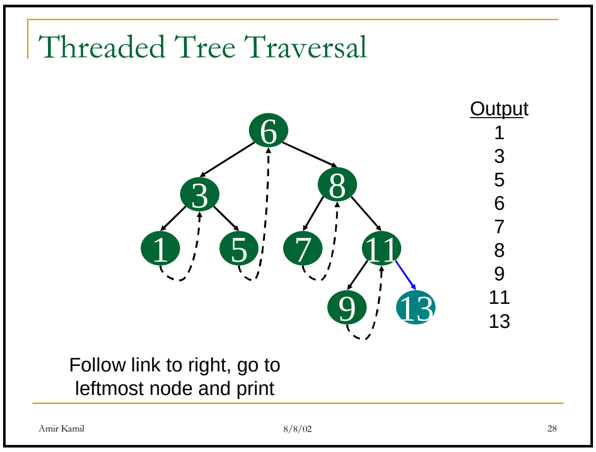
Follow link to right, go to leftmost node and print



Threaded Tree Traversal

Output
1
3
5
6
7
8
9
11

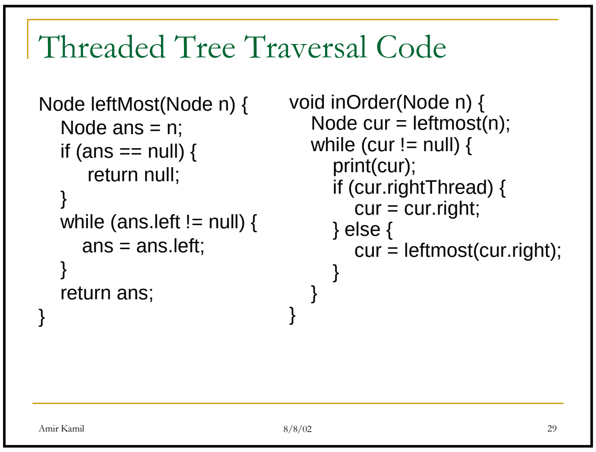
Follow thread to right, print node



Threaded Tree Traversal

Output
1
3
5
6
7
8
9
11
13

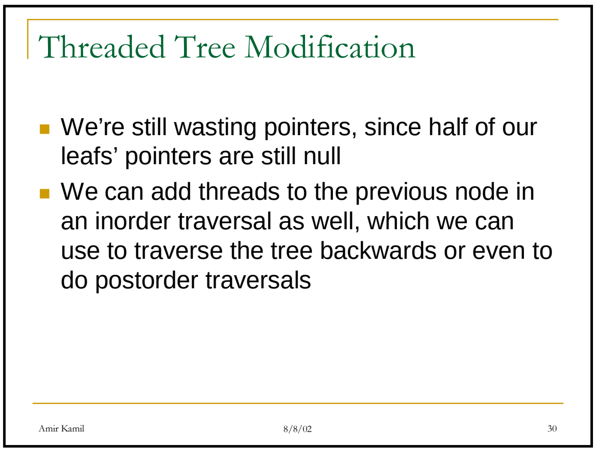
Follow link to right, go to leftmost node and print



Threaded Tree Traversal Code

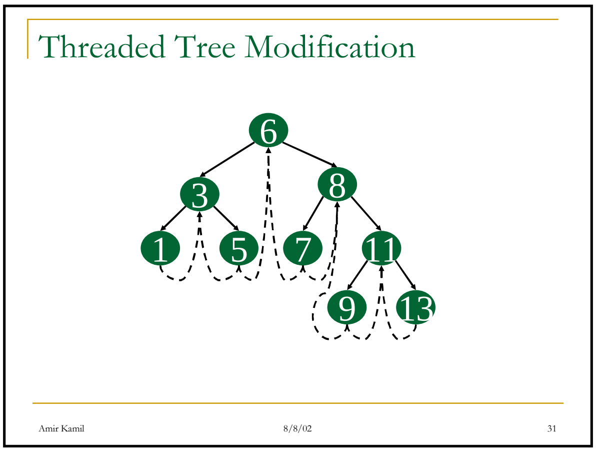
```
Node leftMost(Node n) {
    Node ans = n;
    if (ans == null) {
        return null;
    }
    while (ans.left != null) {
        ans = ans.left;
    }
    return ans;
}
```

```
void inOrder(Node n) {
    Node cur = leftmost(n);
    while (cur != null) {
        print(cur);
        if (cur.rightThread) {
            cur = cur.right;
        } else {
            cur = leftmost(cur.right);
        }
    }
}
```



Threaded Tree Modification

- We’re still wasting pointers, since half of our leafs’ pointers are still null
- We can add threads to the previous node in an inorder traversal as well, which we can use to traverse the tree backwards or even to do postorder traversals



Threaded Tree Modification