COMP90038 Algorithms and Complexity

Lecture 12: More Divide-and-Conquer Algorithms (with thanks to Harald Søndergaard)

Toby Murray

toby.murray@unimelb.edu.au

DMD 8.17 (Level 8, Doug McDonell Bldg)

http://people.eng.unimelb.edu.au/tobym

@tobycmurray

Divide and Conquer

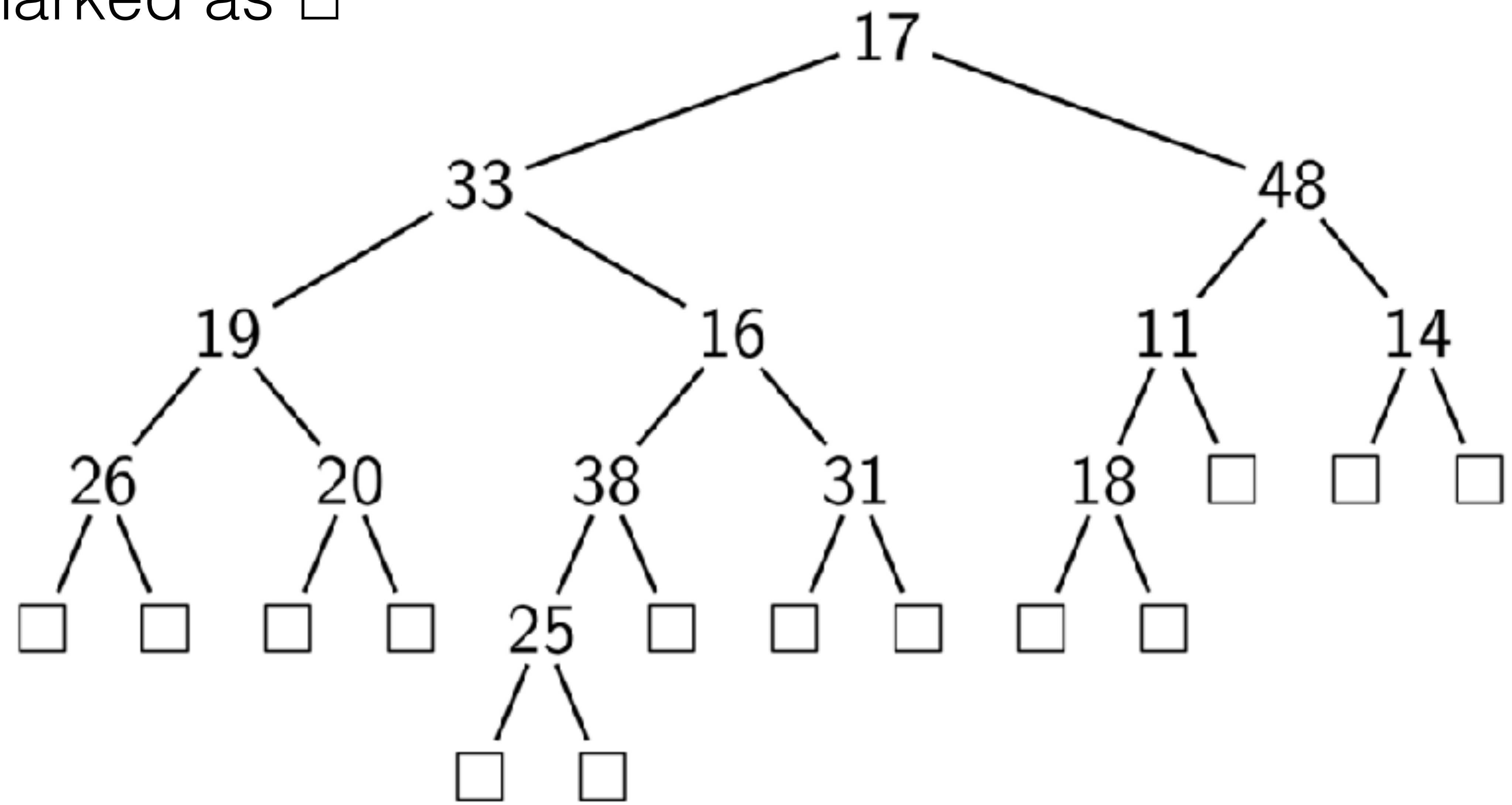
- In the last lecture we studied the archetypal divide-and-conquer sorting algorithms: mergesort and quicksort.

- We also introduced the powerful **master theorem**, providing solutions to a large class of recurrence relations, for free.
  - allows us to quickly determine the complexity of these divide-and-conquer algorithms

- Now we shall look at tree traversal, and then a final example of divide-and-conquer, giving a better solution to the closest-pair problem.

Binary Trees

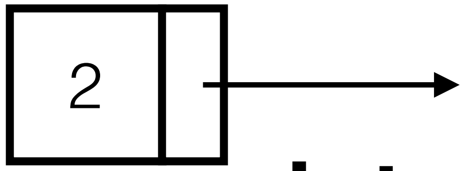
- An example of a **binary tree**, with empty subtrees marked as □



- This tree has **height** 4, the empty tree having height -1

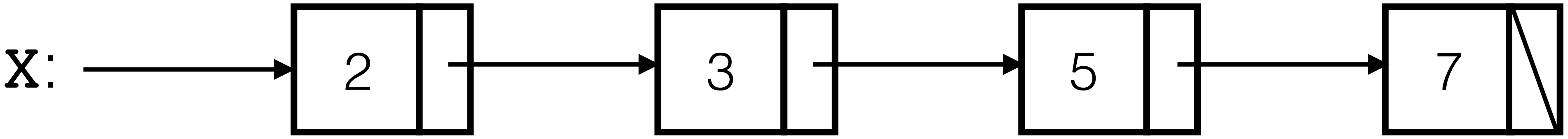
Review of Linked List Terminology

node

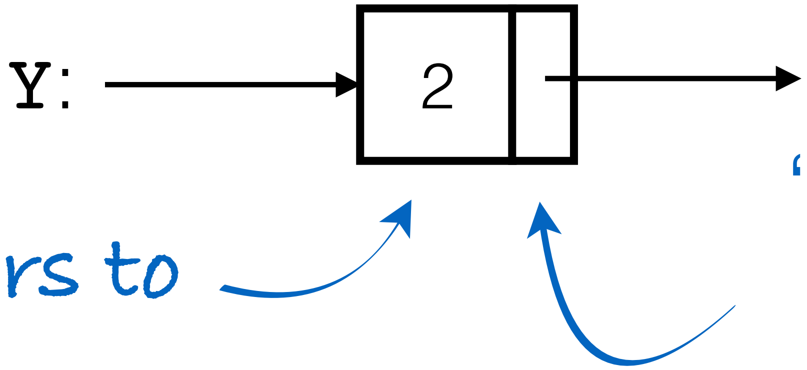


pointer

(in Java: “reference”)



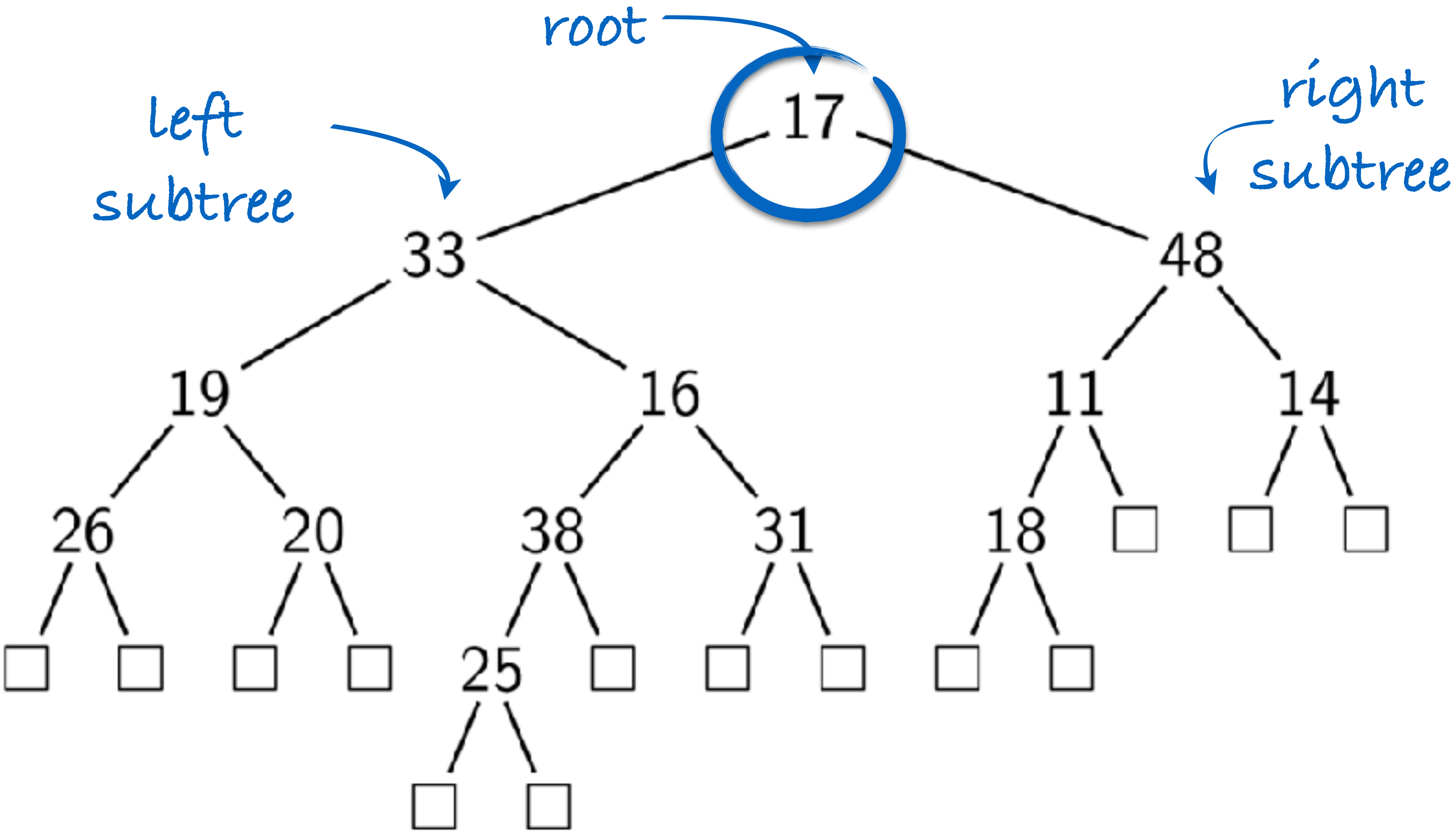
`X` is (a pointer to) the **head node** of the list



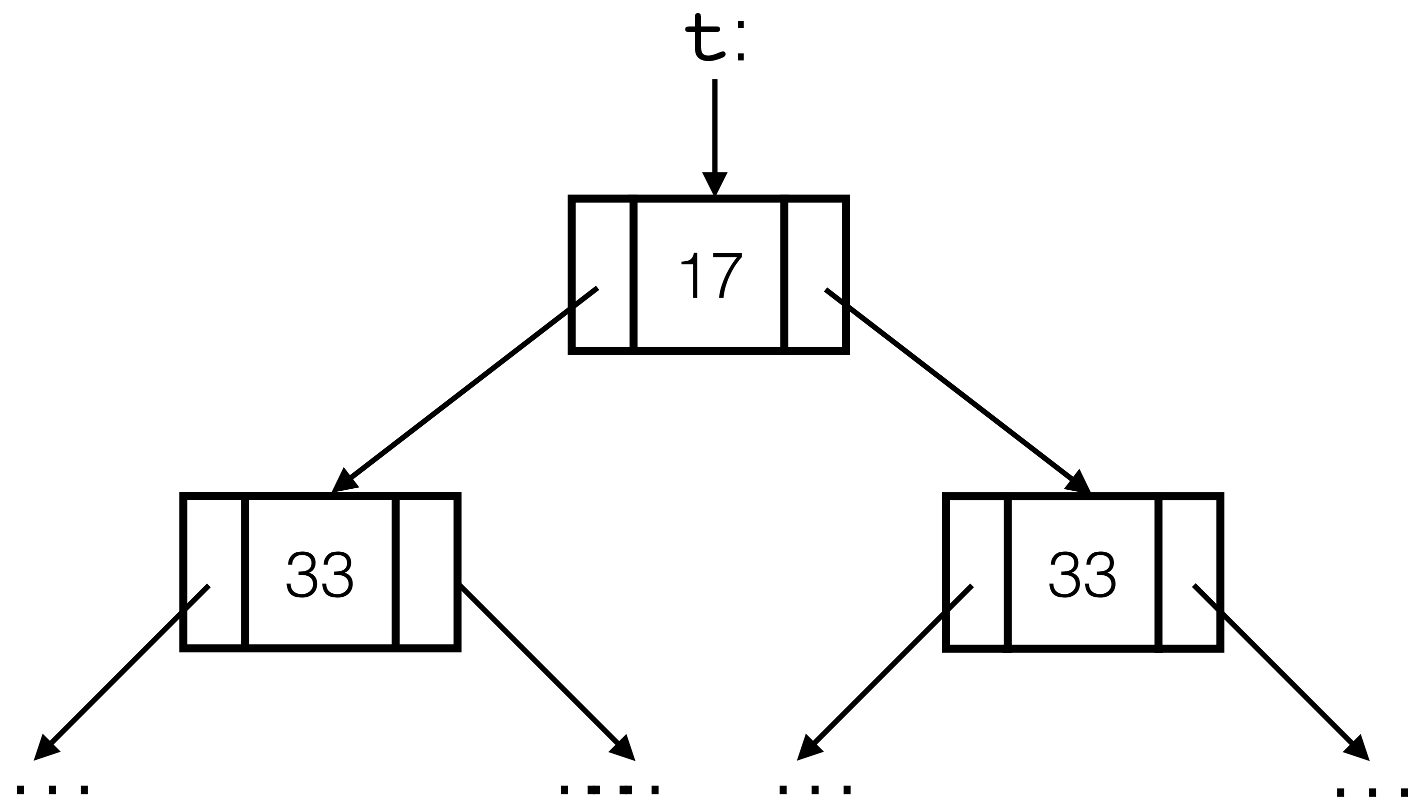
“`Y.val`” *refers to*

“`Y.next`” *refers to*

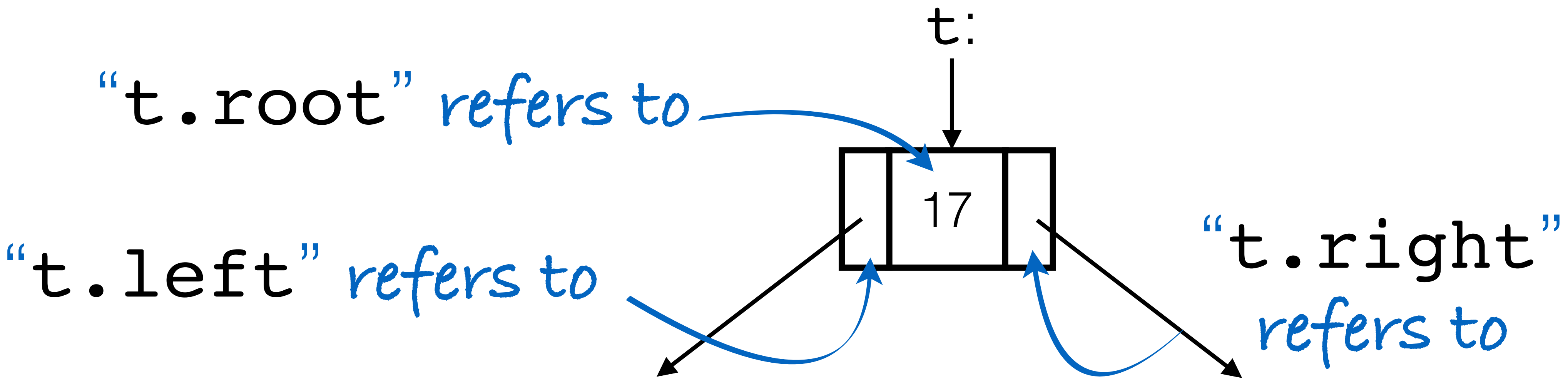
Tree Terminology



Tree Terminology



`t` is (a pointer to) the **root node** of the tree



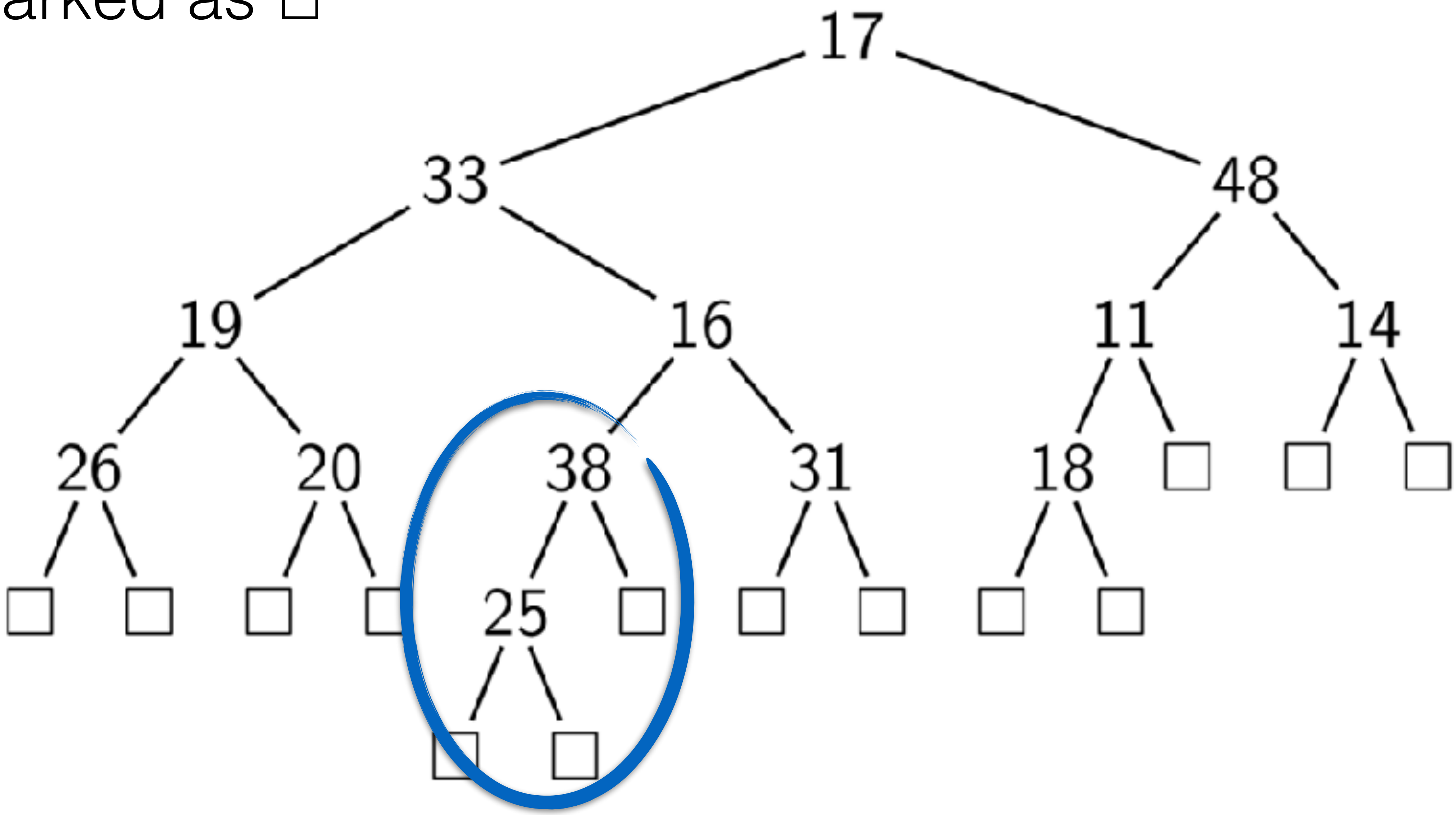
“`t.root`” *refers to*

“`t.left`” *refers to*

“`t.right`” *refers to*

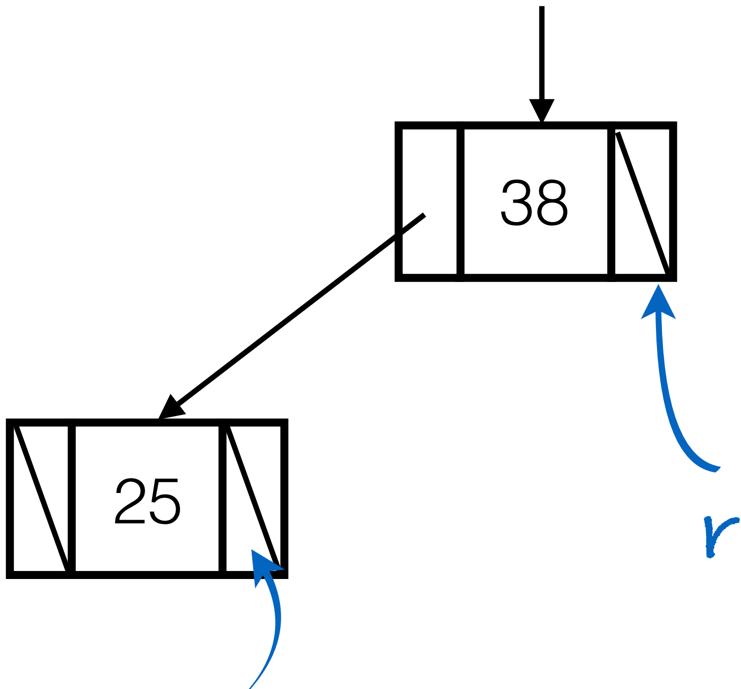
Binary Trees

- An example of a **binary tree**, with empty subtrees marked as □



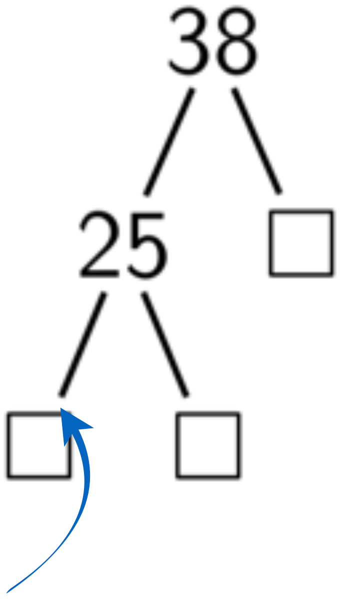
- This tree has **height** 4, the empty tree having height -1

Empty Nodes



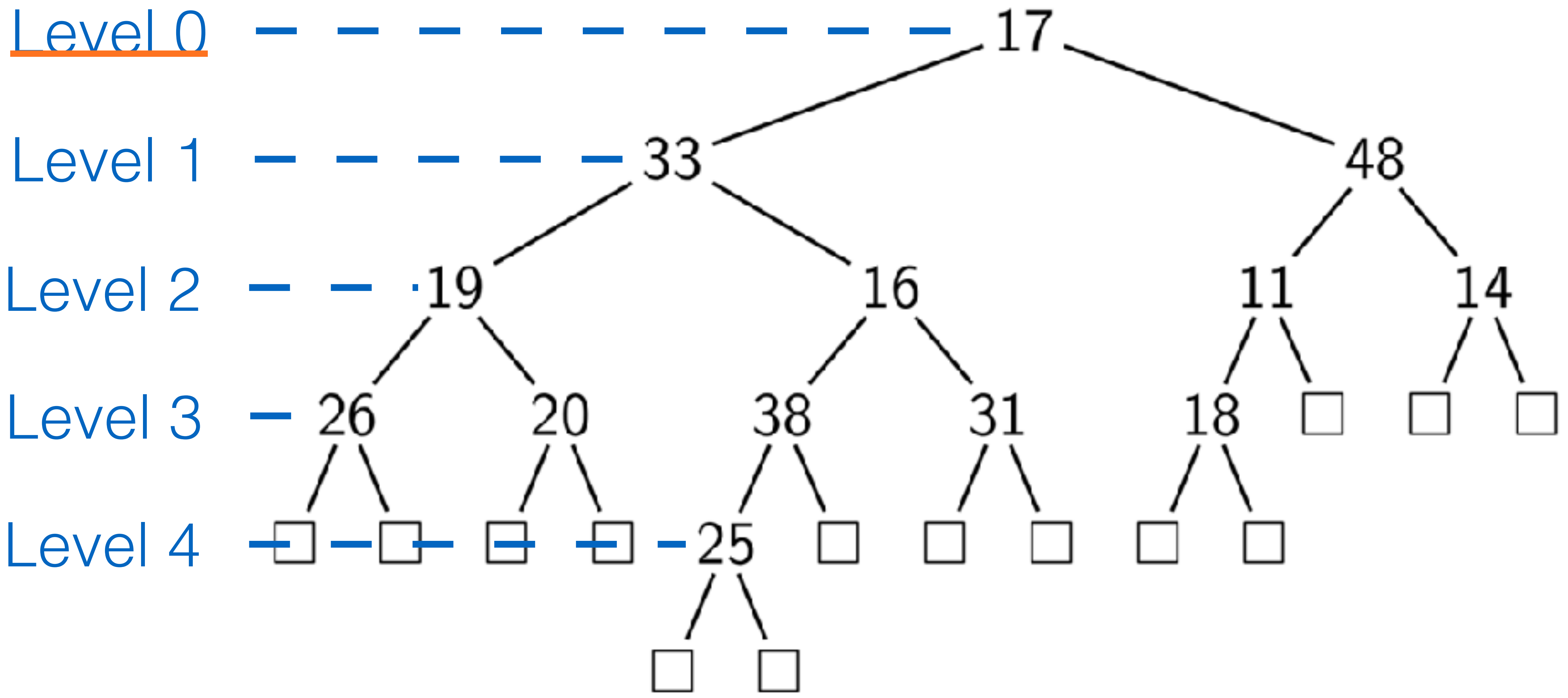
right subtree is empty

empty trees are just null pointers



null pointer

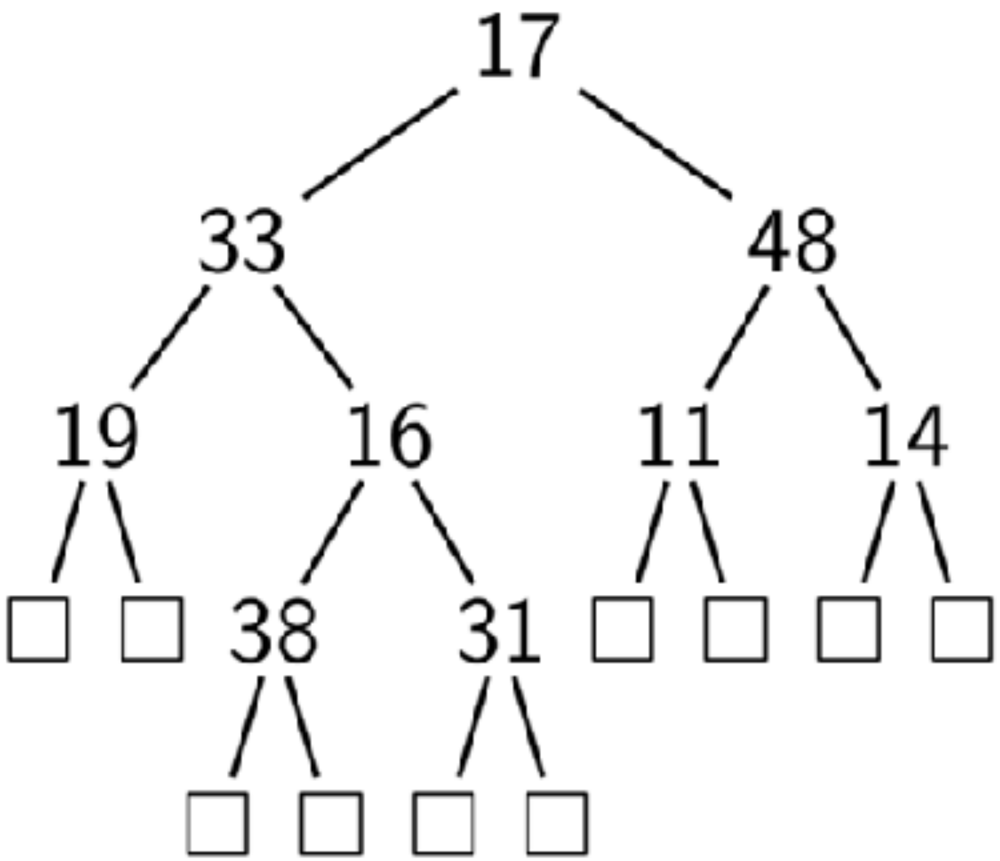
Levels and Height



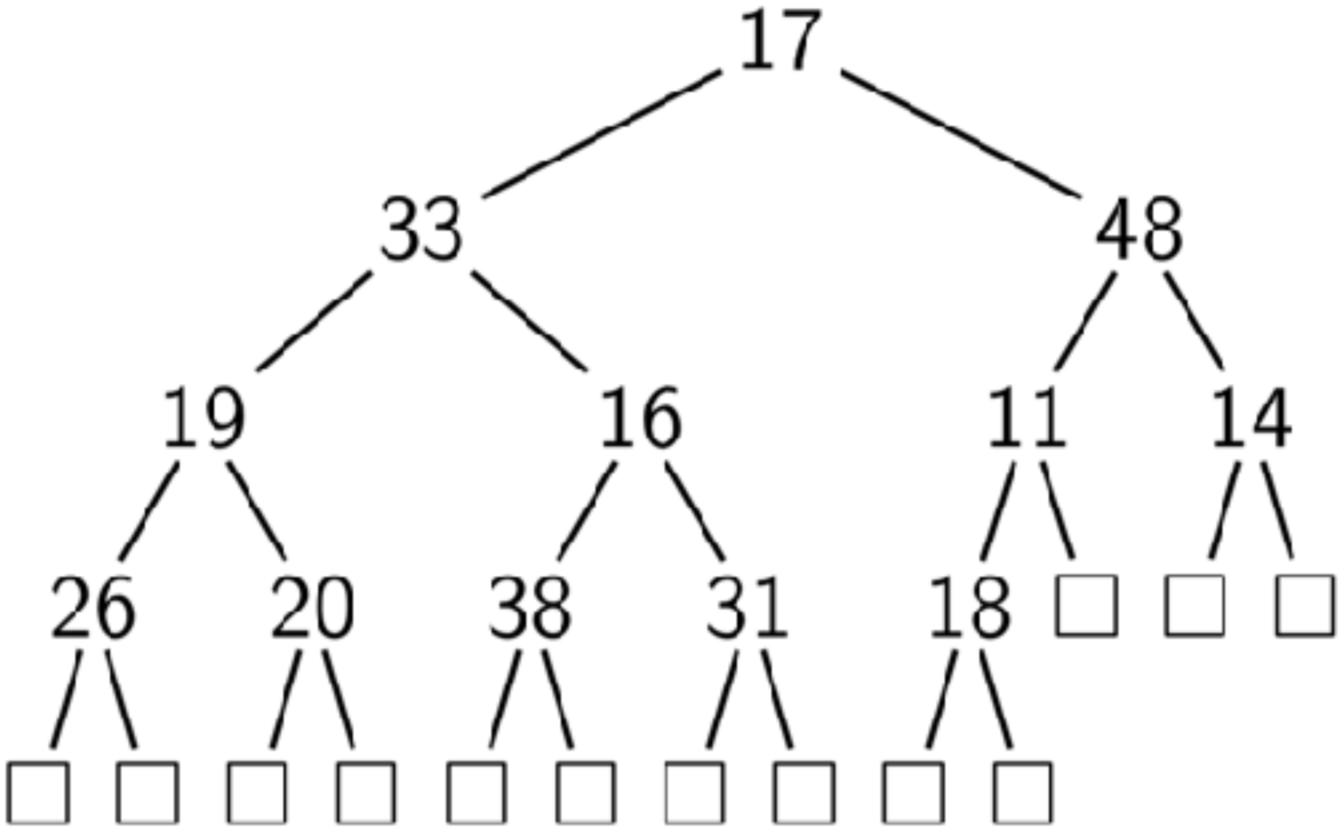
So the tree has **height** 4 (its **maximum level**)

Binary Tree Concepts

- Special trees have their **external nodes** □ only at level *h* and *h*+1 for some *h*:



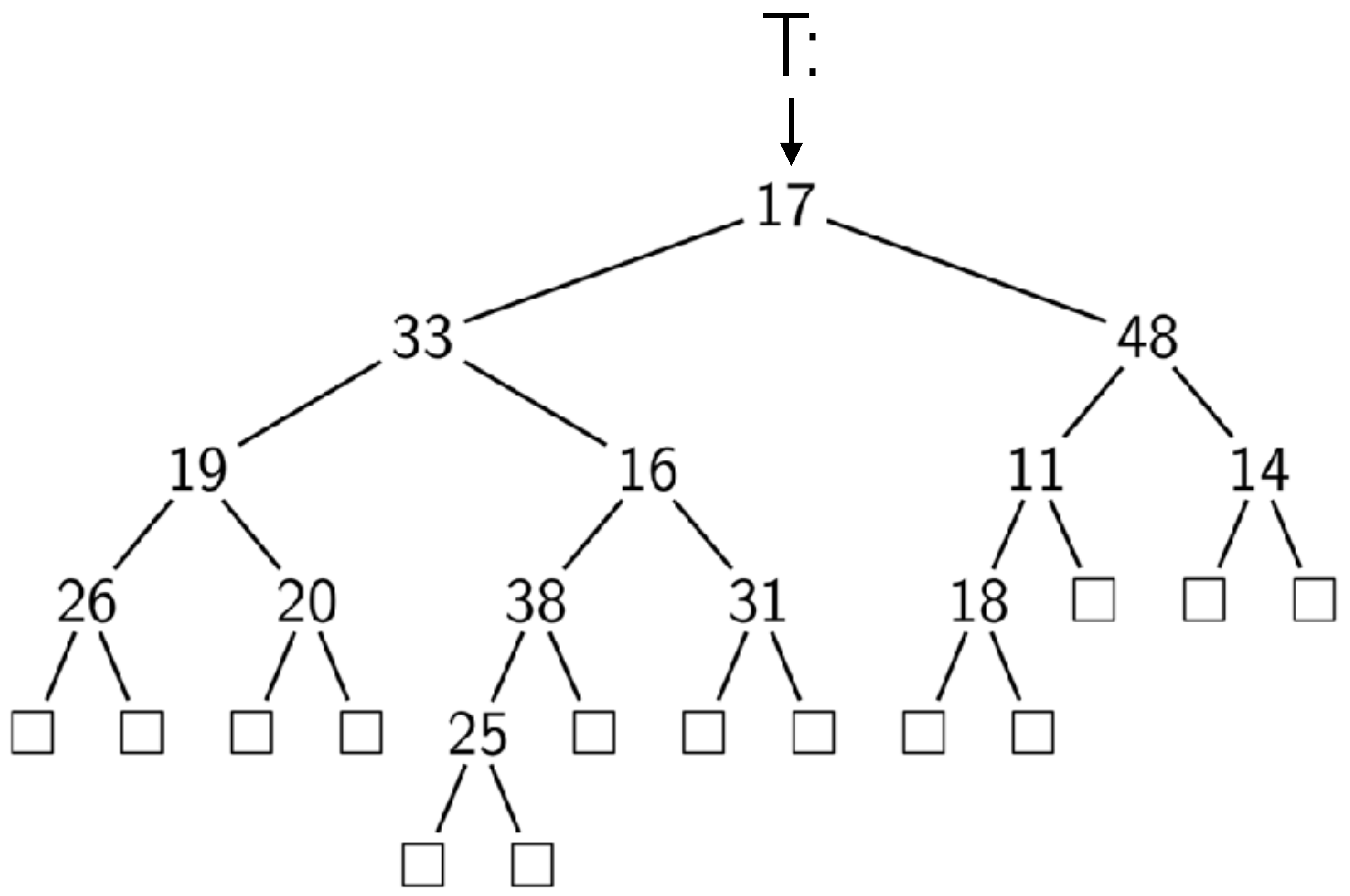
A **full** binary tree: Each node has 0 or 2 (non-empty) children.



A **complete** tree: Each level filled left to right. (Every level except perhaps the last is completely filled.)

Calculating the Height

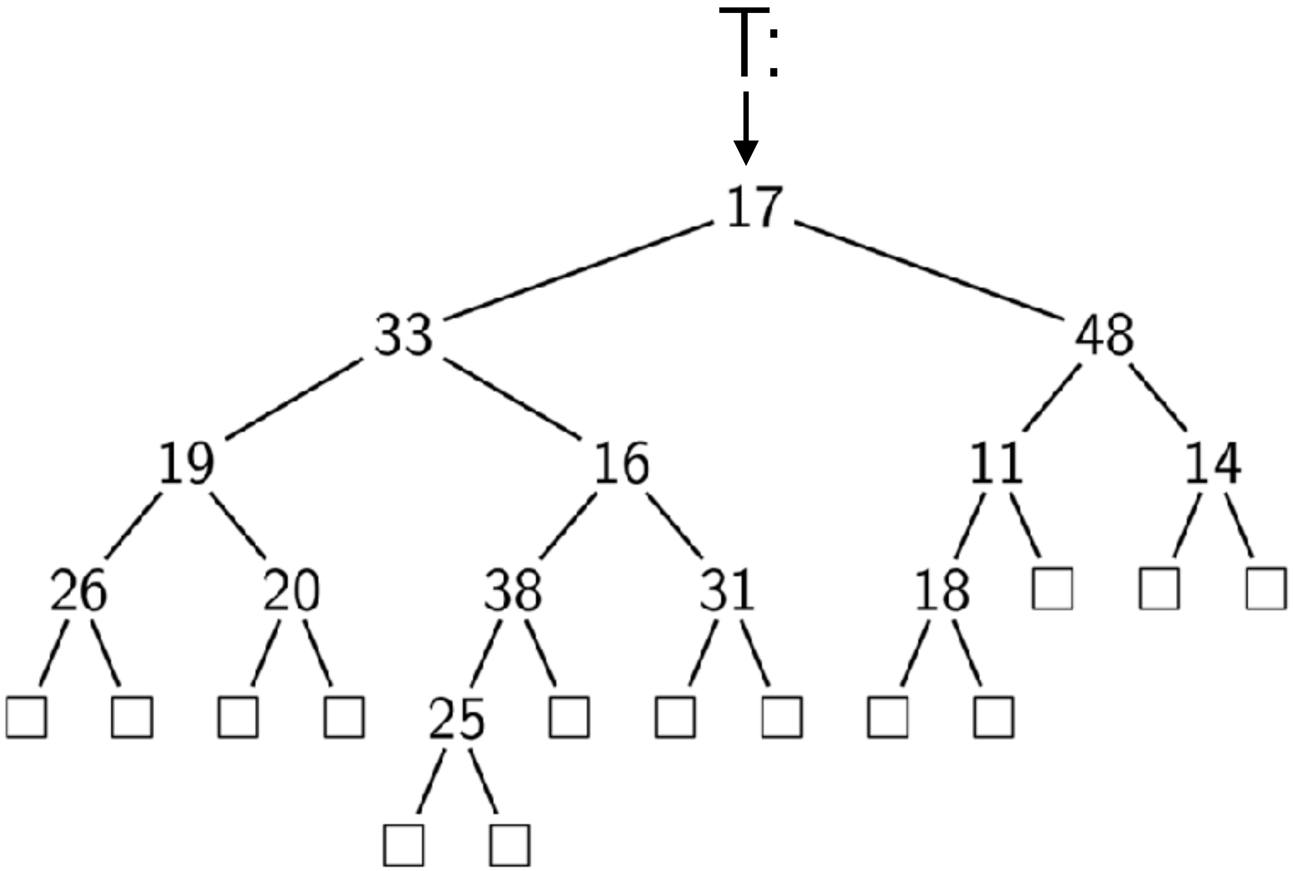
- Recursion is the natural way of calculating the **height**:



```
function HEIGHT($T$)
    if $T = null$ then
        return $-1$
    else
        return $max(\text{HEIGHT}(T.left), \text{HEIGHT}(T.right)) + 1$
```

Height Complexity

- Input size: number *n* of (internal) nodes (e.g. for T *n* is 13)
- Number of external nodes **always** *n*+1 (e.g. for T *x* is 14)
- The function HEIGHT makes one tree comparison (is T null/empty?) per node (internal and external), so altogether 2n + 1 comparisons.



Binary Tree Traversal

- **Preorder** traversal visits the root, then the left subtree, and finally the right subtree.

- **Inorder** traversal visits the left subtree, then the root, and finally the right subtree.

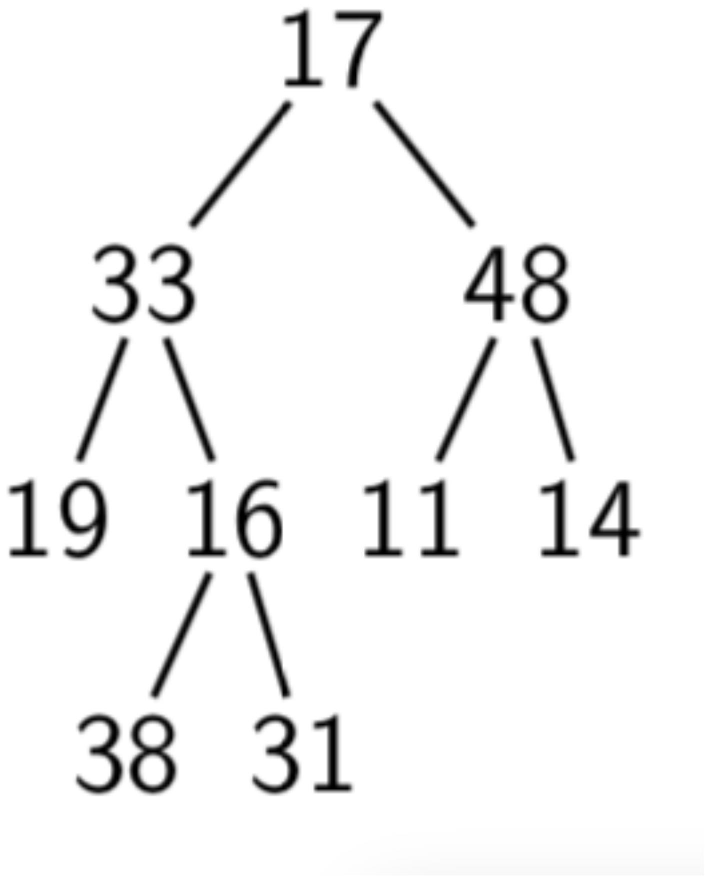
- **Postorder** traversal visits the left subtree, the right subtree, and finally the root.

- **Level-order** traversal visits the nodes, level by level, starting from the root.

Preorder Traversal

Visit order:

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



先左后右

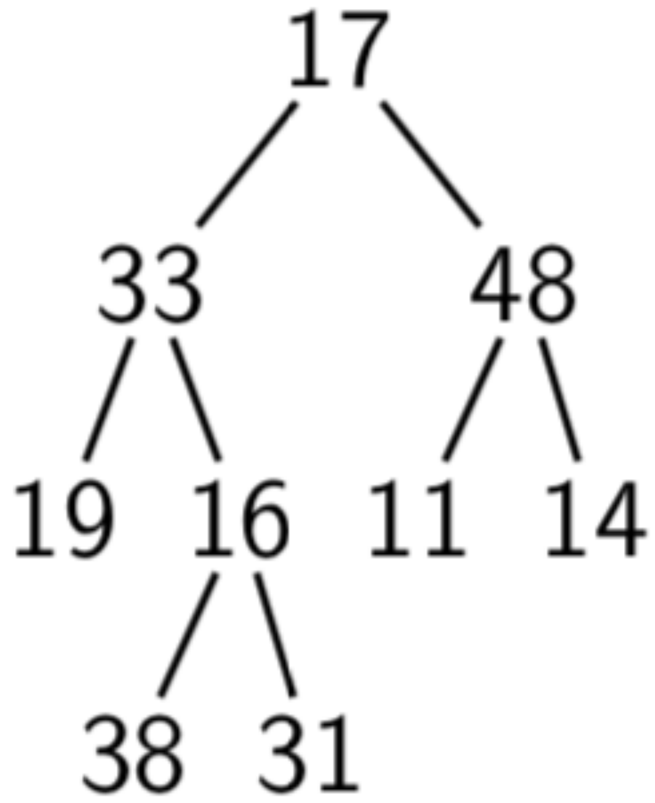
PREORDERTRAVERSE(17)

Call Stack

Preorder Traversal

Visit order: 17

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



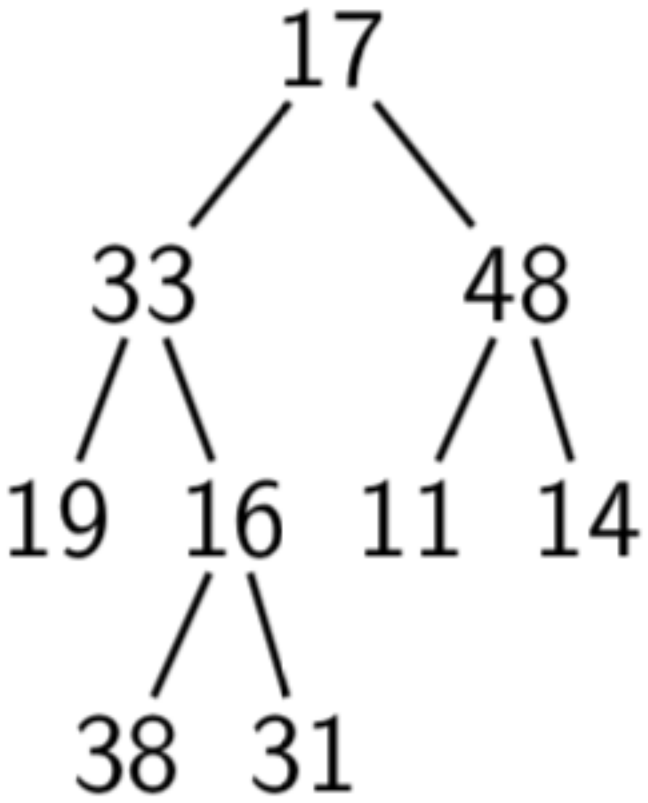
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17

```
procedure PREORDERTRAVERSE($T$)
    if $T \neq null$ then
        visit $T.root$
        PREORDERTRAVERSE($T.left$)
        PREORDERTRAVERSE($T.right$)
```



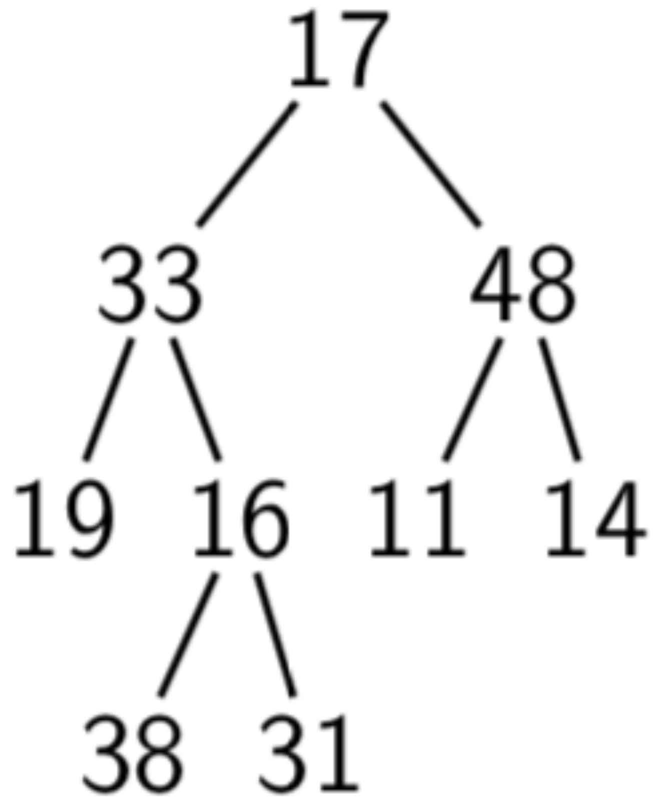
PREORDERTRAVERSE(33)
PREORDERTRAVERSE(17)

Call Stack

Preorder Traversal

Visit order: 17 33

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



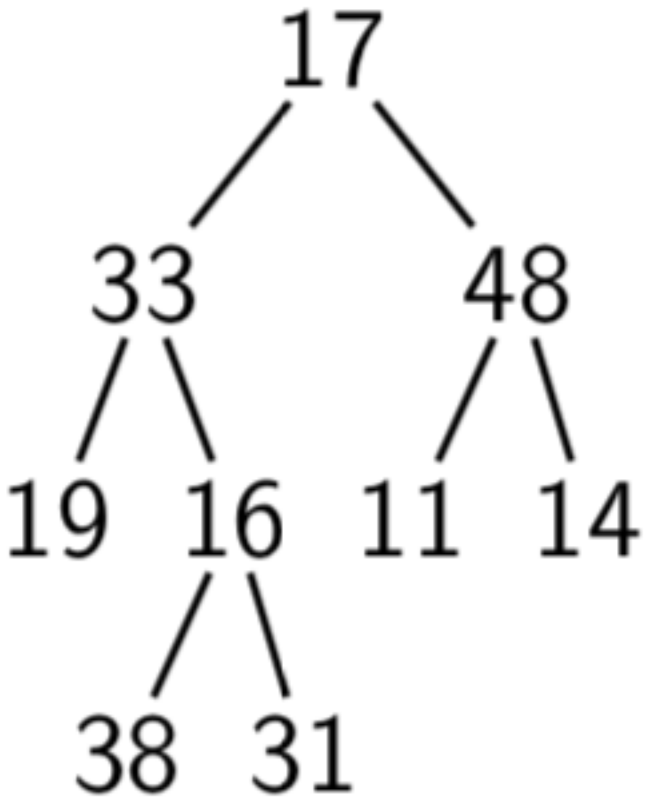
PREORDERTRAVERSE(33)
PREORDERTRAVERSE(17)

Call Stack

Preorder Traversal

Visit order: 17 33

```
procedure PREORDERTRAVERSE(T)
    if T ≠ null then
        visit T.root
        PREORDERTRAVERSE(T.left)
        PREORDERTRAVERSE(T.right)
```



PREORDERTRAVERSE(19)

PREORDERTRAVERSE(33)

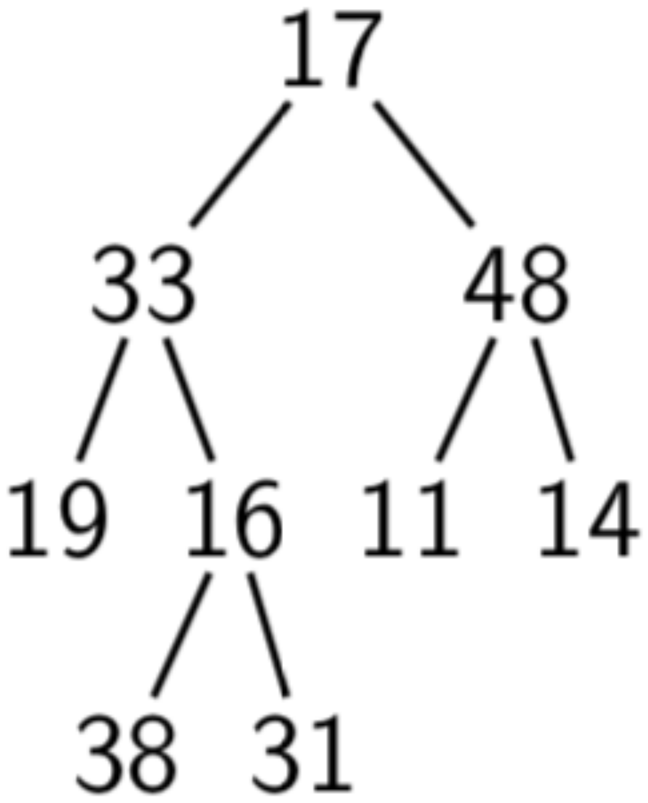
PREORDERTRAVERSE(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



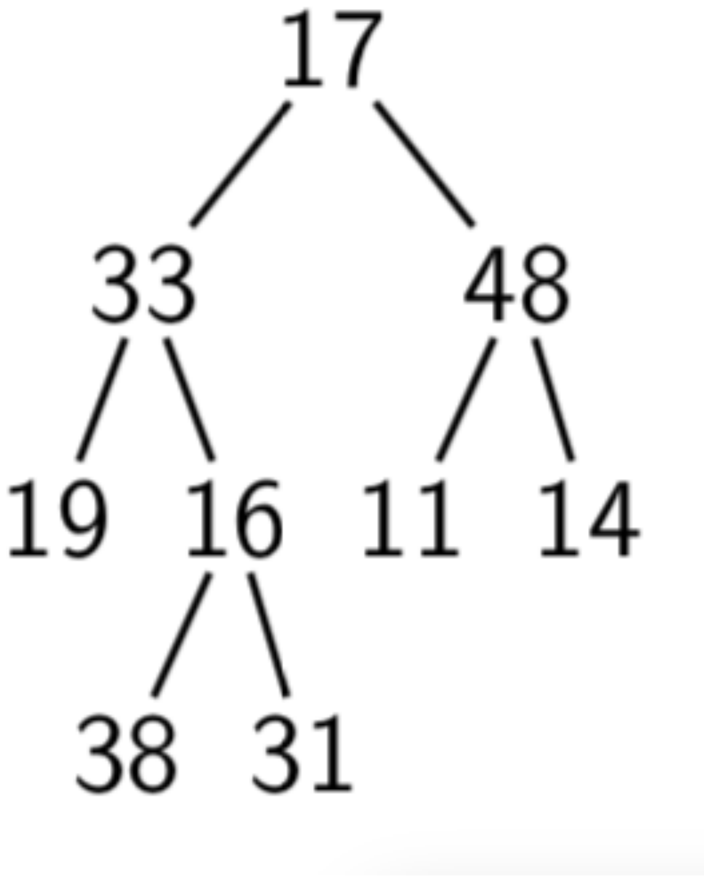
PreorderTraverse(19)
PreorderTraverse(33)
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



PreorderTraverse(null)

PreorderTraverse(19)

PreorderTraverse(33)

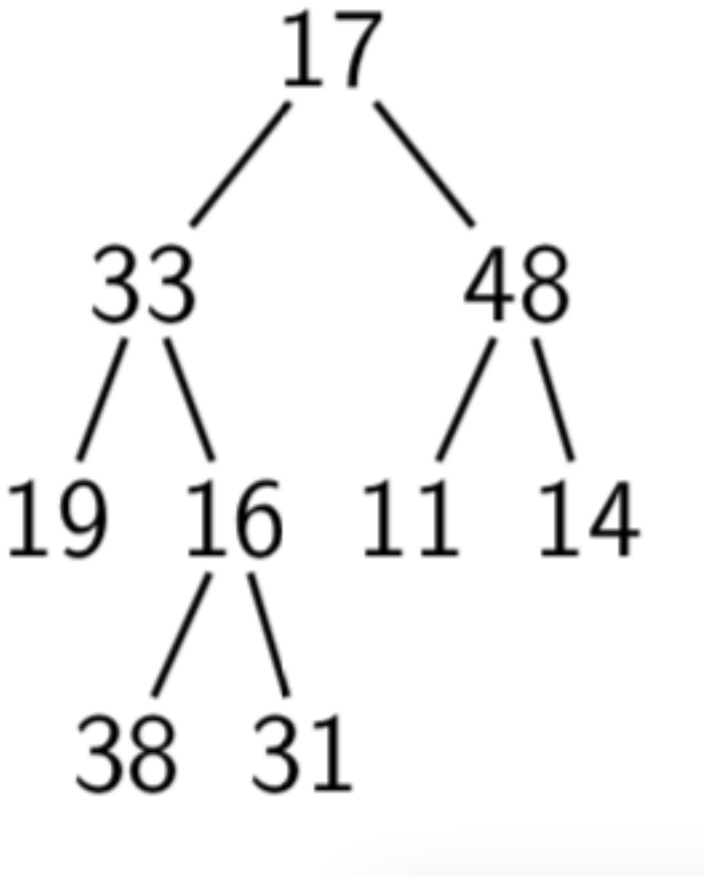
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19

```
procedure PREORDERTRAVERSE($T$)
    if $T \neq null$ then
        visit $T.root$
        PREORDERTRAVERSE($T.left$)
        PREORDERTRAVERSE($T.right$)
```



PREORDERTRAVERSE(19)

PREORDERTRAVERSE(33)

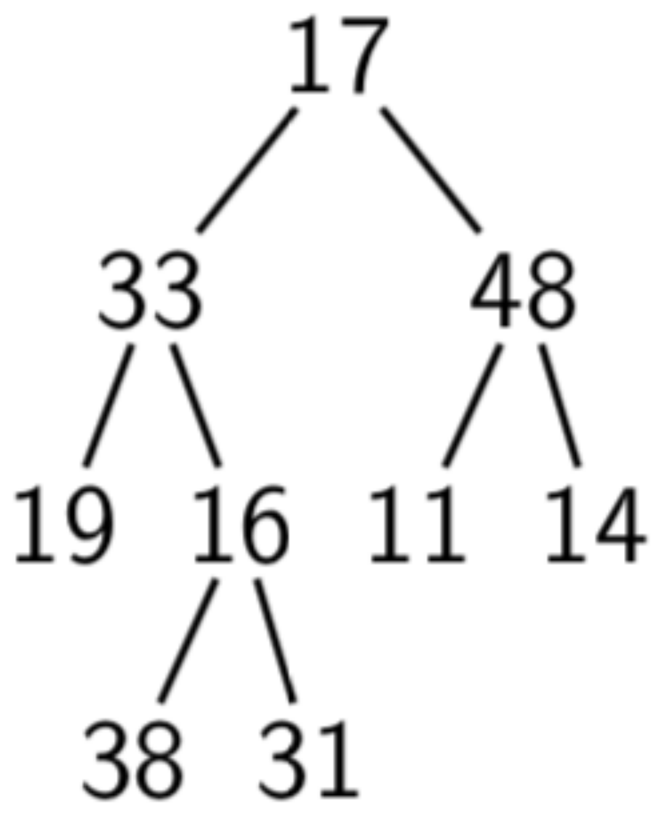
PREORDERTRAVERSE(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



PreorderTraverse(null)

PreorderTraverse(19)

PreorderTraverse(33)

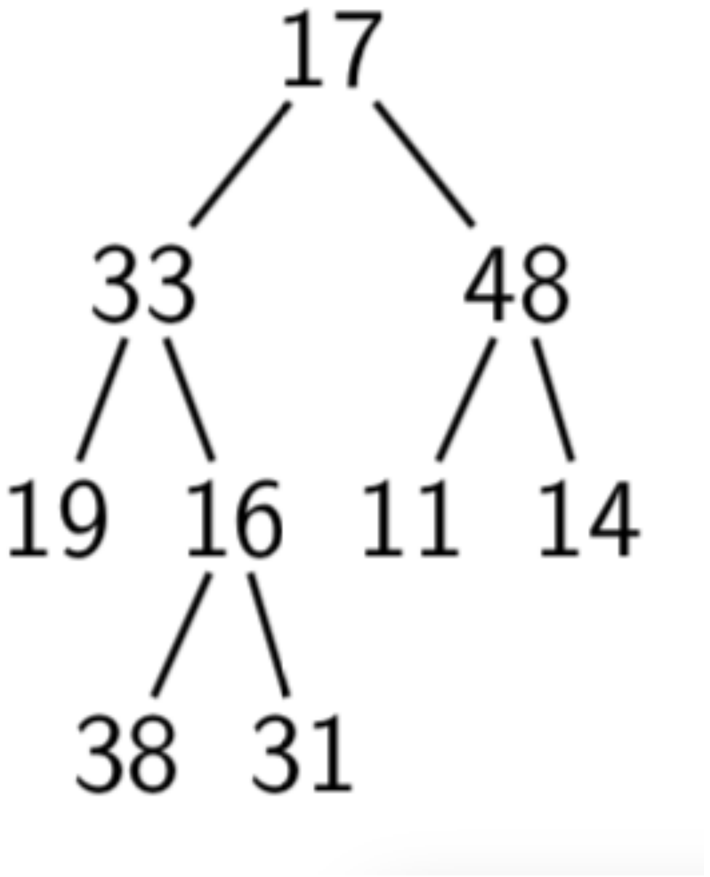
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



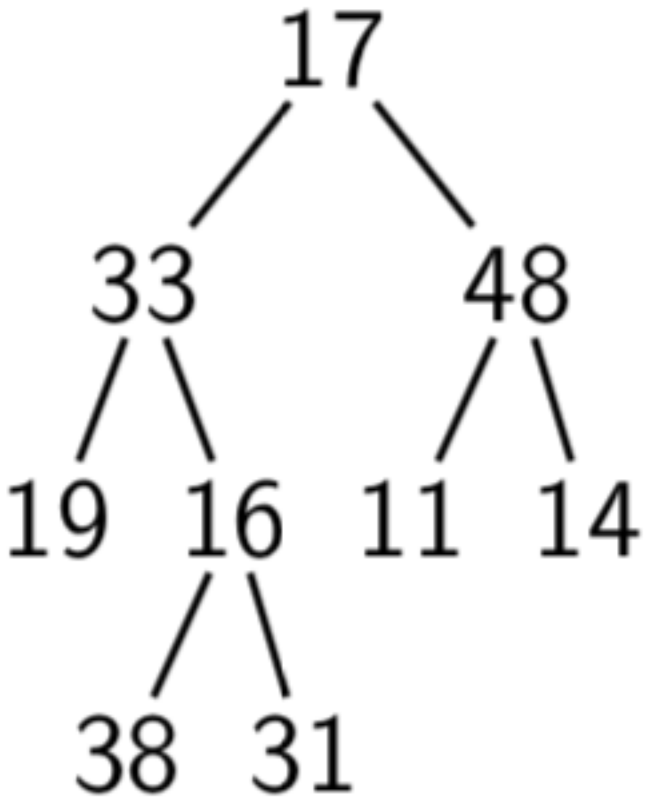
PreorderTraverse(19)
PreorderTraverse(33)
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



PreorderTraverse(33)

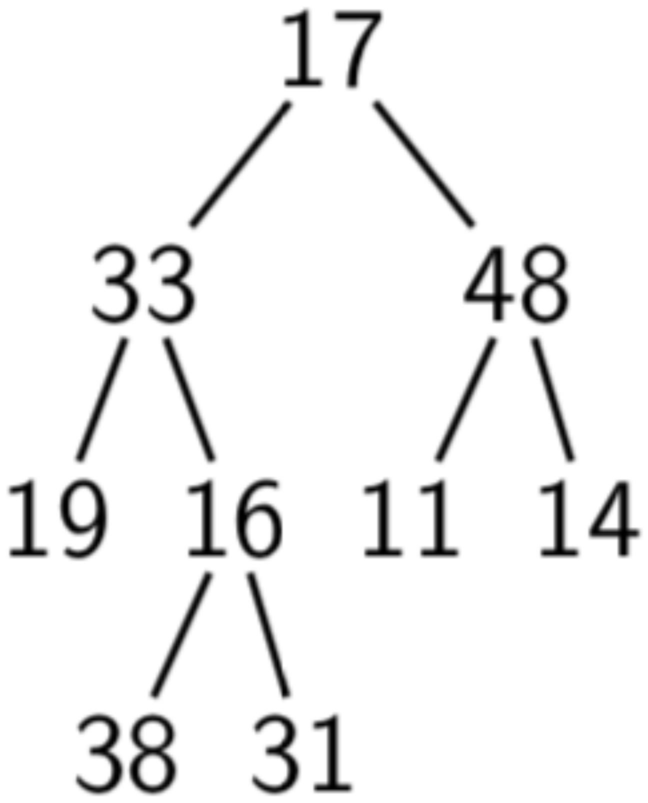
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19

```
procedure PreorderTraverse(T)
    if $T \neq null$ then
        visit T.root
        PreorderTraverse(T.left)
        PreorderTraverse(T.right)
```



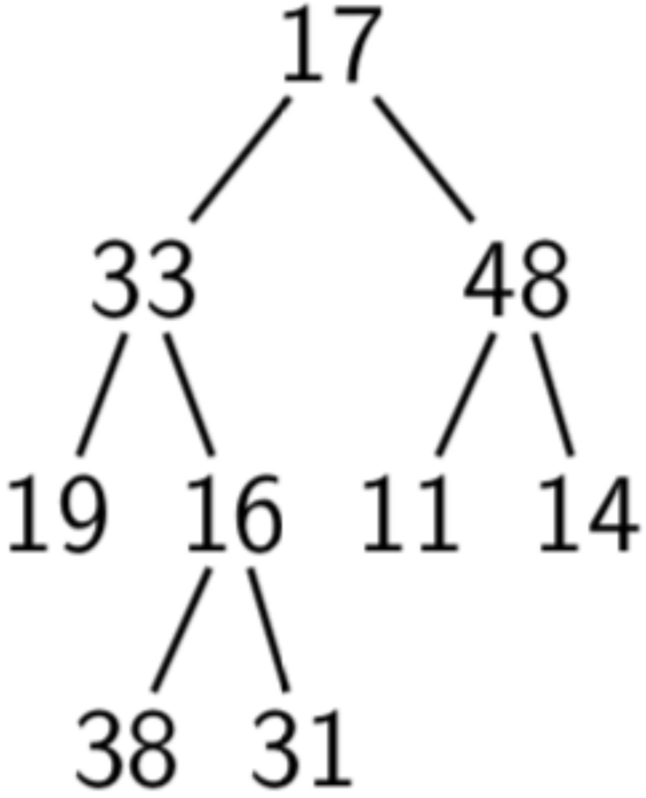
PreorderTraverse(16)
PreorderTraverse(33)
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16

```
procedure PreorderTraverse(T)
    if $T \neq null$ then
        visit T.root
        PreorderTraverse(T.left)
        PreorderTraverse(T.right)
```



PreorderTraverse(16)

PreorderTraverse(33)

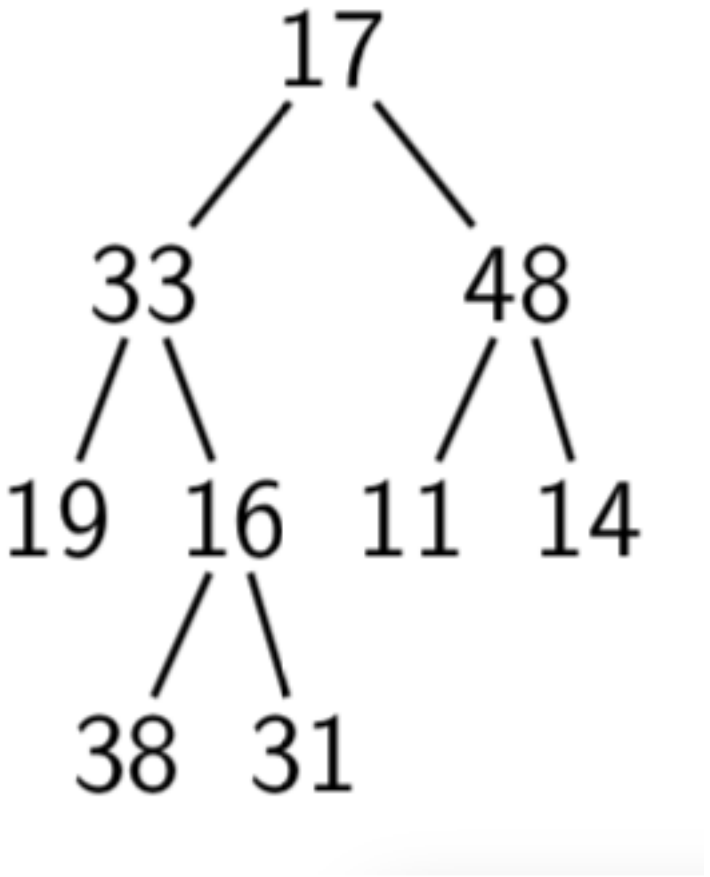
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



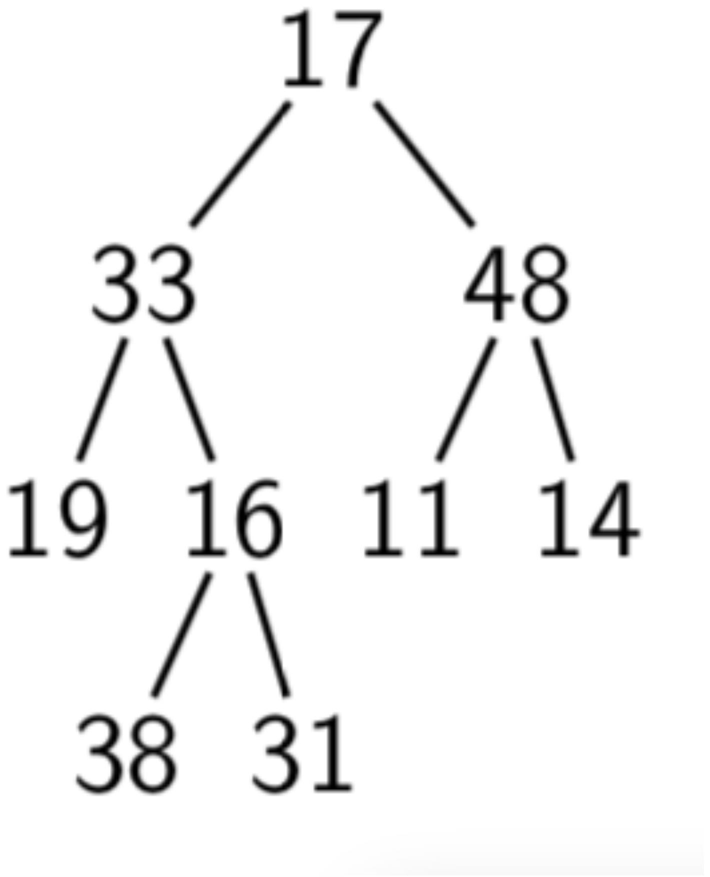
PreorderTraverse(38)
PreorderTraverse(16)
PreorderTraverse(33)
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



PreorderTraverse(38)

PreorderTraverse(16)

PreorderTraverse(33)

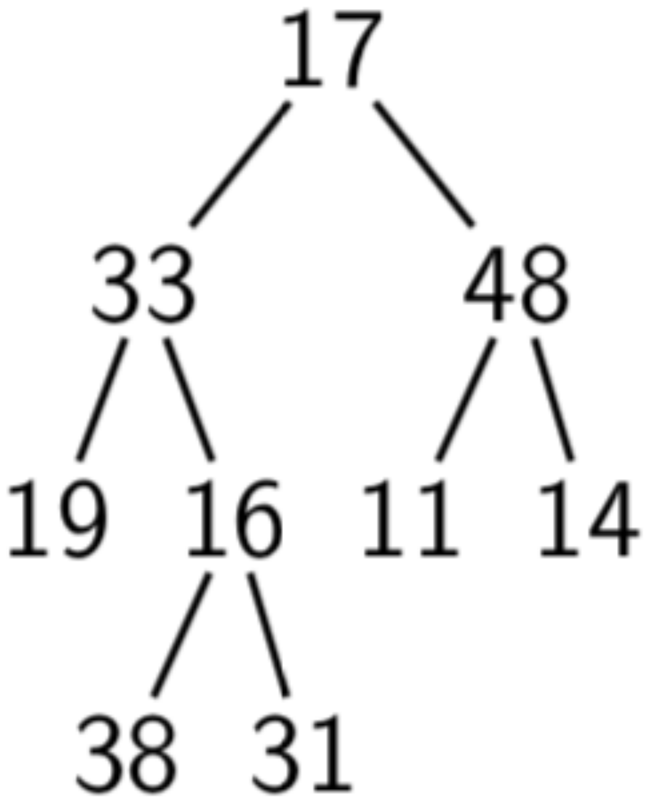
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38

```
procedure PREORDERTRAVERSE($T$)
    if $T \neq null$ then
        visit $T.root$
        PREORDERTRAVERSE($T.left$)
        PREORDERTRAVERSE($T.right$)
```



PREORDERTRAVERSE(38)

PREORDERTRAVERSE(16)

PREORDERTRAVERSE(33)

PREORDERTRAVERSE(17)

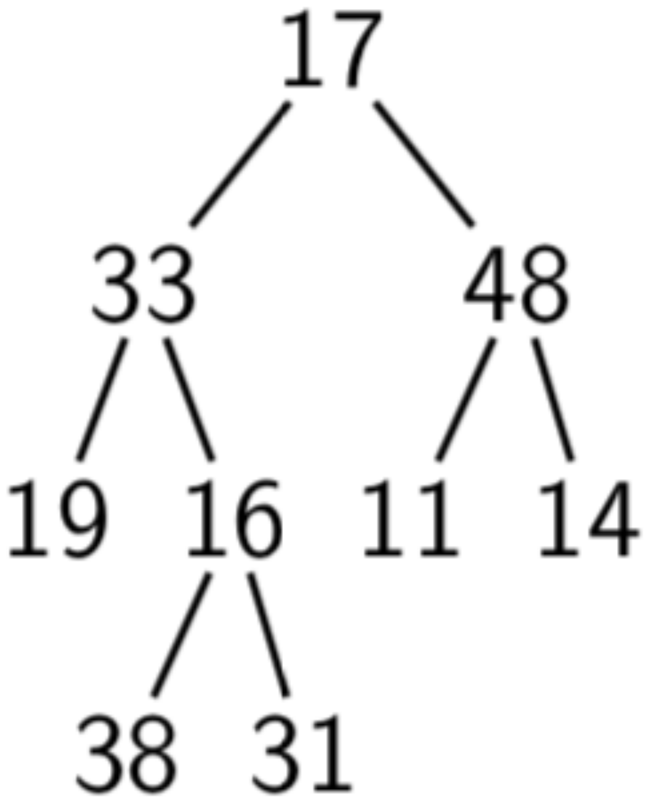
Call Stack

(…skipping the calls to PREORDERTRAVERSE(null)…)

Preorder Traversal

Visit order: 17 33 19 16 38

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



PreorderTraverse(16)

PreorderTraverse(33)

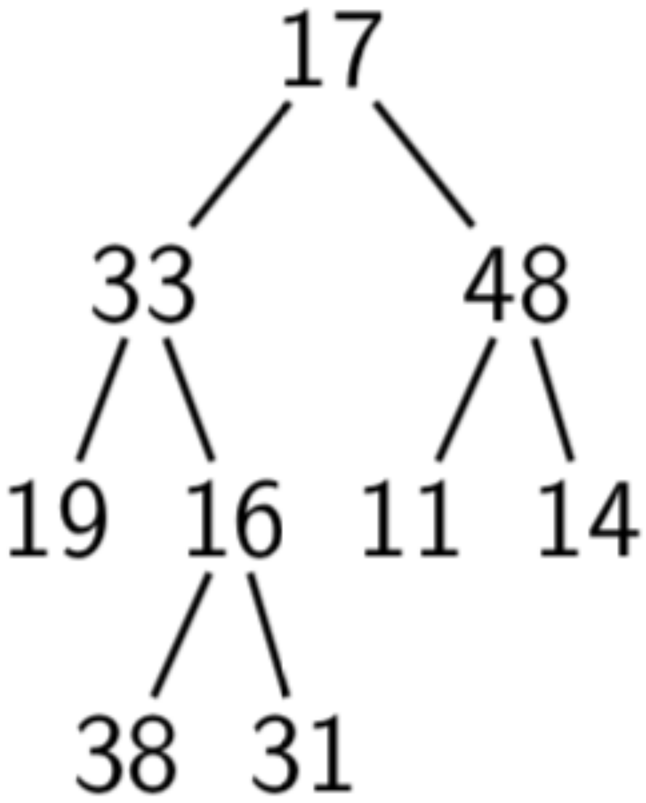
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



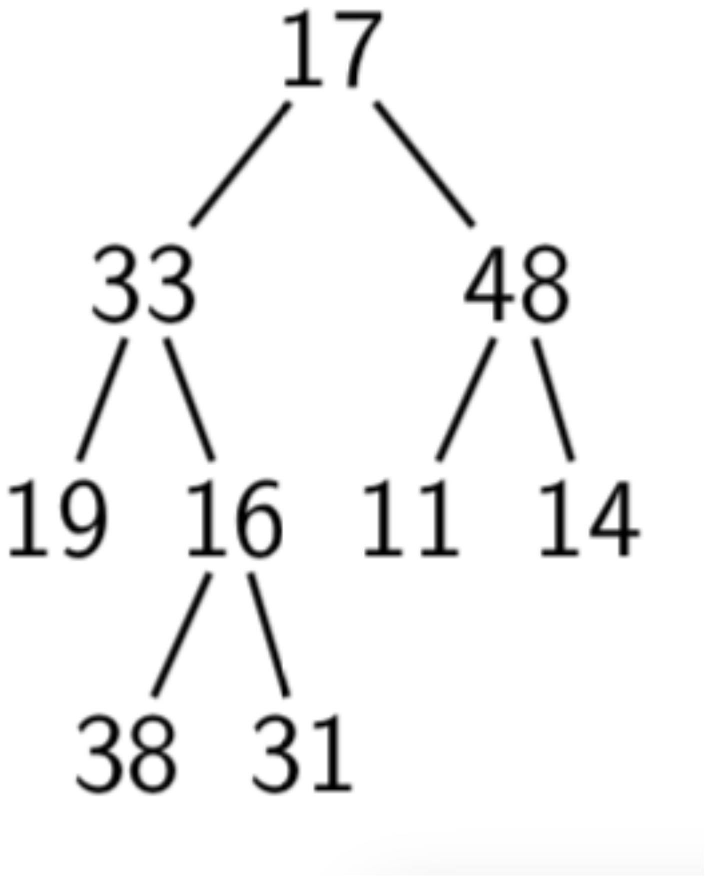
PreorderTraverse(31)
PreorderTraverse(16)
PreorderTraverse(33)
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38 31

```
procedure PREORDERTRAVERSE(T)
    if $T \neq null$ then
        visit T.root
        PREORDERTRAVERSE(T.left)
        PREORDERTRAVERSE(T.right)
```



PREORDERTRAVERSE(31)

PREORDERTRAVERSE(16)

PREORDERTRAVERSE(33)

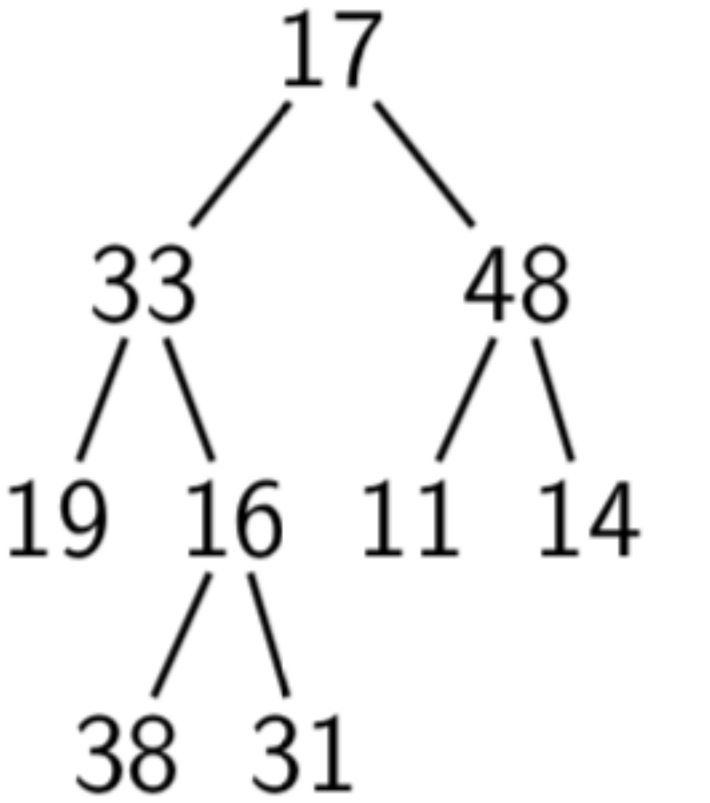
PREORDERTRAVERSE(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38 31

```
procedure PREORDERTRAVERSE($T$)
    if $T \neq null$ then
        visit $T.root$
        PREORDERTRAVERSE($T.left$)
        PREORDERTRAVERSE($T.right$)
```



PREORDERTRAVERSE(31)

PREORDERTRAVERSE(16)

PREORDERTRAVERSE(33)

PREORDERTRAVERSE(17)

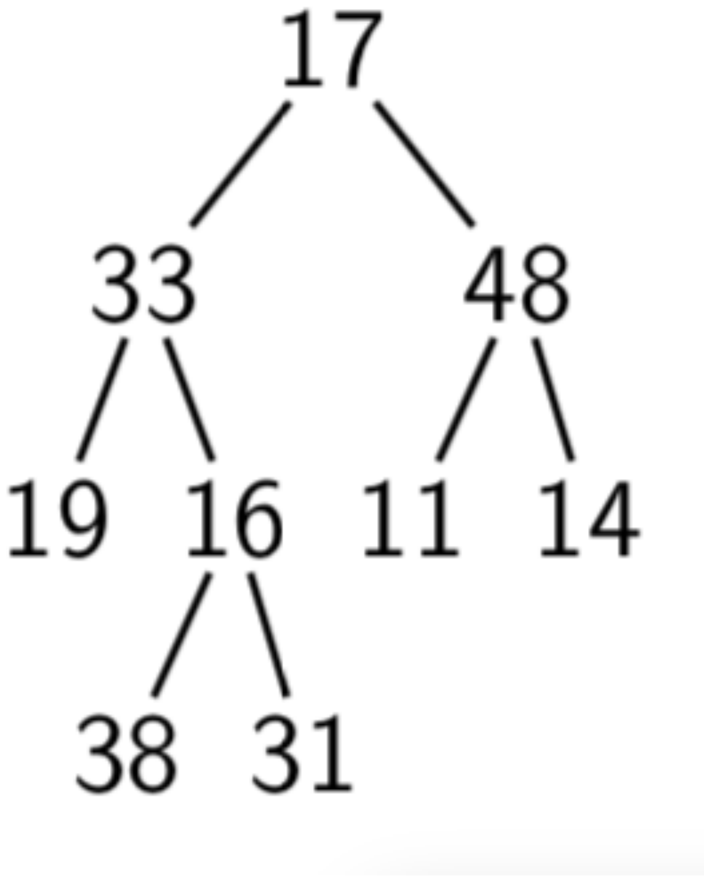
Call Stack

(…skipping the calls to PREORDERTRAVERSE(null)…)

Preorder Traversal

Visit order: 17 33 19 16 38 31

```
procedure PreorderTraverse(T)
    if $T \neq null$ then
        visit T.root
        PreorderTraverse(T.left)
        PreorderTraverse(T.right)
```



PreorderTraverse(16)

PreorderTraverse(33)

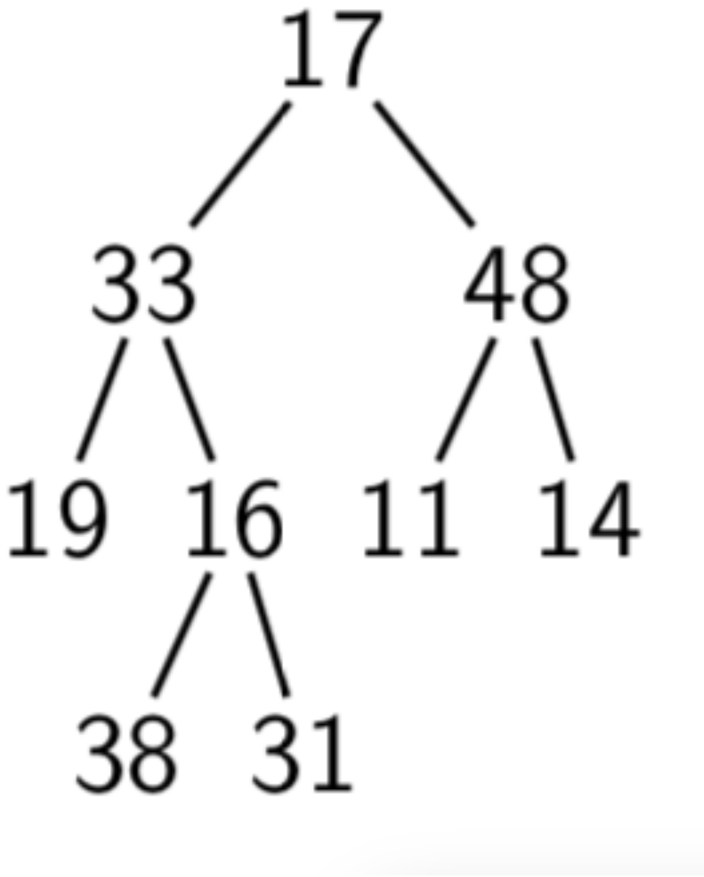
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38 31

```
procedure PreorderTraverse(T)
    if $T \neq null$ then
        visit T.root
        PreorderTraverse(T.left)
        PreorderTraverse(T.right)
```



PREORDERTRAVERSE(33)

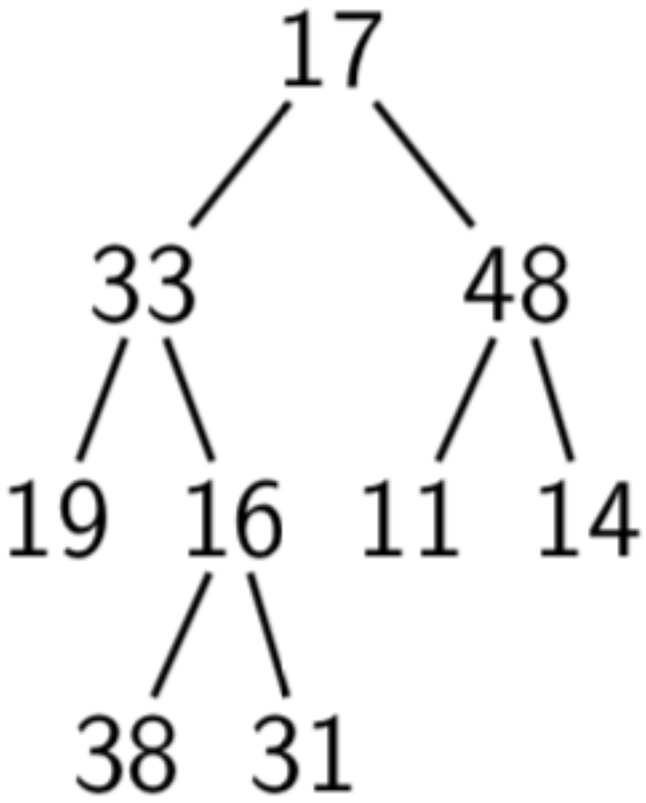
PREORDERTRAVERSE(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38 31

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



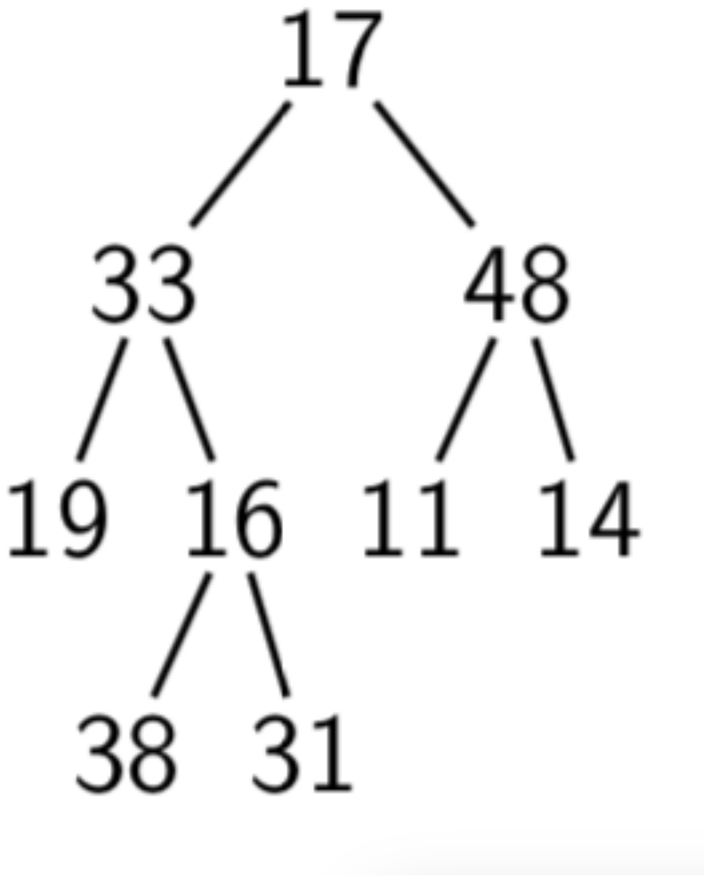
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38 31

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



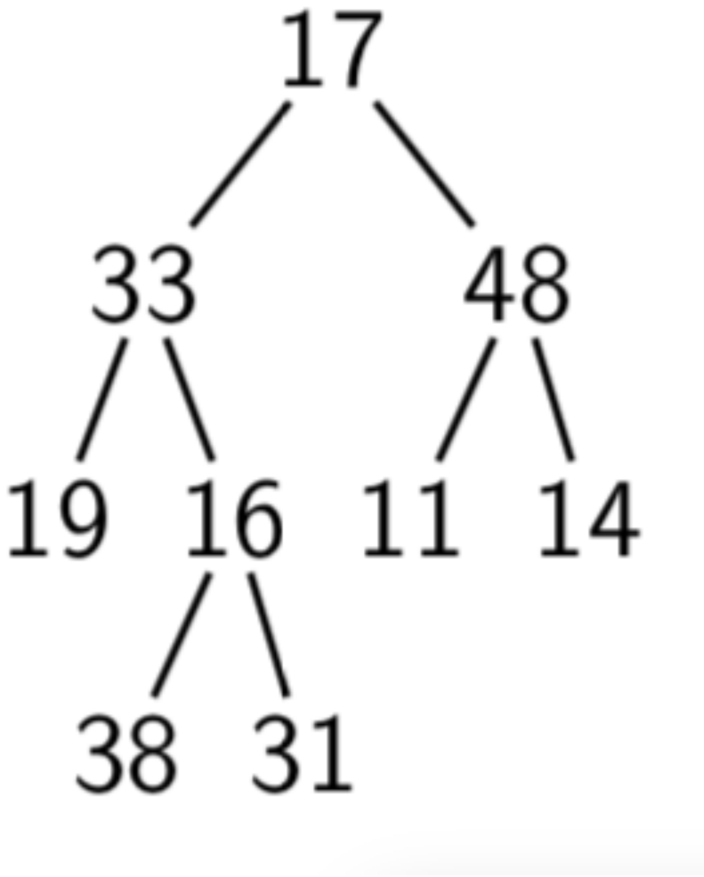
PreorderTraverse(48)
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38 31 48

```
procedure PreorderTraverse(T)
    if $T \neq null$ then
        visit T.root
        PreorderTraverse(T.left)
        PreorderTraverse(T.right)
```



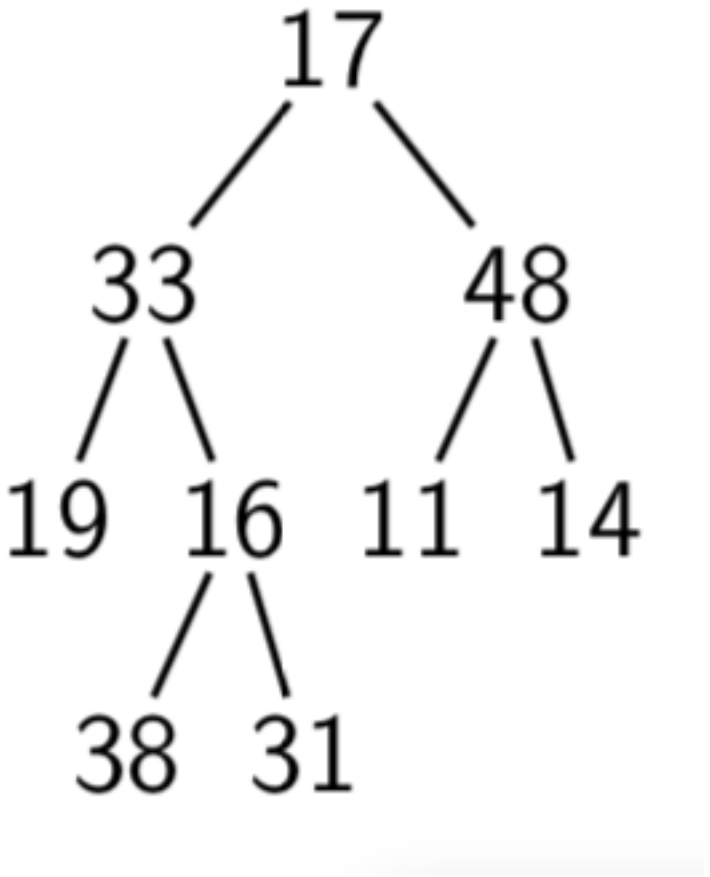
PREORDERTRAVERSE(48)
PREORDERTRAVERSE(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38 31 48

```
procedure PreorderTraverse(T)
    if $T \neq null$ then
        visit T.root
        PreorderTraverse(T.left)
        PreorderTraverse(T.right)
```



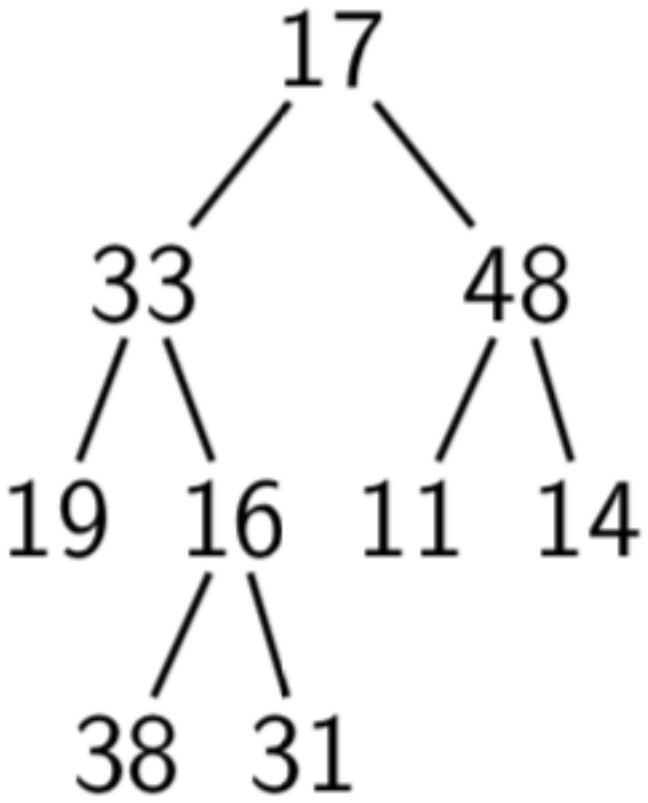
PreorderTraverse(11)
PreorderTraverse(48)
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38 31 48 11

```
procedure PREORDERTRAVERSE($T$)
    if $T \neq null$ then
        visit $T.root$
        PREORDERTRAVERSE($T.left$)
        PREORDERTRAVERSE($T.right$)
```



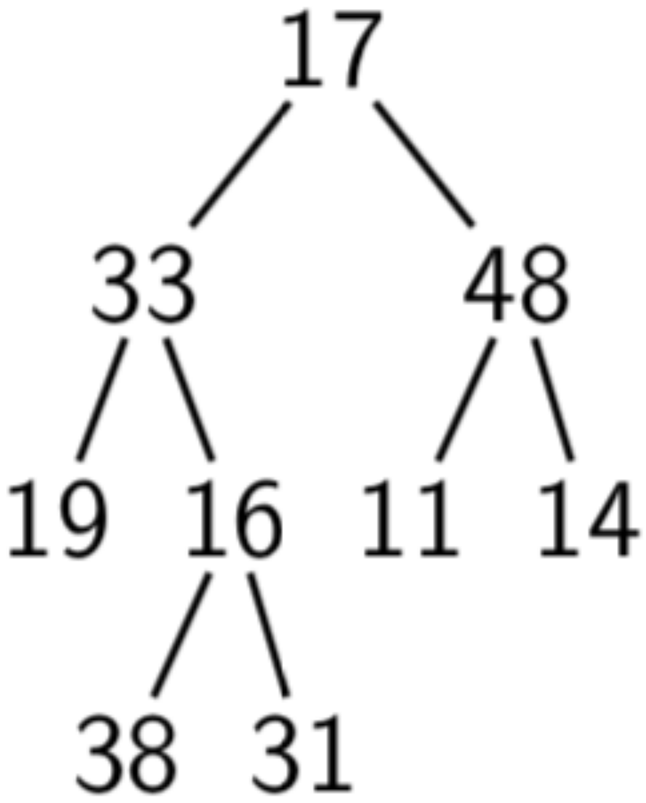
PREORDERTRAVERSE(11)
PREORDERTRAVERSE(48)
PREORDERTRAVERSE(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38 31 48 11

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



PreorderTraverse(11)
PreorderTraverse(48)
PreorderTraverse(17)

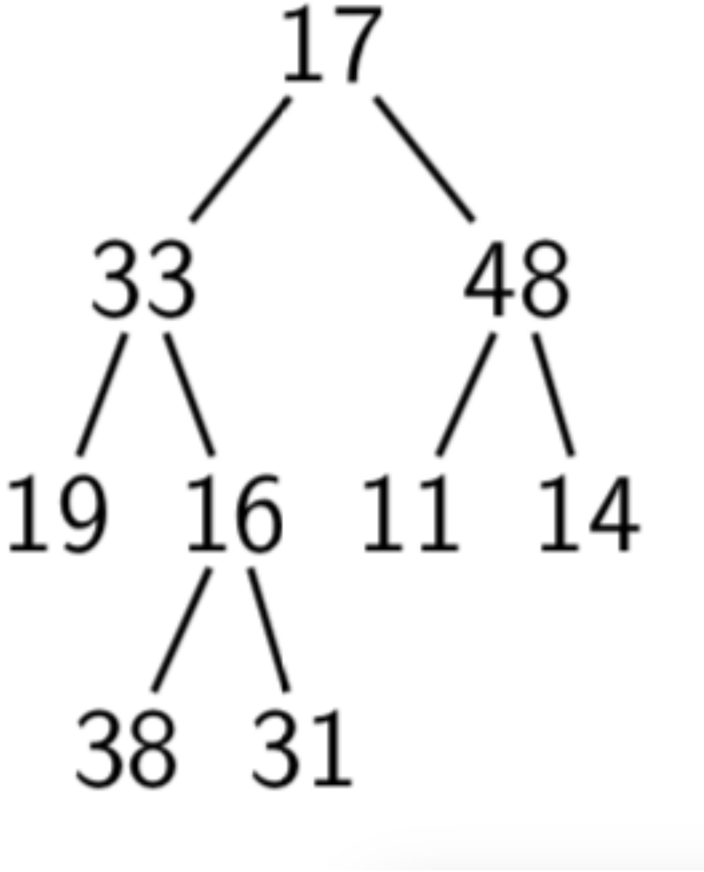
Call Stack

(…skipping the calls to PreorderTraverse(null)…)

Preorder Traversal

Visit order: 17 33 19 16 38 31 48 11

```
procedure PREORDERTRAVERSE($T$)
    if $T \neq null$ then
        visit $T.root$
        PREORDERTRAVERSE($T.left$)
        PREORDERTRAVERSE($T.right$)
```



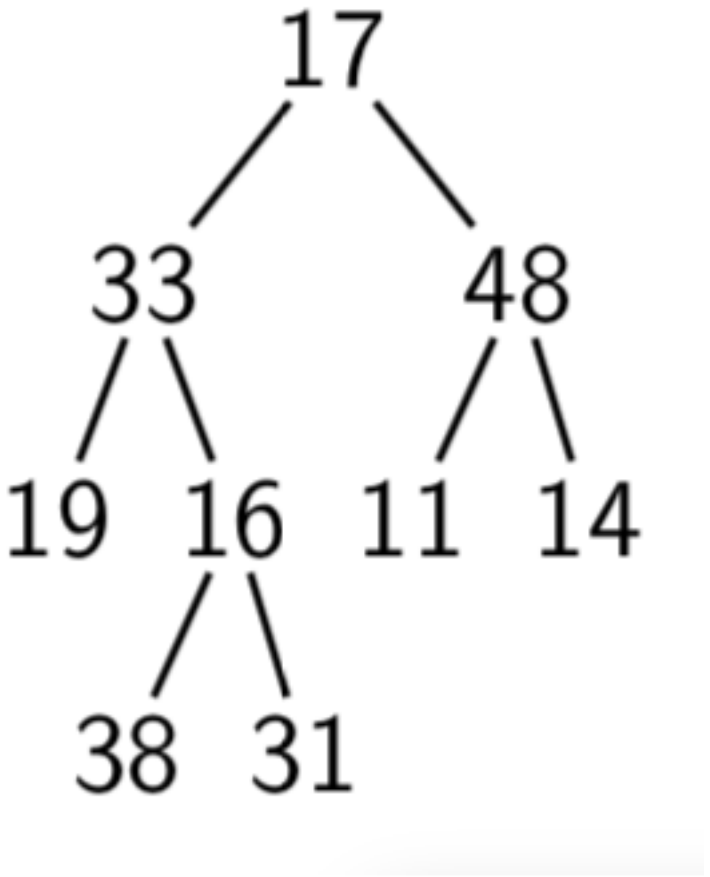
PREORDERTRAVERSE(48)
PREORDERTRAVERSE(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38 31 48 11

```
procedure PreorderTraverse(T)
    if $T \neq null$ then
        visit T.root
        PreorderTraverse(T.left)
        PreorderTraverse(T.right)
```



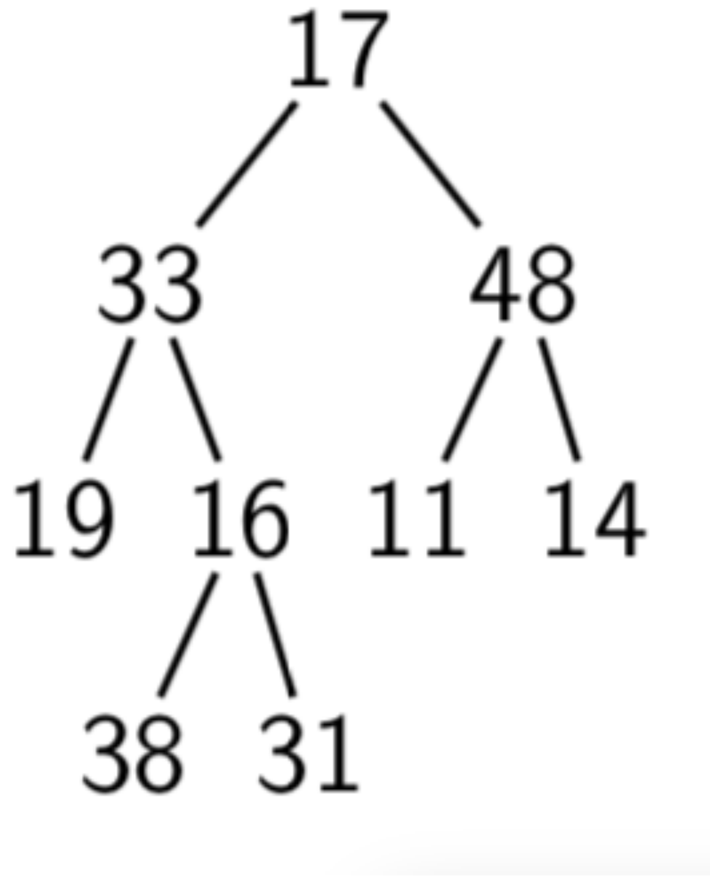
PREORDERTRAVERSE(14)
PREORDERTRAVERSE(48)
PREORDERTRAVERSE(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38 31 48 11 14

```
procedure PREORDERTRAVERSE(T)
    if $T \neq null$ then
        visit T.root
        PREORDERTRAVERSE(T.left)
        PREORDERTRAVERSE(T.right)
```



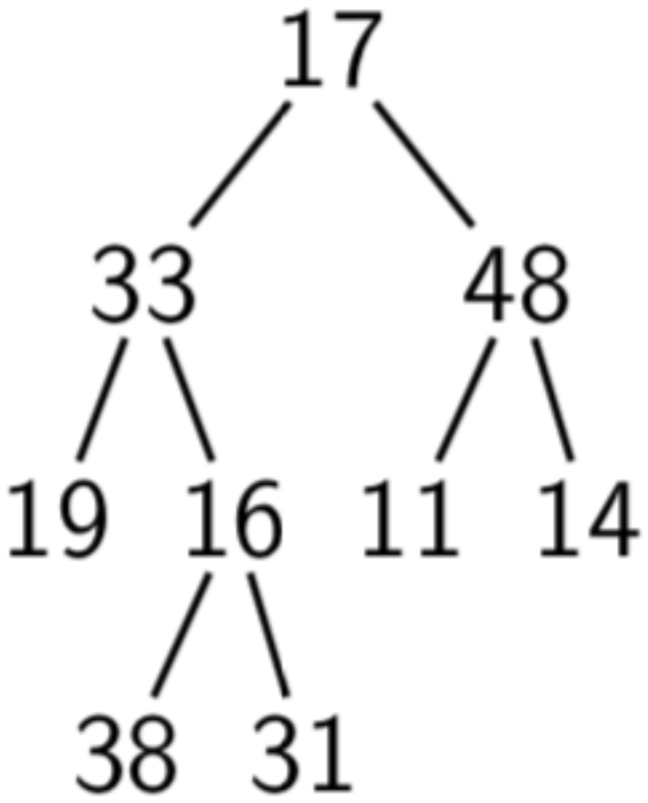
PREORDERTRAVERSE(14)
PREORDERTRAVERSE(48)
PREORDERTRAVERSE(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38 31 48 11 14

```
procedure PreorderTraverse($T$)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse($T.left$)
        PreorderTraverse($T.right$)
```



PreorderTraverse(14)
PreorderTraverse(48)
PreorderTraverse(17)

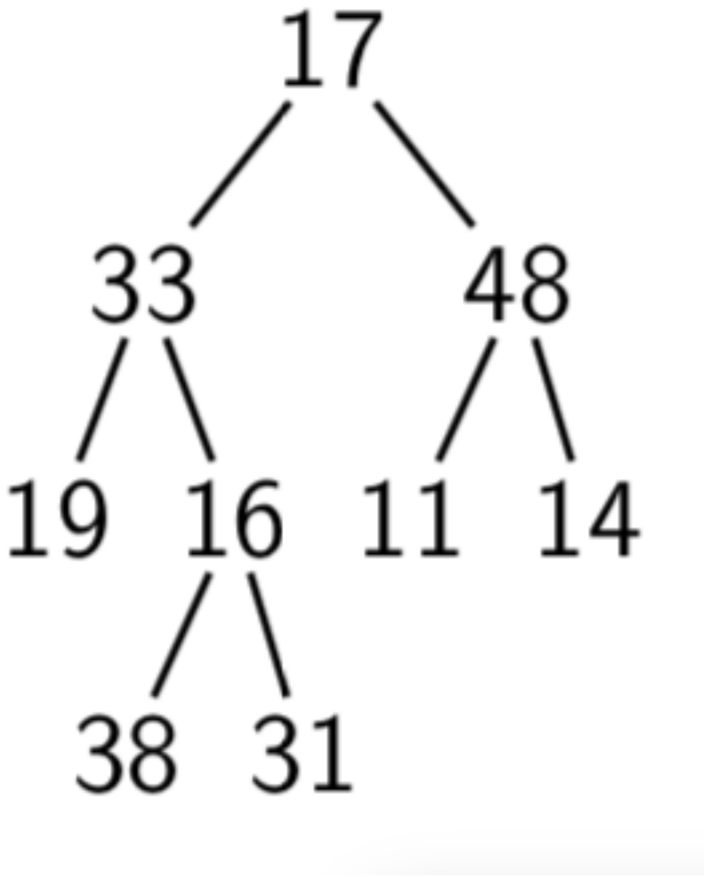
Call Stack

(…skipping the calls to PreorderTraverse(null)…)

Preorder Traversal

Visit order: 17 33 19 16 38 31 48 11 14

```
procedure PreorderTraverse(T)
    if $T \neq null$ then
        visit T.root
        PreorderTraverse(T.left)
        PreorderTraverse(T.right)
```



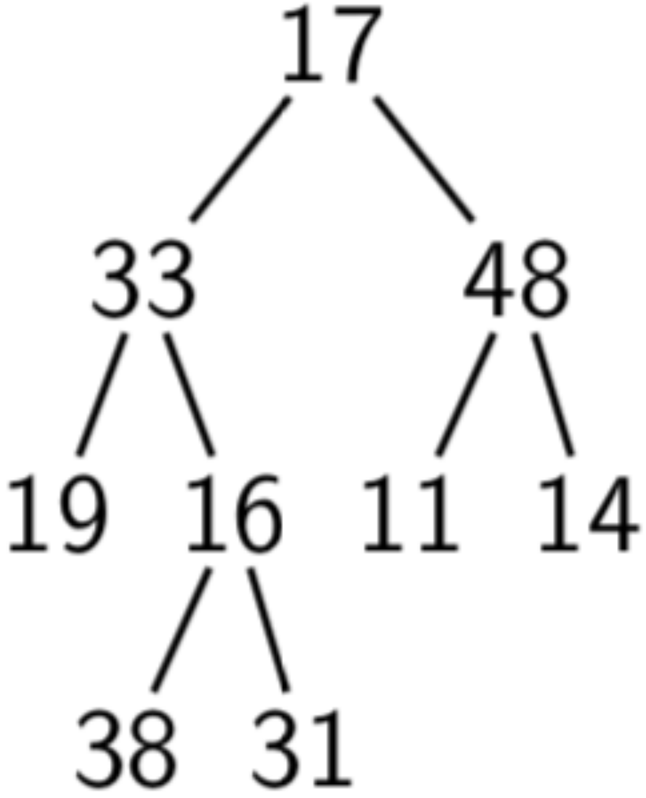
PreorderTraverse(48)
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38 31 48 11 14

```
procedure PreorderTraverse(T)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse(T.left)
        PreorderTraverse(T.right)
```



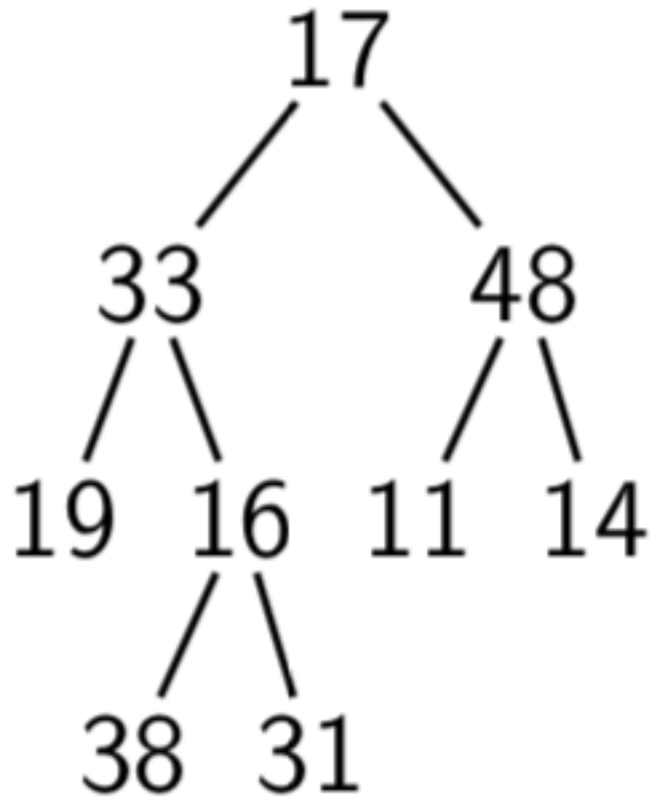
PreorderTraverse(17)

Call Stack

Preorder Traversal

Visit order: 17 33 19 16 38 31 48 11 14

```
procedure PreorderTraverse(T)
    if $T \neq null$ then
        visit $T.root$
        PreorderTraverse(T.left)
        PreorderTraverse(T.right)
```

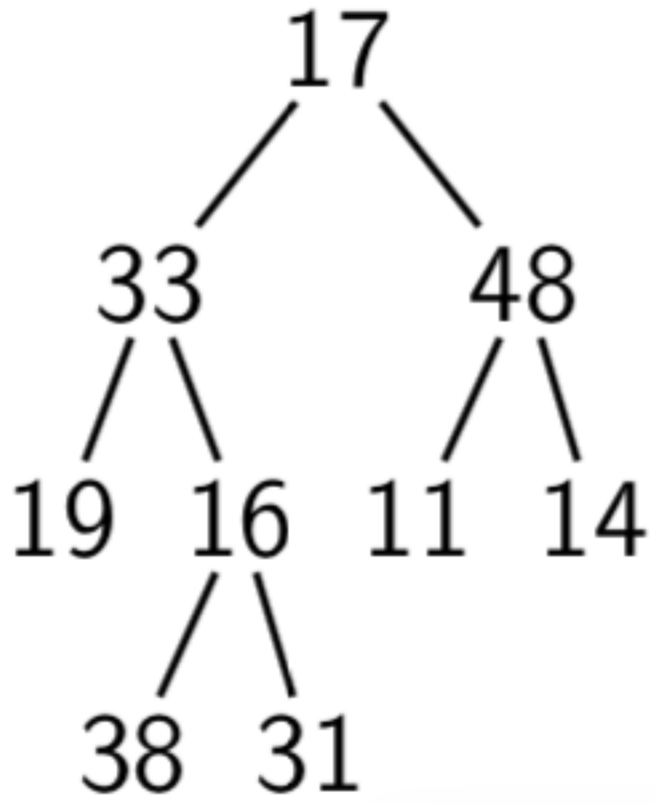


Call Stack

Inorder Traversal

Visit order:

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



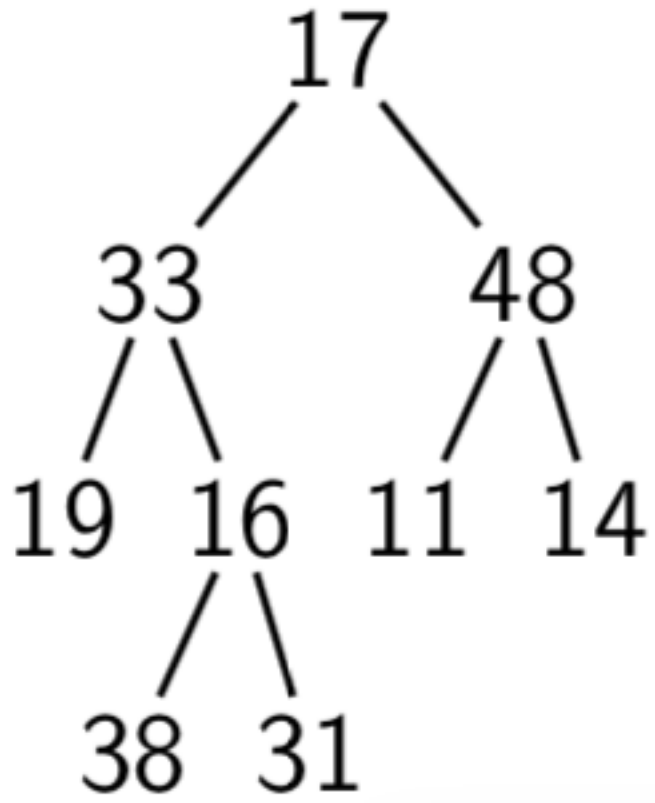
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order:

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



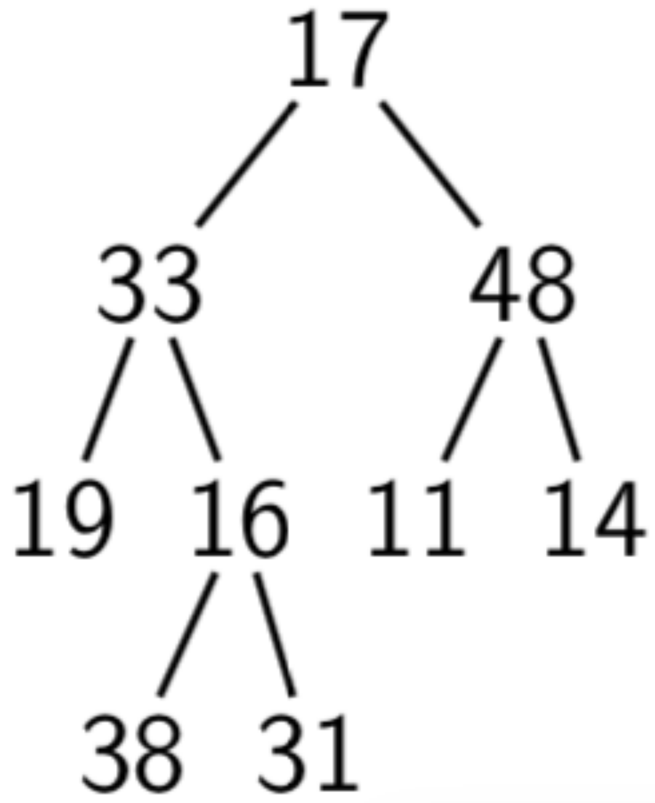
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order:

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



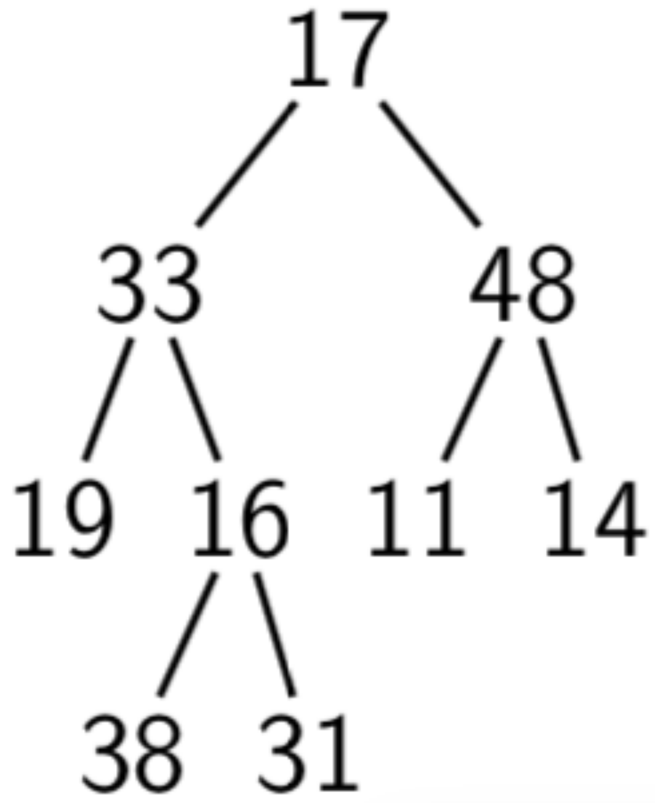
InorderTraverse(19)
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order:

```
procedure INORDERTRAVERSE($T$)
    if $T \neq null$ then
        INORDERTRAVERSE($T.left$)
        visit $T.root$
        INORDERTRAVERSE($T.right$)
```



INORDERTRAVERSE(null)

INORDERTRAVERSE(19)

INORDERTRAVERSE(33)

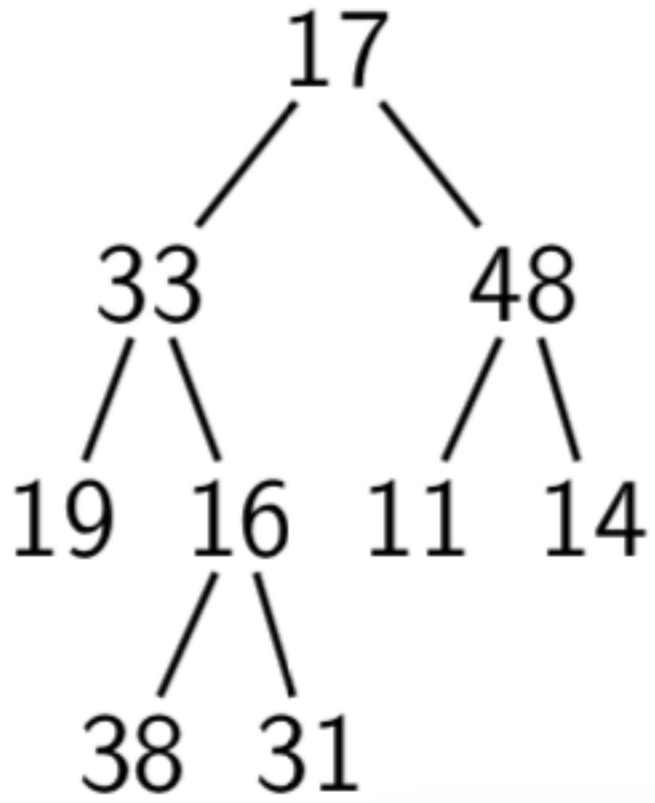
INORDERTRAVERSE(17)

Call Stack

Inorder Traversal

Visit order:

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



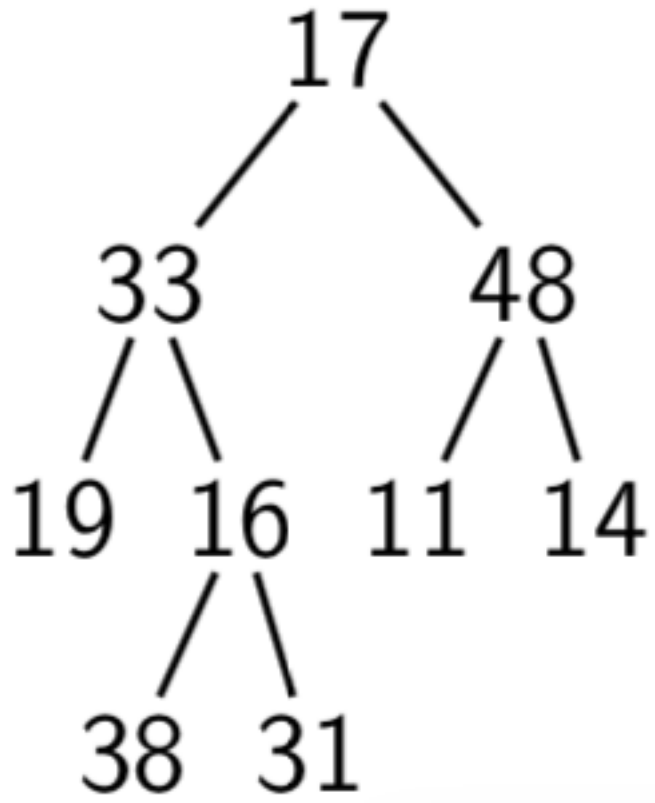
InorderTraverse(19)
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19

```
procedure INORDERTRAVERSE($T$)
    if $T \neq null$ then
        INORDERTRAVERSE($T.left$)
        visit $T.root$
        INORDERTRAVERSE($T.right$)
```



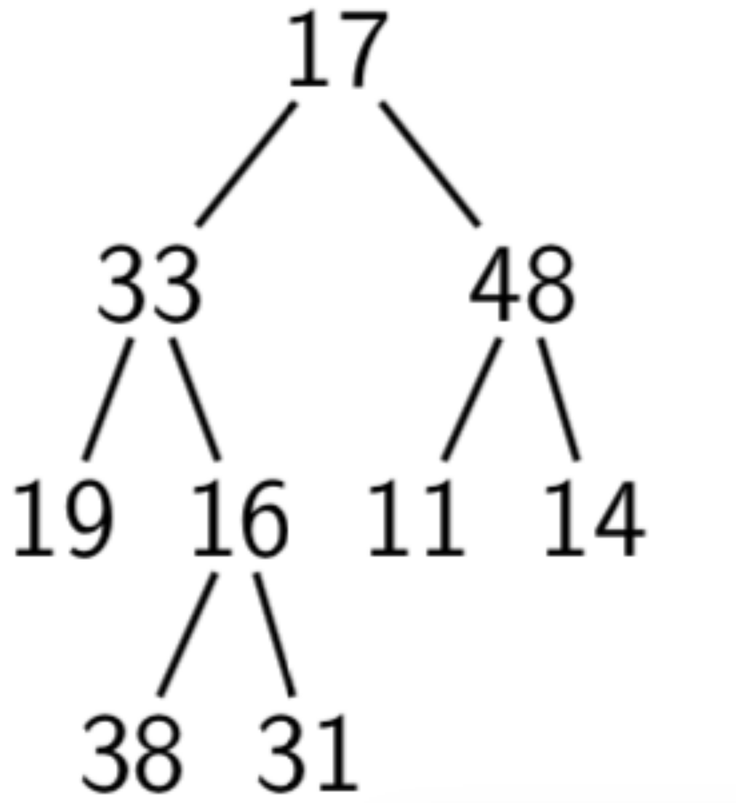
INORDERTRAVERSE(19)
INORDERTRAVERSE(33)
INORDERTRAVERSE(17)

Call Stack

Inorder Traversal

Visit order: 19

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



InorderTraverse(null)

InorderTraverse(19)

InorderTraverse(33)

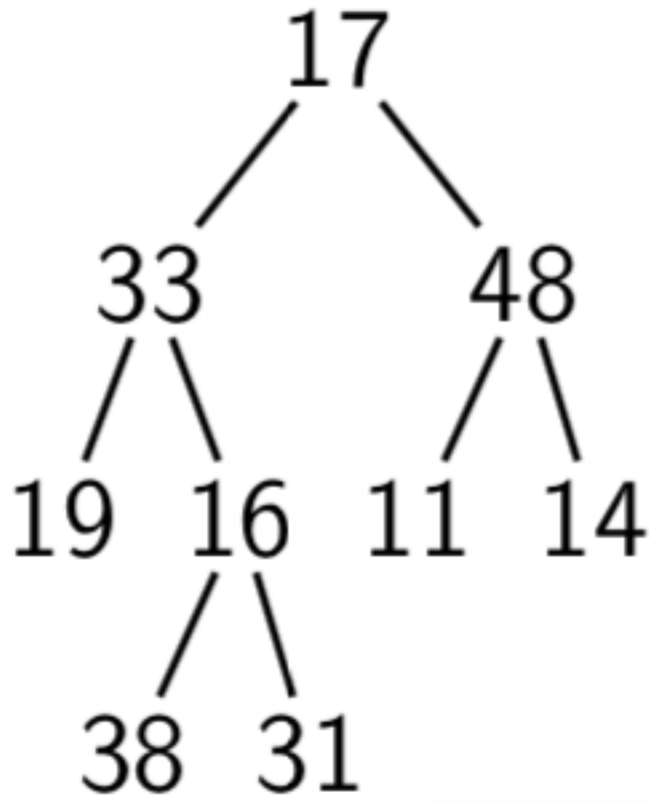
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19

```
procedure INORDERTRAVERSE($T$)
    if $T \neq null$ then
        INORDERTRAVERSE($T.left$)
        visit $T.root$
        INORDERTRAVERSE($T.right$)
```



INORDERTRAVERSE(19)

INORDERTRAVERSE(33)

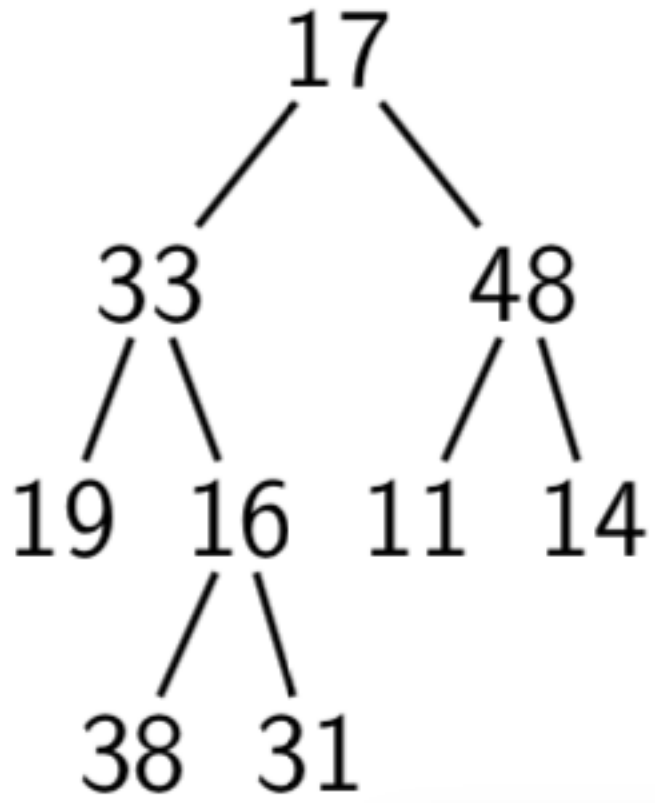
INORDERTRAVERSE(17)

Call Stack

Inorder Traversal

Visit order: 19

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



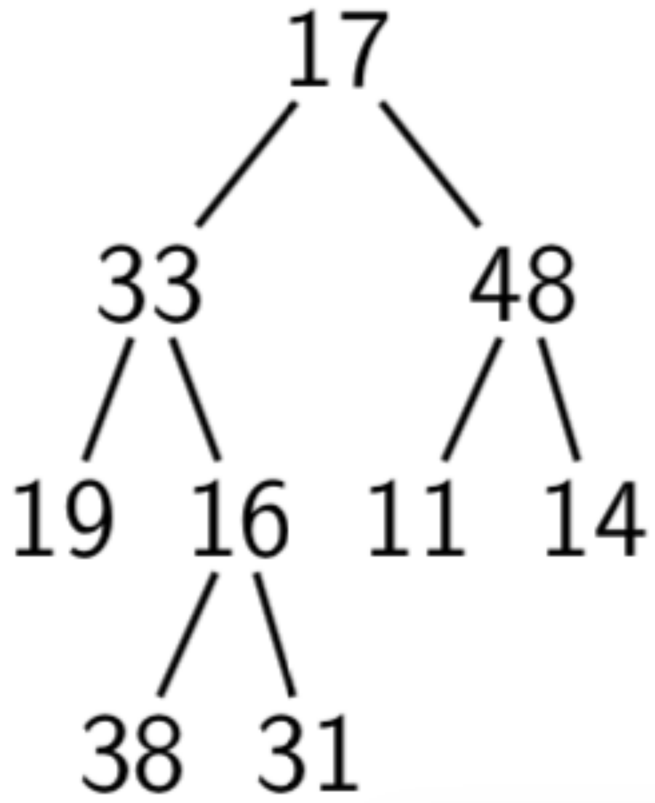
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



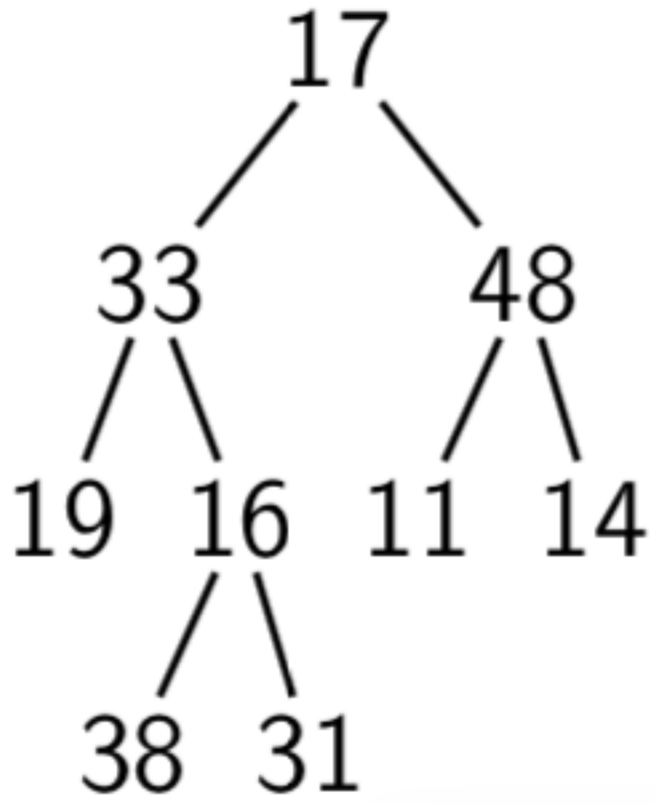
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



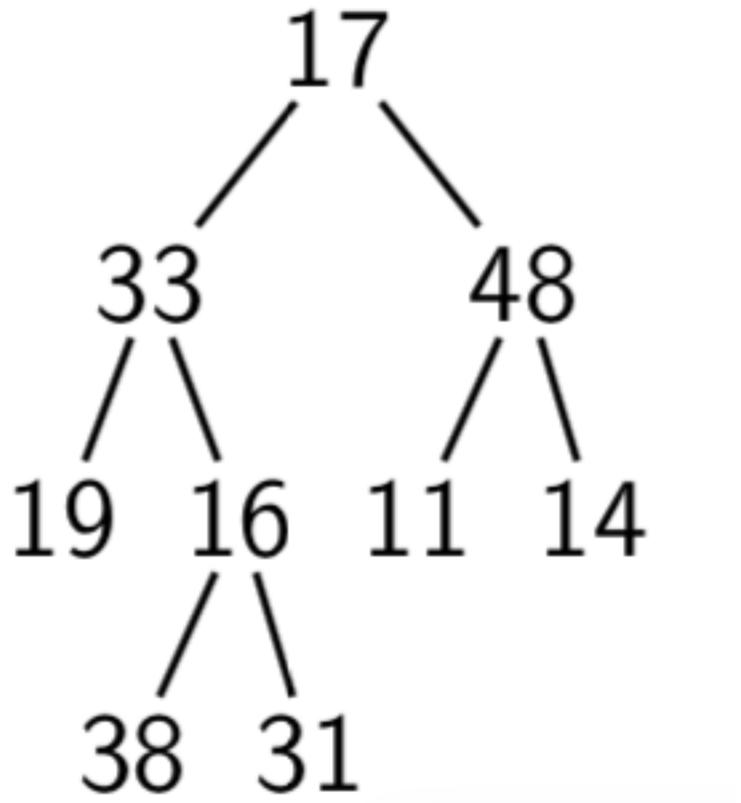
InorderTraverse(16)
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



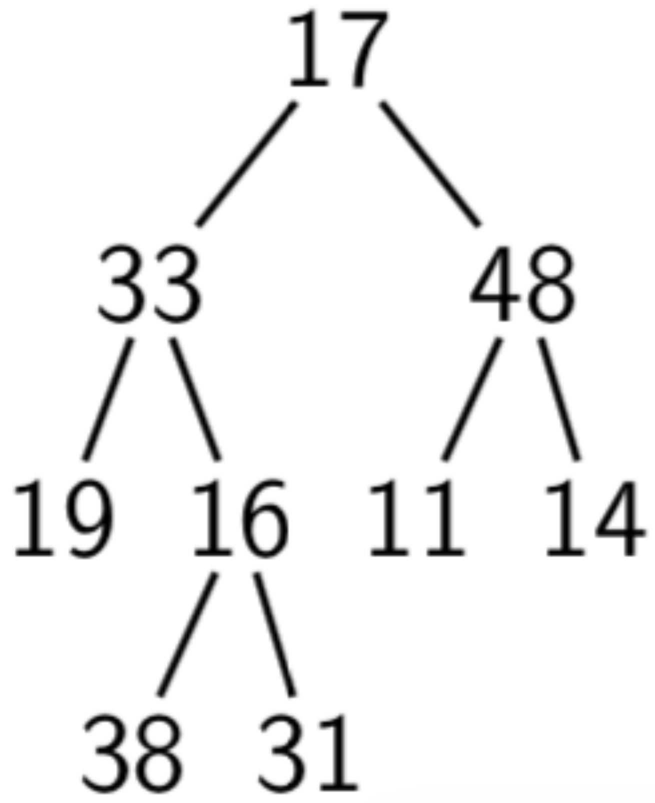
InorderTraverse(38)
InorderTraverse(16)
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33

```
procedure INORDERTRAVERSE(T)
    if $T \neq null$ then
        INORDERTRAVERSE(T.left)
        visit T.root
        INORDERTRAVERSE(T.right)
```



INORDERTRAVERSE(null)

INORDERTRAVERSE(38)

INORDERTRAVERSE(16)

INORDERTRAVERSE(33)

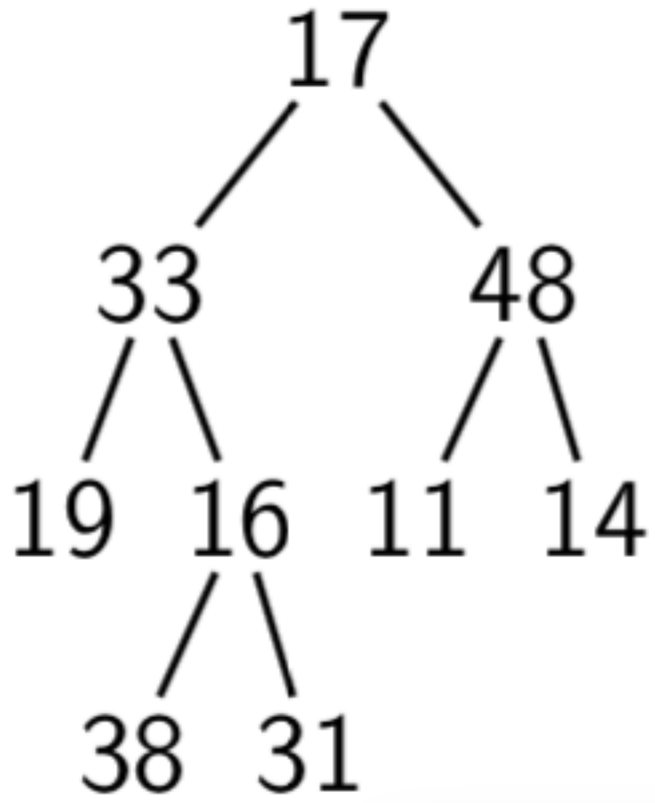
INORDERTRAVERSE(17)

Call Stack

Inorder Traversal

Visit order: 19 33

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



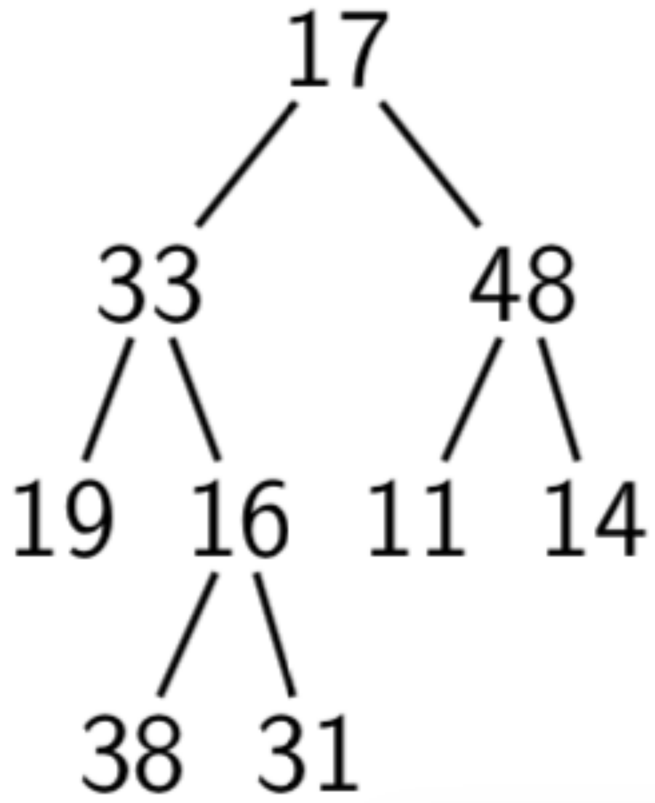
INORDERTRAVERSE(38)
INORDERTRAVERSE(16)
INORDERTRAVERSE(33)
INORDERTRAVERSE(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



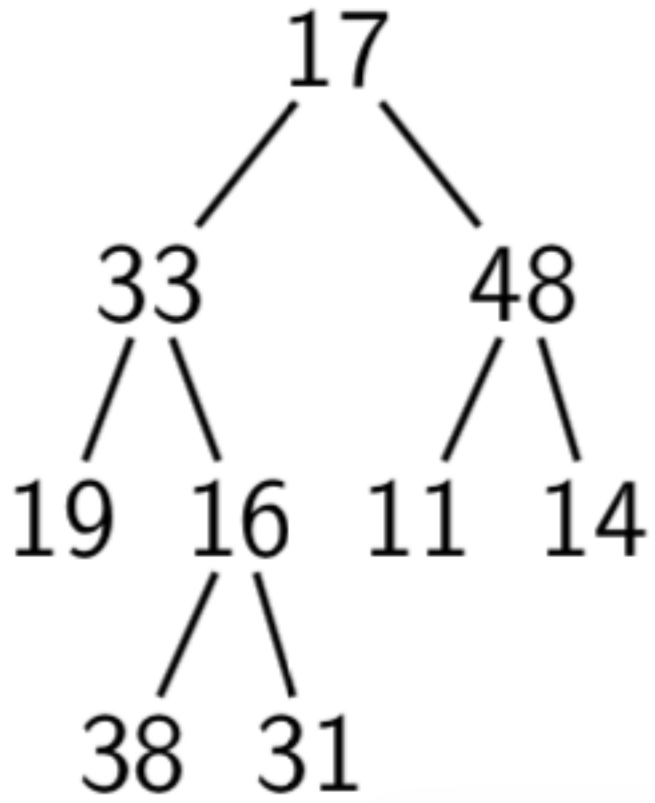
InorderTraverse(38)
InorderTraverse(16)
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



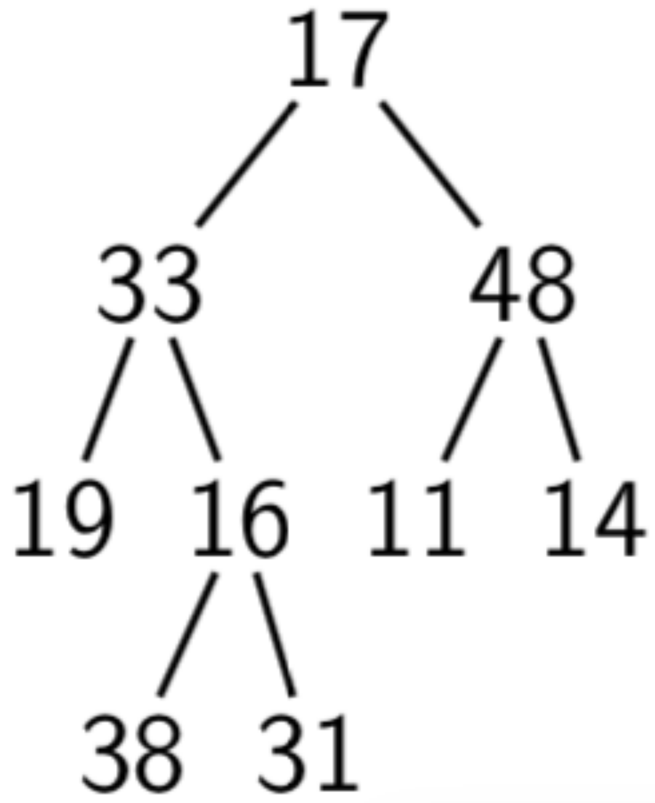
InorderTraverse(null)
InorderTraverse(38)
InorderTraverse(16)
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



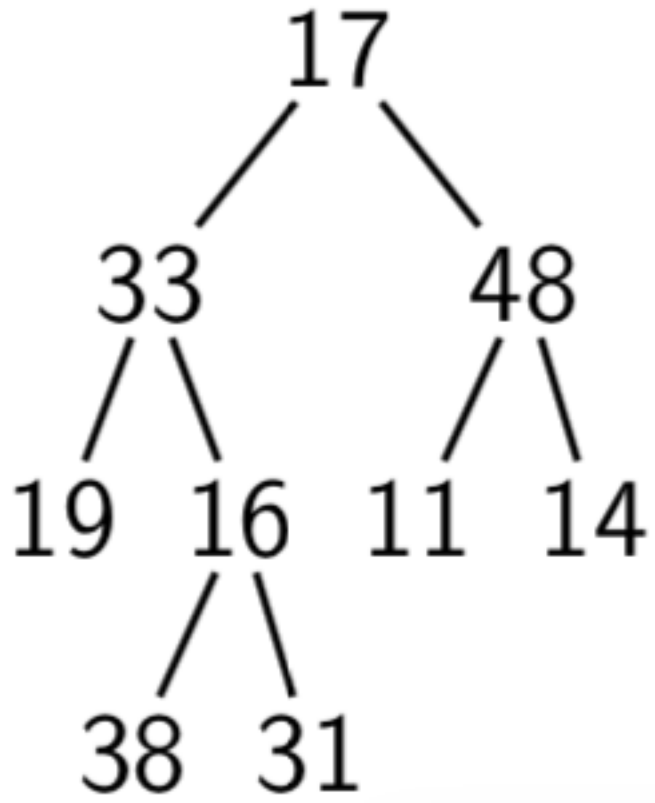
InorderTraverse(38)
InorderTraverse(16)
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



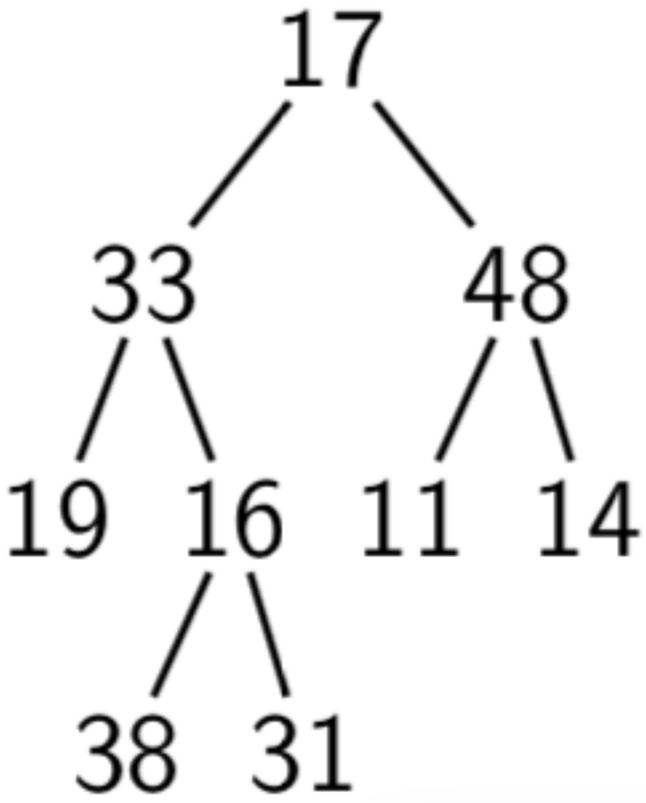
InorderTraverse(16)
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



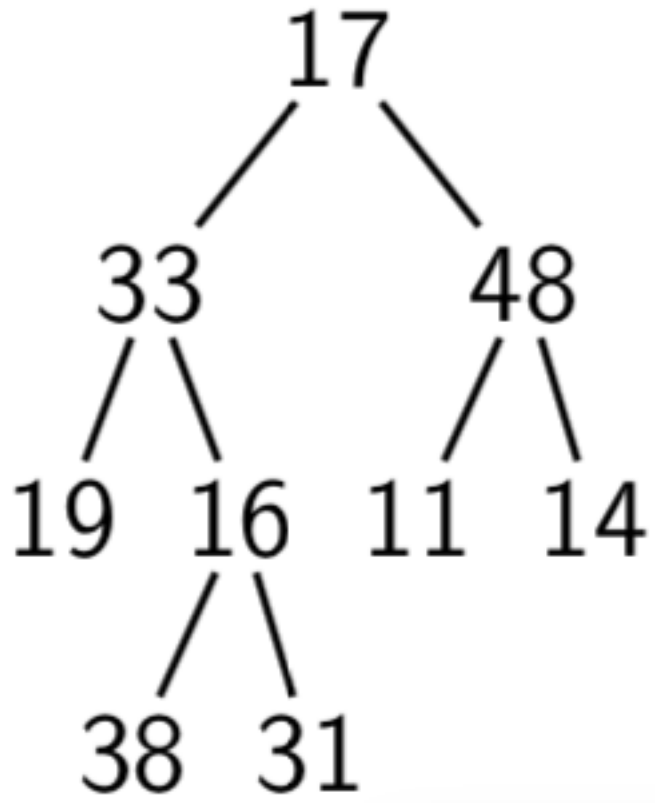
InorderTraverse(16)
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



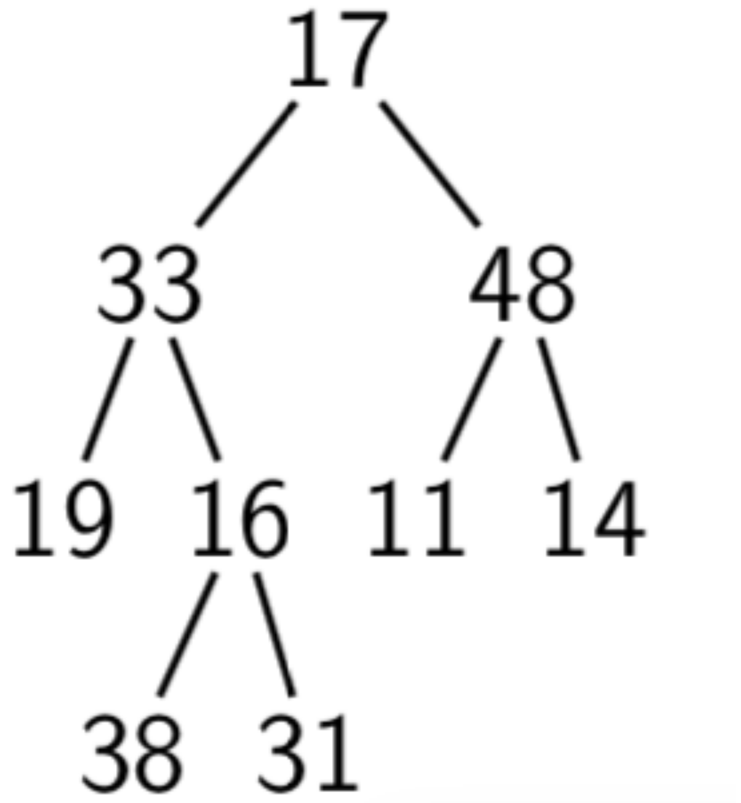
InorderTraverse(31)
InorderTraverse(16)
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16

```
procedure INORDERTRAVERSE(T)
    if $T \neq null$ then
        INORDERTRAVERSE(T.left)
        visit T.root
        INORDERTRAVERSE(T.right)
```



INORDERTRAVERSE(null)

INORDERTRAVERSE(31)

INORDERTRAVERSE(16)

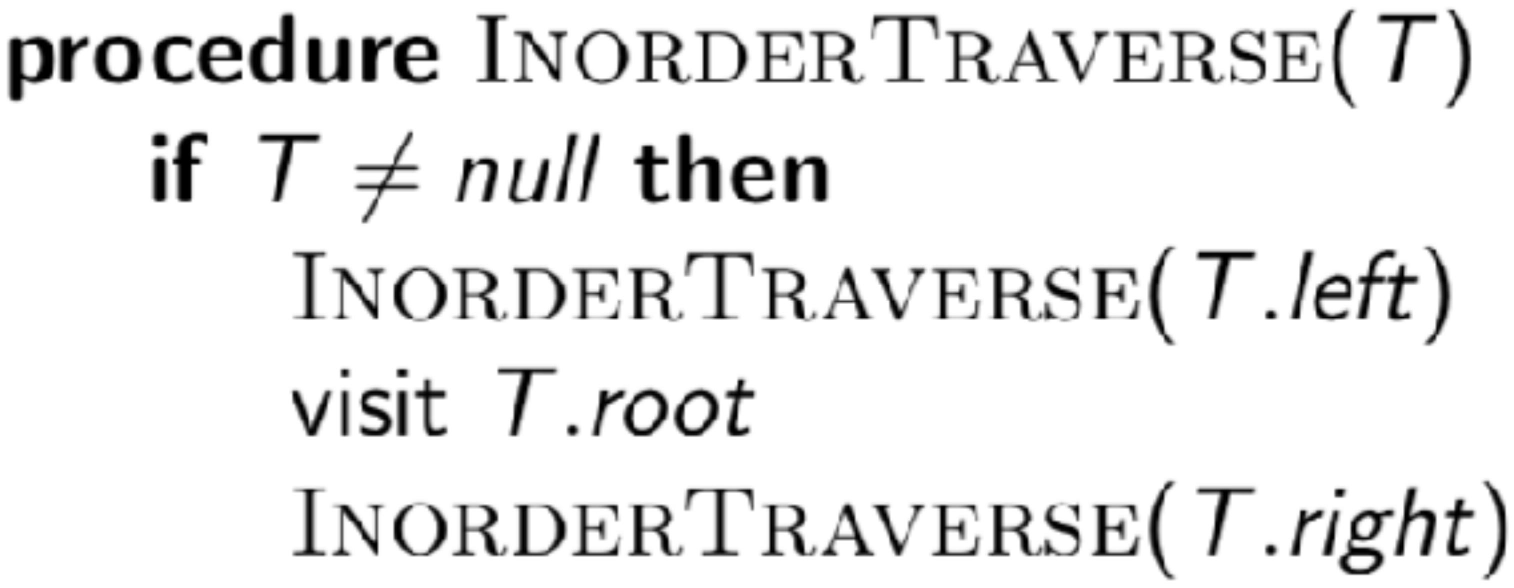
INORDERTRAVERSE(33)

INORDERTRAVERSE(17)

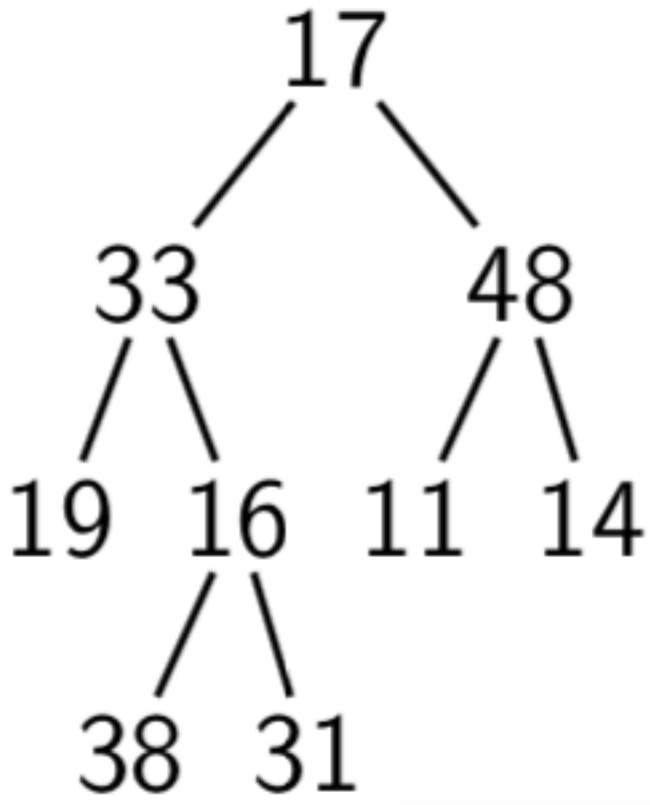
Call Stack

Inorder Traversal

Visit order: 19 33 38 16



```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



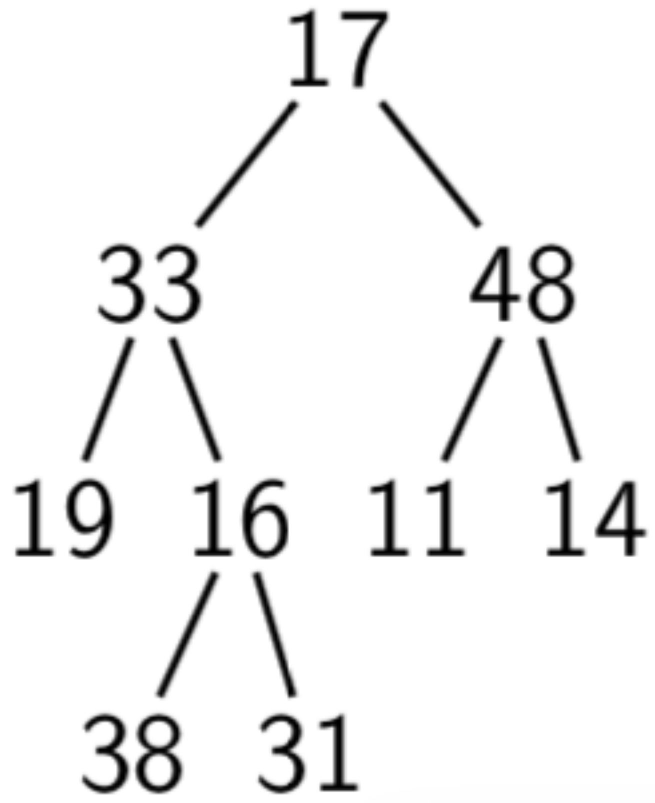
InorderTraverse(31)
InorderTraverse(16)
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31

```
procedure INORDERTRAVERSE(T)
    if $T \neq null$ then
        INORDERTRAVERSE(T.left)
        visit T.root
        INORDERTRAVERSE(T.right)
```



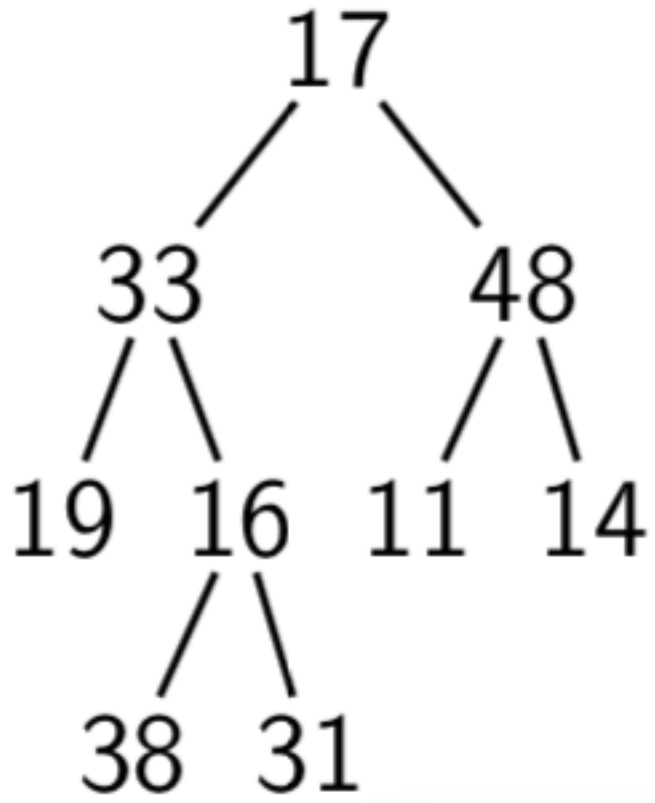
INORDERTRAVERSE(31)
INORDERTRAVERSE(16)
INORDERTRAVERSE(33)
INORDERTRAVERSE(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31

```
procedure InorderTraverse(T)
    if T ≠ null then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



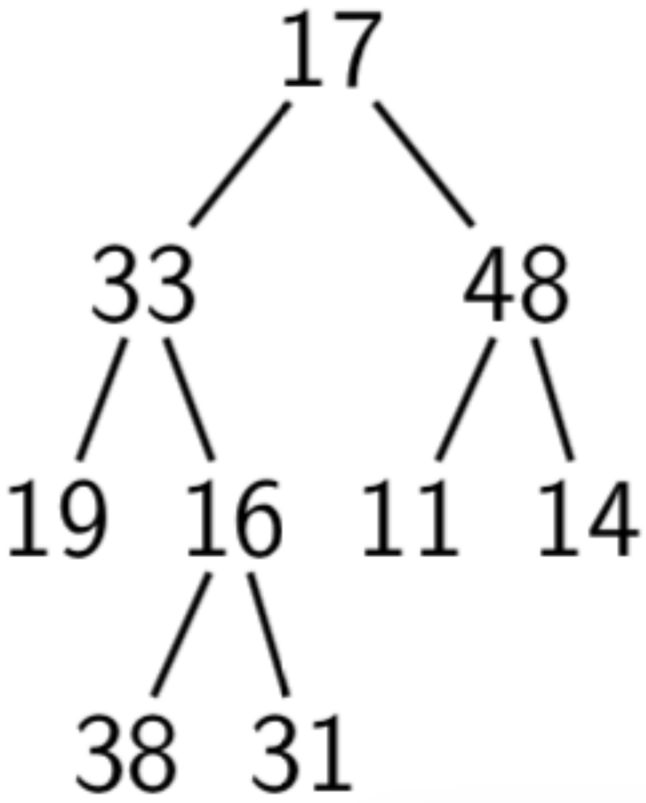
InorderTraverse(null)
InorderTraverse(31)
InorderTraverse(16)
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31

```
procedure InorderTraverse(T)
    if T ≠ null then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



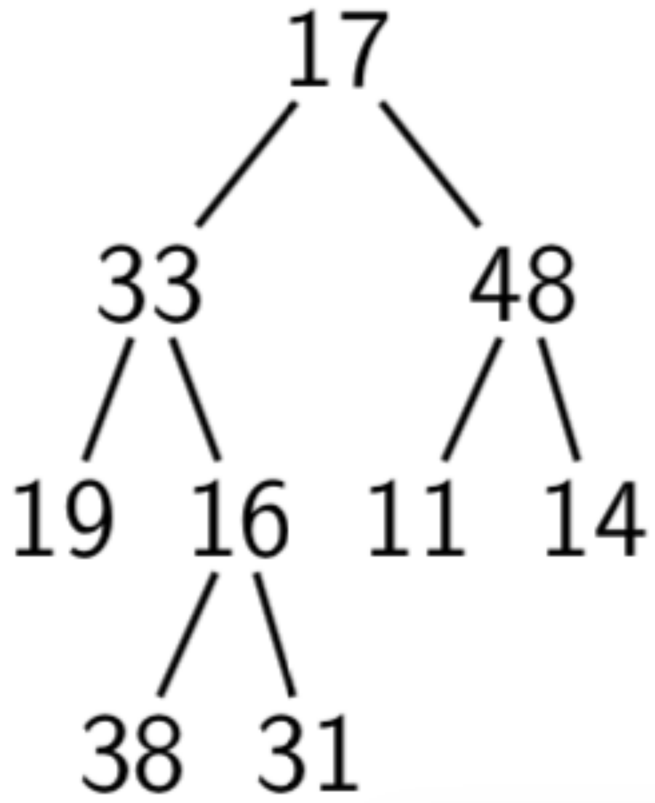
InorderTraverse(31)
InorderTraverse(16)
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



InorderTraverse(16)

InorderTraverse(33)

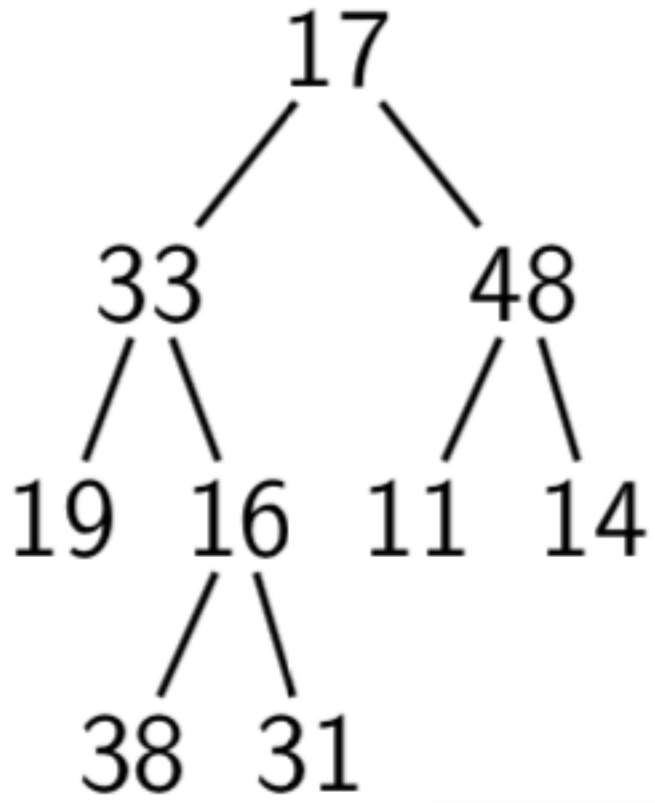
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



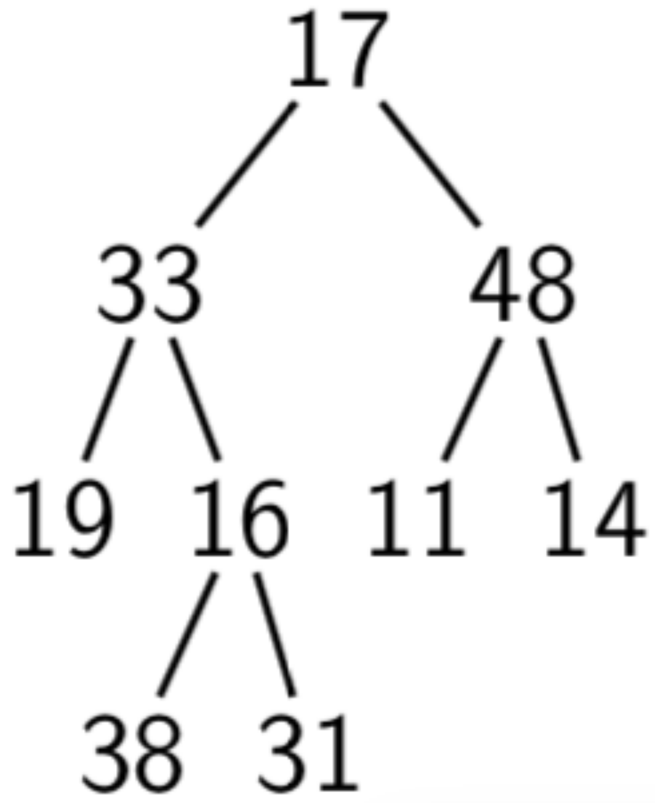
InorderTraverse(33)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



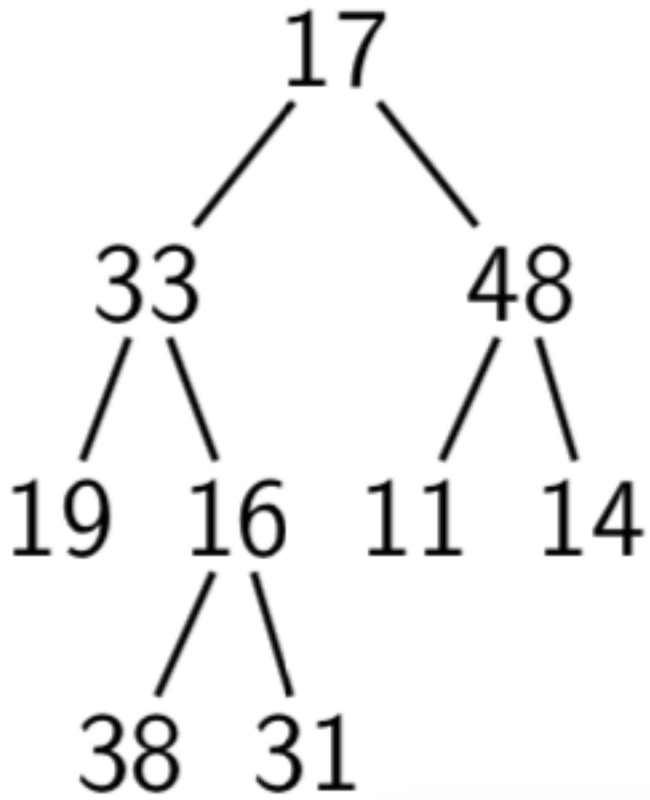
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



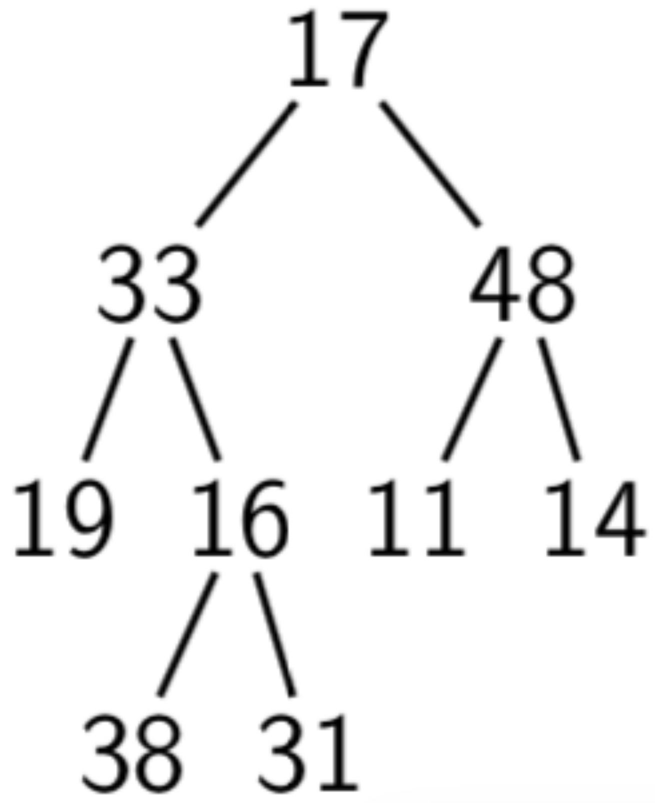
INORDERTRAVERSE(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



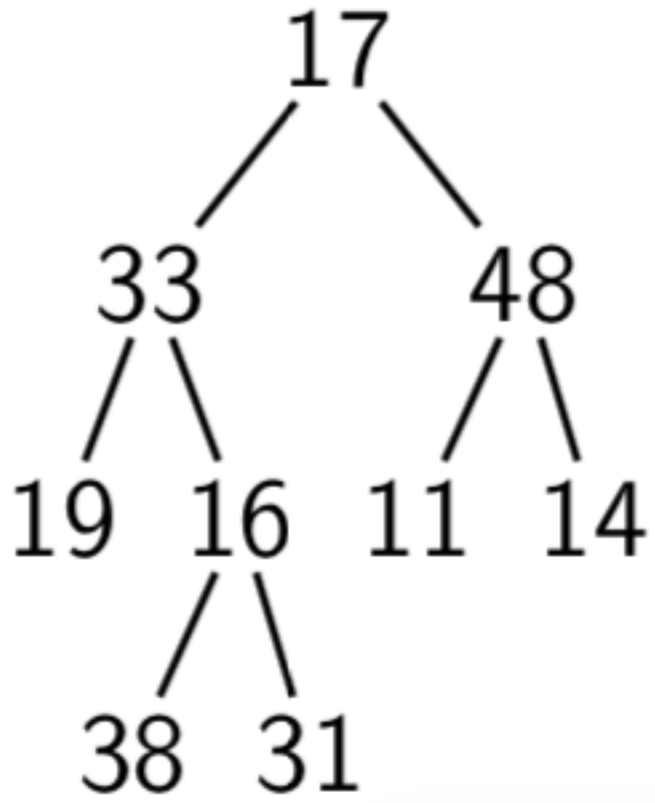
InorderTraverse(48)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



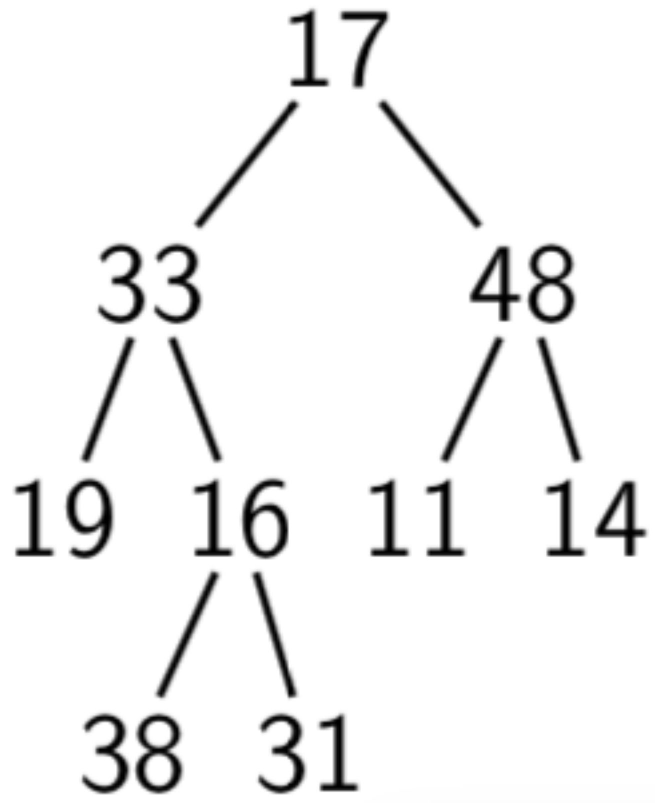
InorderTraverse(11)
InorderTraverse(48)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



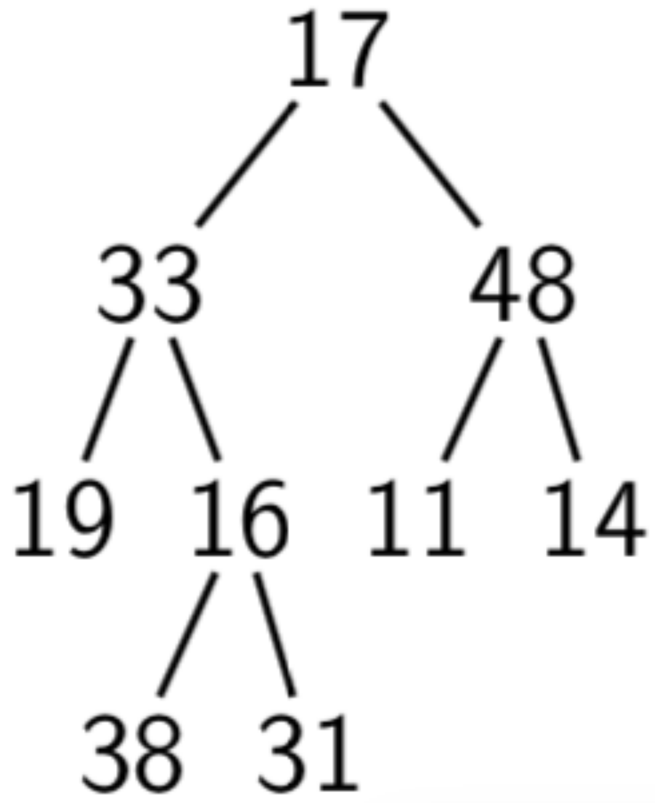
InorderTraverse(null)
InorderTraverse(11)
InorderTraverse(48)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



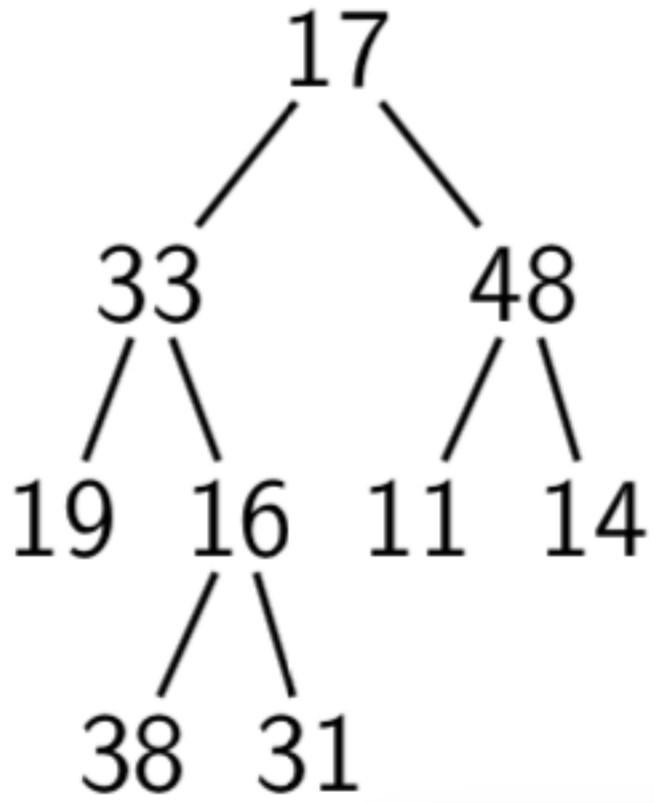
InorderTraverse(11)
InorderTraverse(48)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17 11

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



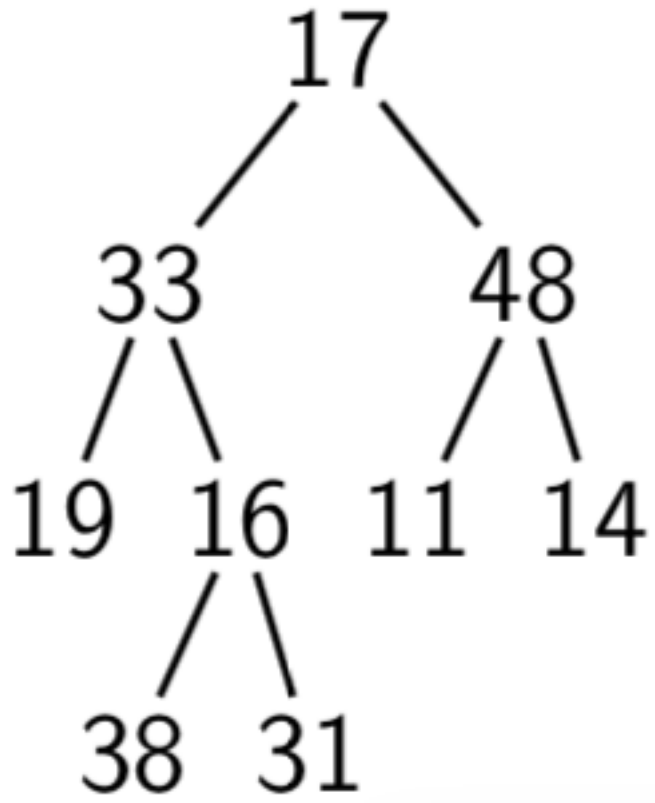
InorderTraverse(11)
InorderTraverse(48)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17 11

```
procedure INORDERTRAVERSE($T$)
    if $T \neq null$ then
        INORDERTRAVERSE($T.left$)
        visit $T.root$
        INORDERTRAVERSE($T.right$)
```



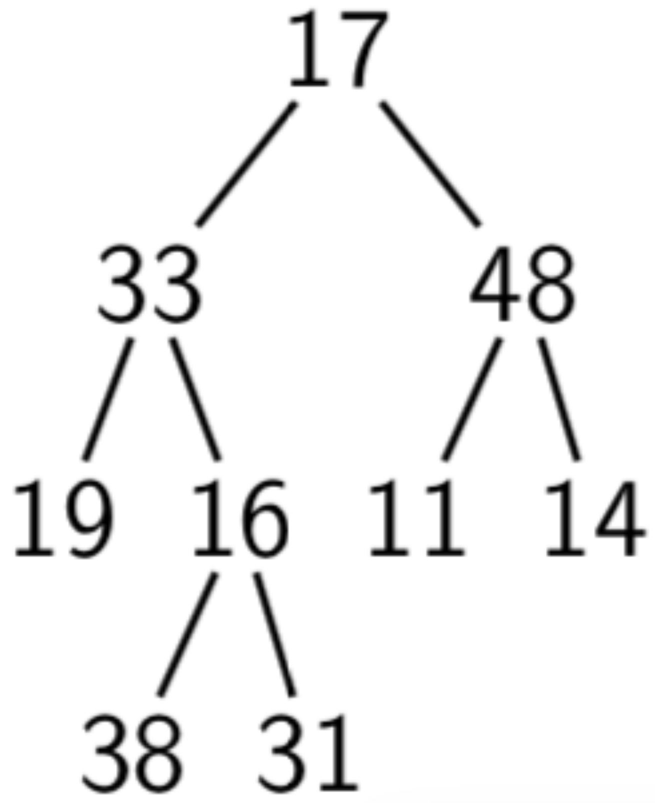
INORDERTRAVERSE(null)
INORDERTRAVERSE(11)
INORDERTRAVERSE(48)
INORDERTRAVERSE(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17 11

```
procedure INORDERTRAVERSE($T$)
    if $T \neq null$ then
        INORDERTRAVERSE($T.left$)
        visit $T.root$
        INORDERTRAVERSE($T.right$)
```



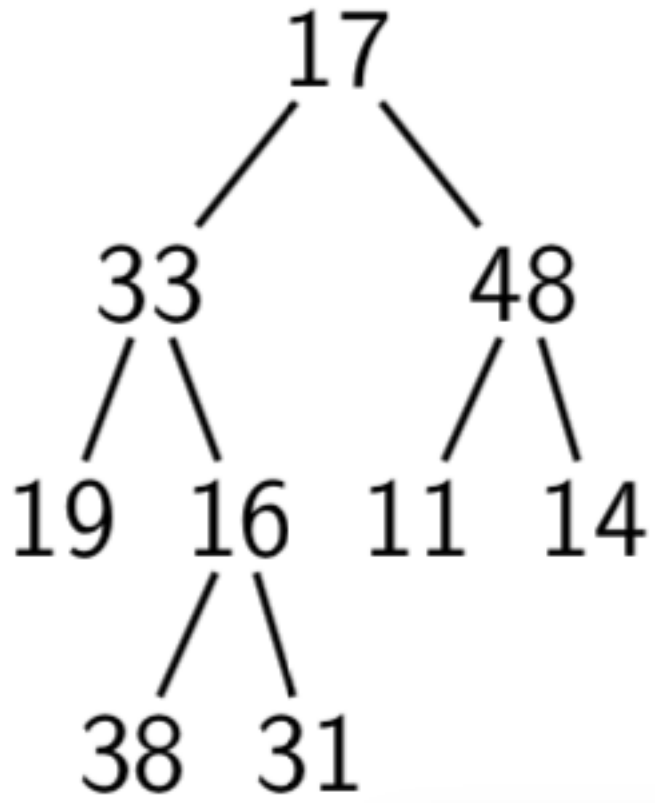
INORDERTRAVERSE(11)
INORDERTRAVERSE(48)
INORDERTRAVERSE(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17 11

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



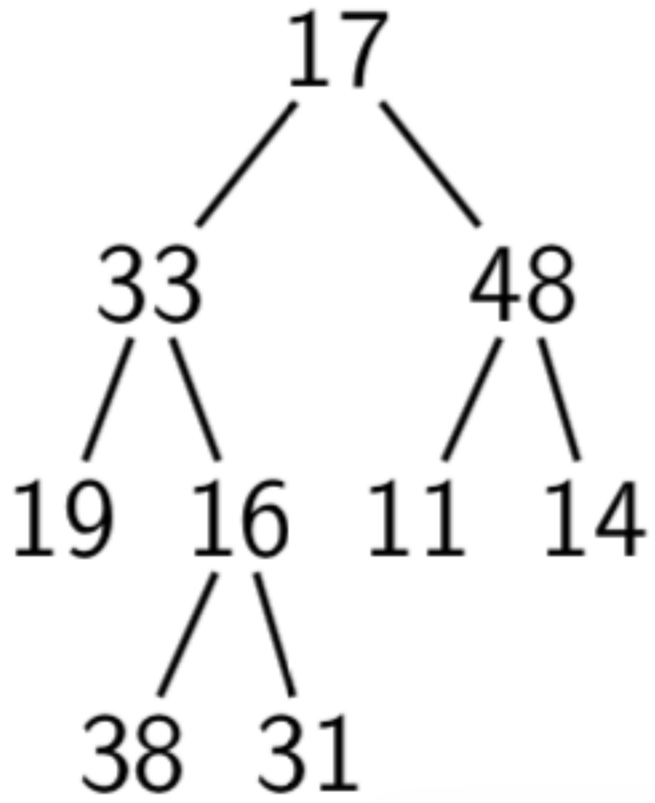
InorderTraverse(48)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17 11 48

```
procedure INORDERTRAVERSE($T$)
    if $T \neq null$ then
        INORDERTRAVERSE($T.left$)
        visit $T.root$
        INORDERTRAVERSE($T.right$)
```



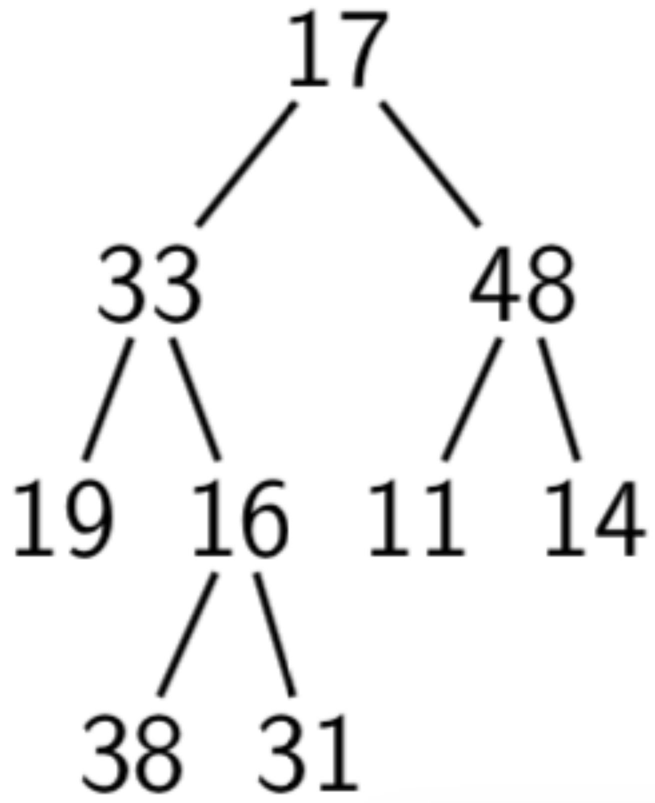
INORDERTRAVERSE(48)
INORDERTRAVERSE(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17 11 48

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



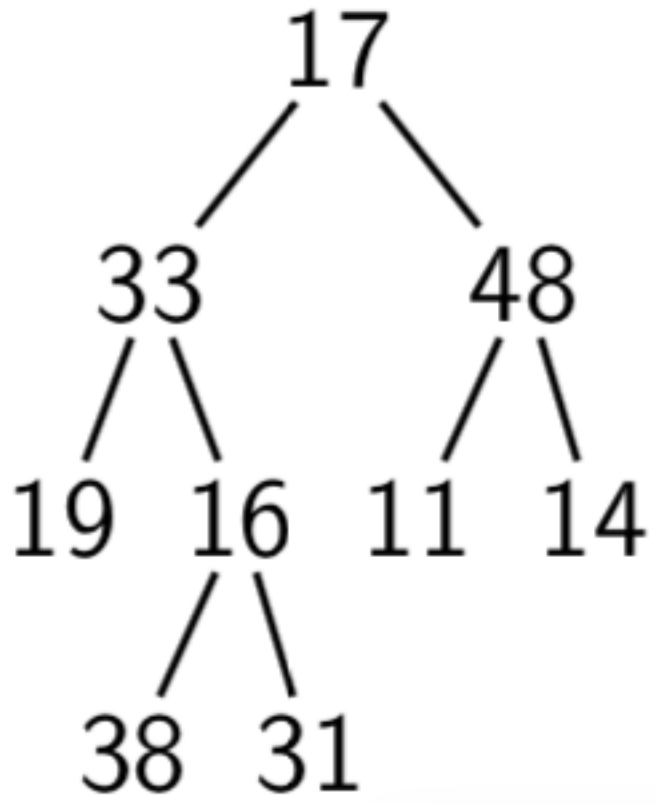
InorderTraverse(14)
InorderTraverse(48)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17 11 48

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



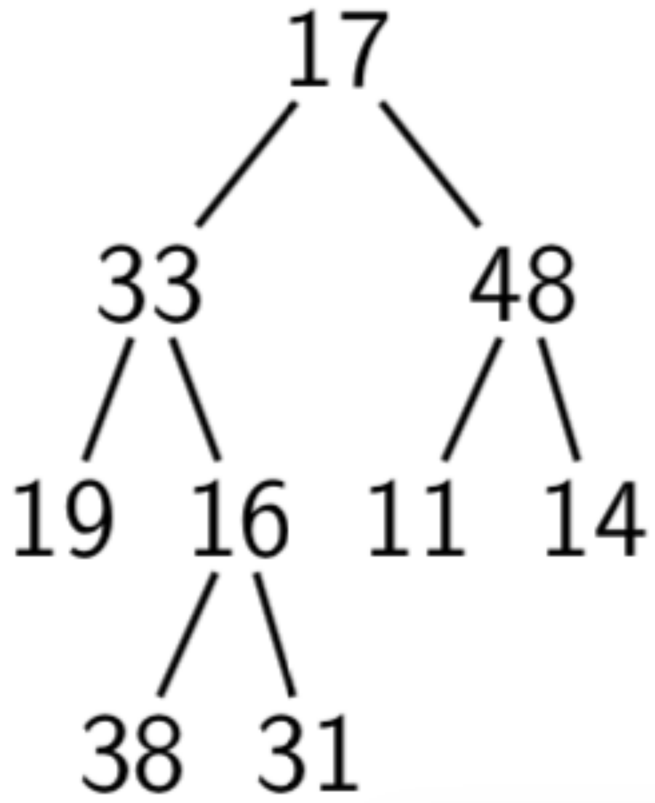
InorderTraverse(null)
InorderTraverse(14)
InorderTraverse(48)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17 11 48

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```



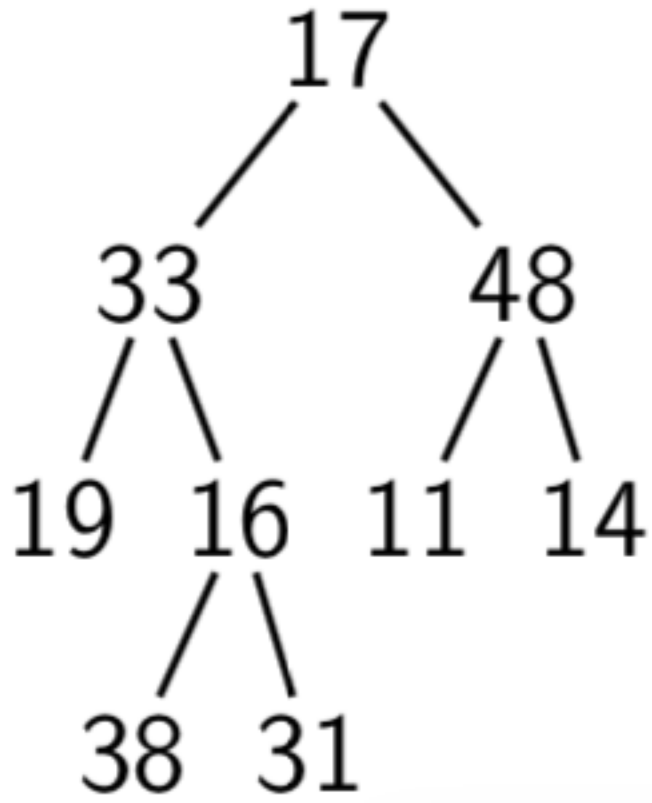
InorderTraverse(14)
InorderTraverse(48)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17 11 48 14

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



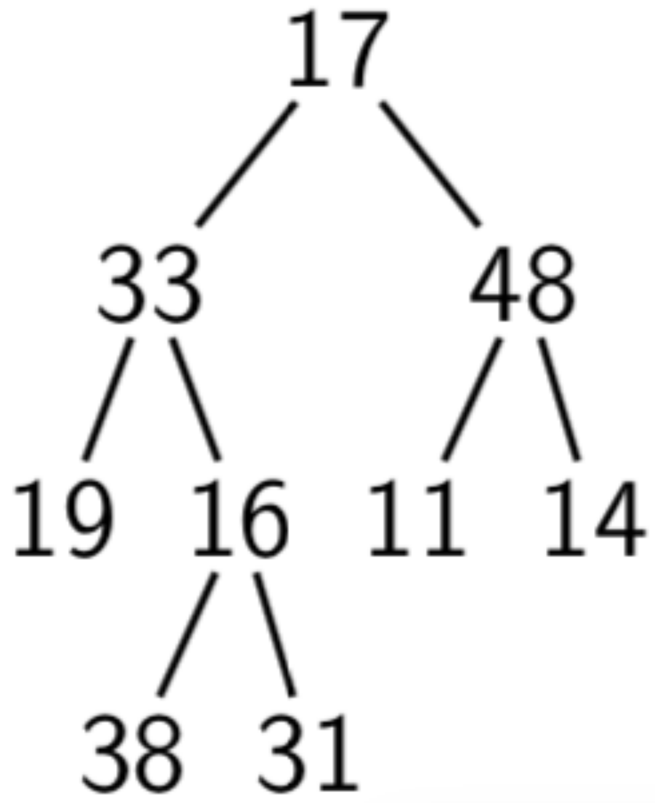
InorderTraverse(14)
InorderTraverse(48)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17 11 48 14

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



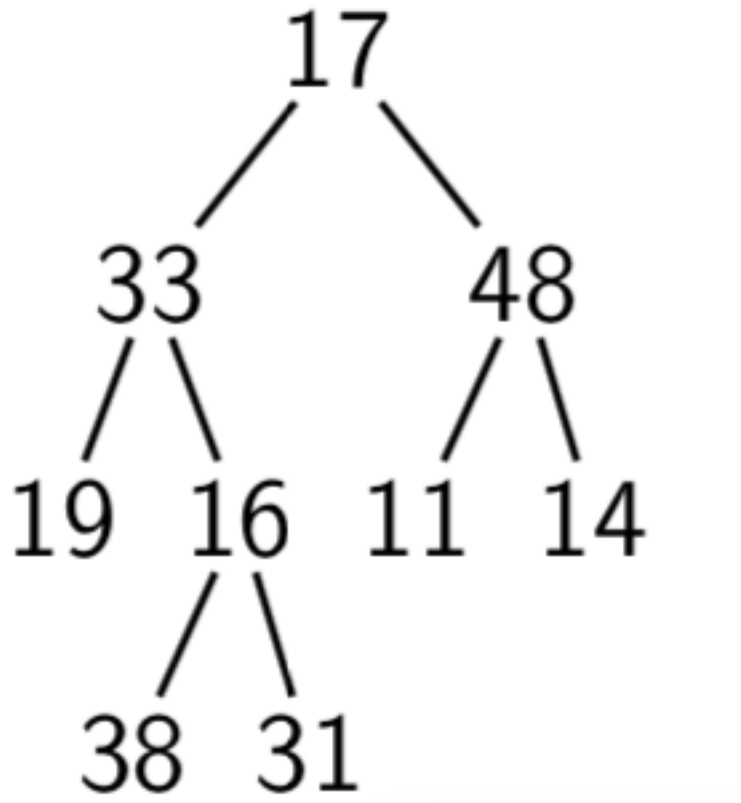
InorderTraverse(null)
InorderTraverse(14)
InorderTraverse(48)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17 11 48 14

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



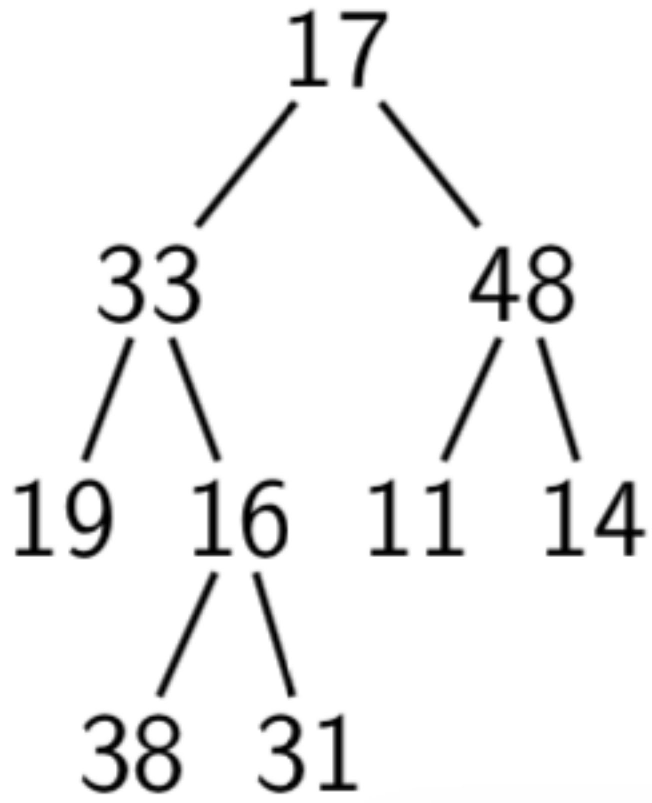
InorderTraverse(14)
InorderTraverse(48)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17 11 48 14

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



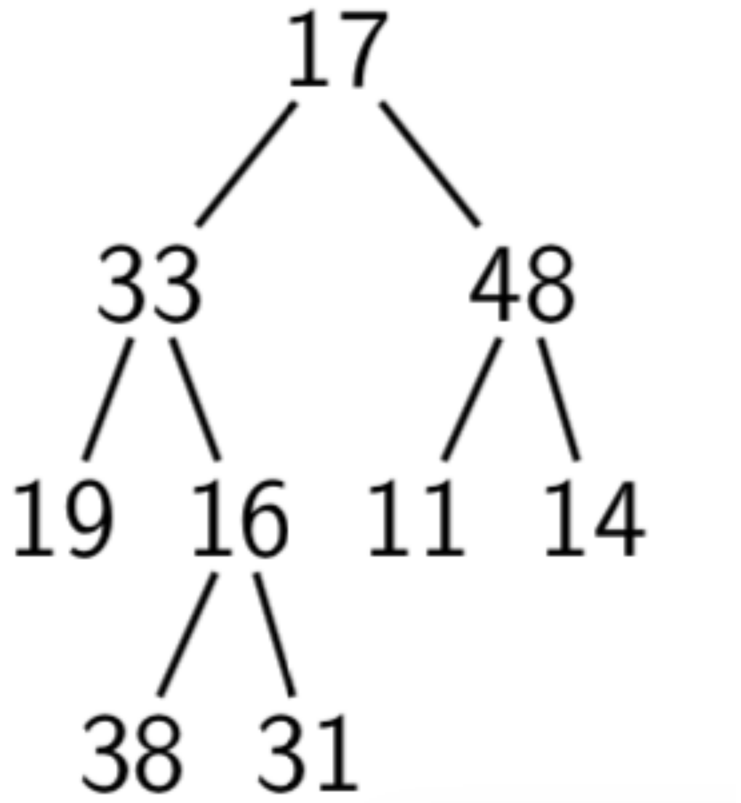
InorderTraverse(48)
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17 11 48 14

```
procedure InorderTraverse($T$)
    if $T \neq null$ then
        InorderTraverse($T.left$)
        visit $T.root$
        InorderTraverse($T.right$)
```



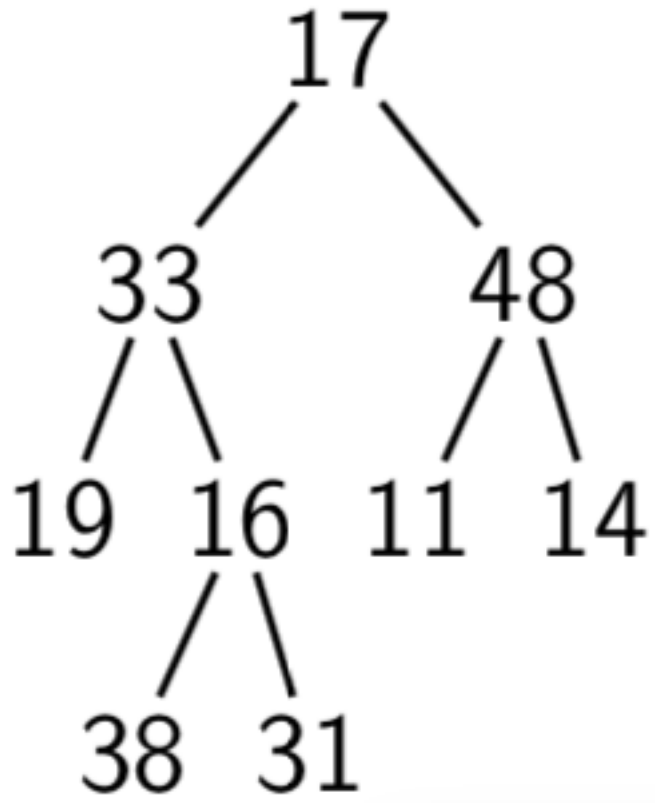
InorderTraverse(17)

Call Stack

Inorder Traversal

Visit order: 19 33 38 16 31 17 11 48 14

```
procedure InorderTraverse(T)
    if $T \neq null$ then
        InorderTraverse(T.left)
        visit T.root
        InorderTraverse(T.right)
```

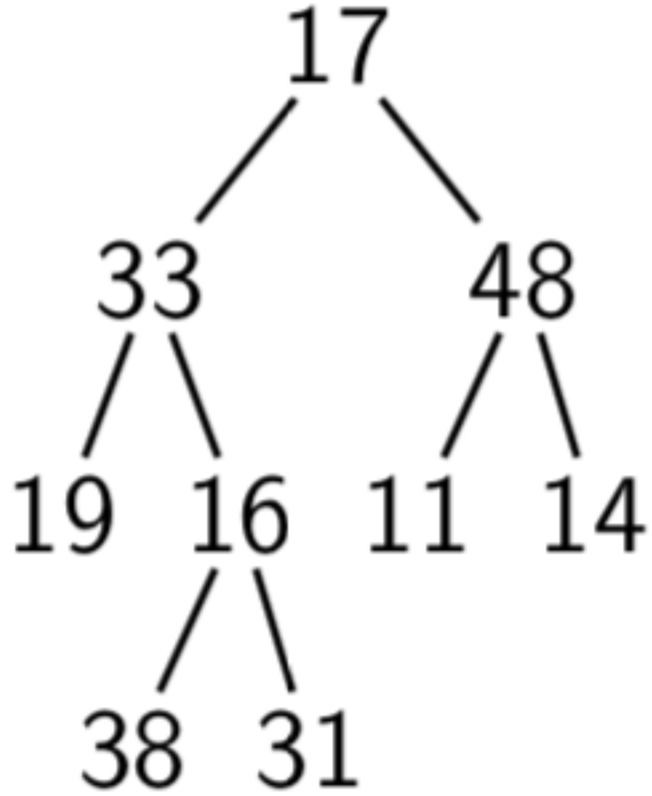


Call Stack

Postorder Traversal

Visit order:

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



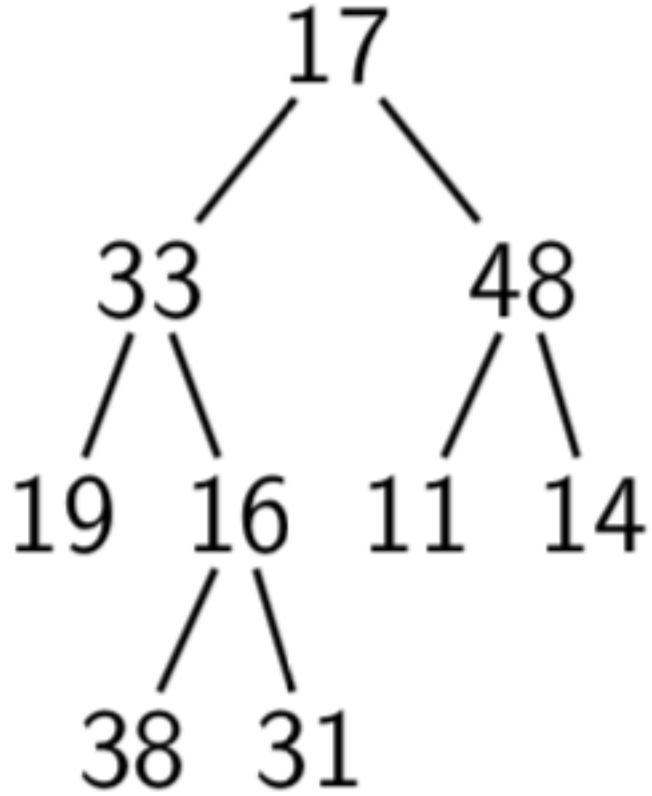
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order:

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



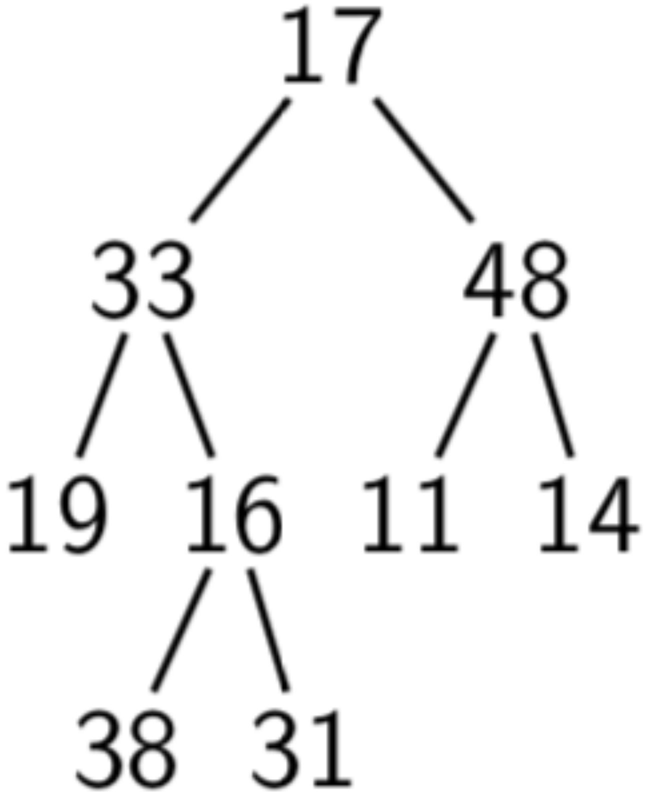
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order:

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



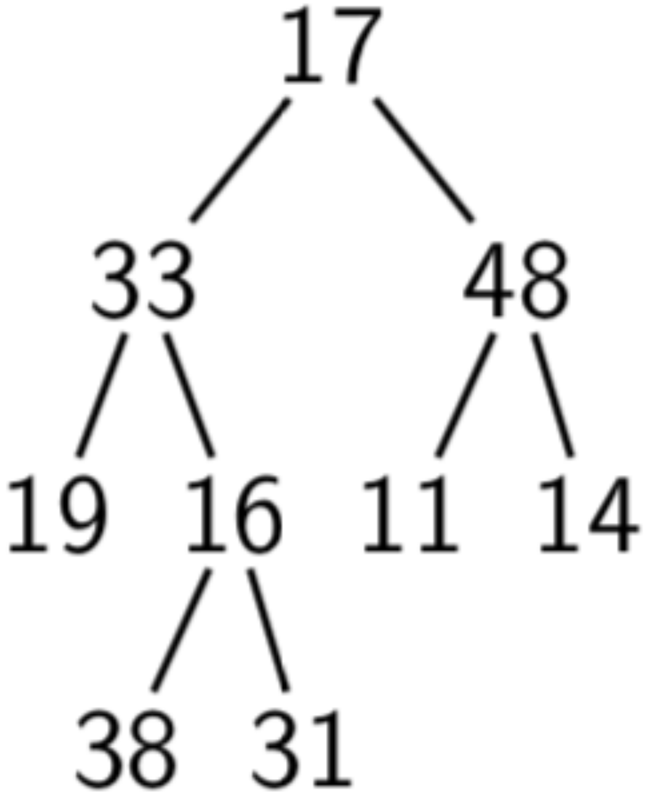
PostorderTraverse(19)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order:

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



PostorderTraverse(null)

PostorderTraverse(19)

PostorderTraverse(33)

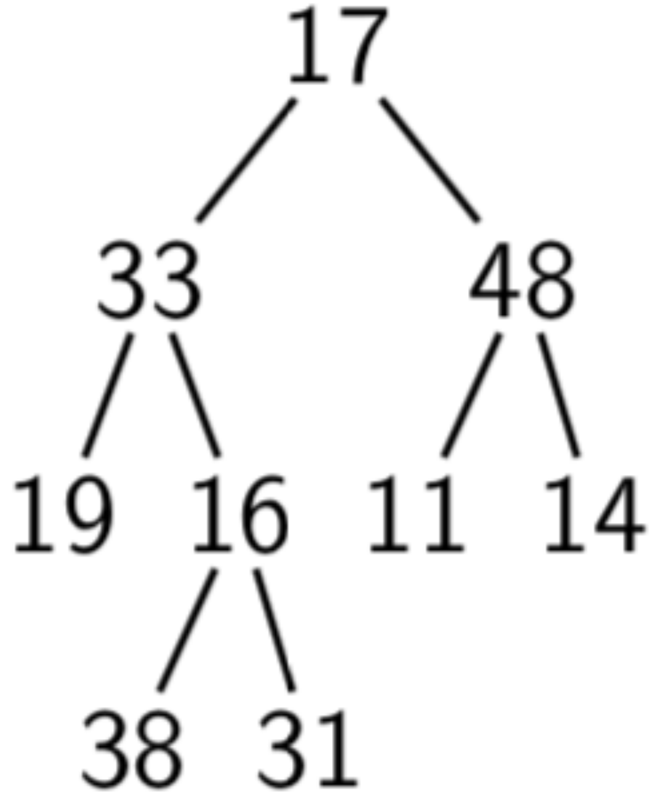
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order:

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



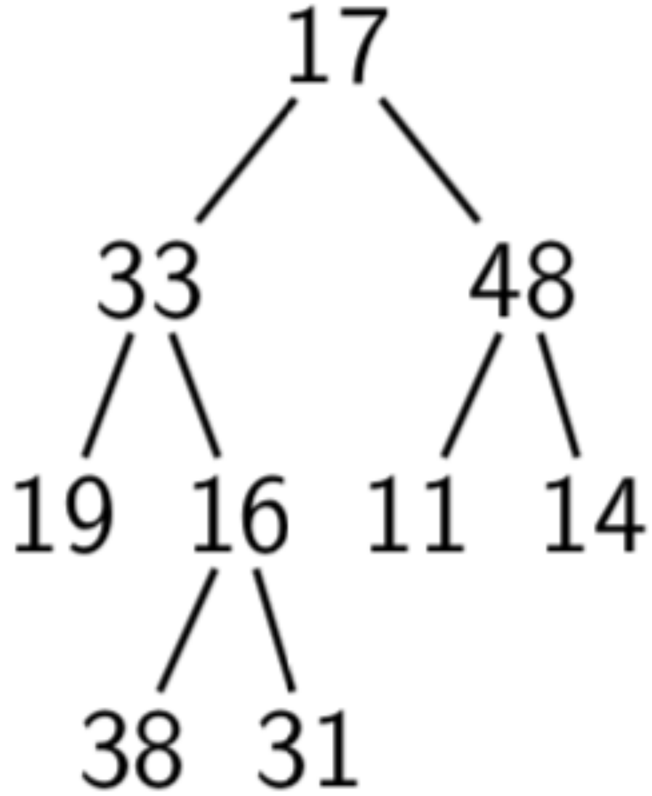
PostorderTraverse(19)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order:

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



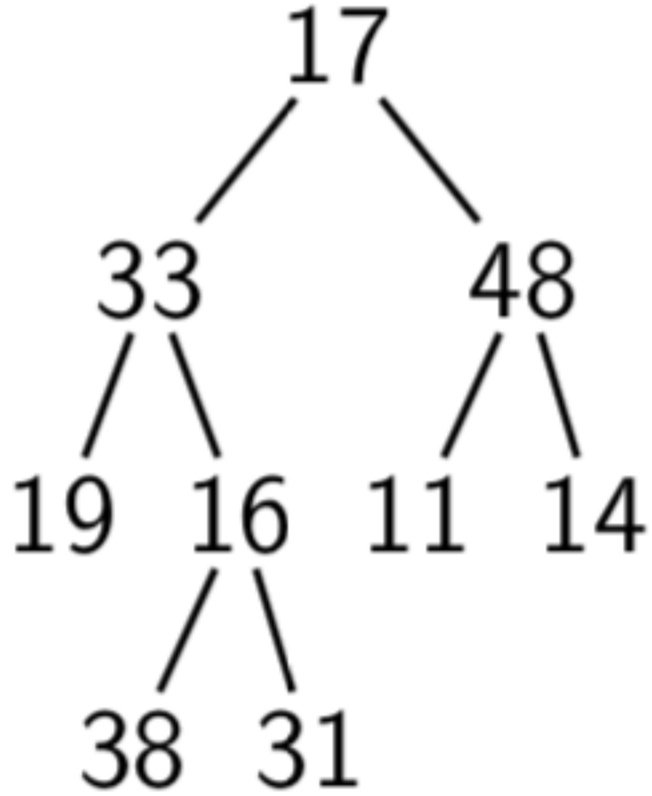
PostorderTraverse(null)
PostorderTraverse(19)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order:

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



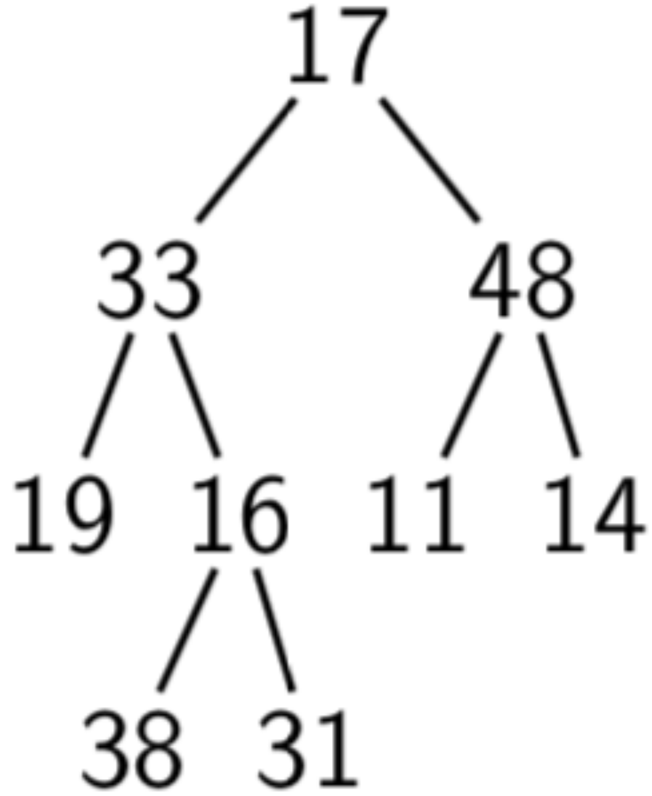
PostorderTraverse(19)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



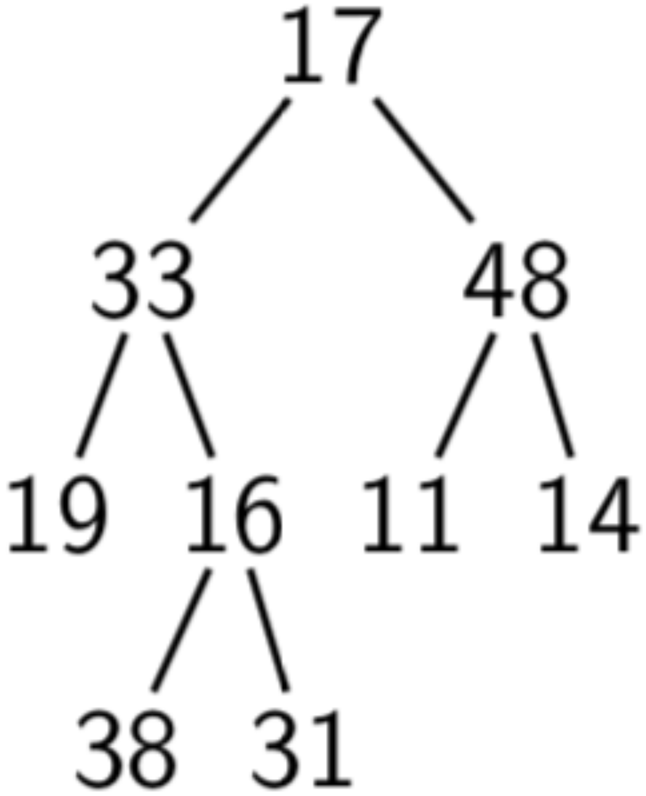
PostorderTraverse(19)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



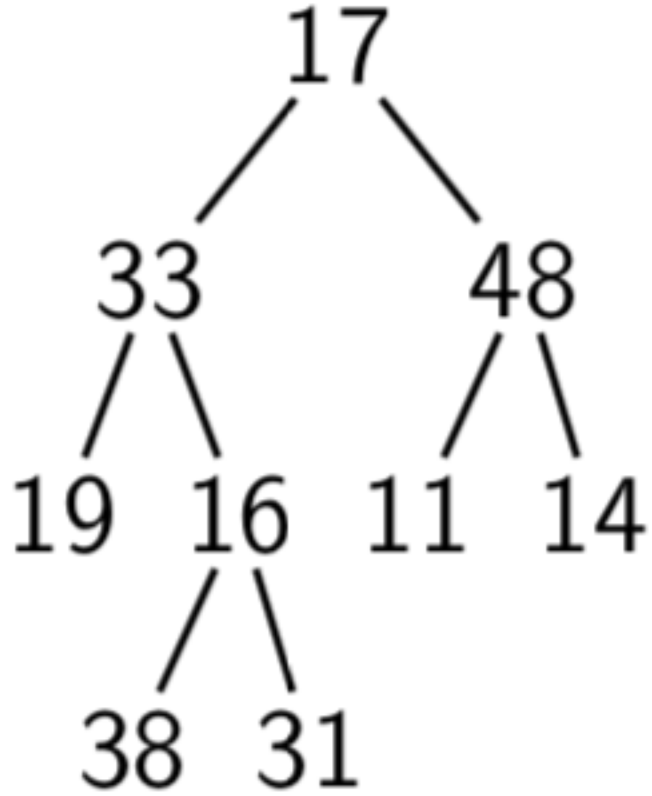
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



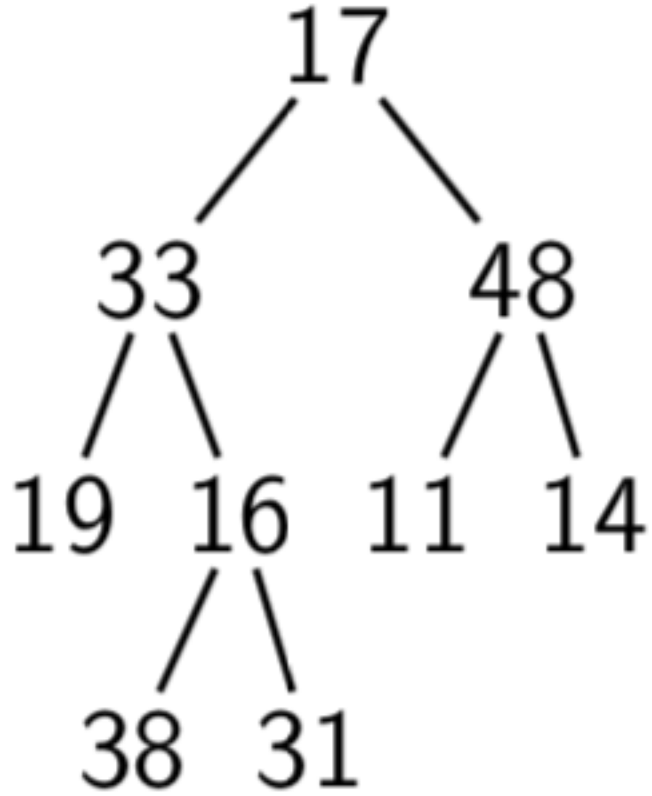
PostorderTraverse(16)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



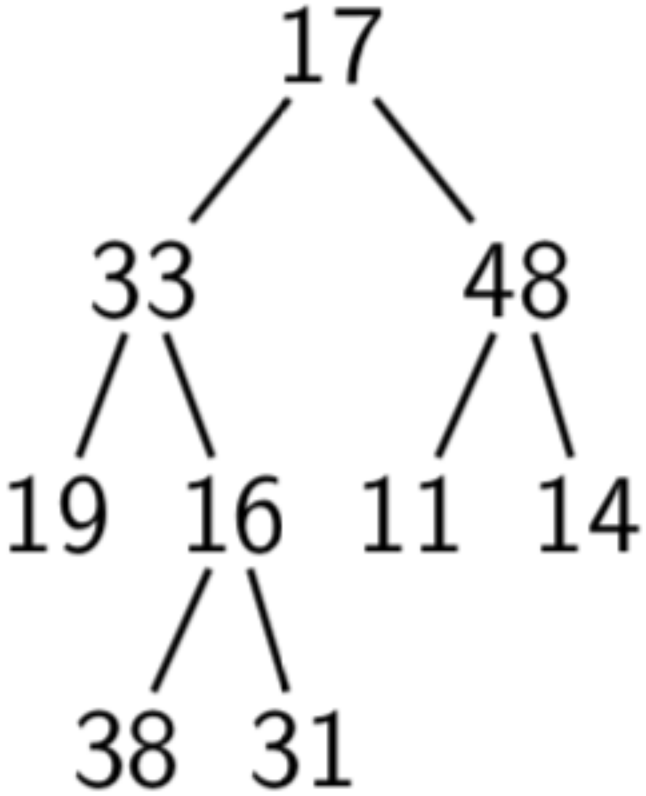
PostorderTraverse(38)
PostorderTraverse(16)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



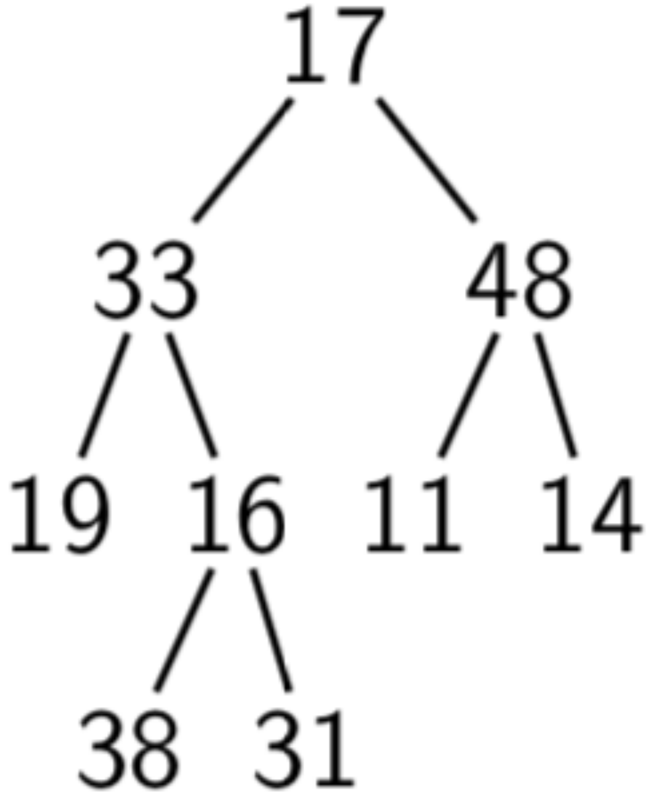
PostorderTraverse(null)
PostorderTraverse(38)
PostorderTraverse(16)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



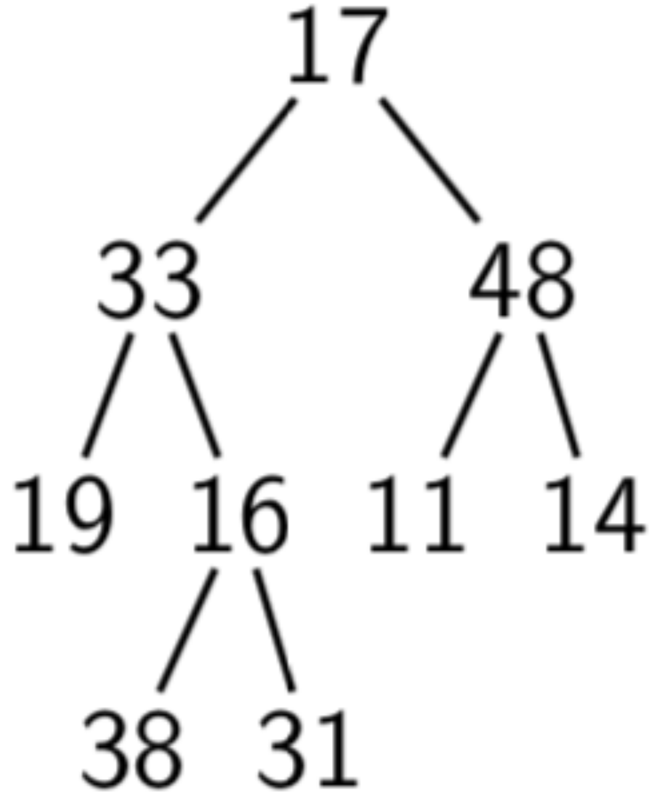
PostorderTraverse(38)
PostorderTraverse(16)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



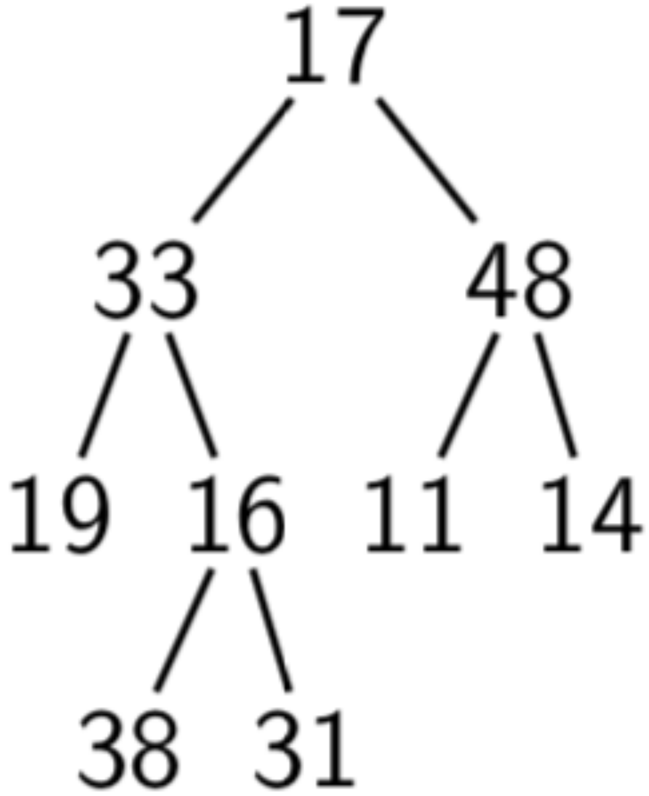
POSTORDERTRAVERSE(null)
POSTORDERTRAVERSE(38)
POSTORDERTRAVERSE(16)
POSTORDERTRAVERSE(33)
POSTORDERTRAVERSE(17)

Call Stack

Postorder Traversal

Visit order: 19

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



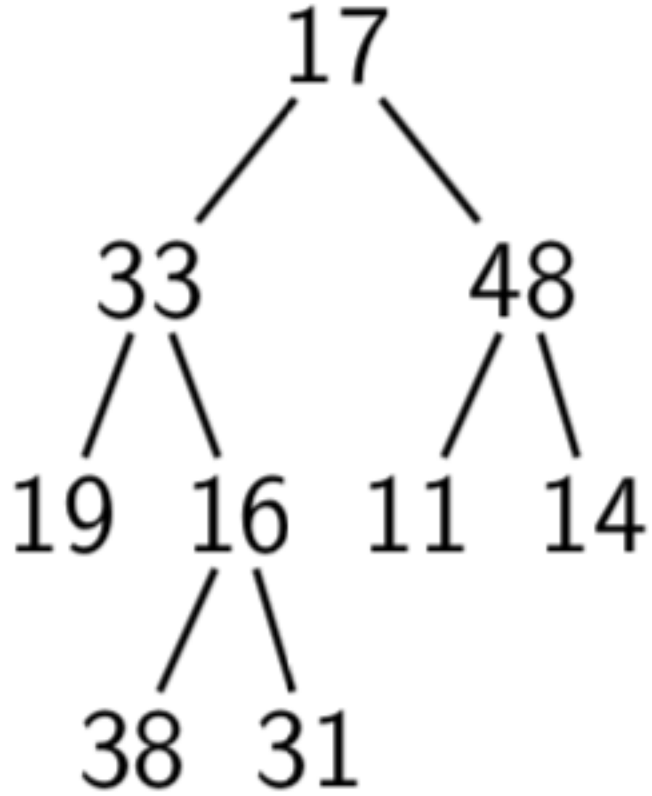
PostorderTraverse(38)
PostorderTraverse(16)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



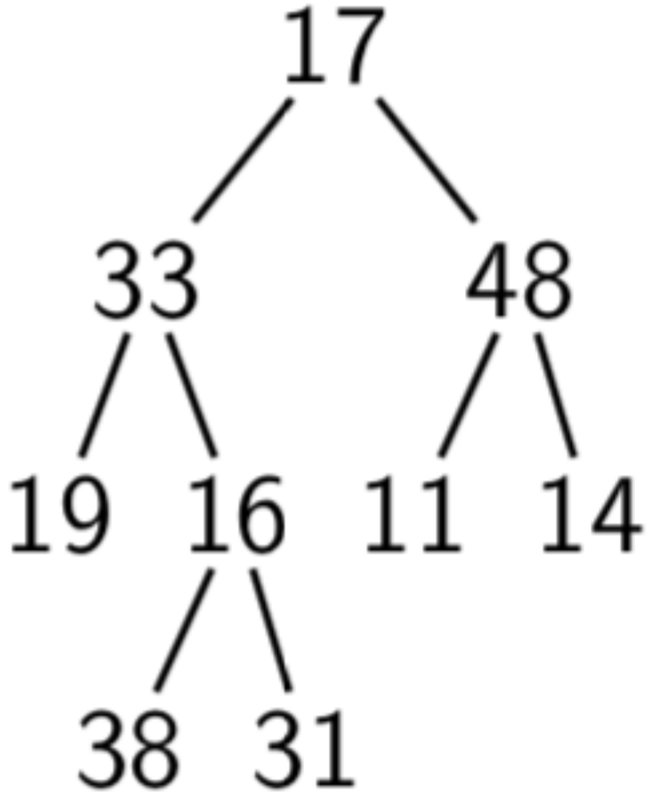
PostorderTraverse(38)
PostorderTraverse(16)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



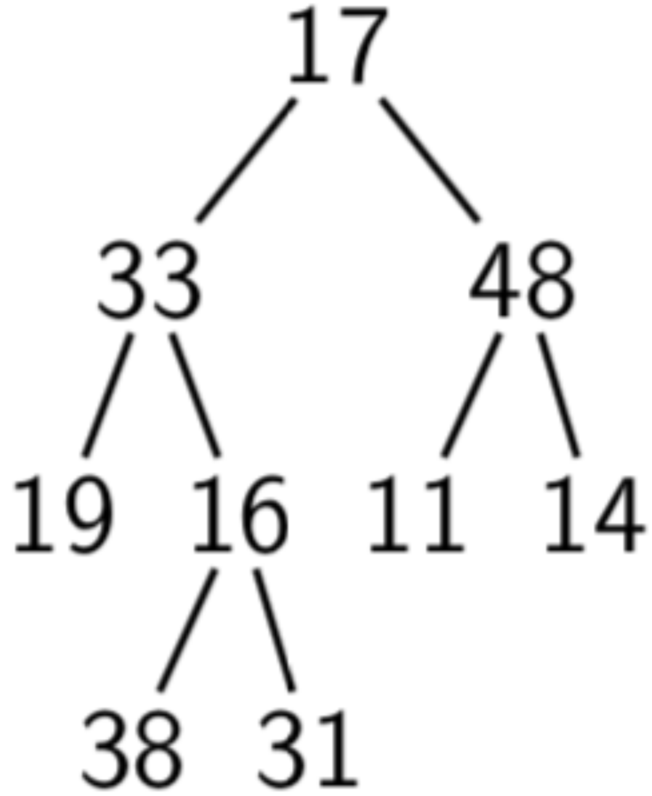
PostorderTraverse(16)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



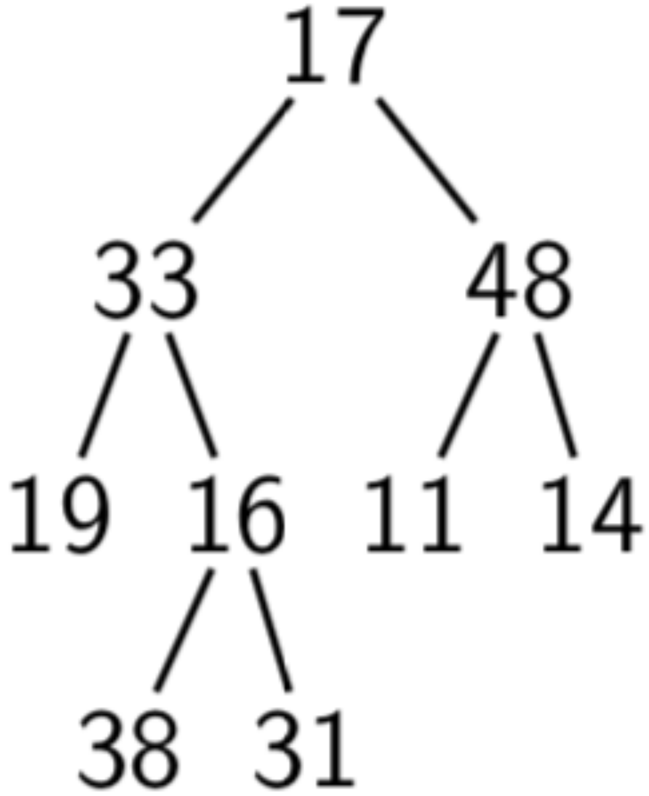
PostorderTraverse(31)
PostorderTraverse(16)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



PostorderTraverse(31)
PostorderTraverse(16)
PostorderTraverse(33)
PostorderTraverse(17)

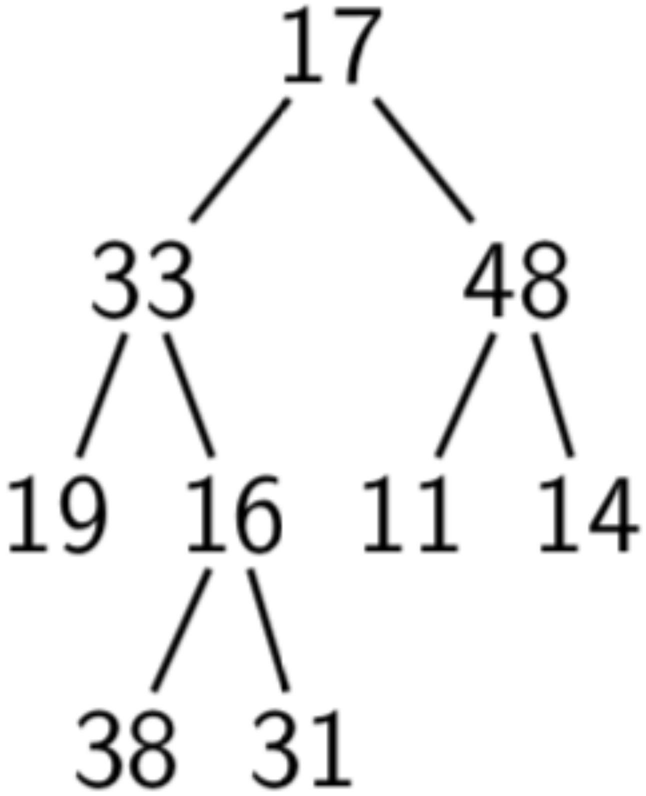
Call Stack

(…skipping the calls to PostorderTraverse(null)…)

Postorder Traversal

Visit order: 19 38 31

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



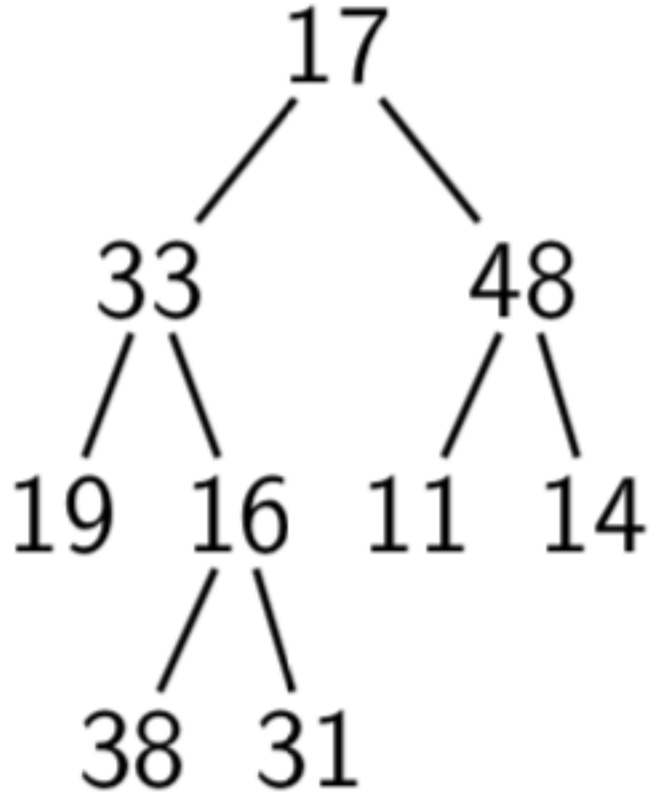
PostorderTraverse(31)
PostorderTraverse(16)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



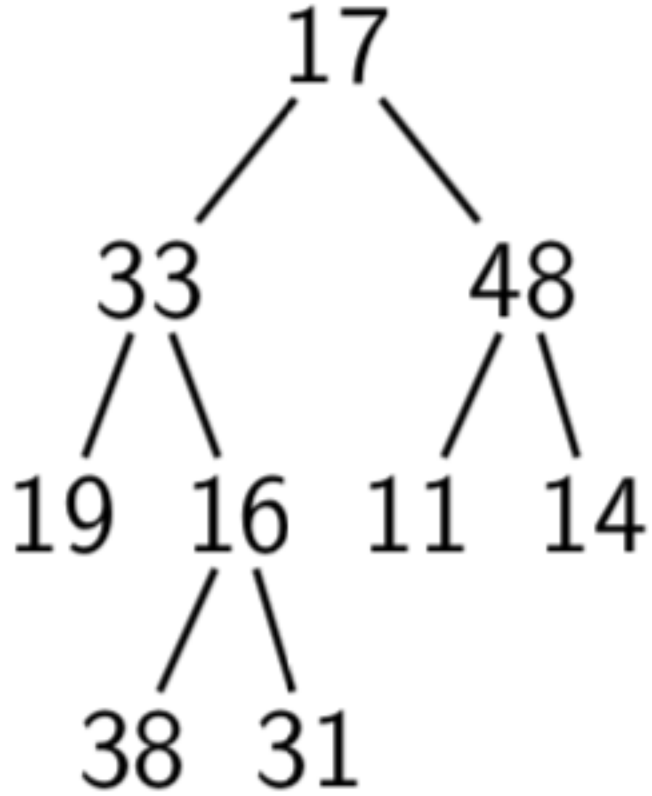
PostorderTraverse(16)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



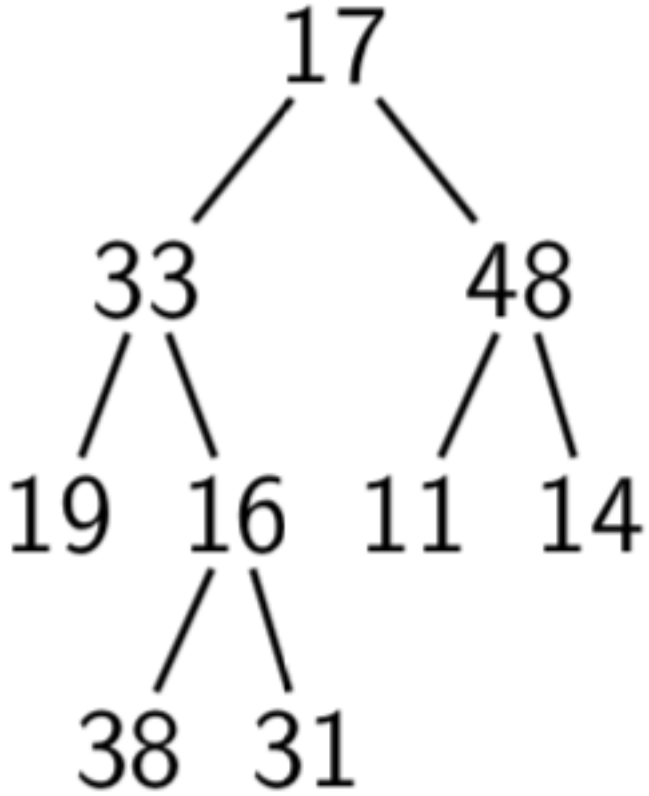
PostorderTraverse(16)
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



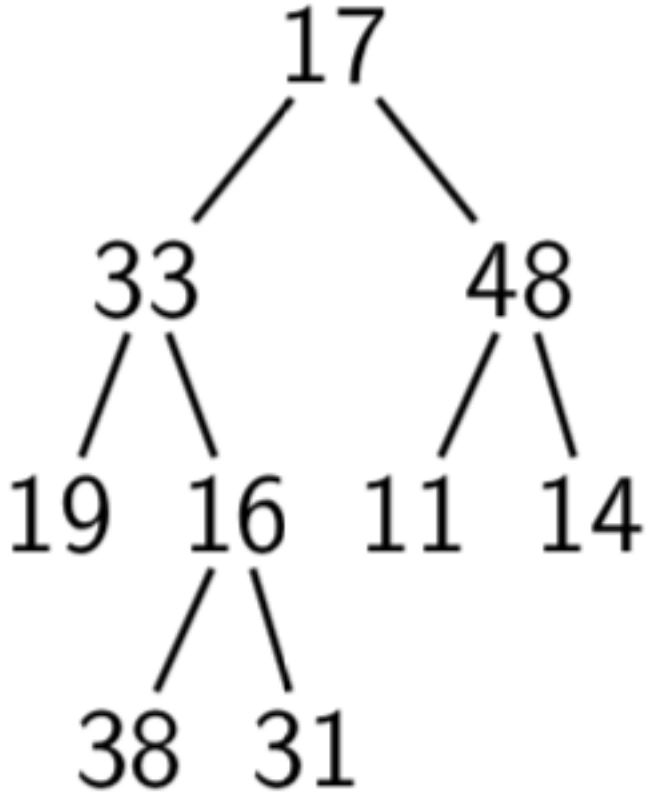
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16 33

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



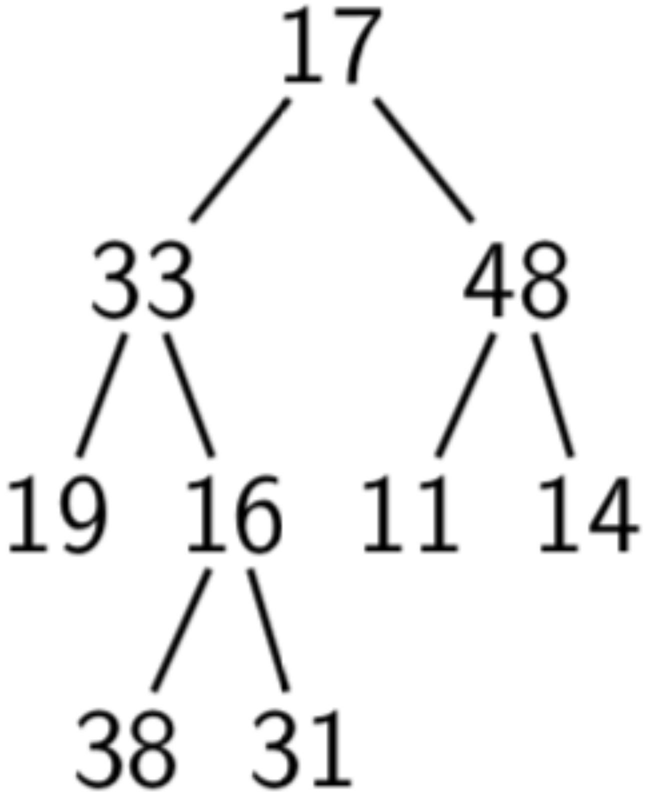
PostorderTraverse(33)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16 33

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



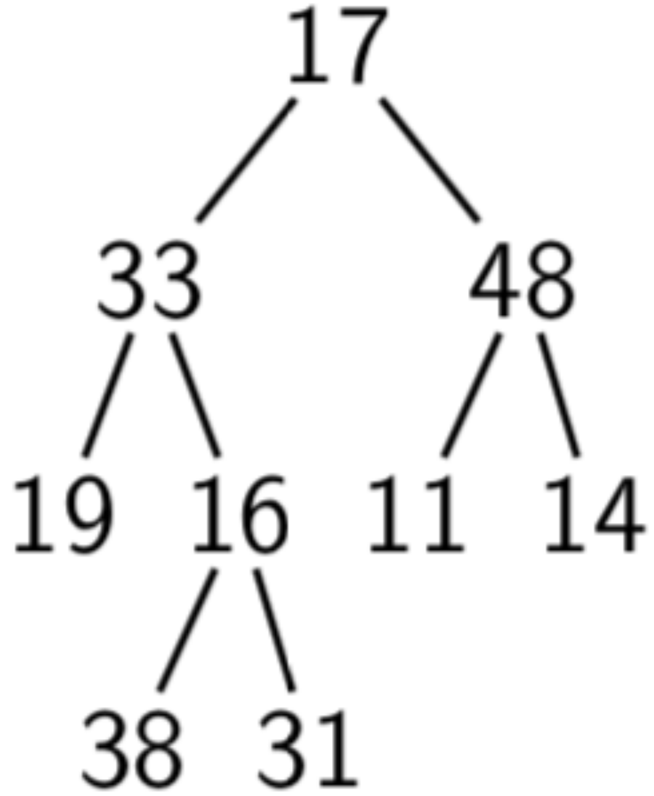
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16 33

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



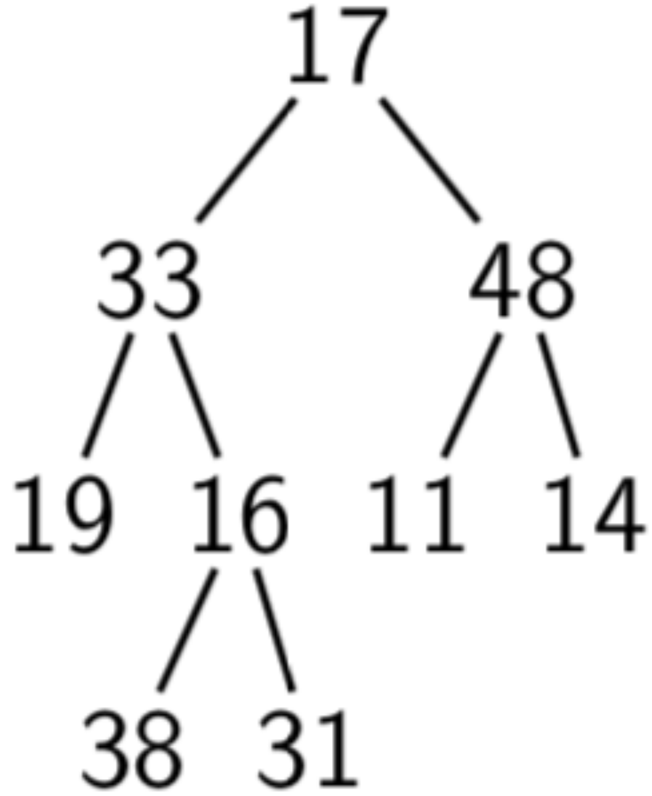
PostorderTraverse(48)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16 33

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



PostorderTraverse(11)

PostorderTraverse(48)

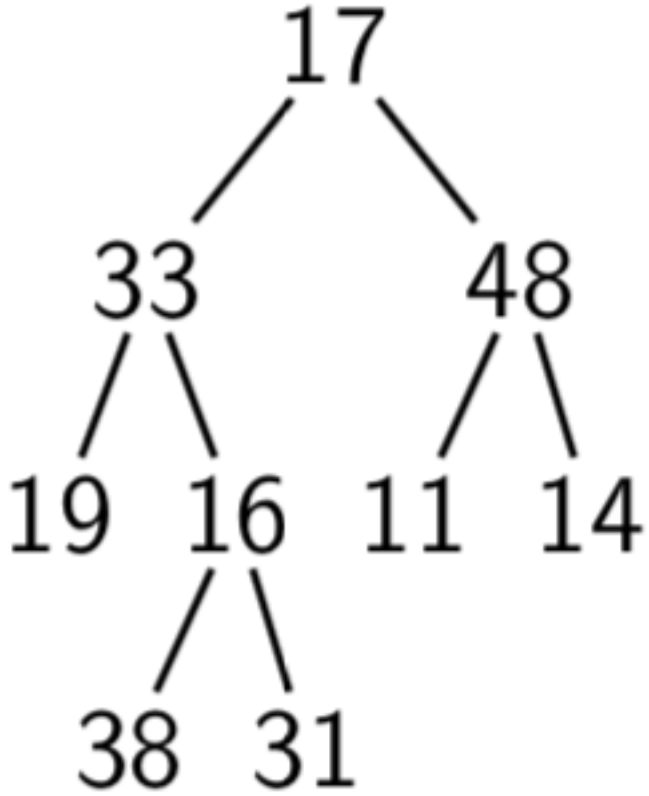
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16 33

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



PostorderTraverse(11)

PostorderTraverse(48)

PostorderTraverse(17)

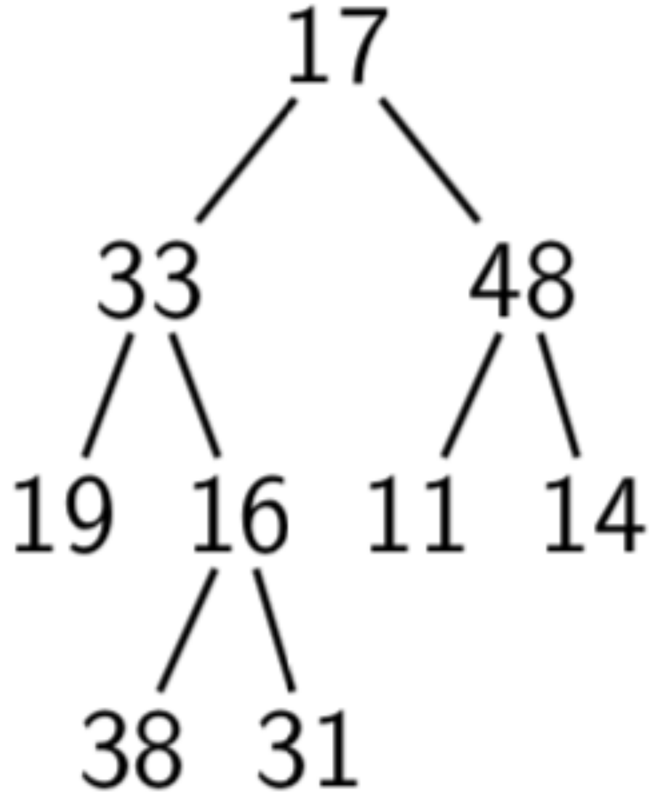
Call Stack

(…skipping the calls to PostorderTraverse(null)…)

Postorder Traversal

Visit order: 19 38 31 16 33 11

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



PostorderTraverse(11)

PostorderTraverse(48)

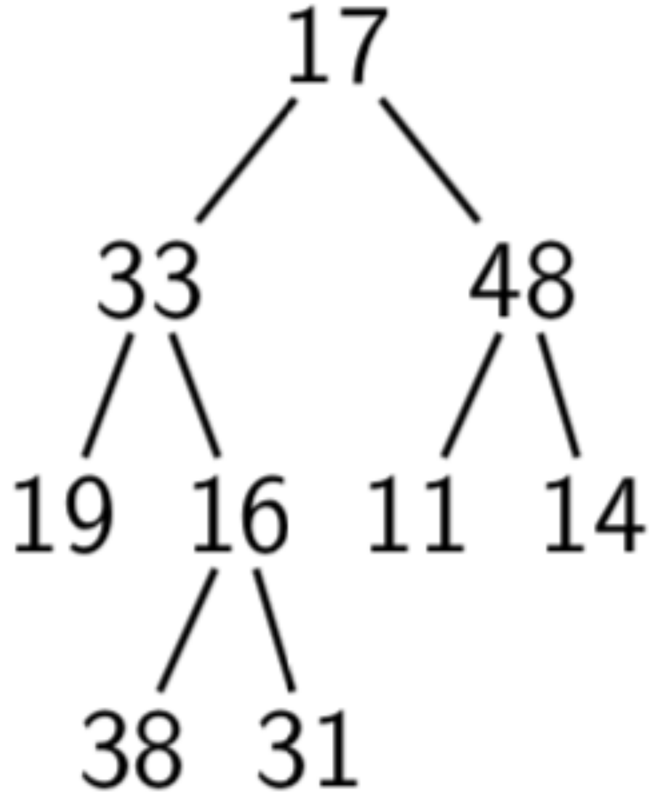
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16 33 11

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



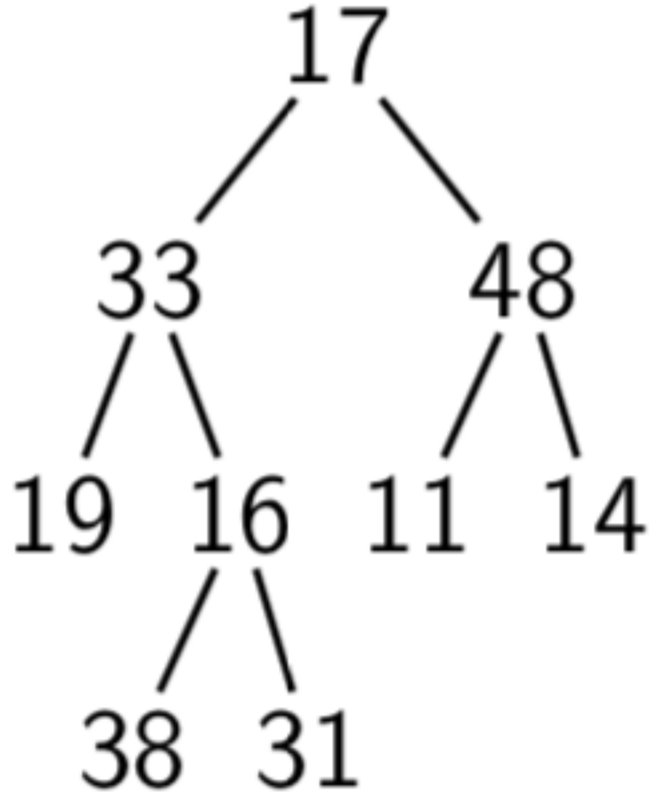
PostorderTraverse(48)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16 33 11

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



PostorderTraverse(14)

PostorderTraverse(48)

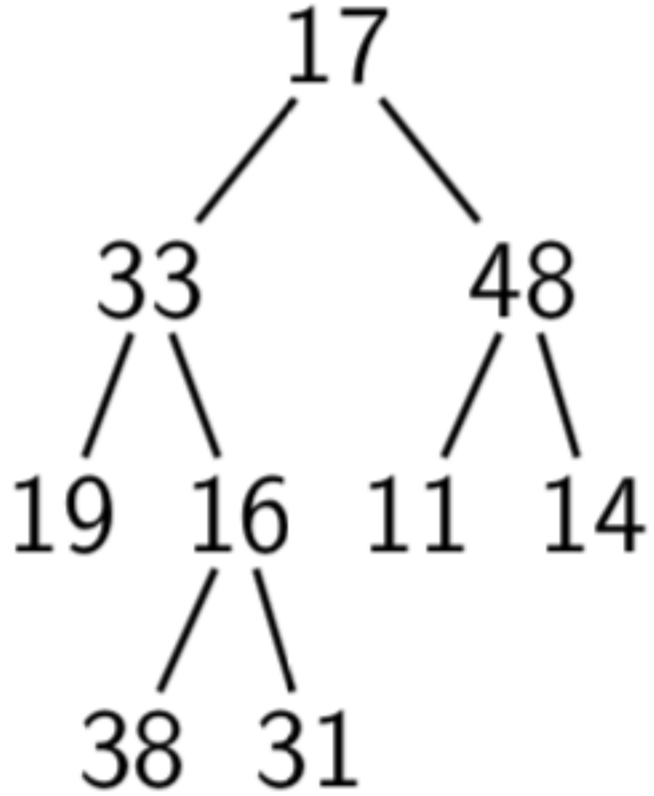
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16 33 11

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



PostorderTraverse(14)

PostorderTraverse(48)

PostorderTraverse(17)

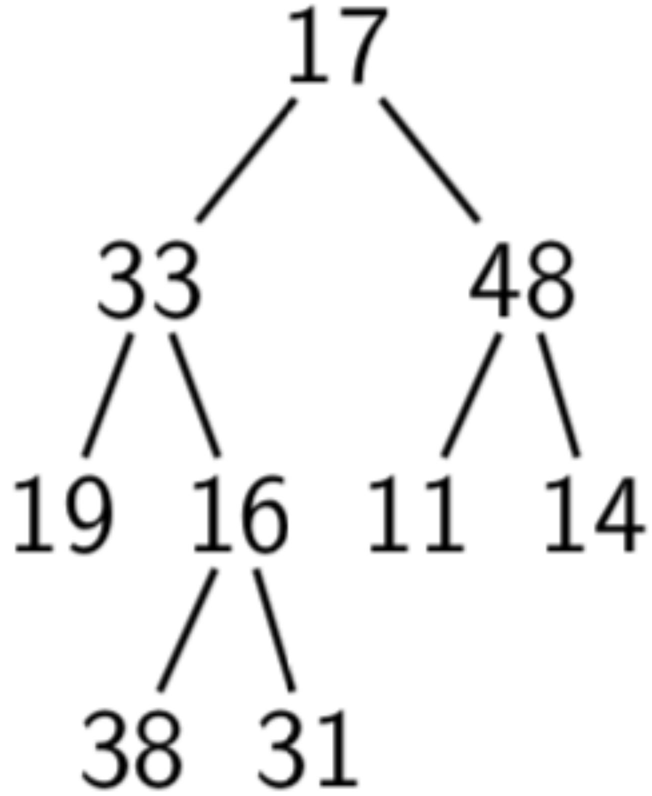
Call Stack

(…skipping the calls to PostorderTraverse(null)…)

Postorder Traversal

Visit order: 19 38 31 16 33 11 14

```
procedure PostorderTraverse(T)
    if T ≠ null then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



PostorderTraverse(14)

PostorderTraverse(48)

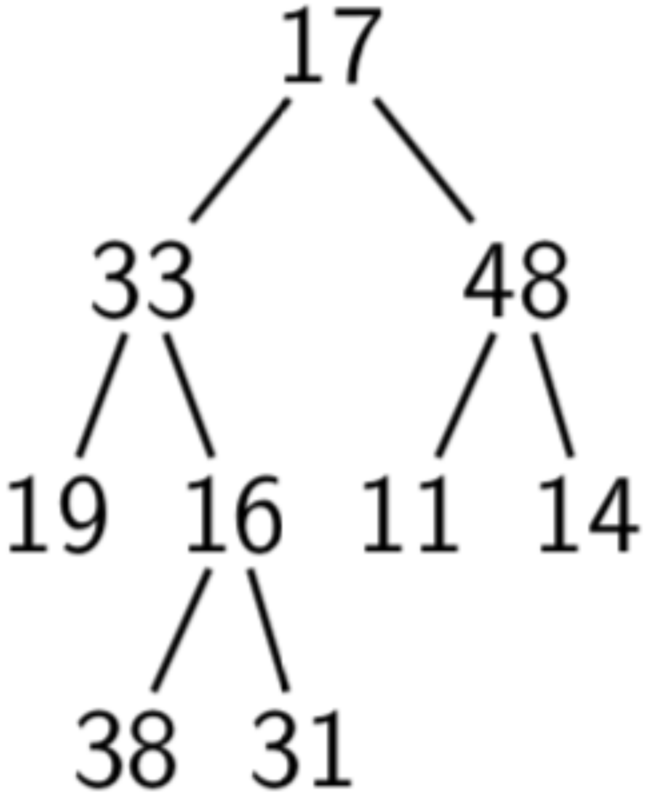
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16 33 11 14

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



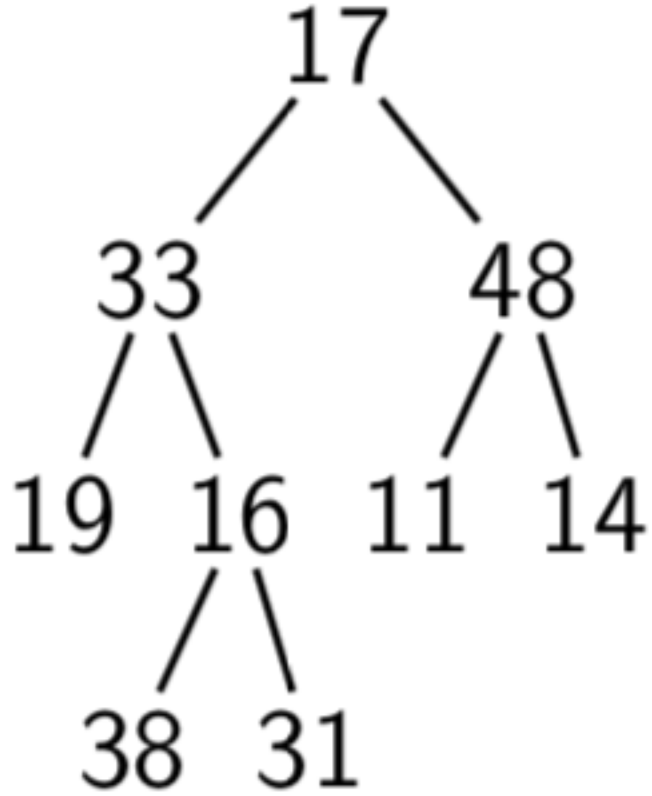
PostorderTraverse(48)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16 33 11 14 48

```
procedure PostorderTraverse($T$)
    if $T \neq null$ then
        PostorderTraverse($T.left$)
        PostorderTraverse($T.right$)
        visit $T.root$
```



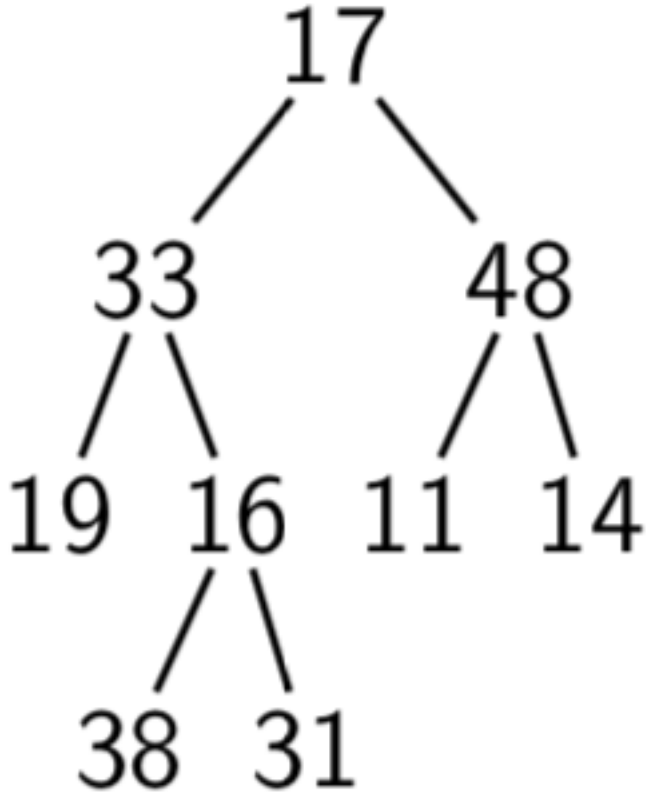
PostorderTraverse(48)
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16 33 11 14 48

```
procedure POSTORDERTRAVERSE($T$)
    if $T \neq null$ then
        POSTORDERTRAVERSE($T.left$)
        POSTORDERTRAVERSE($T.right$)
        visit $T.root$
```



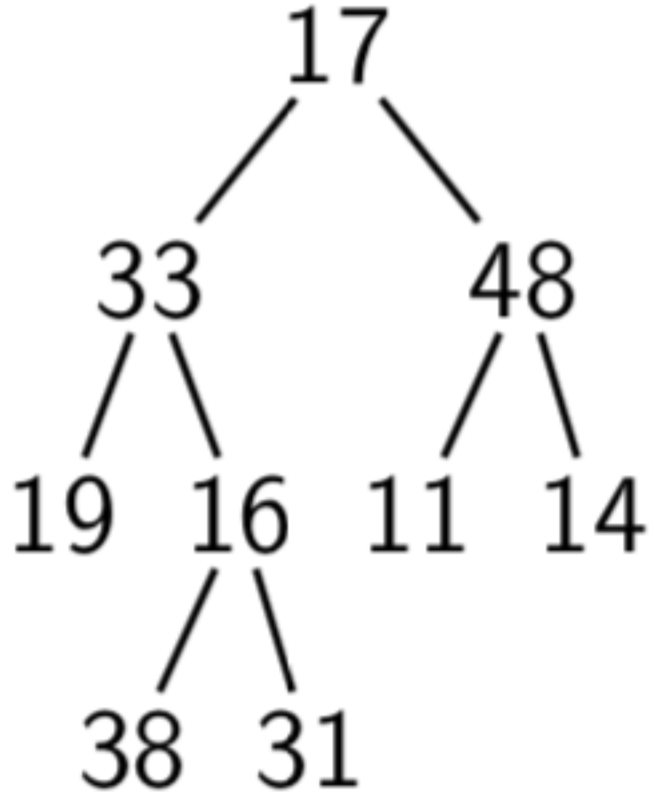
POSTORDERTRAVERSE(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16 33 11 14 48 17

```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



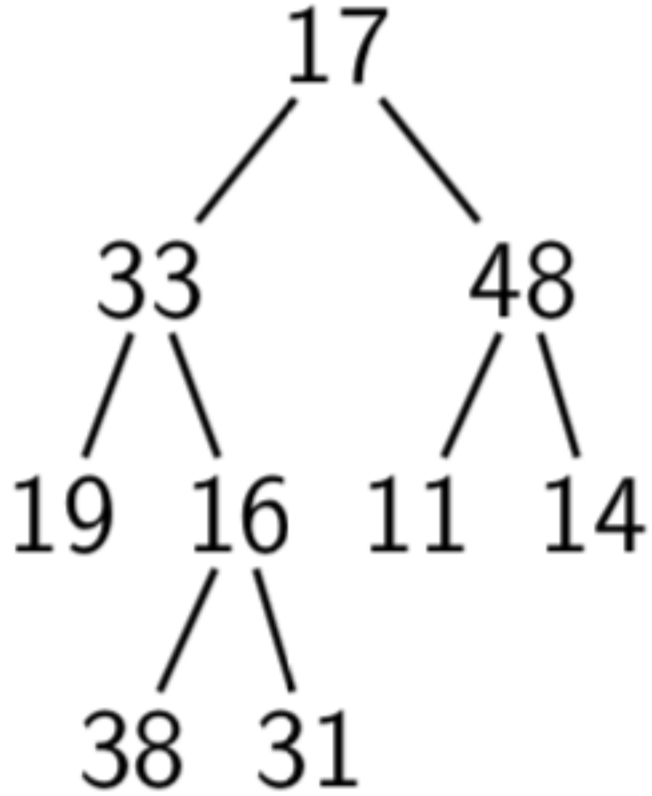
PostorderTraverse(17)

Call Stack

Postorder Traversal

Visit order: 19 38 31 16 33 11 14 48 17

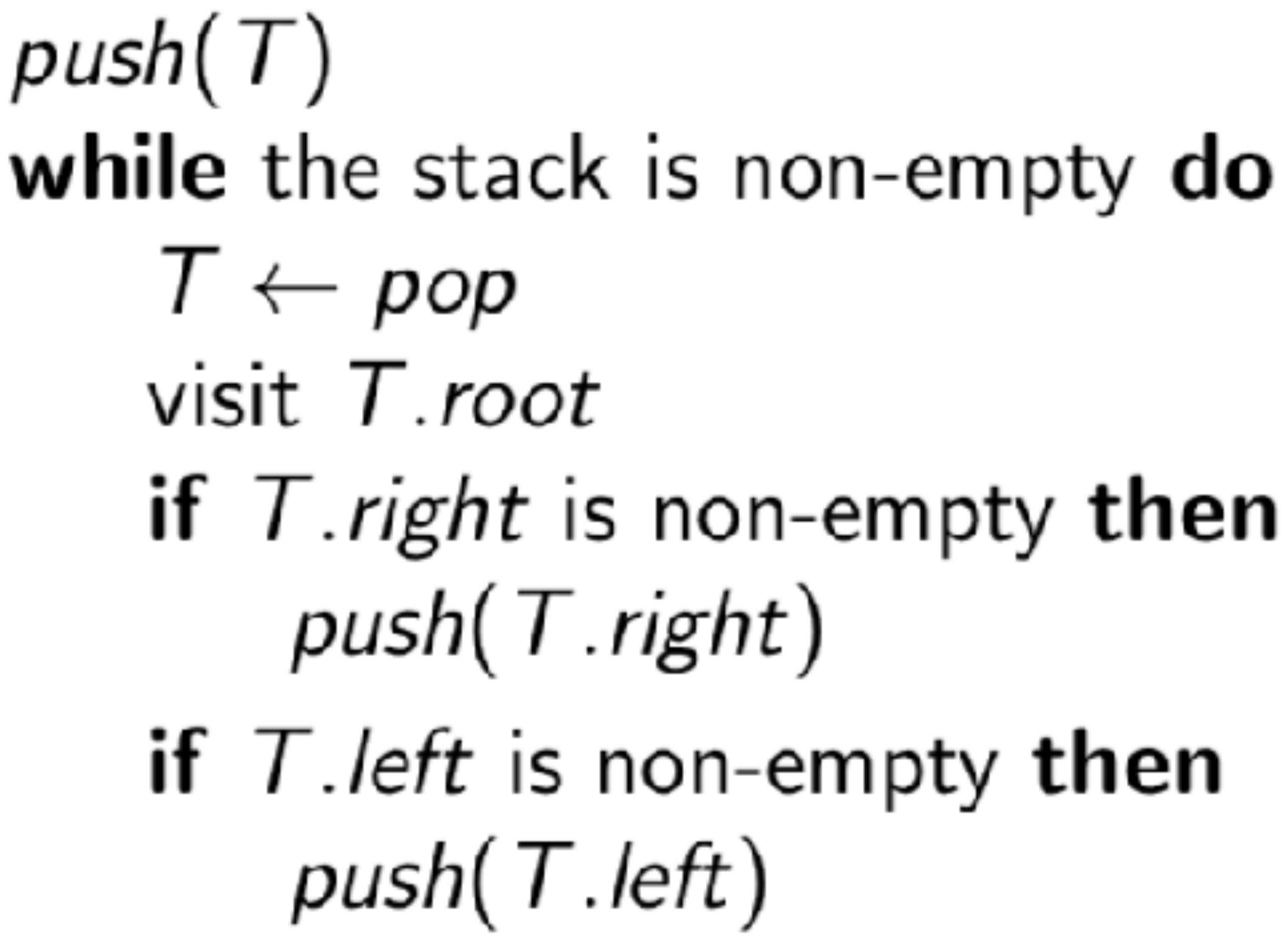
```
procedure PostorderTraverse(T)
    if $T \neq null$ then
        PostorderTraverse(T.left)
        PostorderTraverse(T.right)
        visit T.root
```



Call Stack

Preorder Traversal Using a Stack

- Explicitly maintain a stack of nodes



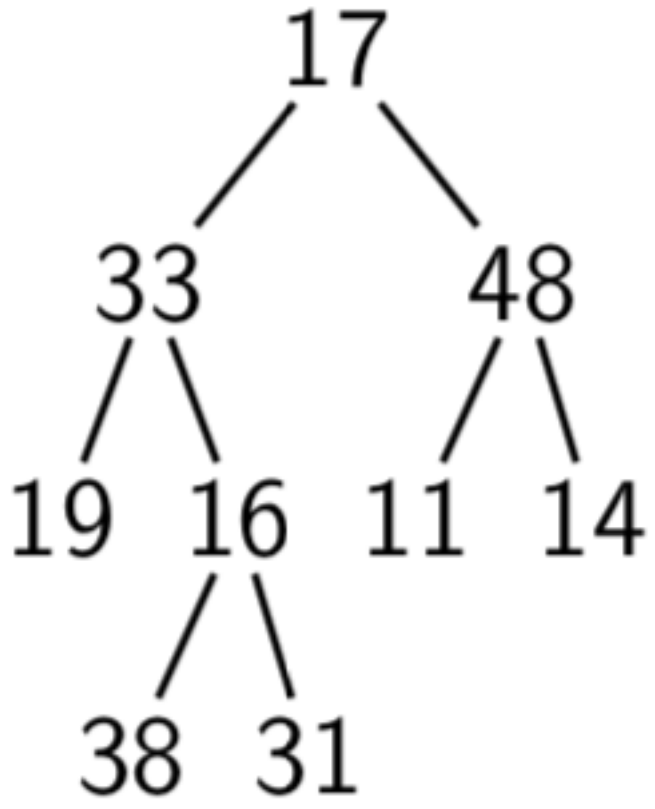
```
$push(T)$
while the stack is non-empty do
    $T \leftarrow pop$
    visit $T.root$
    if $T.right$ is non-empty then
        $push(T.right)$
    if $T.left$ is non-empty then
        $push(T.left)$
```

- In an implementation, the elements placed onto the stack would not be whole trees, but **pointers** to the corresponding internal nodes

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
$inject(T)$
while the queue is non-empty do
    $T \leftarrow eject$
    visit $T.root$
    if $T.left$ is non-empty then
        $inject(T.left)$
    if $T.right$ is non-empty then
        $inject(T.right)$
```



Queue:

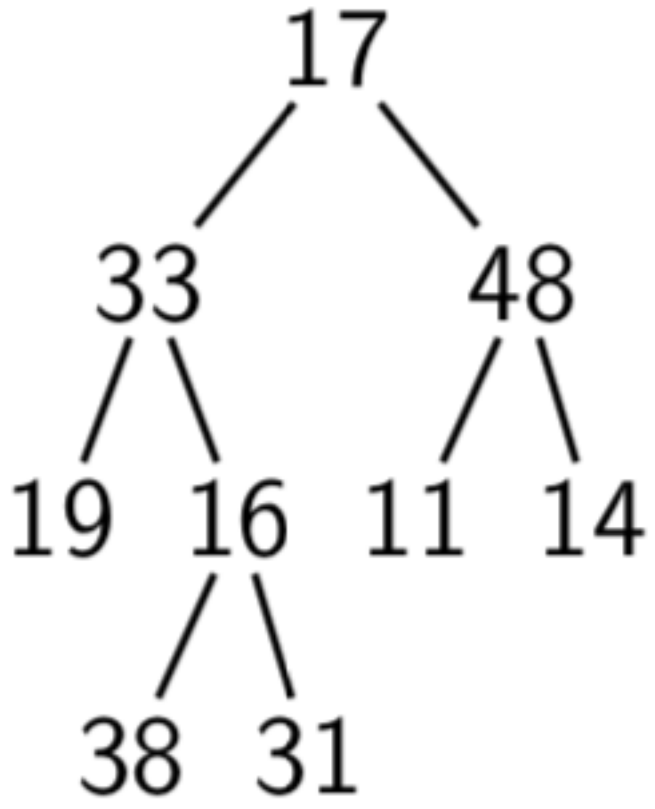
Traversal order:

breadth first

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
$inject(T)$
while the queue is non-empty do
    $T \leftarrow eject$
    visit $T.root$
    if $T.left$ is non-empty then
        $inject(T.left)$
    if $T.right$ is non-empty then
        $inject(T.right)$
```



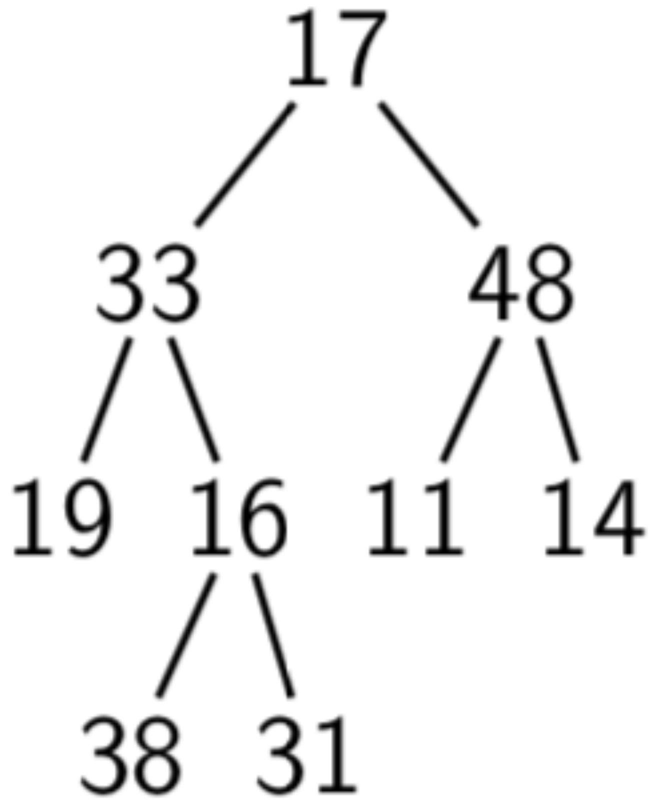
Queue: 17

Traversal order:

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject(T)
while the queue is non-empty do
    T ← eject
    visit T.root
    if T.left is non-empty then
        inject(T.left)
    if T.right is non-empty then
        inject(T.right)
```



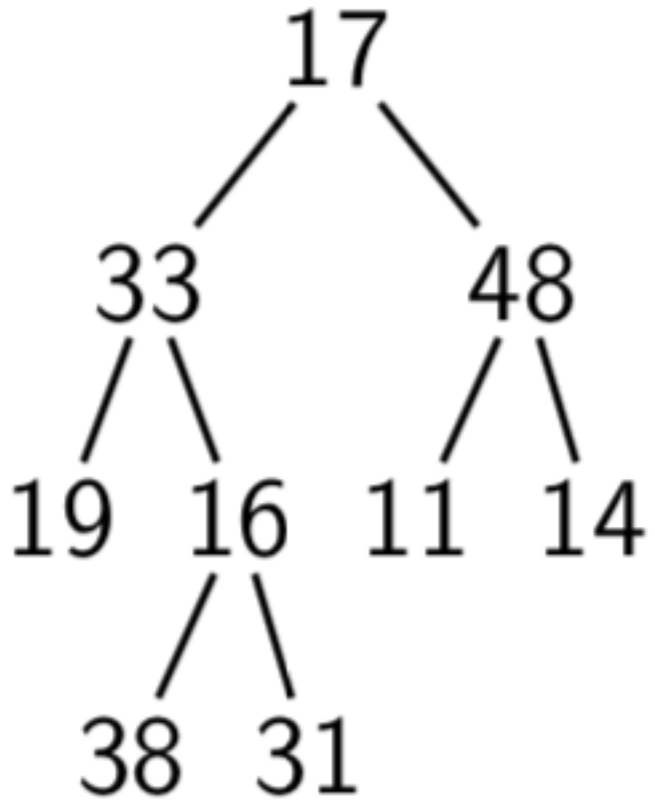
Queue:

Traversal order:

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject($T$)
**while** the queue is non-empty **do**
    $T \leftarrow eject$
    visit $T.root$
    **if** $T.left$ is non-empty **then**
        inject($T.left$)
    **if** $T.right$ is non-empty **then**
        inject($T.right$)
```



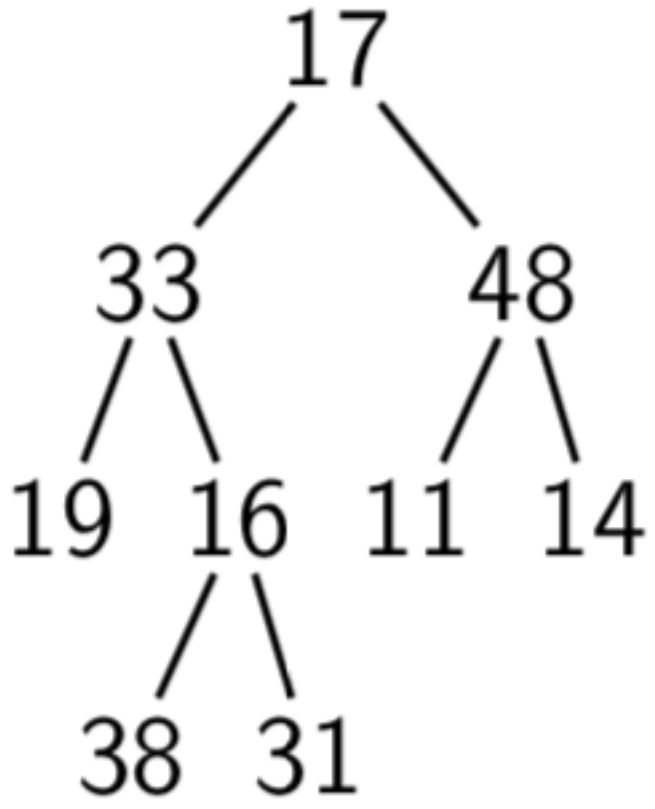
Queue:

Traversal order: 17

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject($T$)
while the queue is non-empty do
    $T \leftarrow$ eject
    visit $T$.root
    if $T$.left is non-empty then
        inject($T$.left)
    if $T$.right is non-empty then
        inject($T$.right)
```



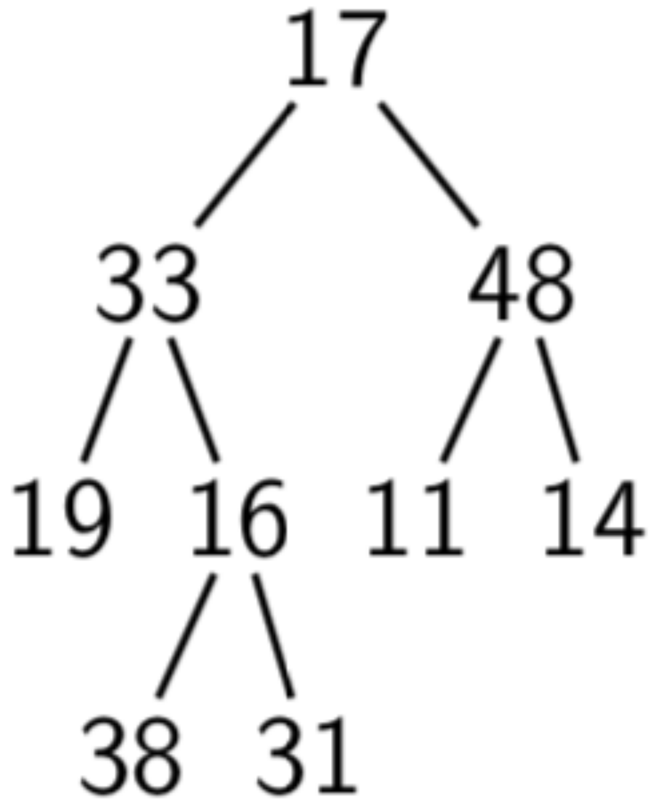
Queue: 33

Traversal order: 17

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject(T)
while the queue is non-empty do
    T ← eject
    visit T.root
    if T.left is non-empty then
        inject(T.left)
    if T.right is non-empty then
        inject(T.right)
```



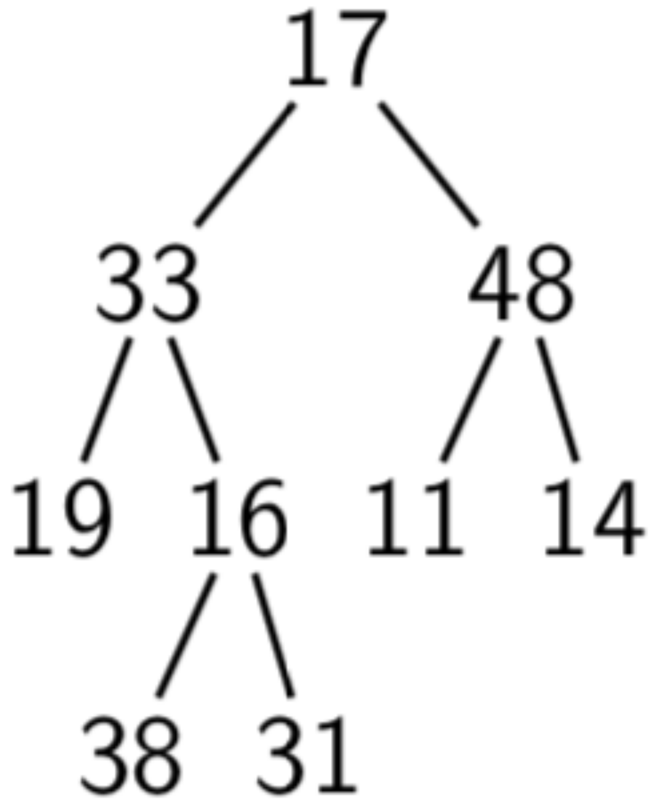
Queue: 33 48

Traversal order: 17

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
$inject(T)$
while the queue is non-empty do
    $T \leftarrow eject$
    visit $T.root$
    if $T.left$ is non-empty then
        $inject(T.left)$
    if $T.right$ is non-empty then
        $inject(T.right)$
```



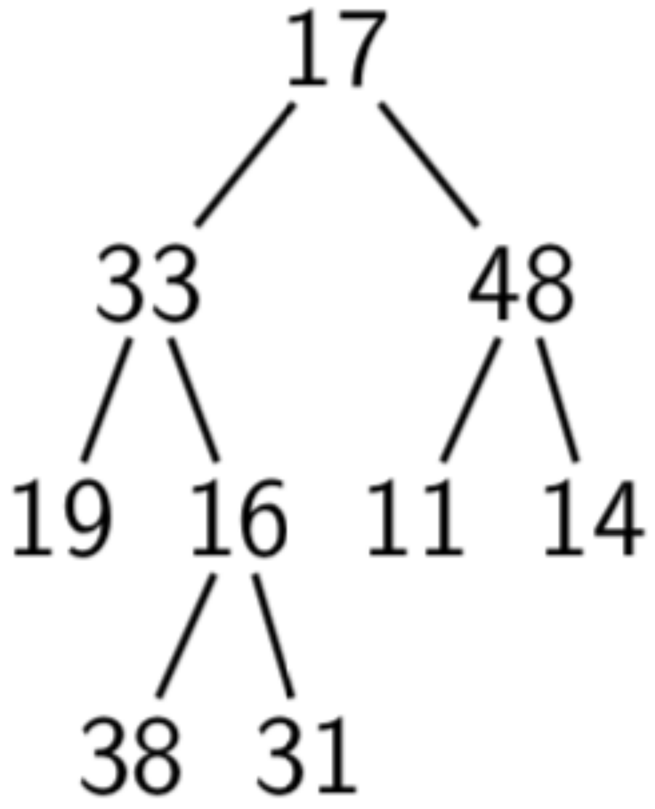
Queue: 48

Traversal order: 17

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject($T$)
while the queue is non-empty do
    $T \leftarrow$ eject
    visit $T.root$
    if $T.left$ is non-empty then
        inject($T.left$)
    if $T.right$ is non-empty then
        inject($T.right$)
```



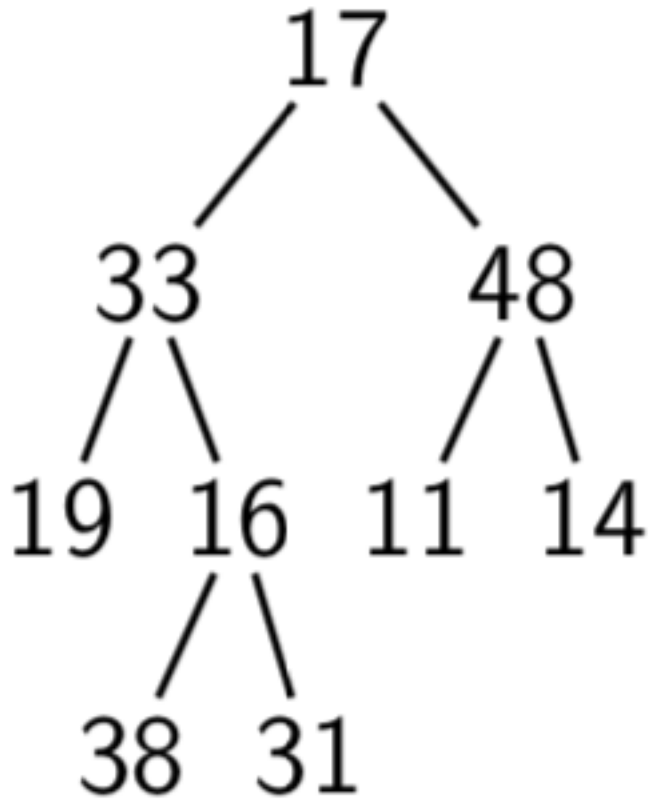
Queue: 48

Traversal order: 17 33

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject(T)
while the queue is non-empty do
    T ← eject
    visit T.root
    if T.left is non-empty then
        inject(T.left)
    if T.right is non-empty then
        inject(T.right)
```



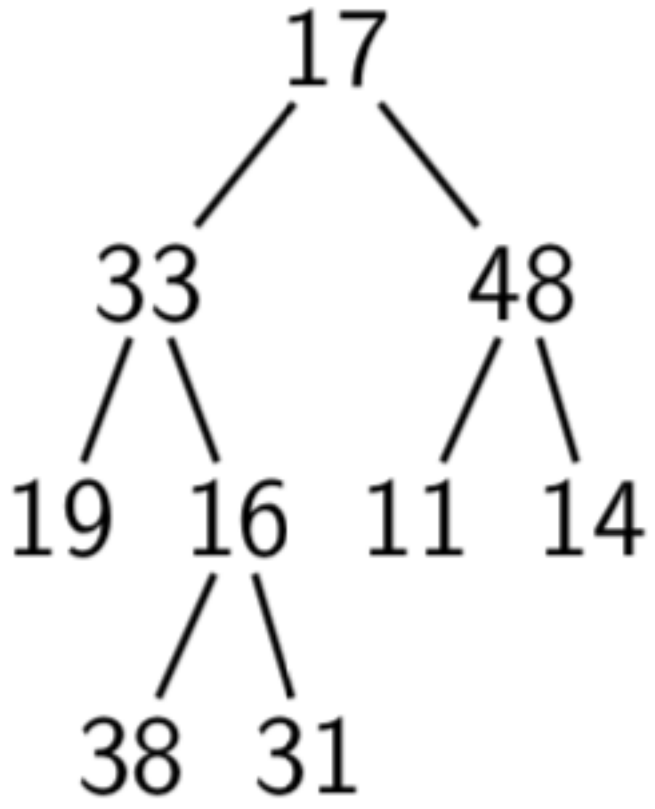
Queue: 48 19

Traversal order: 17 33

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject($T$)
while the queue is non-empty do
    $T \leftarrow eject$
    visit $T.root$
    if $T.left$ is non-empty then
        inject($T.left$)
    if $T.right$ is non-empty then
        inject($T.right$)
```



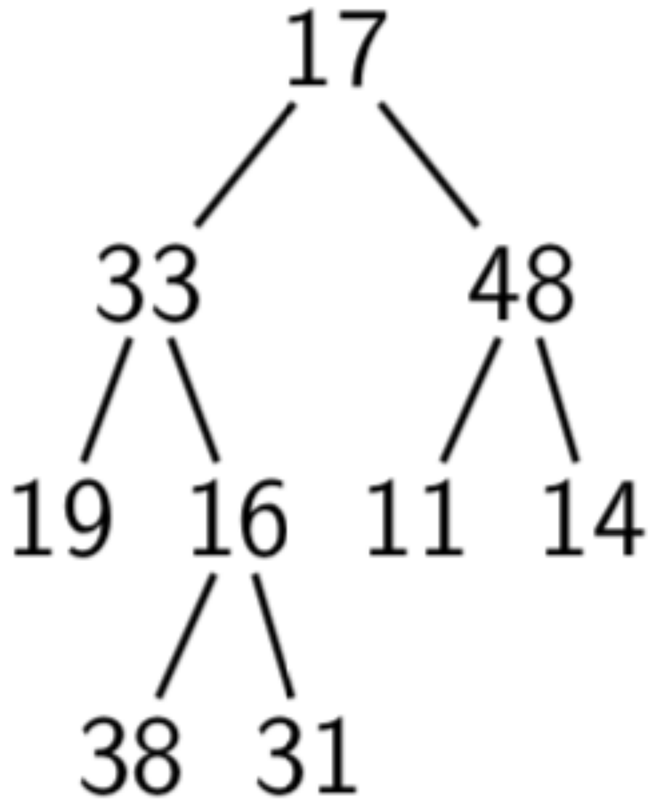
Queue: 48 19 16

Traversal order: 17 33

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
$inject(T)$
while the queue is non-empty do
    $T \leftarrow eject$
    visit $T.root$
    if $T.left$ is non-empty then
        $inject(T.left)$
    if $T.right$ is non-empty then
        $inject(T.right)$
```



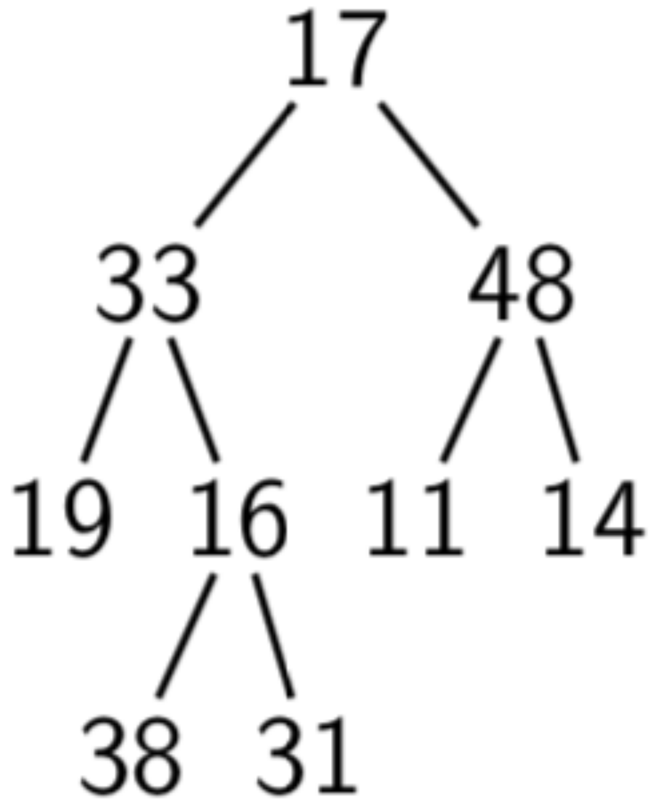
Queue: 19 16

Traversal order: 17 33

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject(T)
while the queue is non-empty do
    T ← eject
    visit T.root
    if T.left is non-empty then
        inject(T.left)
    if T.right is non-empty then
        inject(T.right)
```



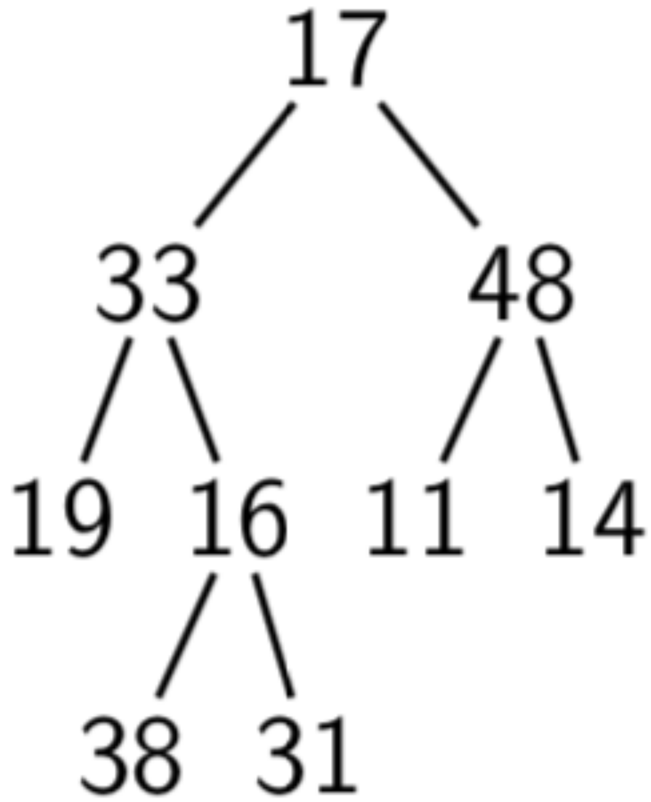
Queue: 19 16

Traversal order: 17 33 48

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject(T)
while the queue is non-empty do
    $T \leftarrow eject$
    visit T.root
    if T.left is non-empty then
        inject(T.left)
    if T.right is non-empty then
        inject(T.right)
```



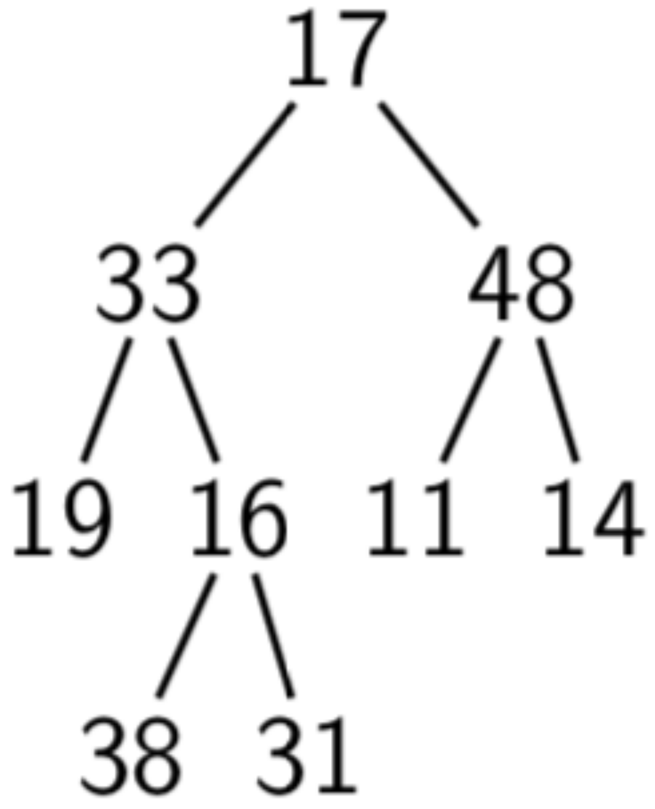
Queue: 19 16 11

Traversal order: 17 33 48

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject($T$)
while the queue is non-empty do
    $T \leftarrow$ eject
    visit $T.root$
    if $T.left$ is non-empty then
        inject($T.left$)
    if $T.right$ is non-empty then
        inject($T.right$)
```



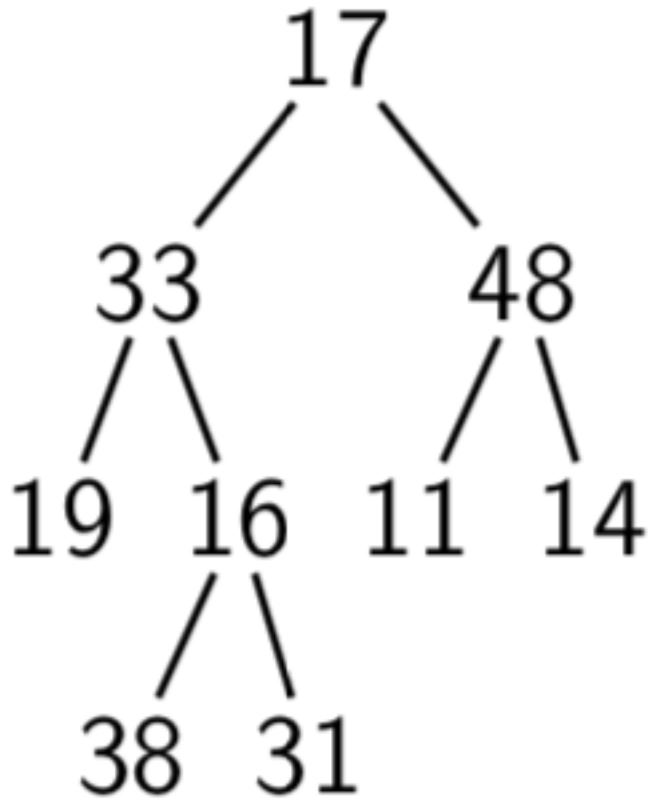
Queue: 19 16 11 14

Traversal order: 17 33 48

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
$inject(T)$
while the queue is non-empty do
    $T \leftarrow eject$
    visit $T.root$
    if $T.left$ is non-empty then
        $inject(T.left)$
    if $T.right$ is non-empty then
        $inject(T.right)$
```



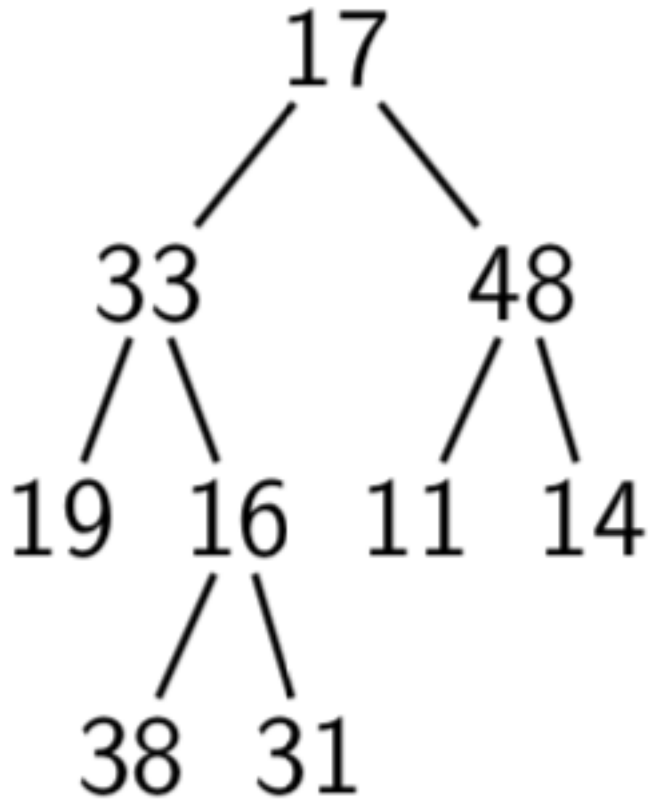
Queue: 16 11 14

Traversal order: 17 33 48

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
$inject(T)$
while the queue is non-empty do
    $T \leftarrow eject$
    visit $T.root$
    if $T.left$ is non-empty then
        $inject(T.left)$
    if $T.right$ is non-empty then
        $inject(T.right)$
```



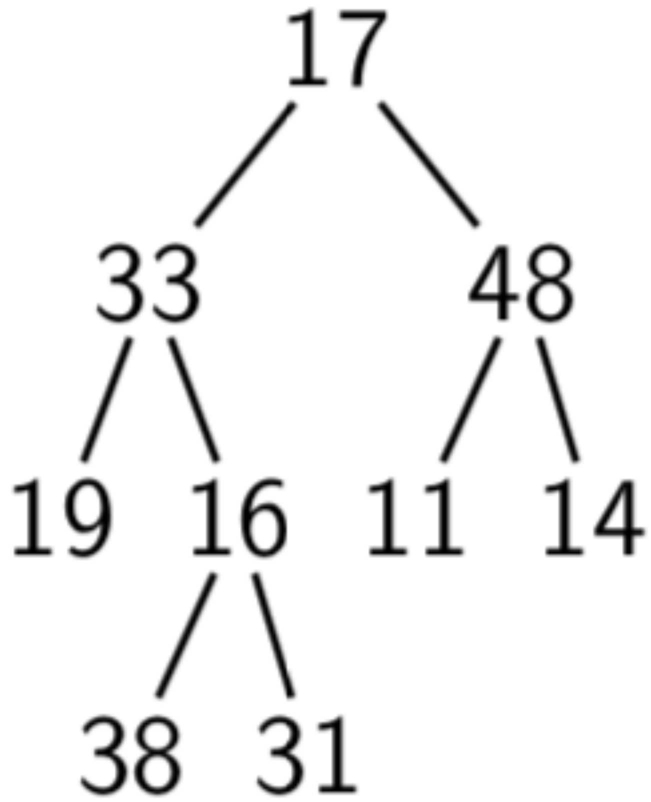
Queue: 16 11 14

Traversal order: 17 33 48 19

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
$inject(T)$
while the queue is non-empty do
    $T \leftarrow eject$
    visit $T.root$
    if $T.left$ is non-empty then
        $inject(T.left)$
    if $T.right$ is non-empty then
        $inject(T.right)$
```



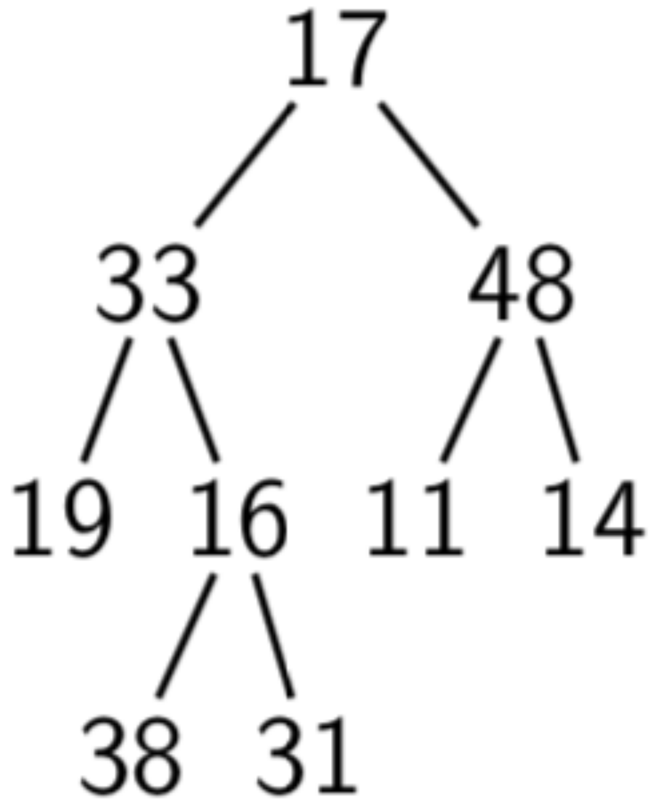
Queue: 11 14

Traversal order: 17 33 48 19

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
$inject(T)$
while the queue is non-empty do
    $T \leftarrow eject$
    visit $T.root$
    if $T.left$ is non-empty then
        $inject(T.left)$
    if $T.right$ is non-empty then
        $inject(T.right)$
```



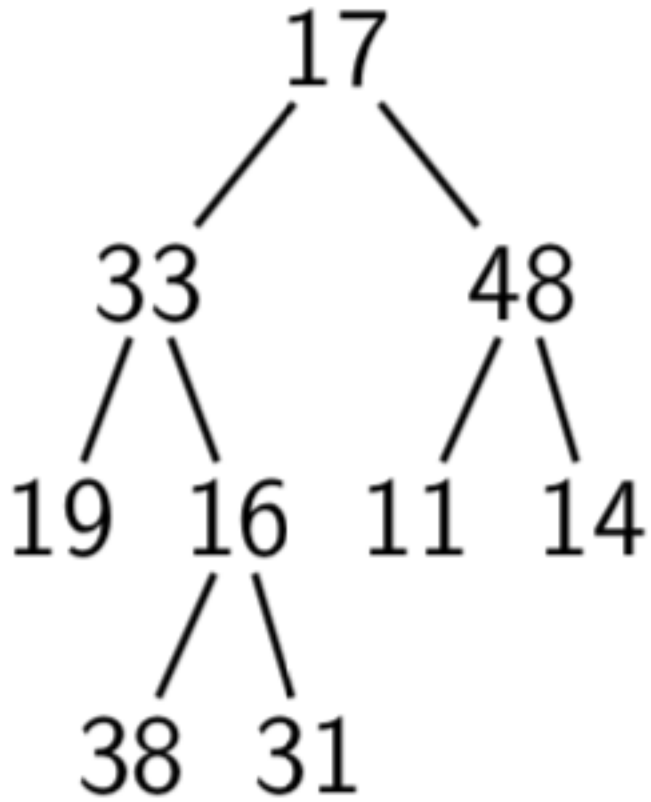
Queue: 11 14

Traversal order: 17 33 48 19 16

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject(T)
while the queue is non-empty do
    $T \leftarrow eject$
    visit $T.root$
    if $T.left$ is non-empty then
        inject($T.left$)
    if $T.right$ is non-empty then
        inject($T.right$)
```



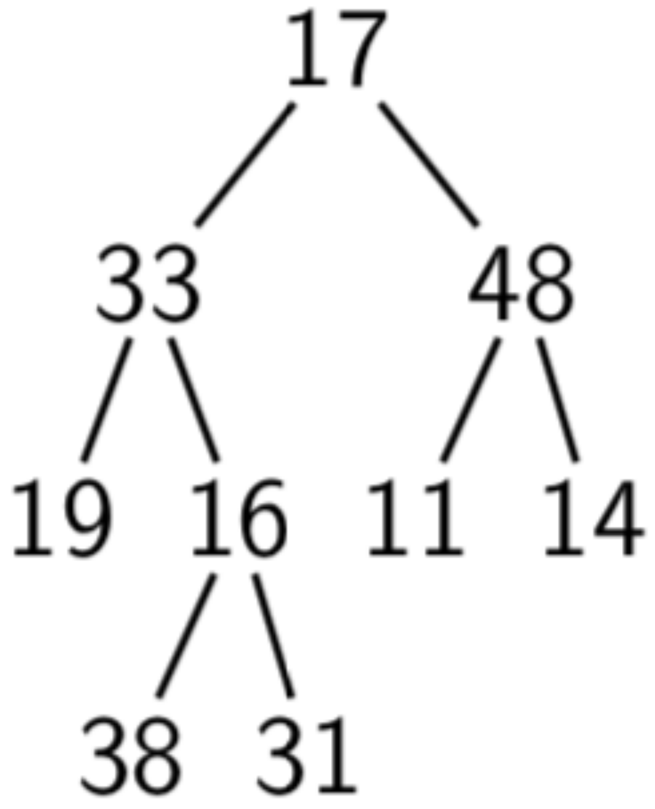
Queue: 11 14 38

Traversal order: 17 33 48 19 16

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject($T$)
while the queue is non-empty do
    $T \leftarrow$ eject
    visit $T.root$
    if $T.left$ is non-empty then
        inject($T.left$)
    if $T.right$ is non-empty then
        inject($T.right$)
```



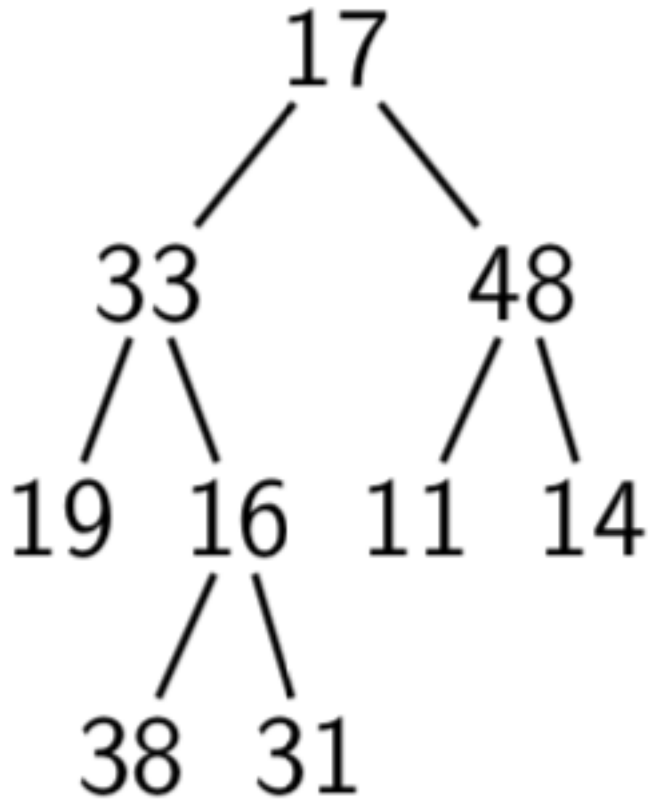
Queue: 11 14 38 31

Traversal order: 17 33 48 19 16

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject(T)
while the queue is non-empty do
    T ← eject
    visit T.root
    if T.left is non-empty then
        inject(T.left)
    if T.right is non-empty then
        inject(T.right)
```



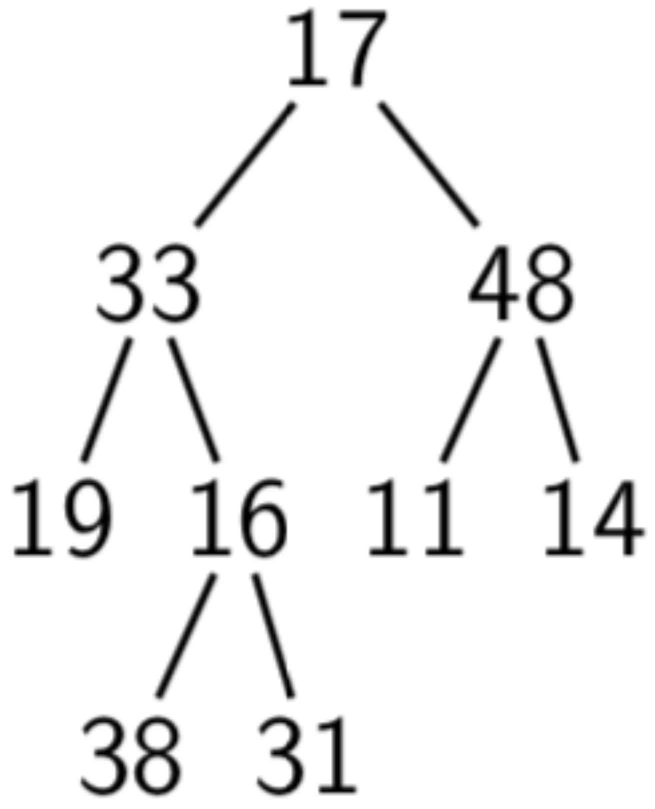
Queue: 14 38 31

Traversal order: 17 33 48 19 16

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject(T)
while the queue is non-empty do
    T ← eject
    visit T.root
    if T.left is non-empty then
        inject(T.left)
    if T.right is non-empty then
        inject(T.right)
```



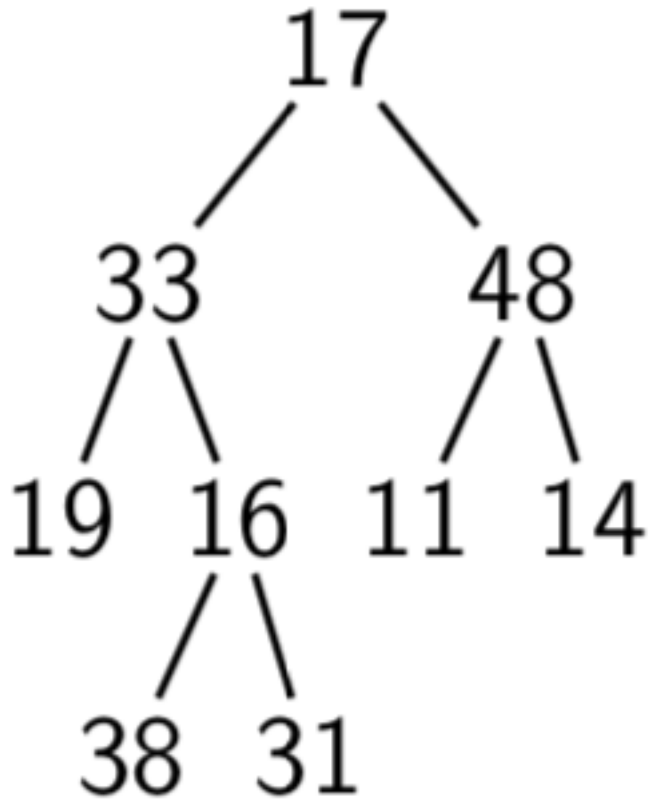
Queue: 14 38 31

Traversal order: 17 33 48 19 16 11

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject(T)
while the queue is non-empty do
    T ← eject
    visit T.root
    if T.left is non-empty then
        inject(T.left)
    if T.right is non-empty then
        inject(T.right)
```



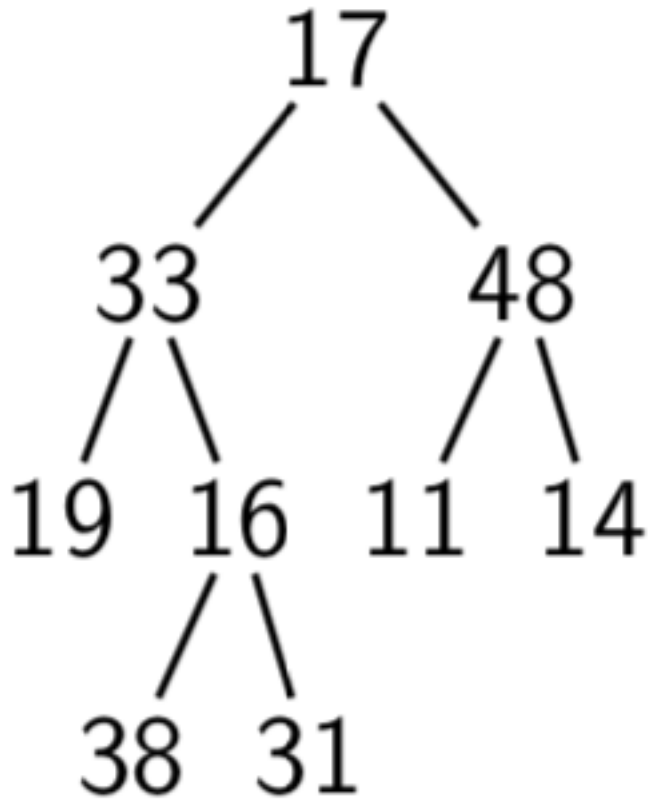
Queue: 38 31

Traversal order: 17 33 48 19 16 11

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject($T$)
while the queue is non-empty do
    $T \leftarrow eject$
    visit $T.root$
    if $T.left$ is non-empty then
        inject($T.left$)
    if $T.right$ is non-empty then
        inject($T.right$)
```

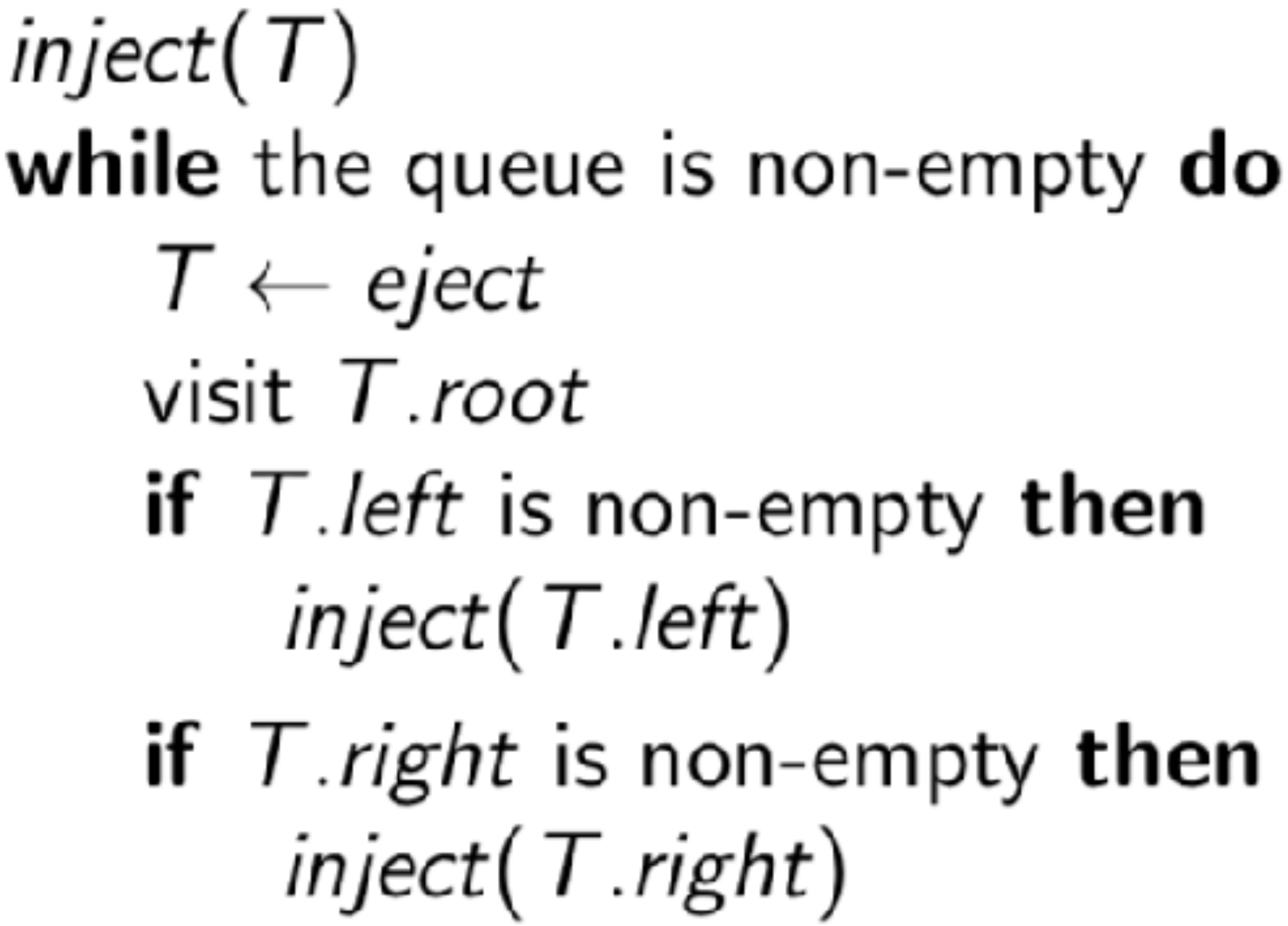


Queue: 38 31

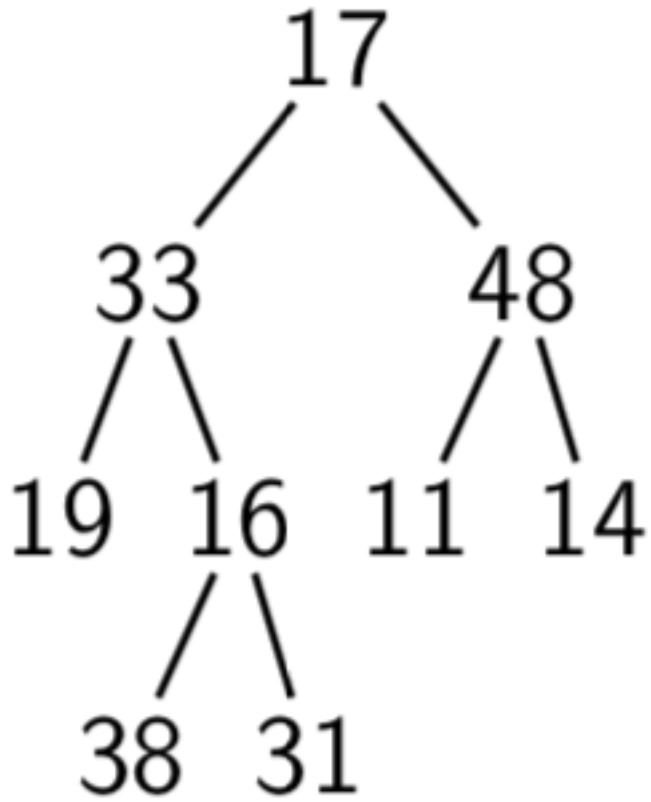
Traversal order: 17 33 48 19 16 11 14

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**



```
inject(T)
while the queue is non-empty do
    T ← eject
    visit T.root
    if T.left is non-empty then
        inject(T.left)
    if T.right is non-empty then
        inject(T.right)
```



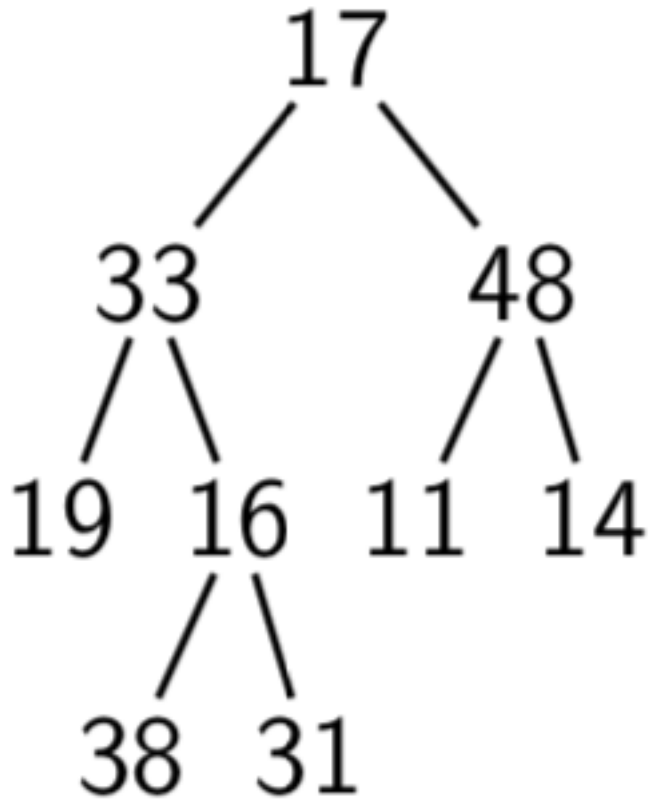
Queue: 31

Traversal order: 17 33 48 19 16 11 14

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject(T)
while the queue is non-empty do
    T ← eject
    visit T.root
    if T.left is non-empty then
        inject(T.left)
    if T.right is non-empty then
        inject(T.right)
```



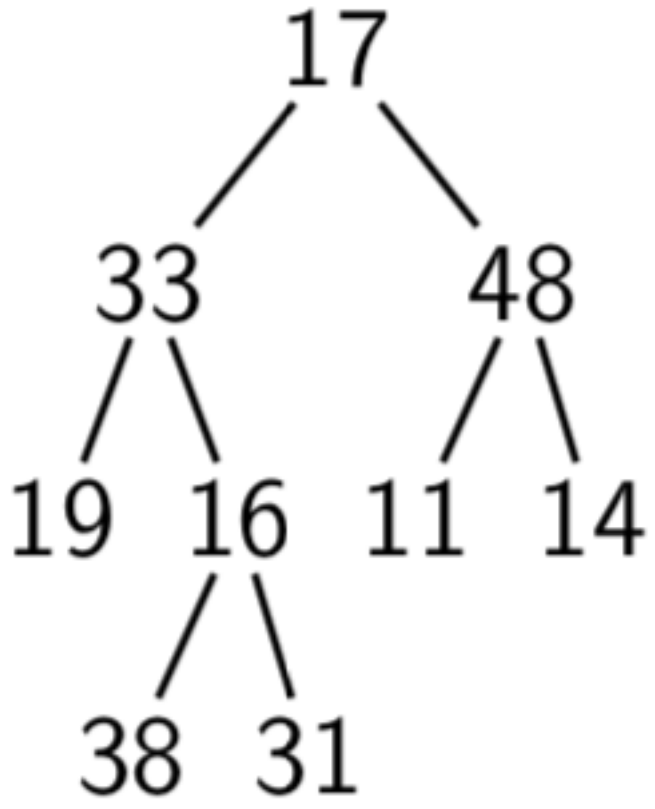
Queue: 31

Traversal order: 17 33 48 19 16 11 14 38

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

```
inject(T)
while the queue is non-empty do
    T ← eject
    visit T.root
    if T.left is non-empty then
        inject(T.left)
    if T.right is non-empty then
        inject(T.right)
```



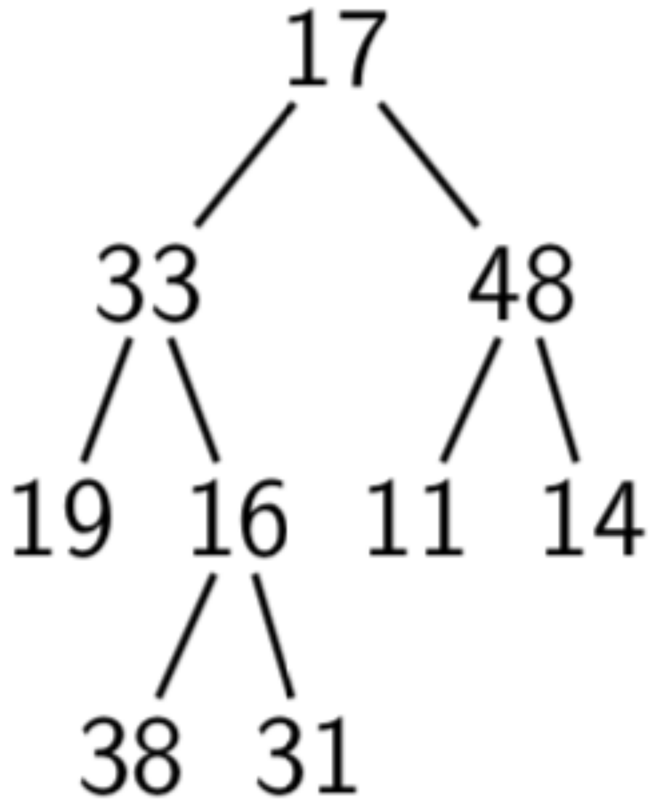
Queue:

Traversal order: 17 33 48 19 16 11 14 38

Level-Order Traversal Using a Queue

- Replace the stack with a **queue**

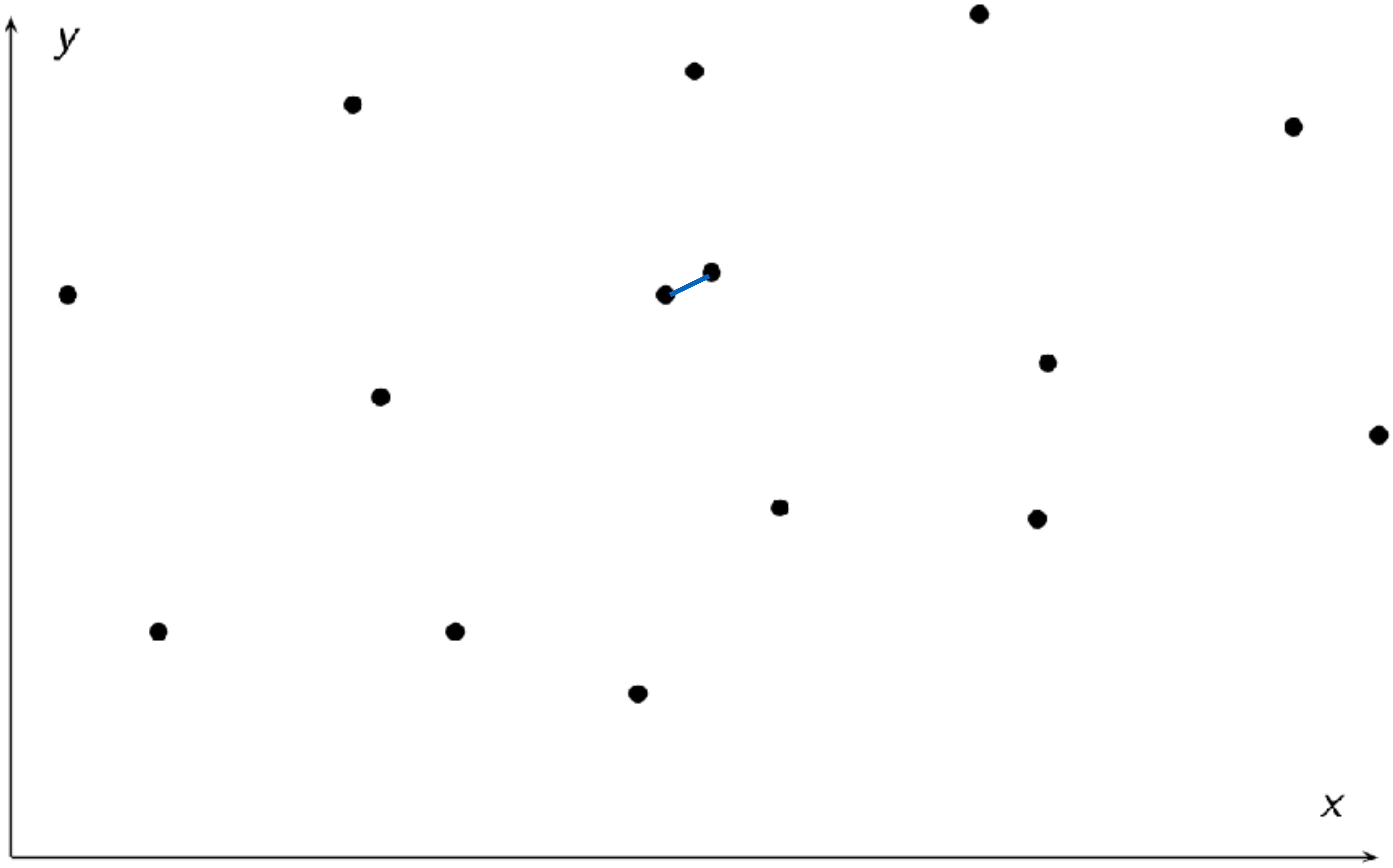
```
$inject(T)$
**while** the queue is non-empty **do**
    $T \leftarrow eject$
    visit $T.root$
    **if** $T.left$ is non-empty **then**
        $inject(T.left)$
    **if** $T.right$ is non-empty **then**
        $inject(T.right)$
```



Queue:

Traversal order: 17 33 48 19 16 11 14 38 31

Closest Pair Problem (2D) Revisited (see Lecture 5)



Closest Pair Problem Revisited

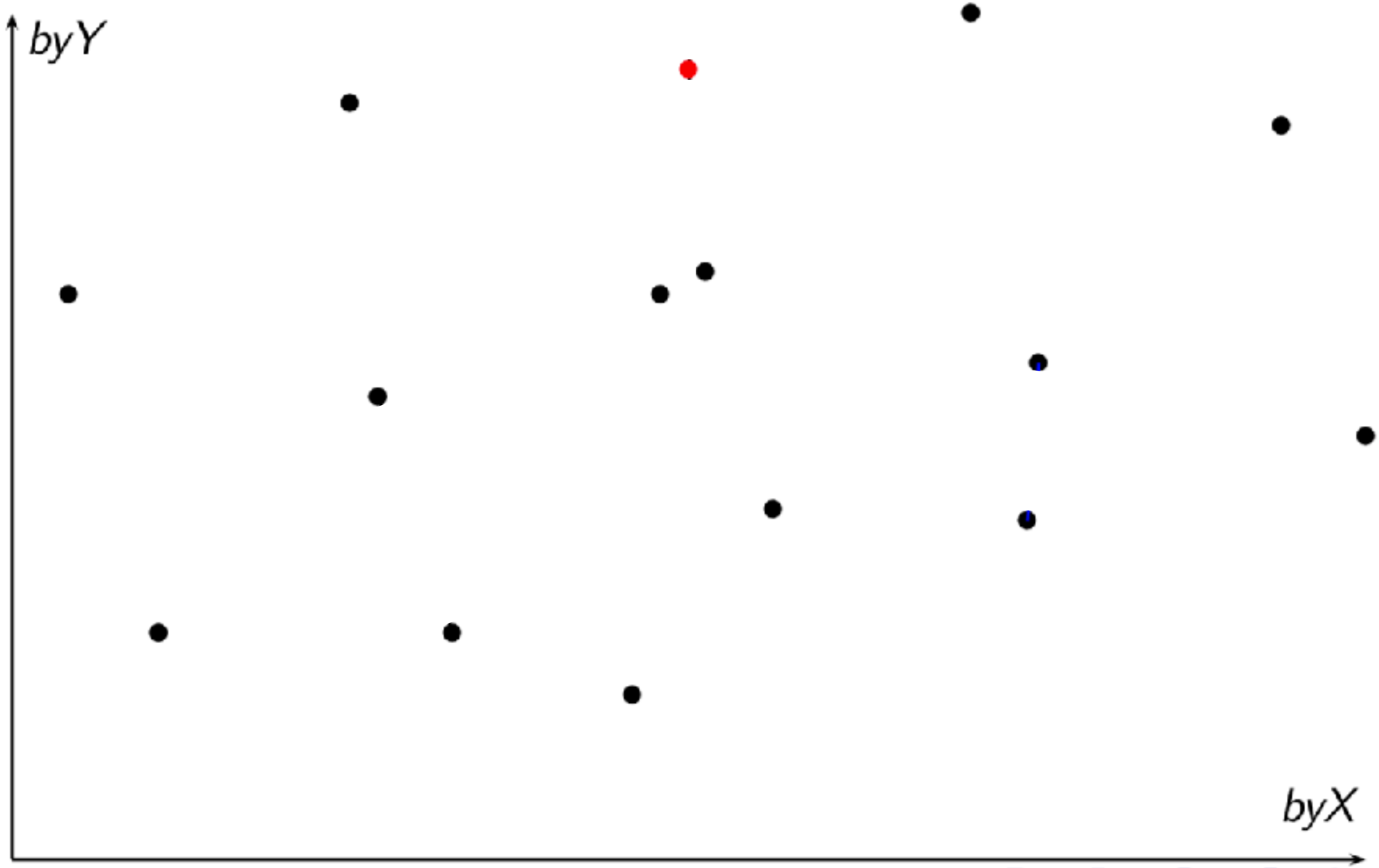
- In Lecture 5 we gave a brute-force algorithm for the closest pair problem: Given n points in the Cartesian plane, find a pair with minimal distance.

- The brute-force method had complexity $\Theta(n^2)$. We can use divide-and-conquer to do better, namely $\Theta(n \log n)$.

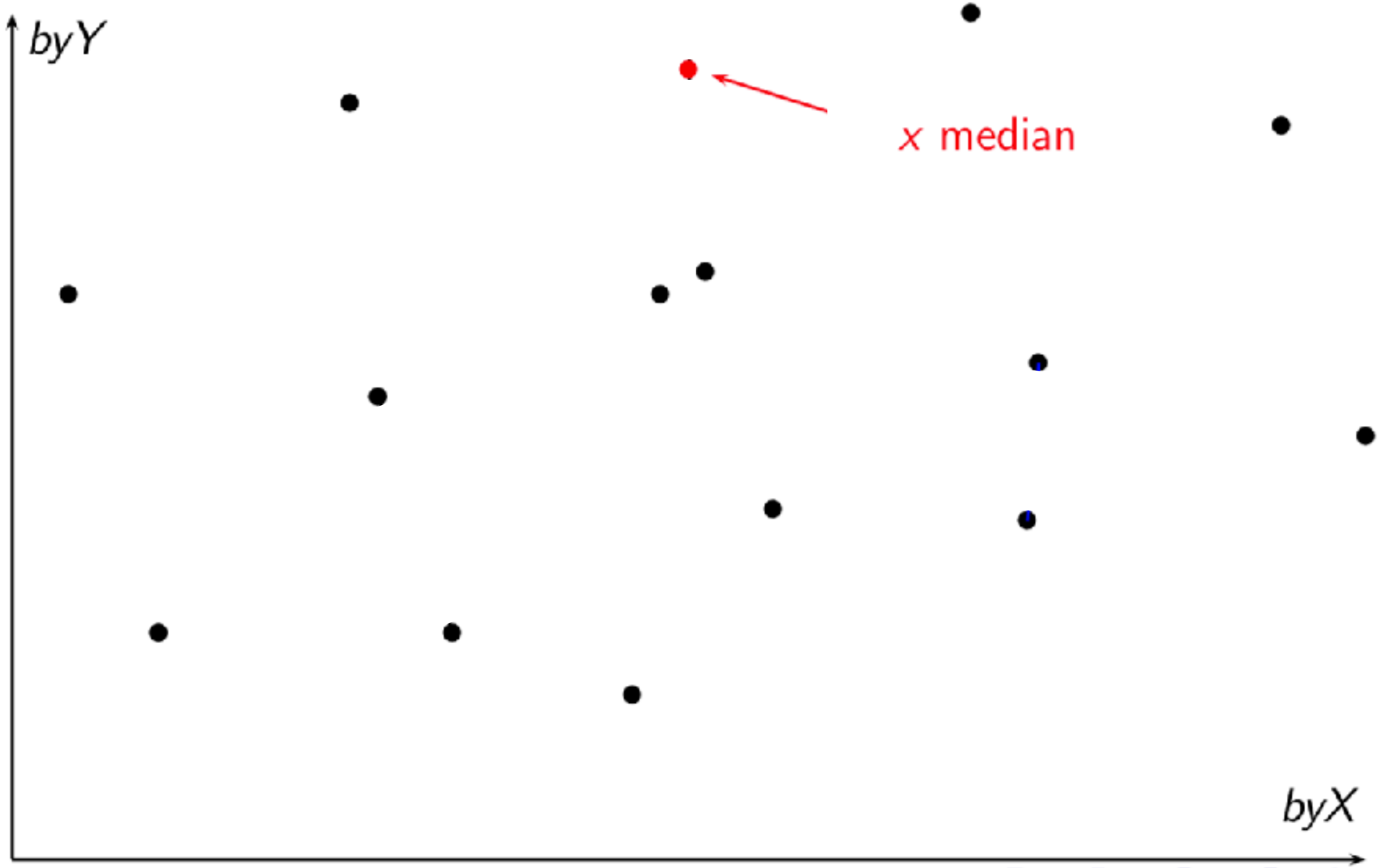
- First, sort the points by x value and store the result in array **byX**. Also sort the points by y value and store the result in array **byY**.

- Now we can identify the x median, and recursively process the set P_L of points with lower x values, as well as the set P_R with higher x values.

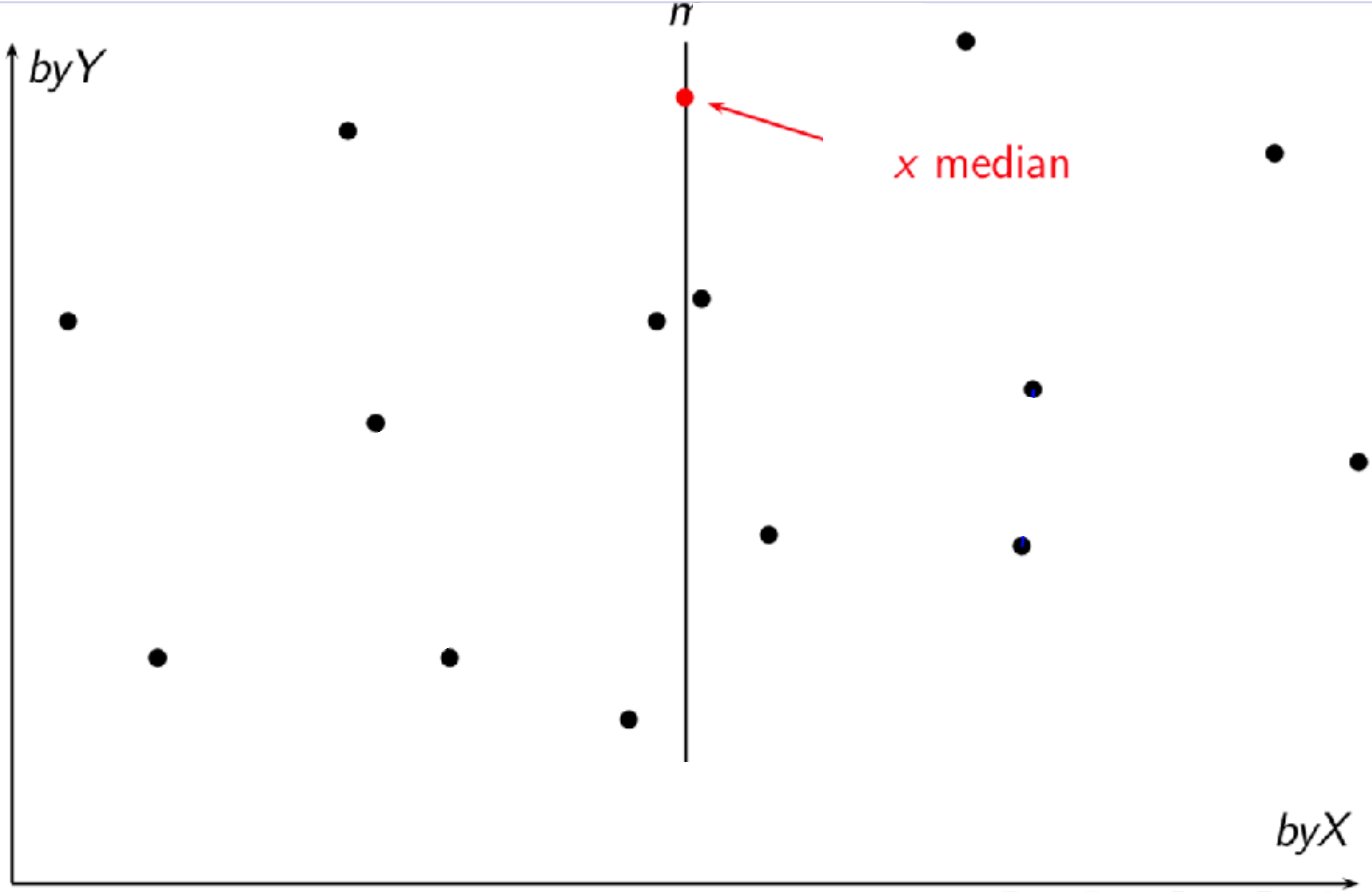
Closest Pair Problem Revisited



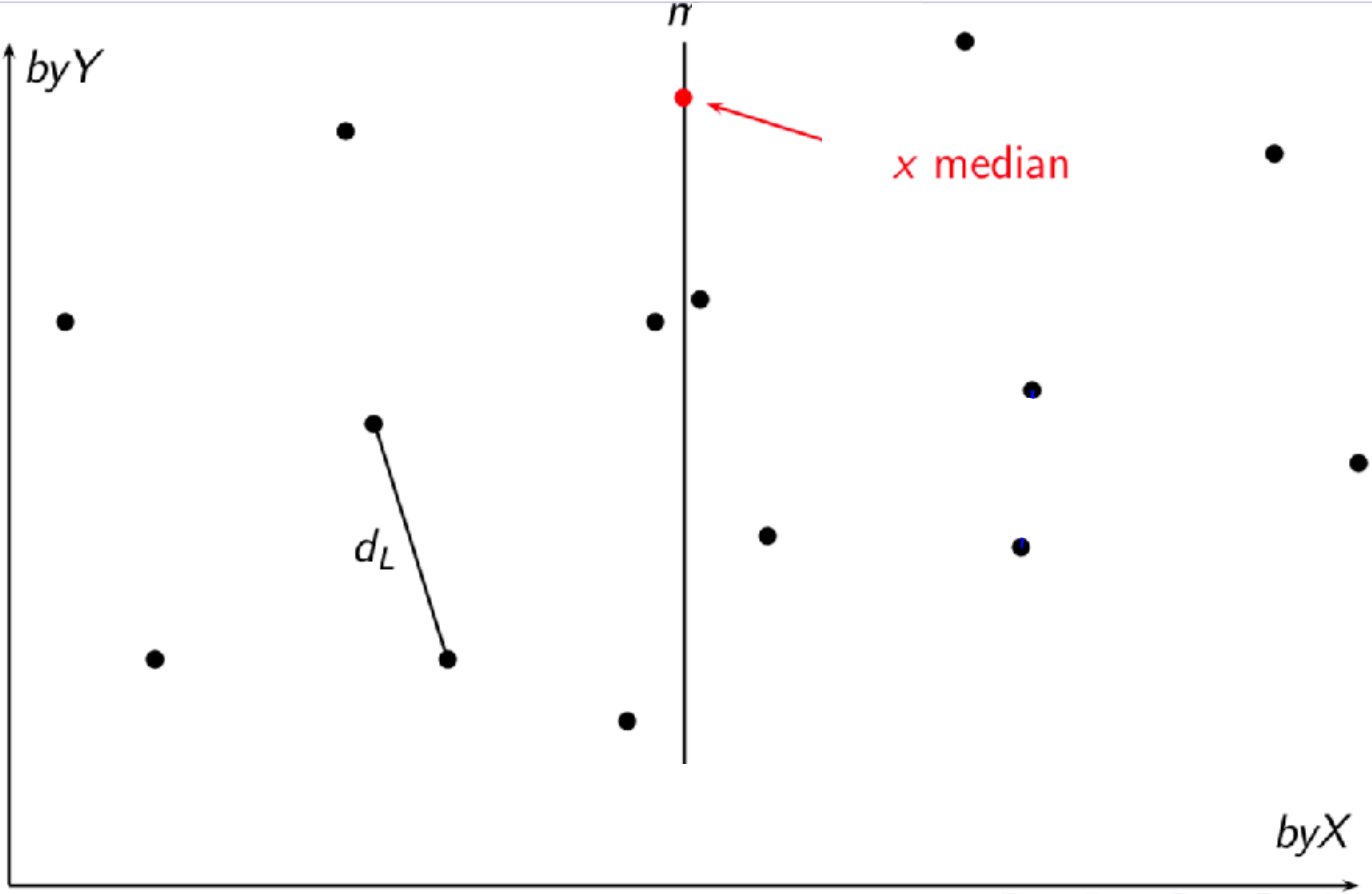
Closest Pair Problem Revisited



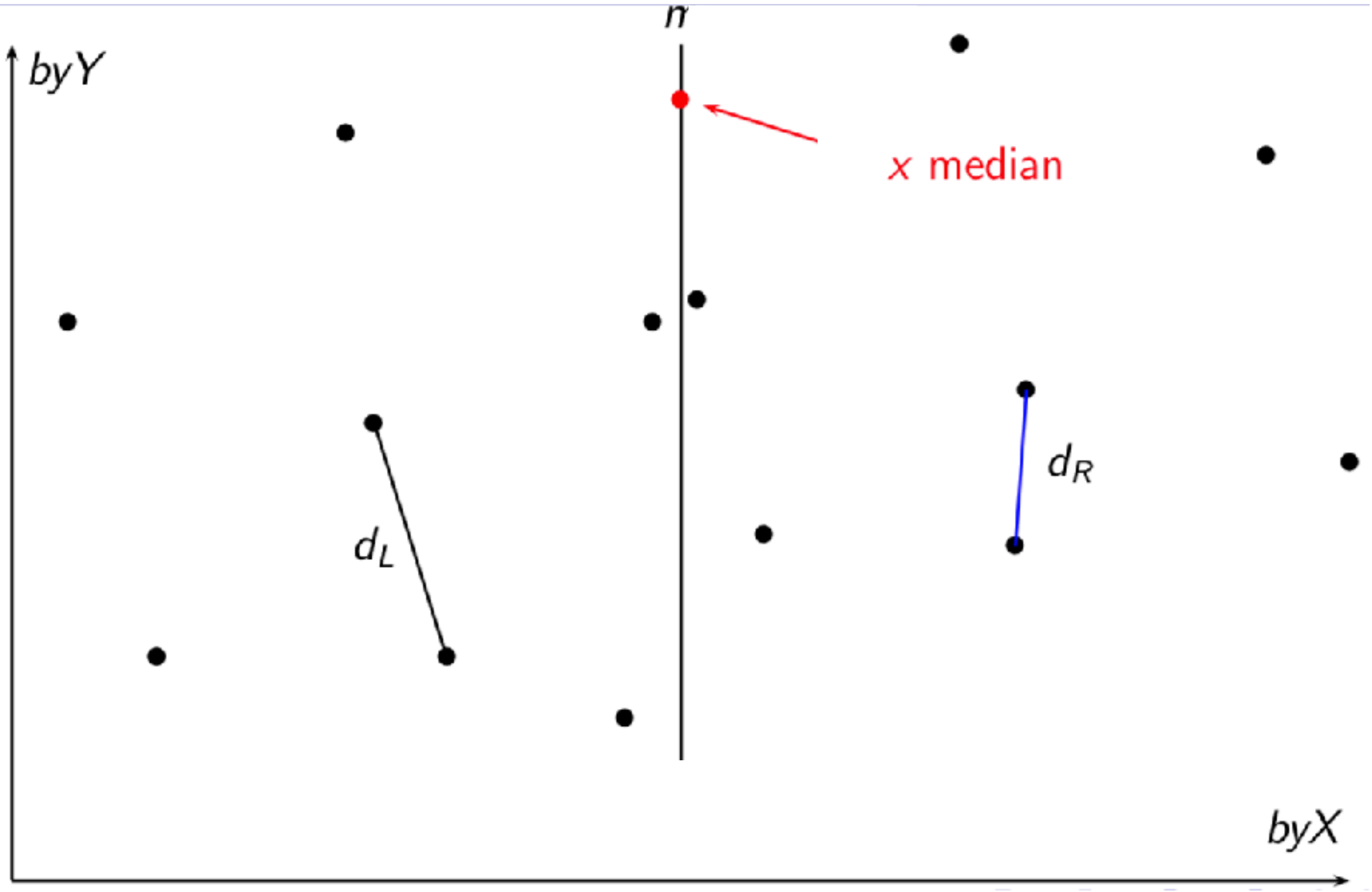
Closest Pair Problem Revisited



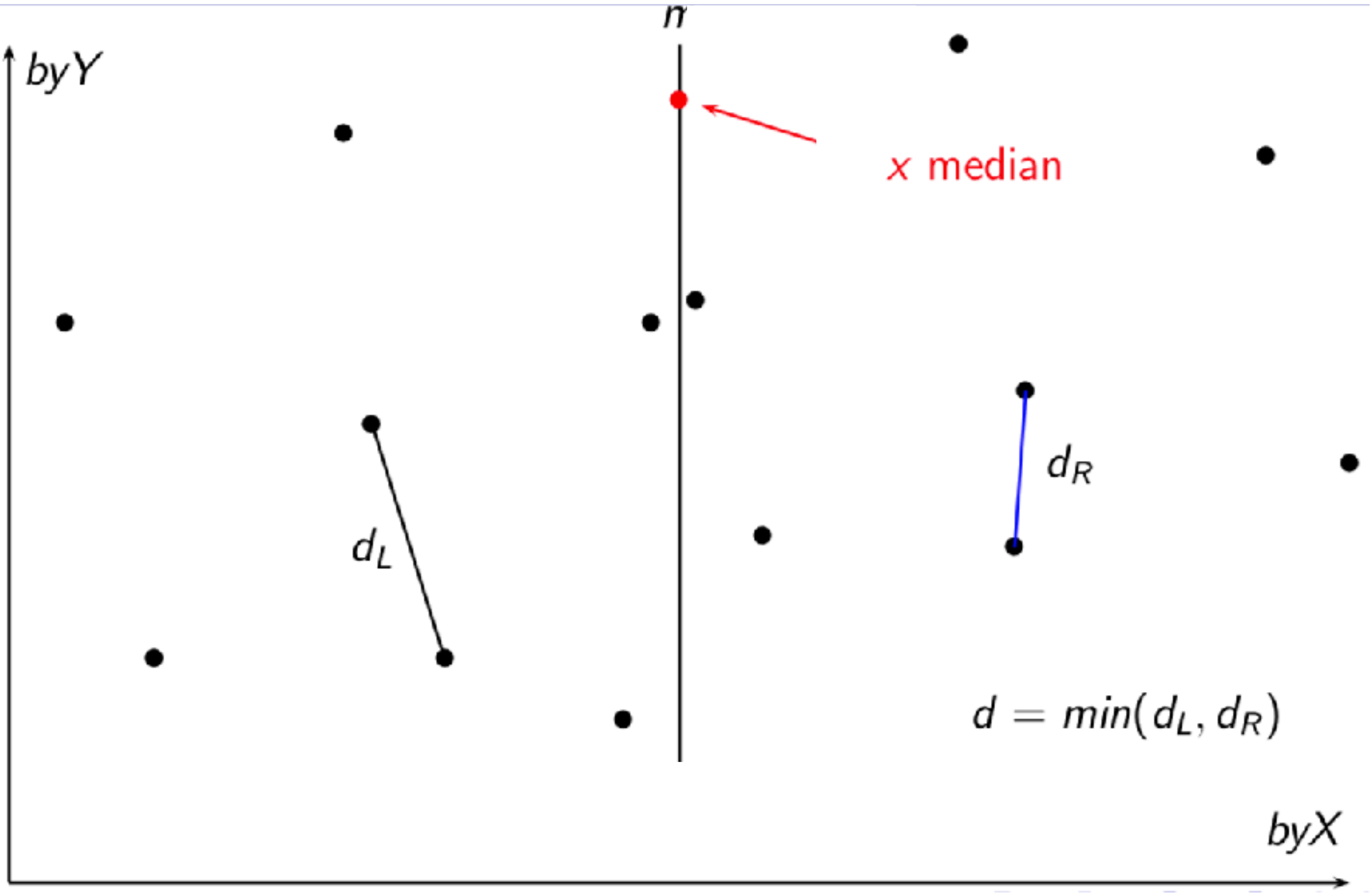
Closest Pair Problem Revisited



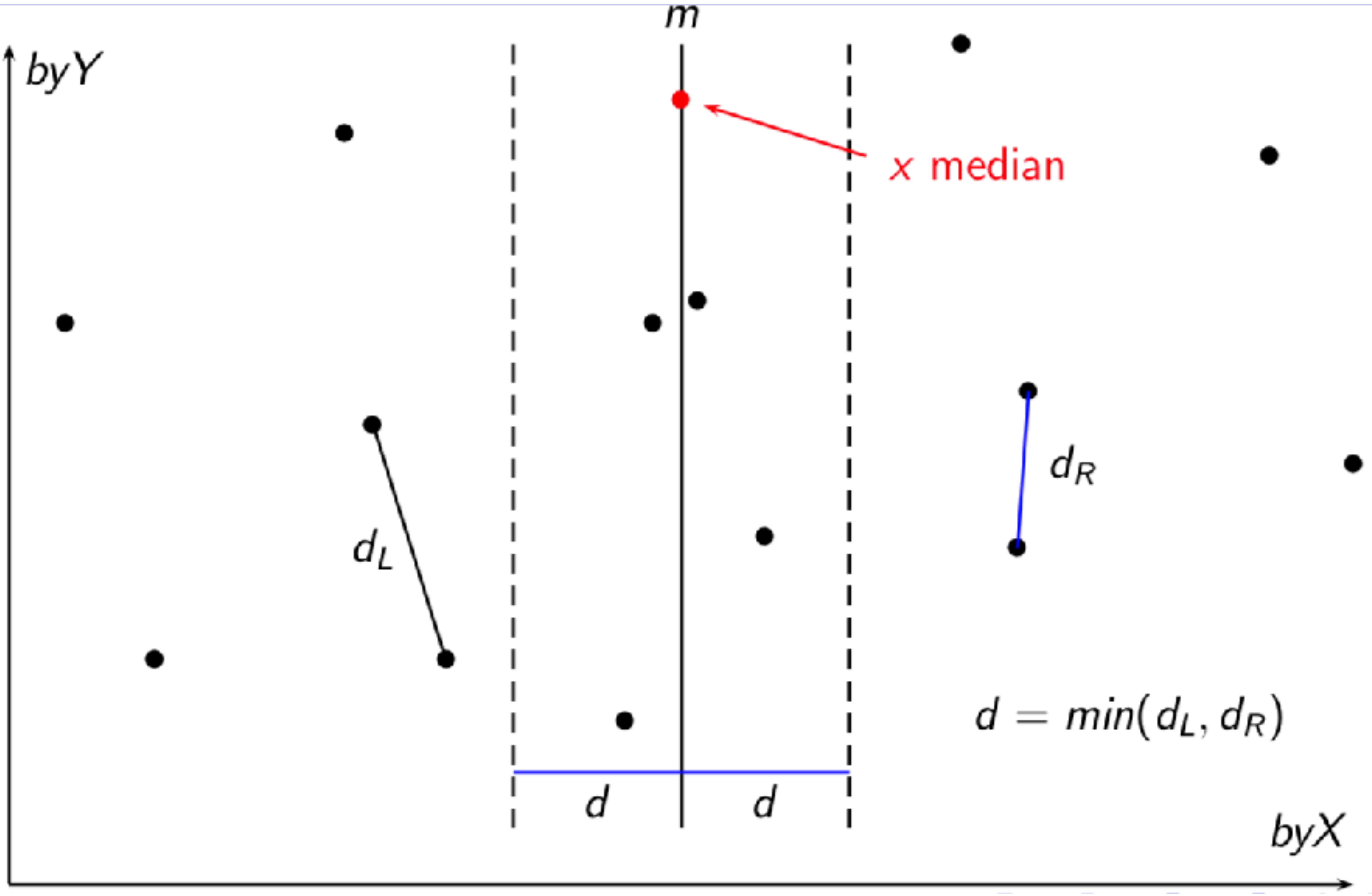
Closest Pair Problem Revisited



Closest Pair Problem Revisited



Closest Pair Problem Revisited



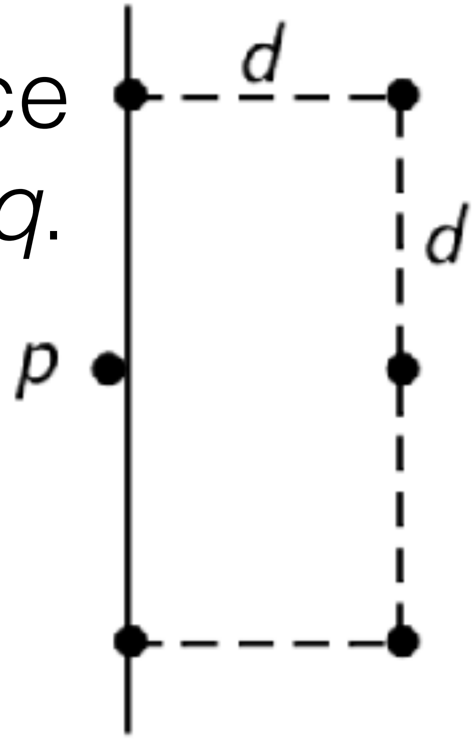
Closest Pair Problem Revisited

- The recursive calls will identify d_L, the shortest distance for pairs in P_L, and d_R, the shortest distance for pairs in P_R.
- Let m be the x median and let $d = \min(d_L, d_R)$. This d is a candidate for the smallest distance.
- But d may not be the global minimum—there could be some close pair whose points are on opposite sides of the median line $x = m$.
- For candidates that may improve on d we only need to look at those in the band $m - d \le x \le m + d$.
- So pick out, from array byY, each point p with x-coordinate between $m-d$ and $m+d$, and keep these in array S.
- For each point in S, consider just its “close” neighbours in S.

Closest Pair Problem Revisited

- The following calculates the smallest distance and leaves the (square of the) result in *minsq*.

- It can be shown that the while loop can execute **at most 5 times** for each *i* value—see diagram.



```
$minsq \leftarrow d^2$
copy all points of $byY$ with $|x - m| < d$ to array $S$
$k \leftarrow |S|$
for $i \leftarrow 0$ to $k - 2$ do
    $j \leftarrow i + 1$
    while $j \leq k - 1$ and $(S[j].y - S[i].y)^2 < minsq$ do
        $minsq \leftarrow min(minsq, (S[j].x - S[i].x)^2 + (S[j].y - S[i].y)^2)$
        $j \leftarrow j + 1$
```

You’re Learning Heaps!

- Next up: Priority queues, heaps and heapsort.