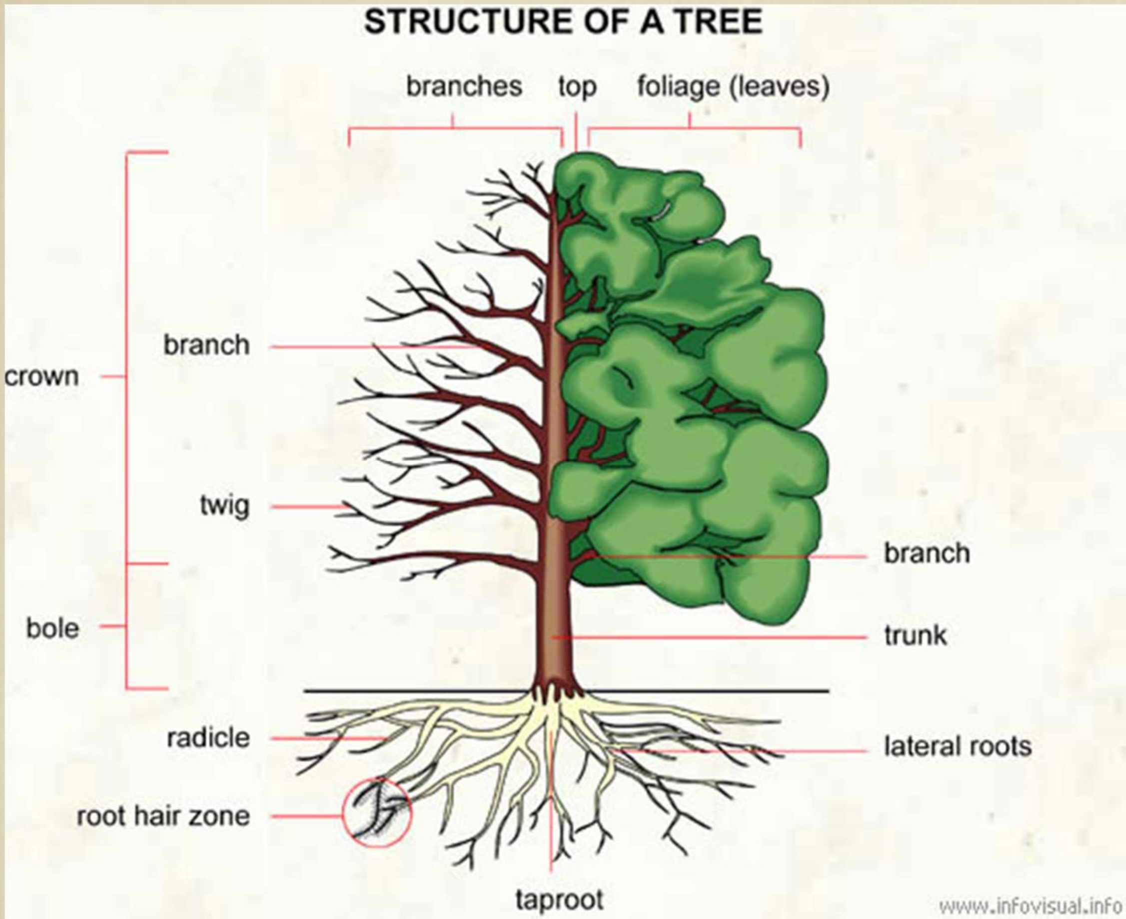
Linked Structures Songs, Games, Movies Part III

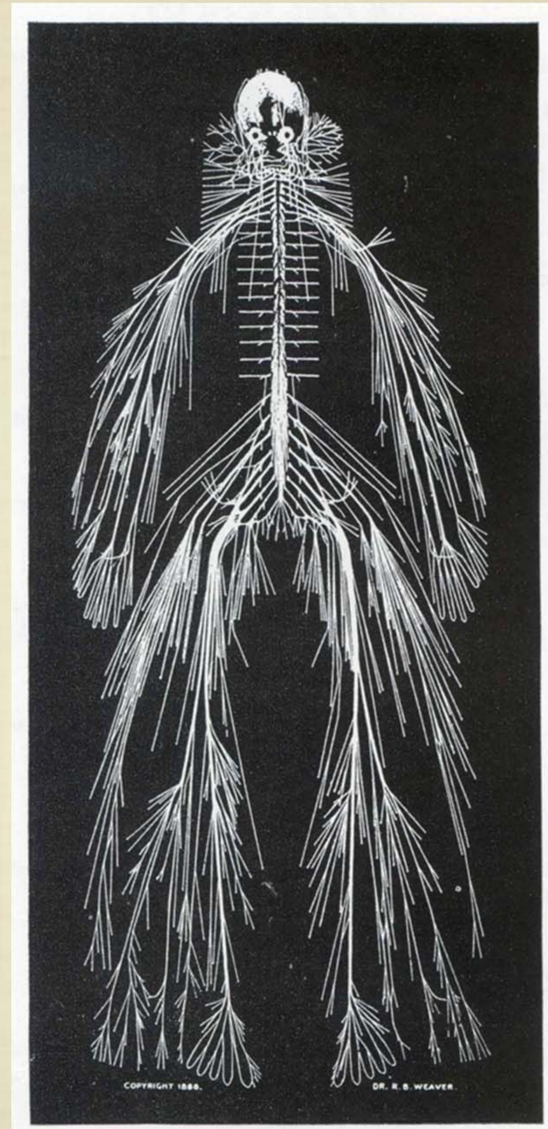
Fall 2013

Carola Wenk

Biological Structures

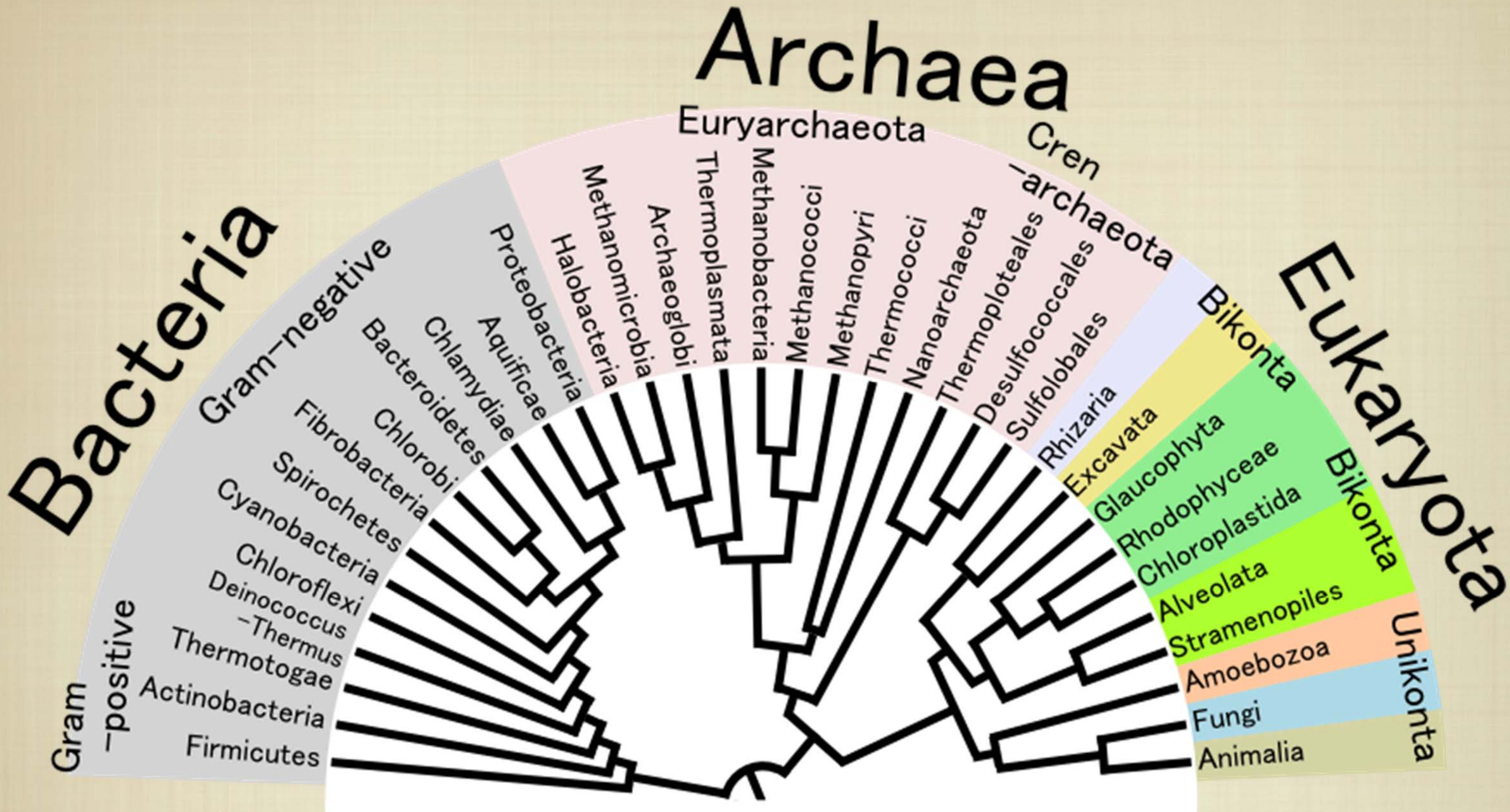


Nature has evolved vascular and nervous systems in a hierarchical manner so that nutrients and signals can quickly reach their destinations.

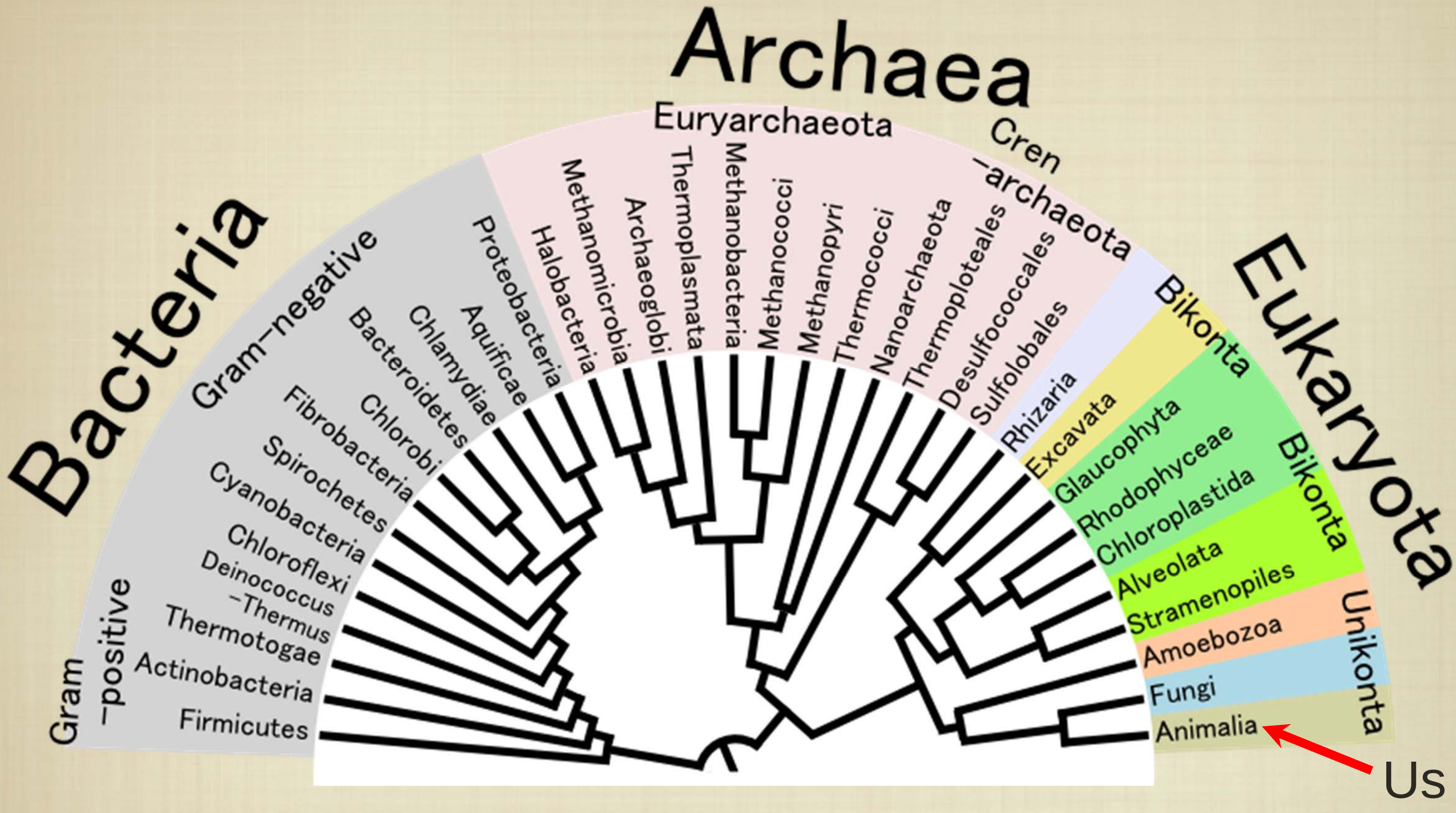


Human Nervous System

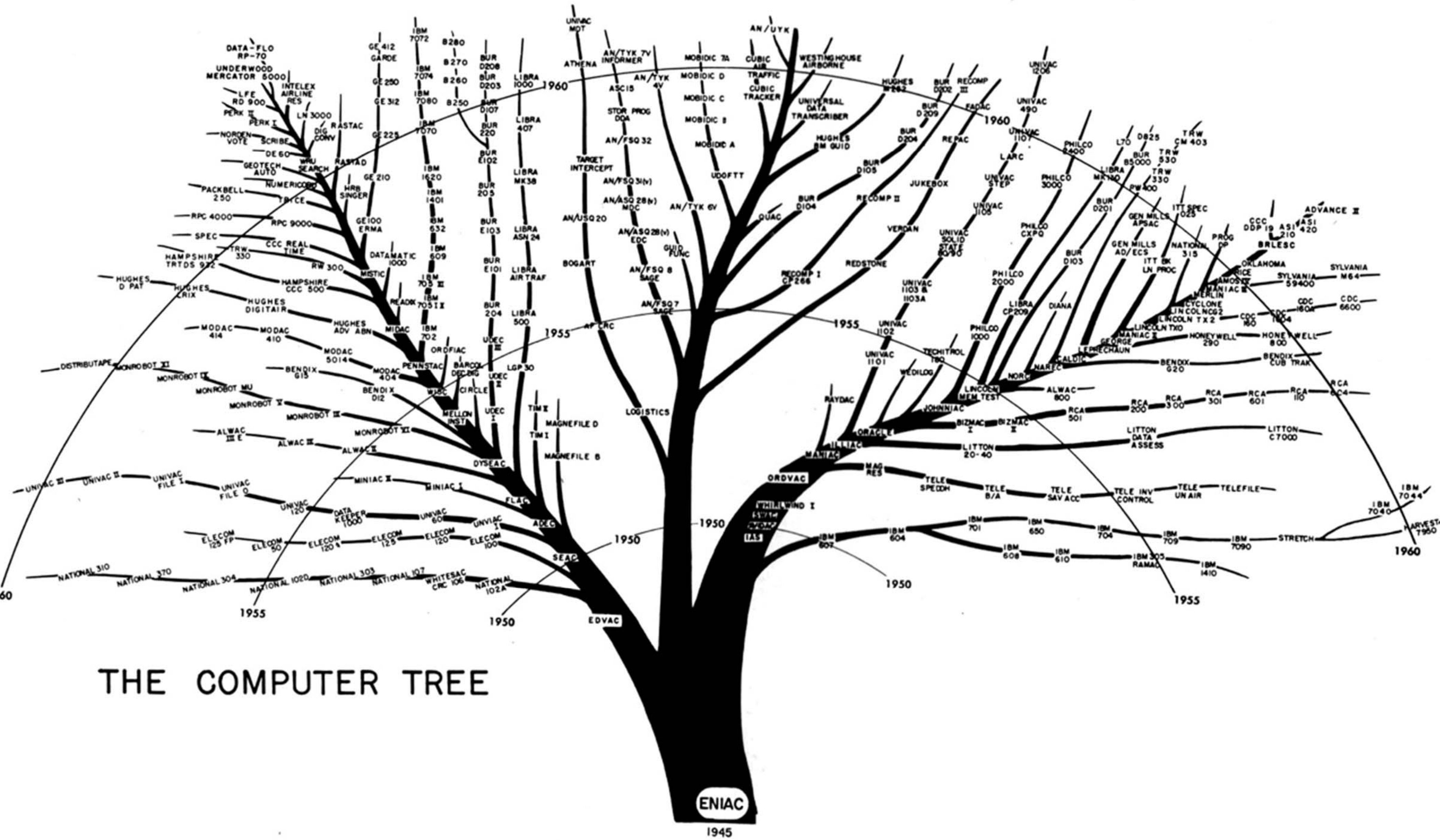
A large amount of information can be viewed more easily when it is laid out in a hiearchical fashion.



A large amount of information can be viewed more easily when it is laid out in a hiearchical fashion.

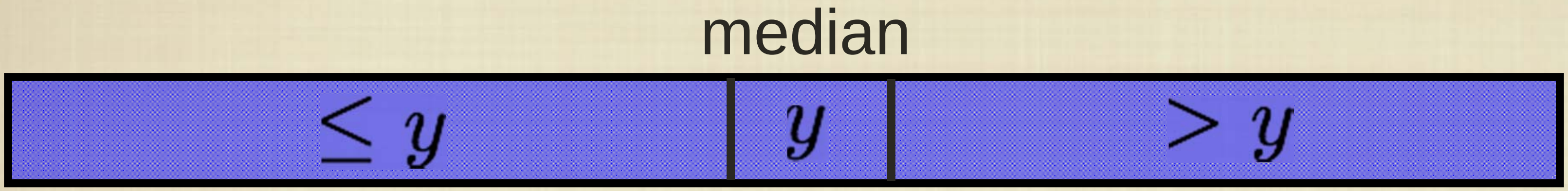


A large amount of information can be viewed more easily when it is laid out in a hiearchical fashion.



A large amount of information can be viewed more easily when it is laid out in a hiearchical fashion.

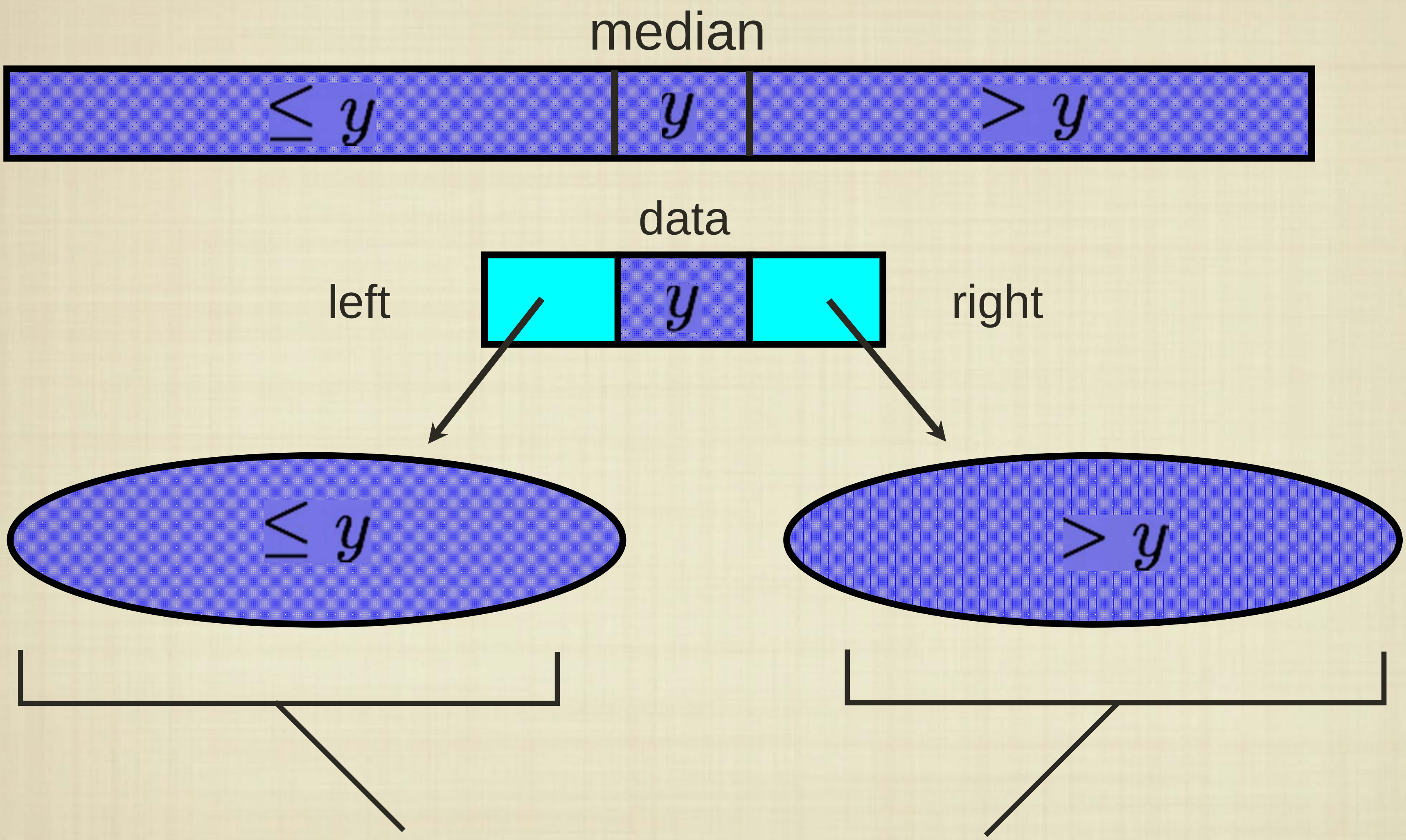
Remember Binary Search?



In binary search, immediate access to the median allowed us to quickly direct our search to smaller or larger elements.

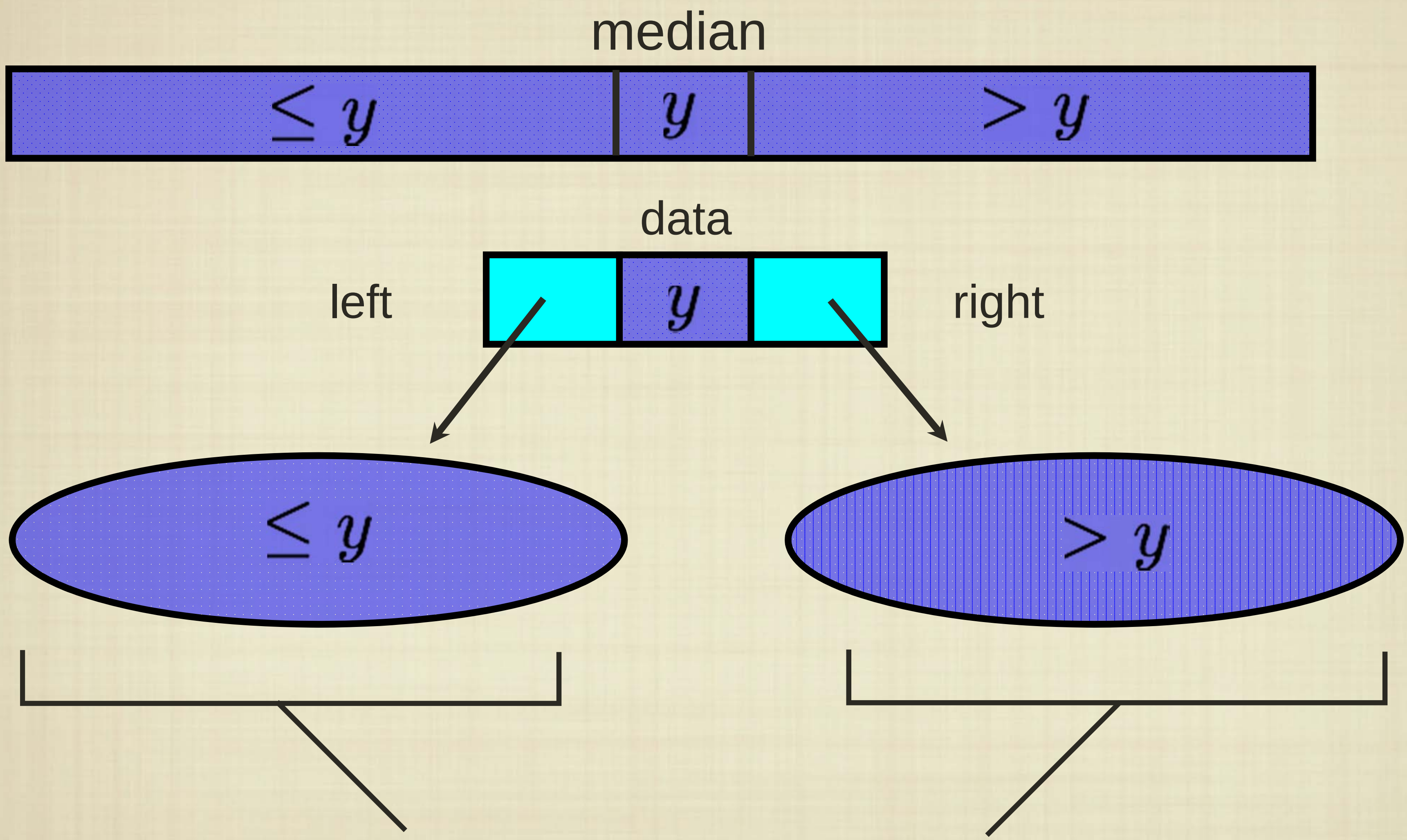
Can we use this idea to develop a linked structure?

Remember Binary Search?



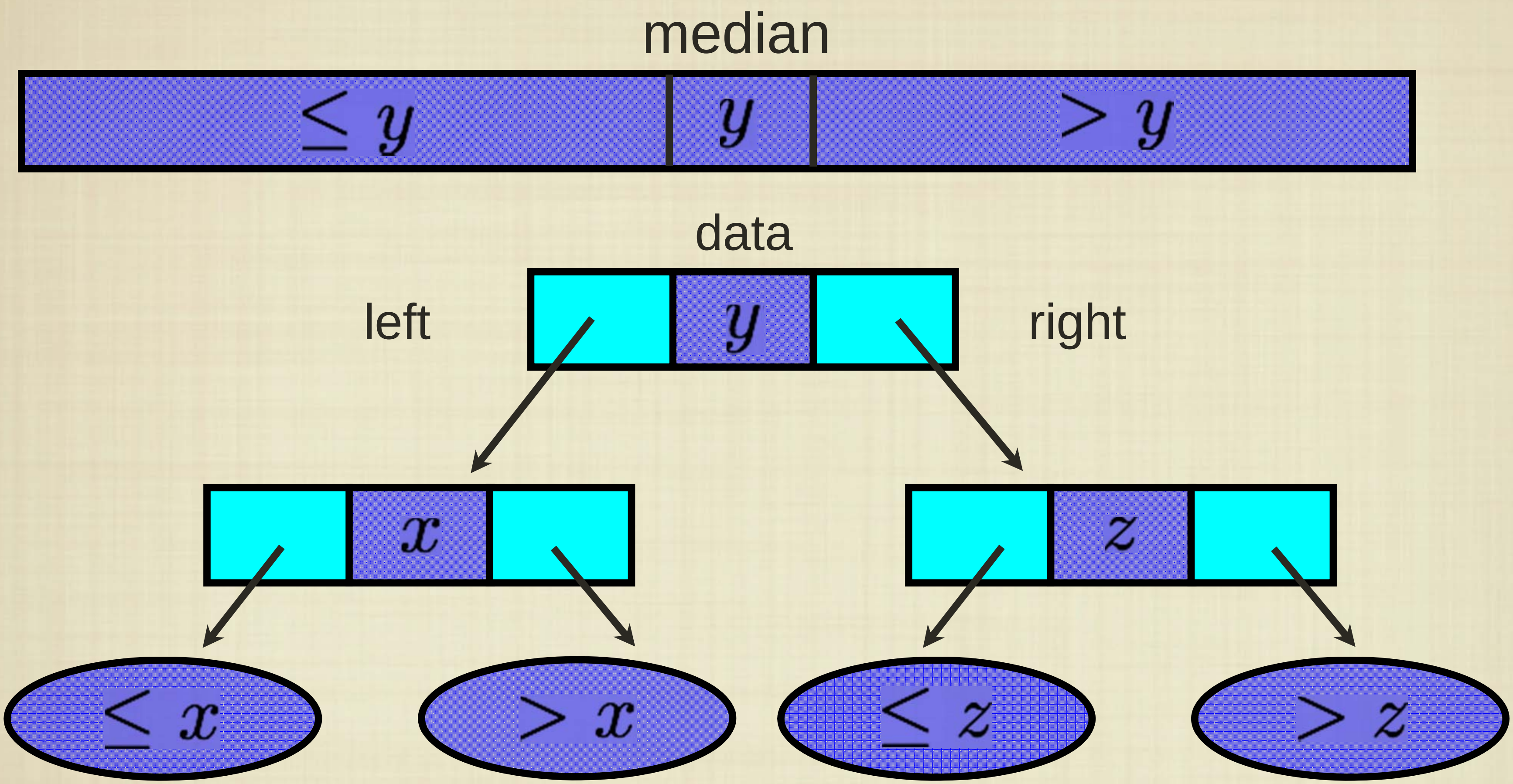
- We can split the list into two “halves” of a linked structure.

Remember Binary Search?



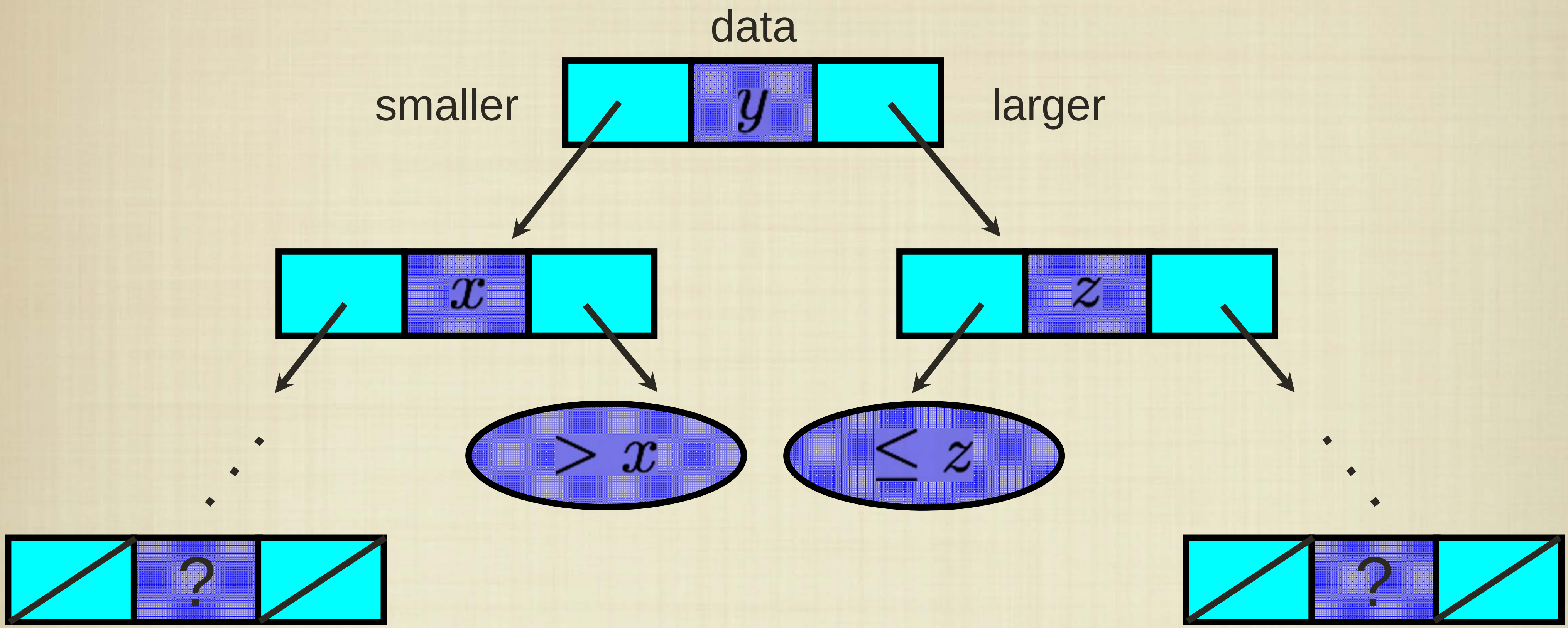
The two halves of a binary search tree can be defined recursively.

Binary Search Trees



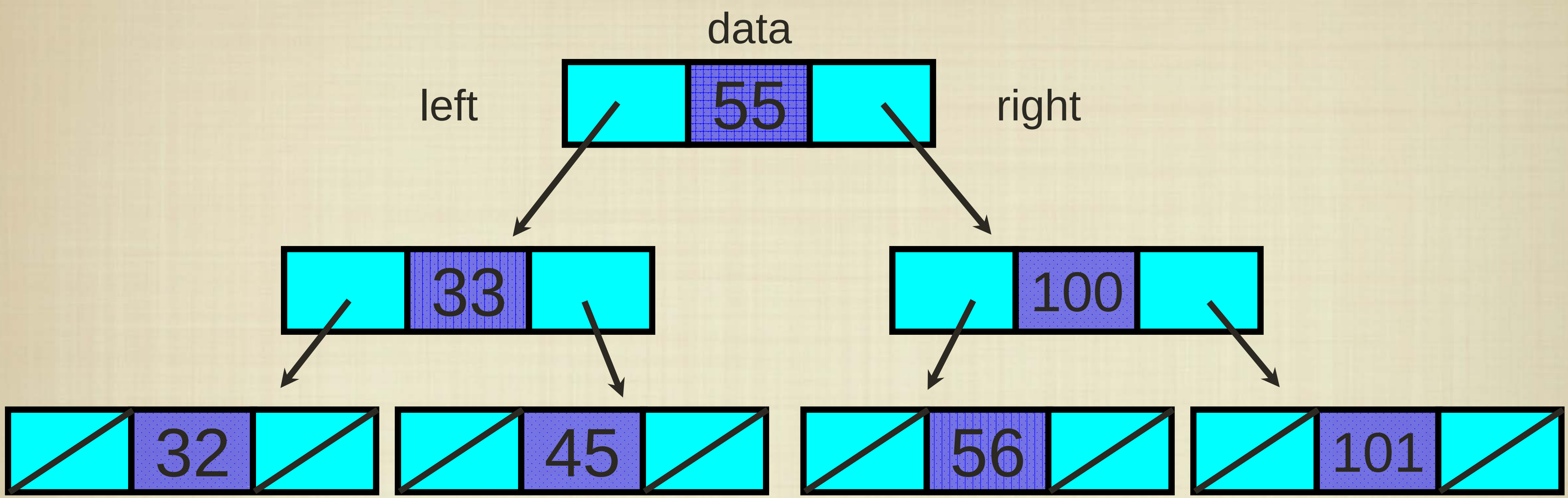
- How do we define this type of structure in Python?

Binary Search Trees



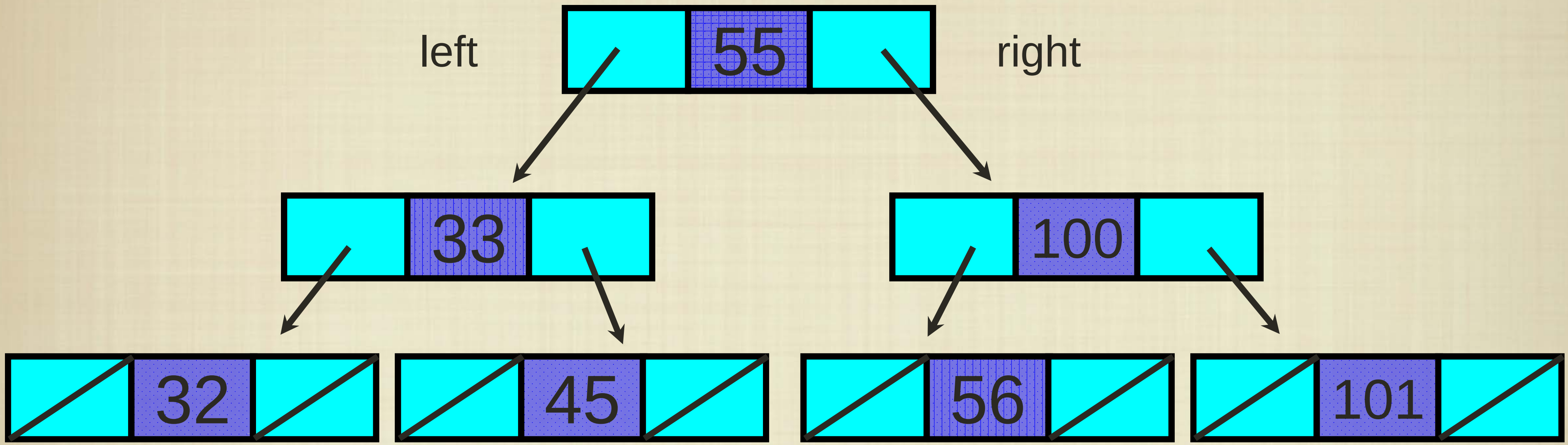
- A node in this linked structure is called a “leaf” if it has no “children.” The topmost element is the “root”.

Binary Search Tree Example



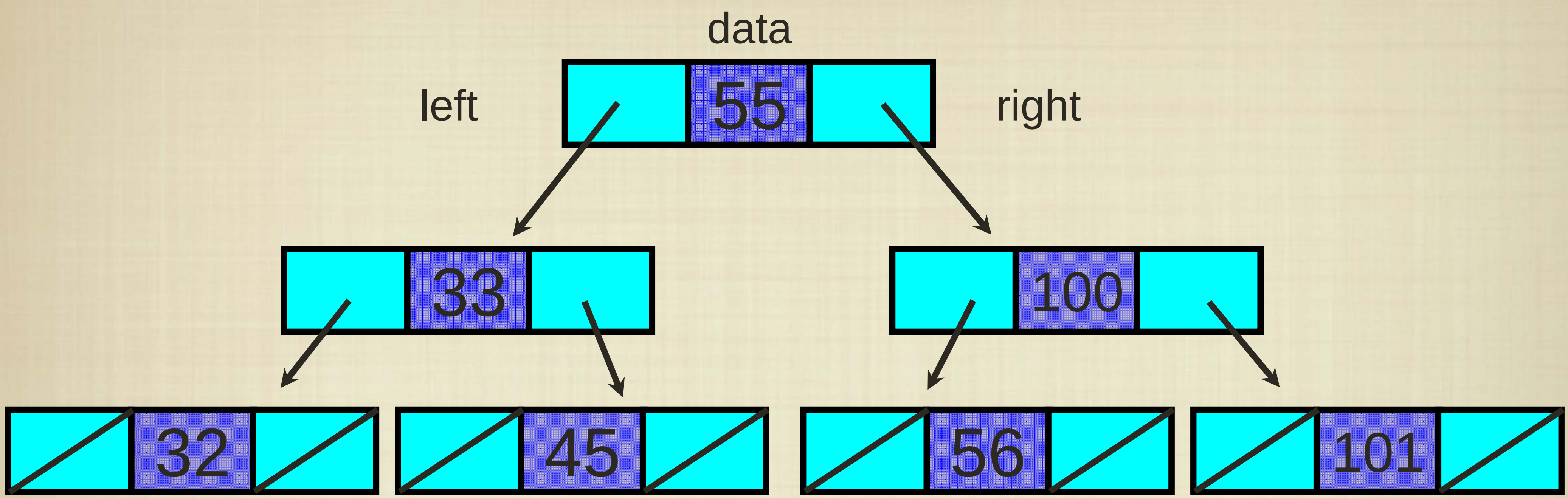
- Where are the minimum and maximum of the element?
- Can we quickly find any element?
- What about adding/removing?

Finding the Minimum/Maximum



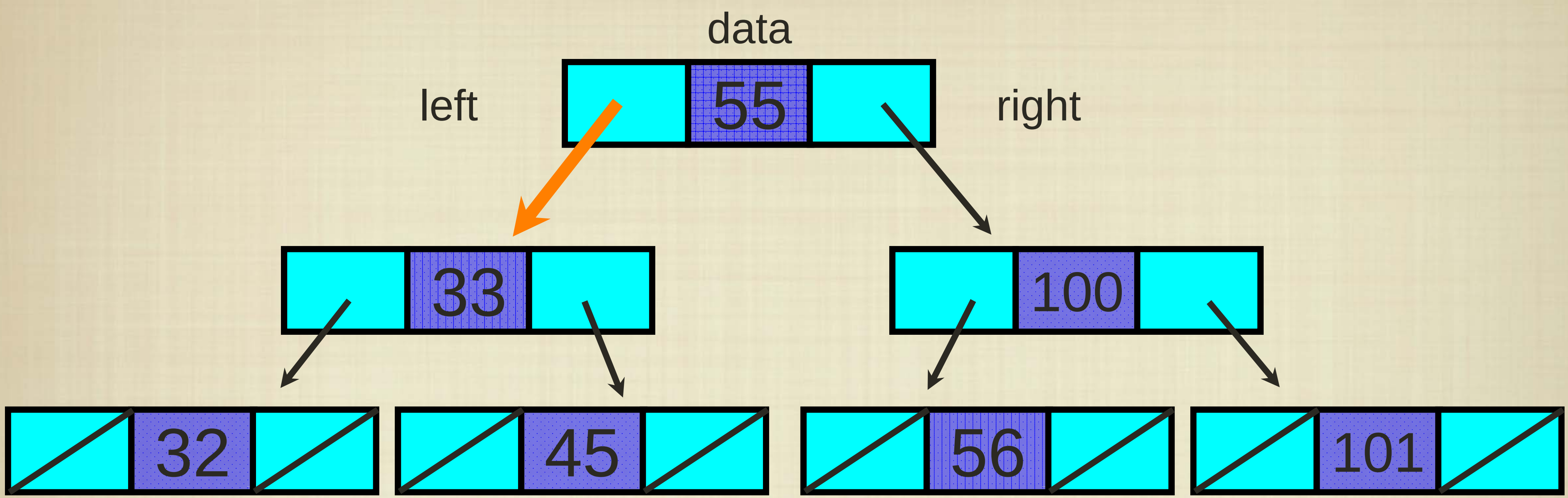
- The minimum element can be found by following the `left` references, and the maximum element can be found by following `right` references.

Finding Any Item (Quickly)



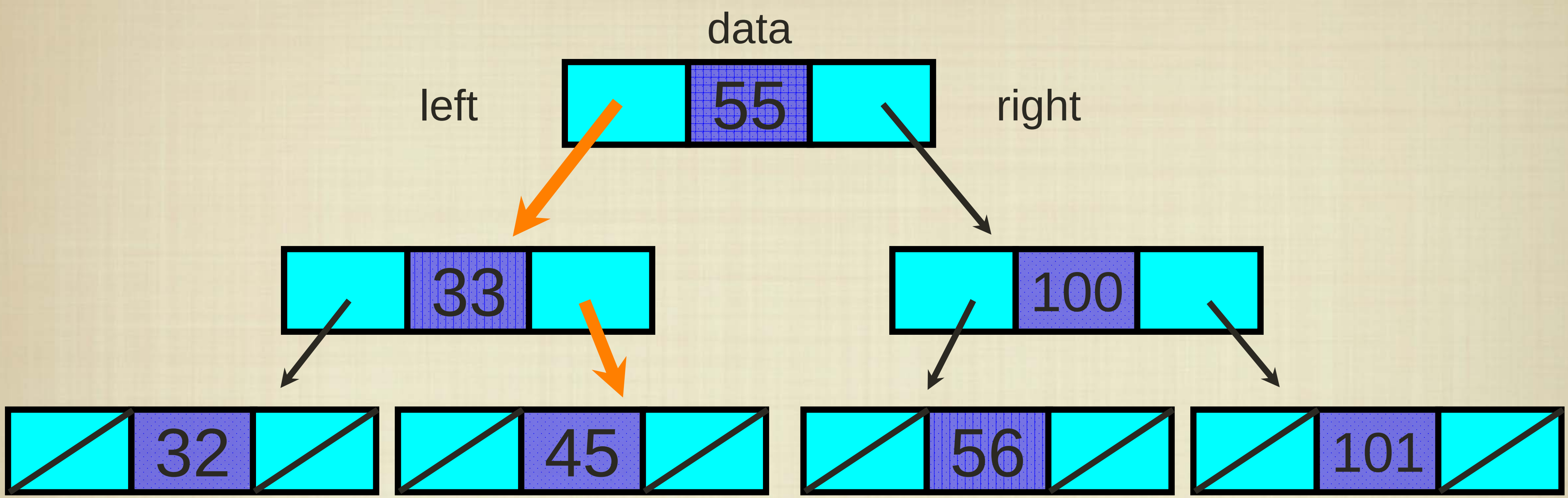
- In exactly the same way as binary search, we can recursively focus on one “side” of the tree by checking the root element. How long does this take?

Finding Any Item (Quickly)



- In exactly the same way as binary search, we can recursively focus on one “side” of the tree by checking the root element. How long does this take?

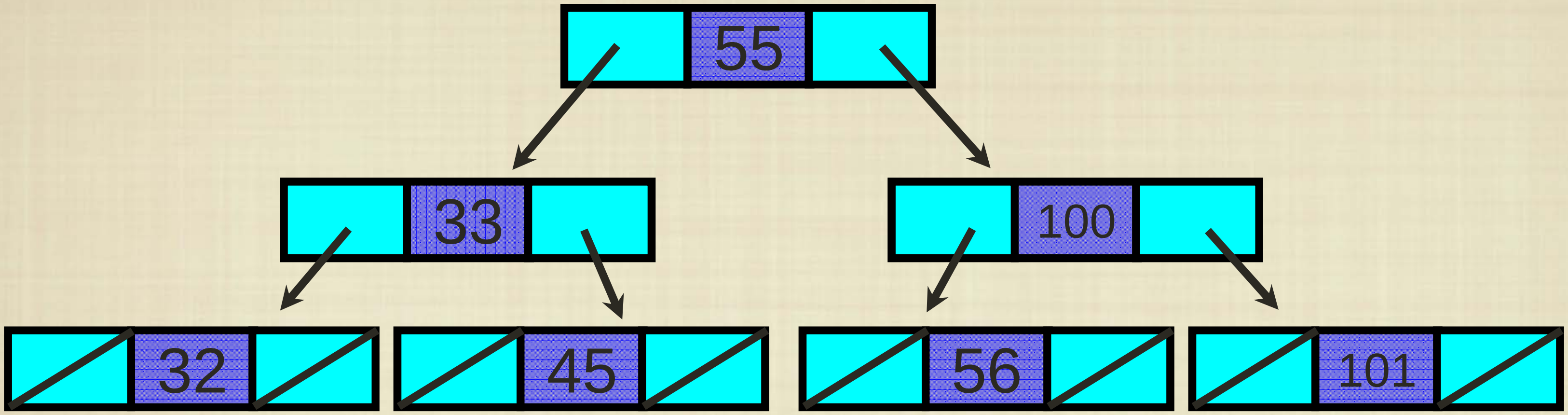
Finding Any Item (Quickly)



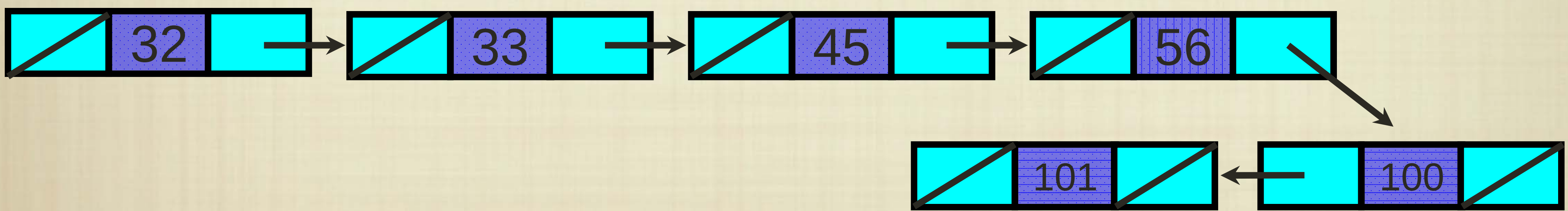
- In exactly the same way as binary search, we can recursively focus on one “side” of the tree by checking the root element. How long does this take?

The (worst-case) time to find an item depends on the height of the tree. How large can the height be?

Worst Case Scenario

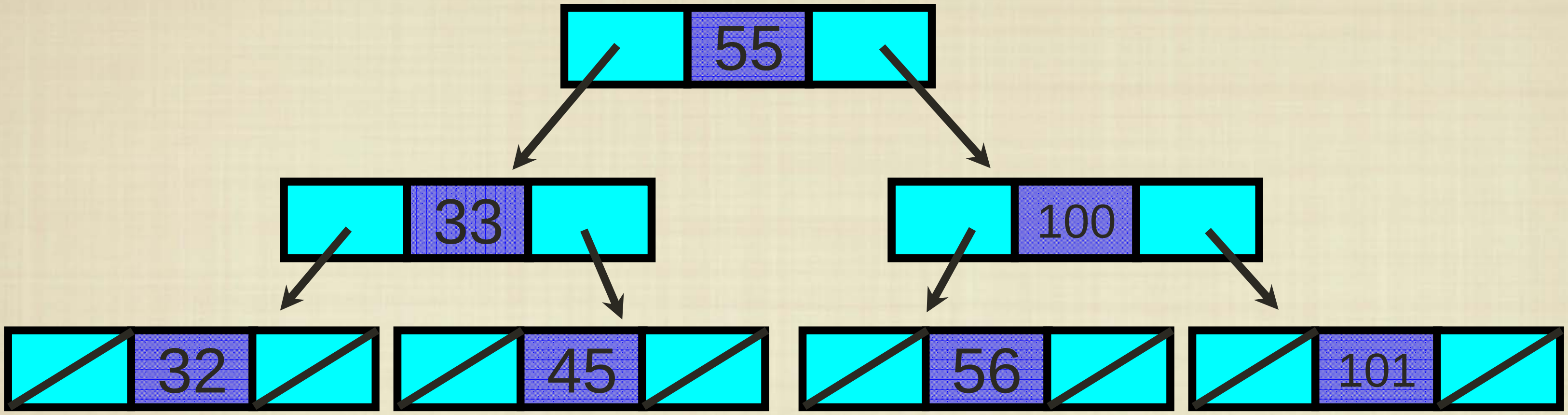


- In this particular tree, we can find any element in two steps.

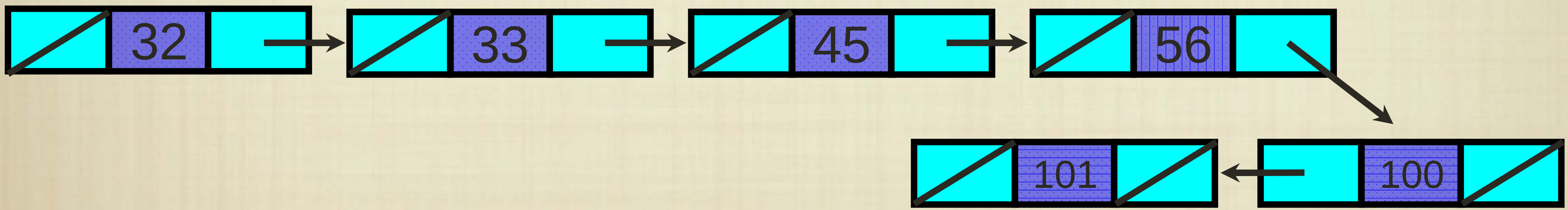


Is this a tree? Why or why not?

Worst Case Scenario

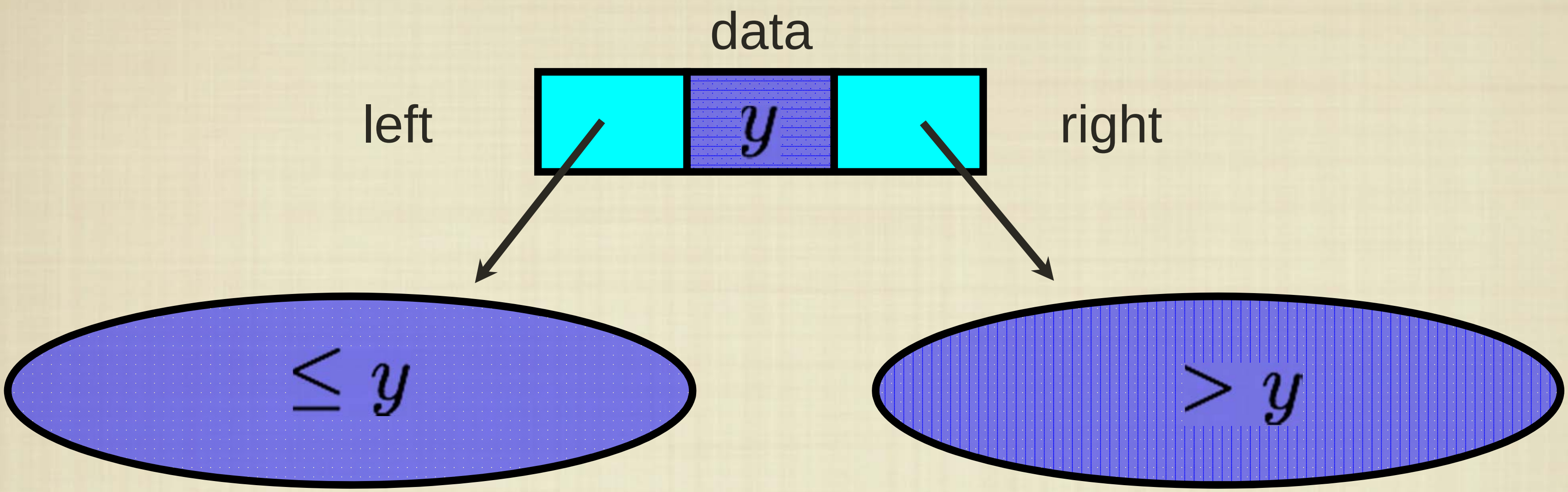


- In this particular tree, we can find any element in two steps.



This is a binary tree that's not very tree-like. Why would a tree look like this?

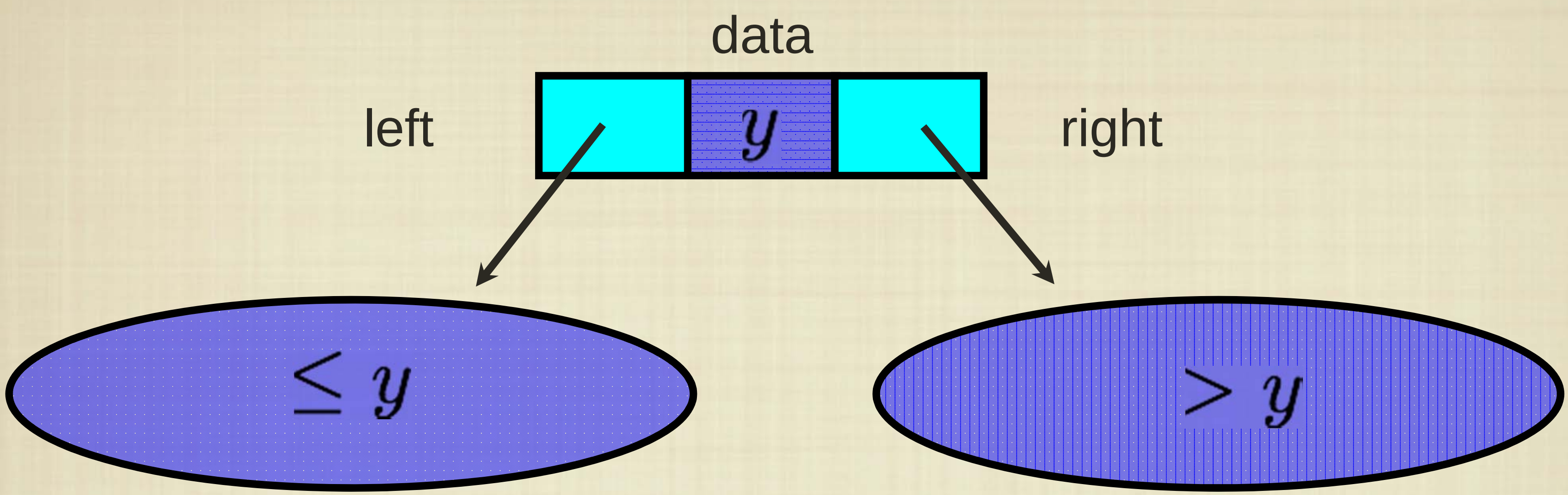
Adding Items



According to our definition, we know at least which side of the tree to insert.

Algorithm: Recursively determine which side of the tree to insert, and create a new element at the bottom.

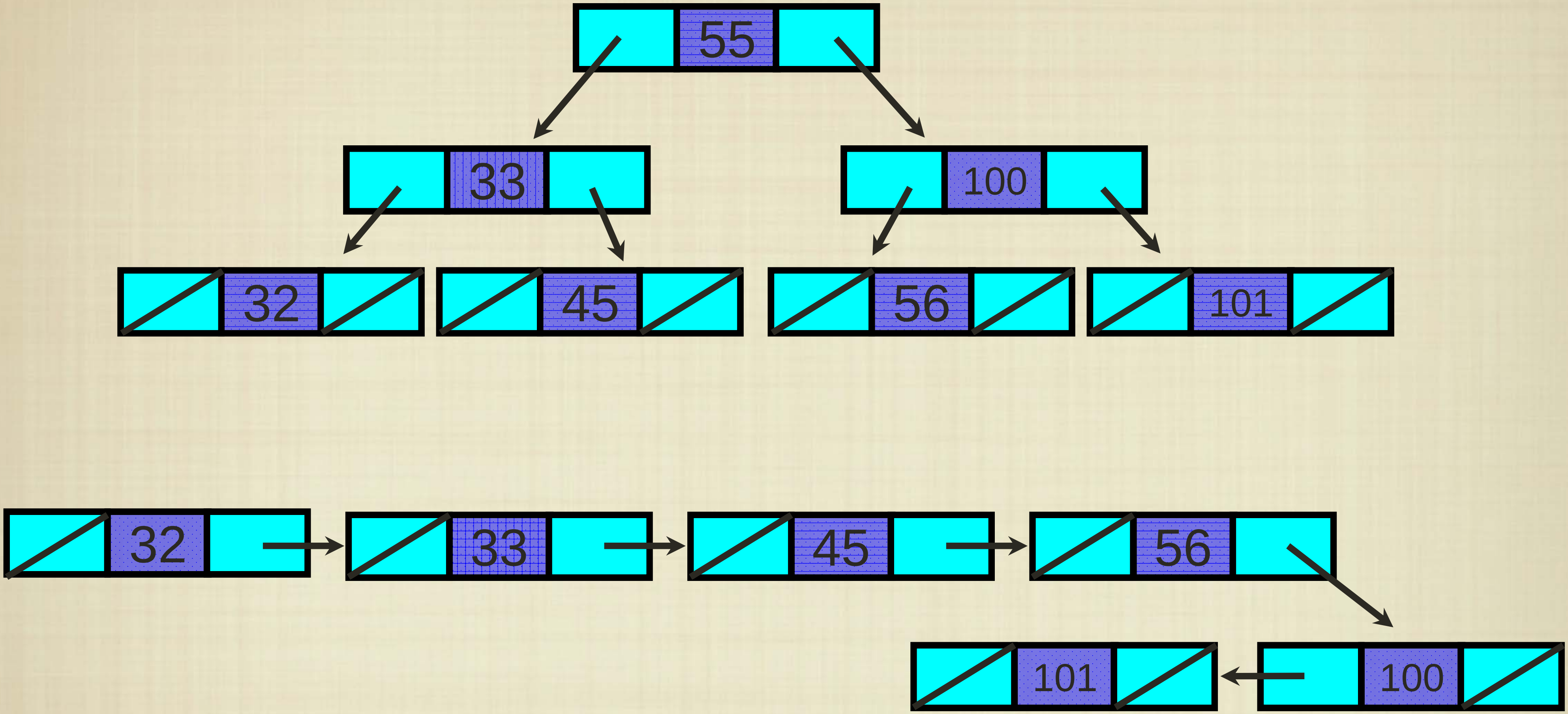
Adding Items



Algorithm: Recursively determine which side of the tree to insert, and create a new element at the bottom.

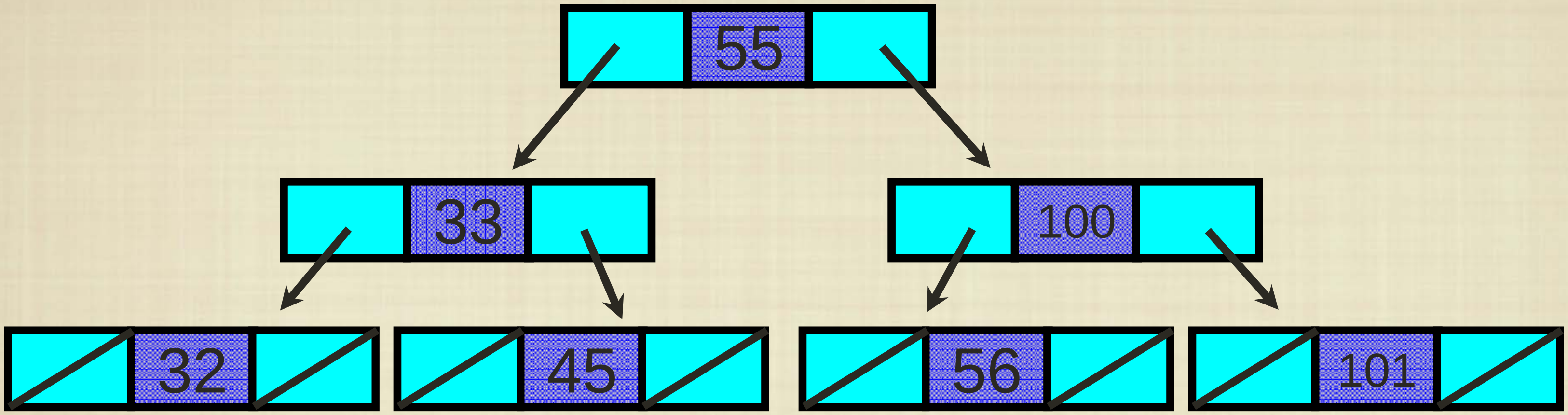
Unfortunately, we can't control the order in which things are added.

Worst Case Scenario

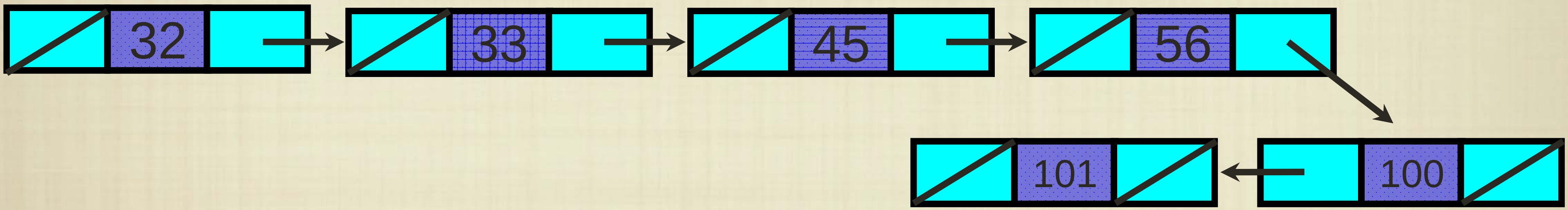


How were these two trees created?

Worst Case Scenario



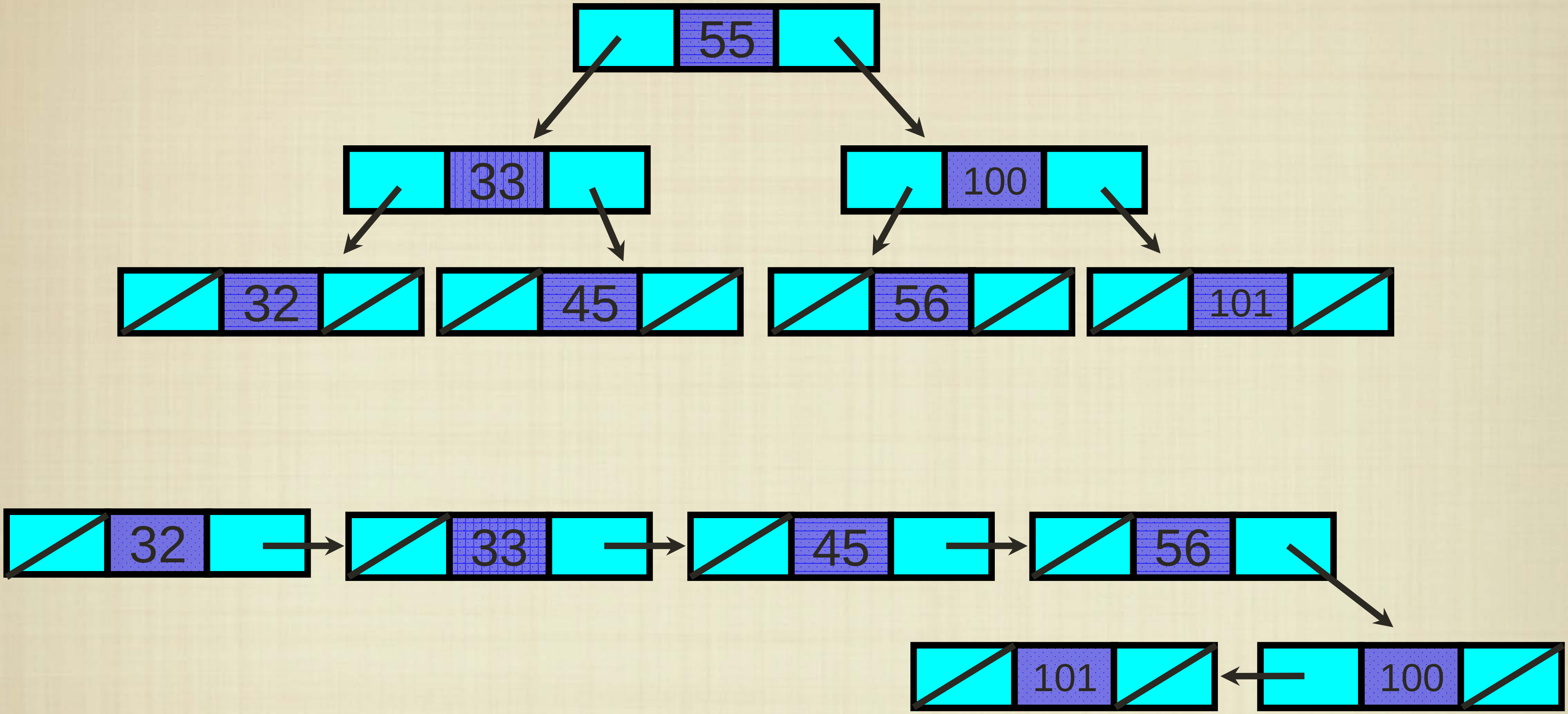
Insertion Order: 55, 33, 100, 32, 45, 56, 101



Insertion Order: 32, 33, 45, 56, 100, 101

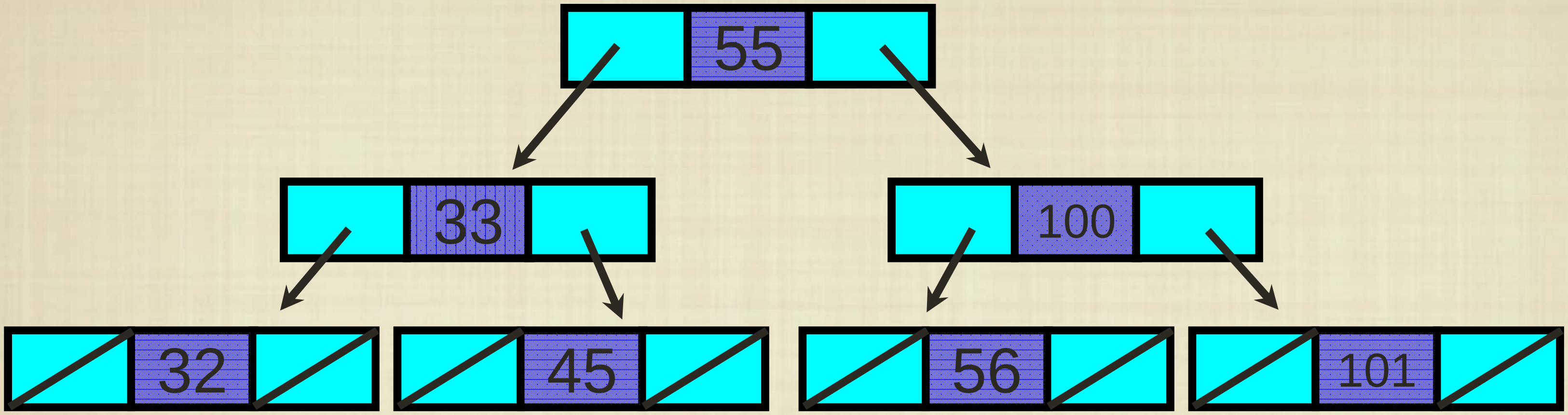
Ironically, inserting items in sorted order produces an extremely “imbalanced” tree.

Worst Case Scenario



The <u>height</u> of a binary search tree is the longest root-to-leaf path; researchers have studied how to minimize this.

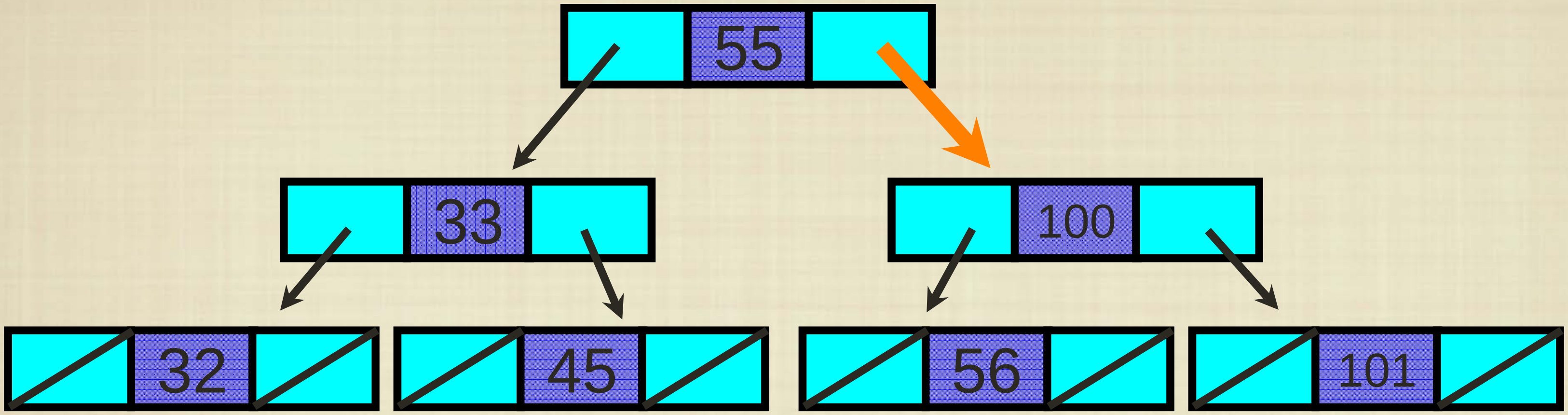
Removing Items



We may want to remove any item in the tree - what if we want to delete the “root”?

We need to find a substitute; where is the next largest item in the tree located?

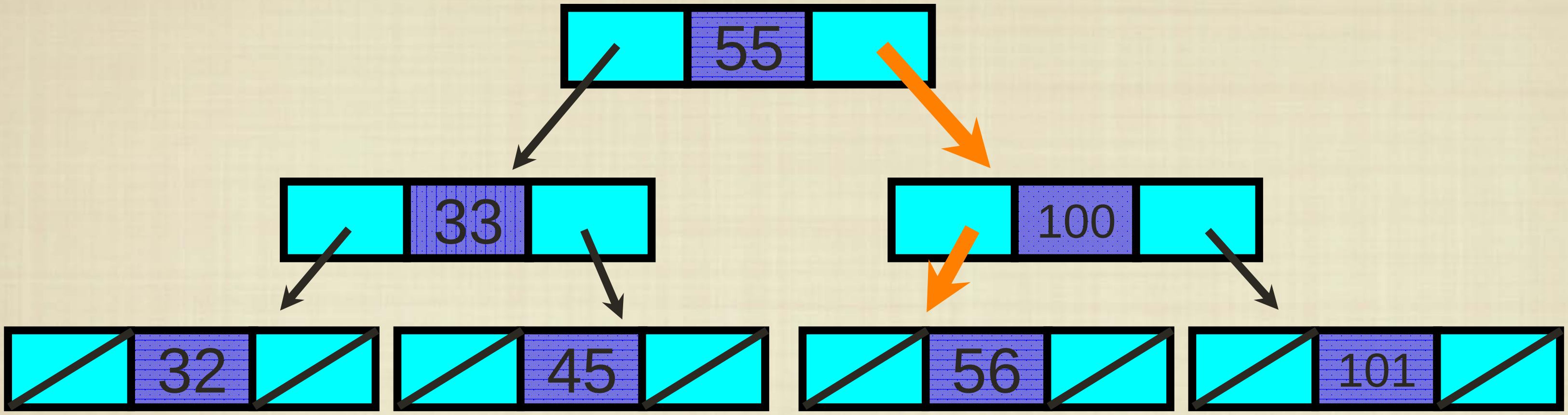
Removing Items



We may want to remove any item in the tree - what if we want to delete the “root”?

We need to find a substitute; where is the next largest item in the tree located?

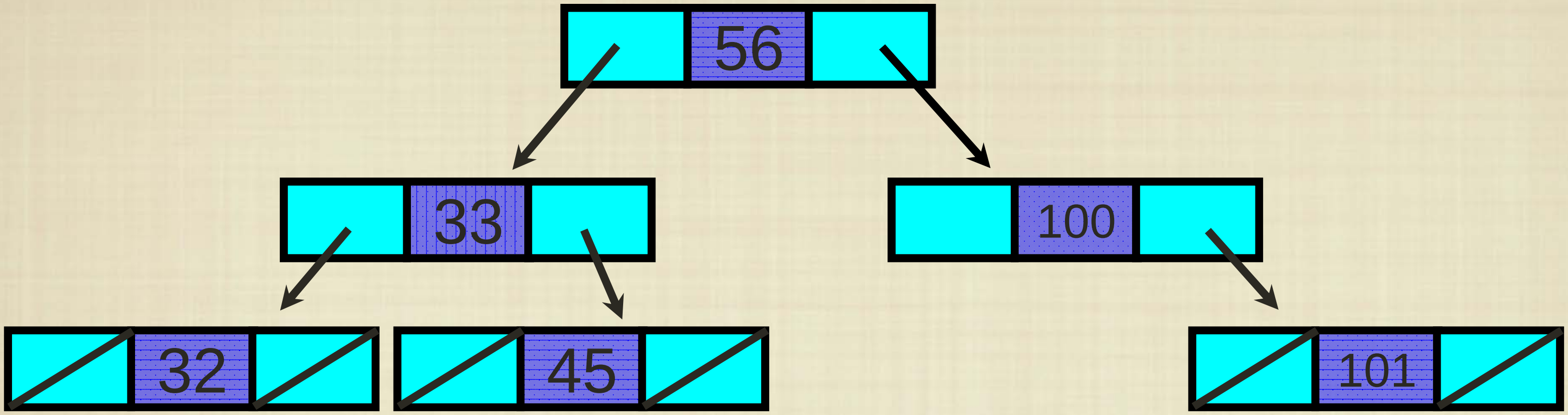
Removing Items



Which two elements can take the place of the item to be removed?

Algorithm: Replace the item to be deleted with the smallest item larger than it. (The minimum element in the right subtree.)

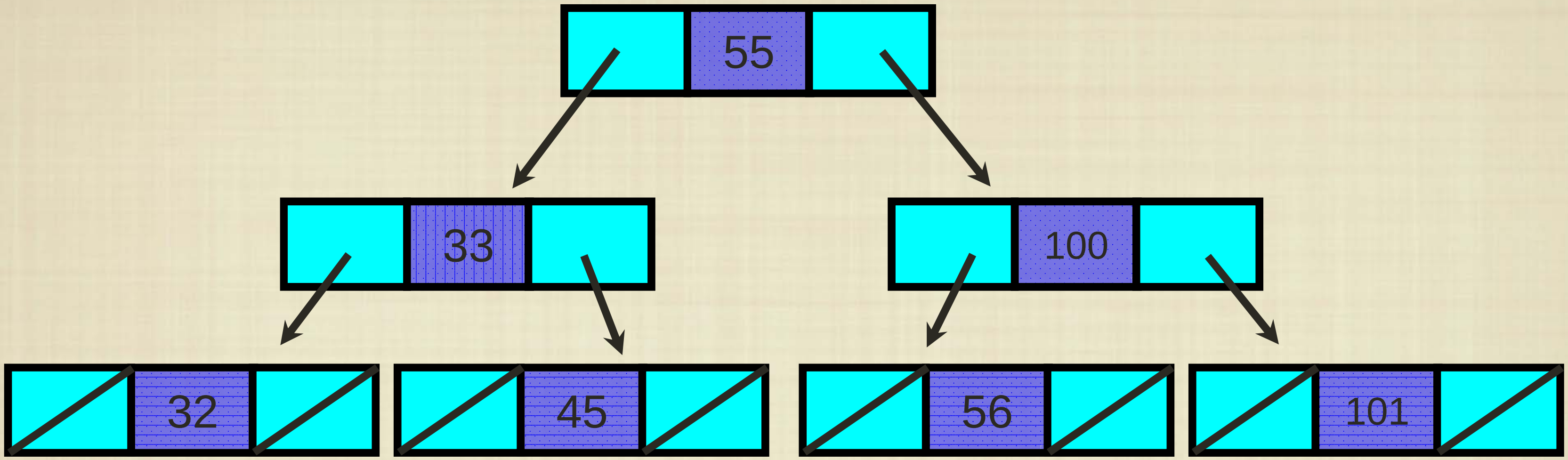
Removing Items



Which two elements can take the place of the item to be removed?

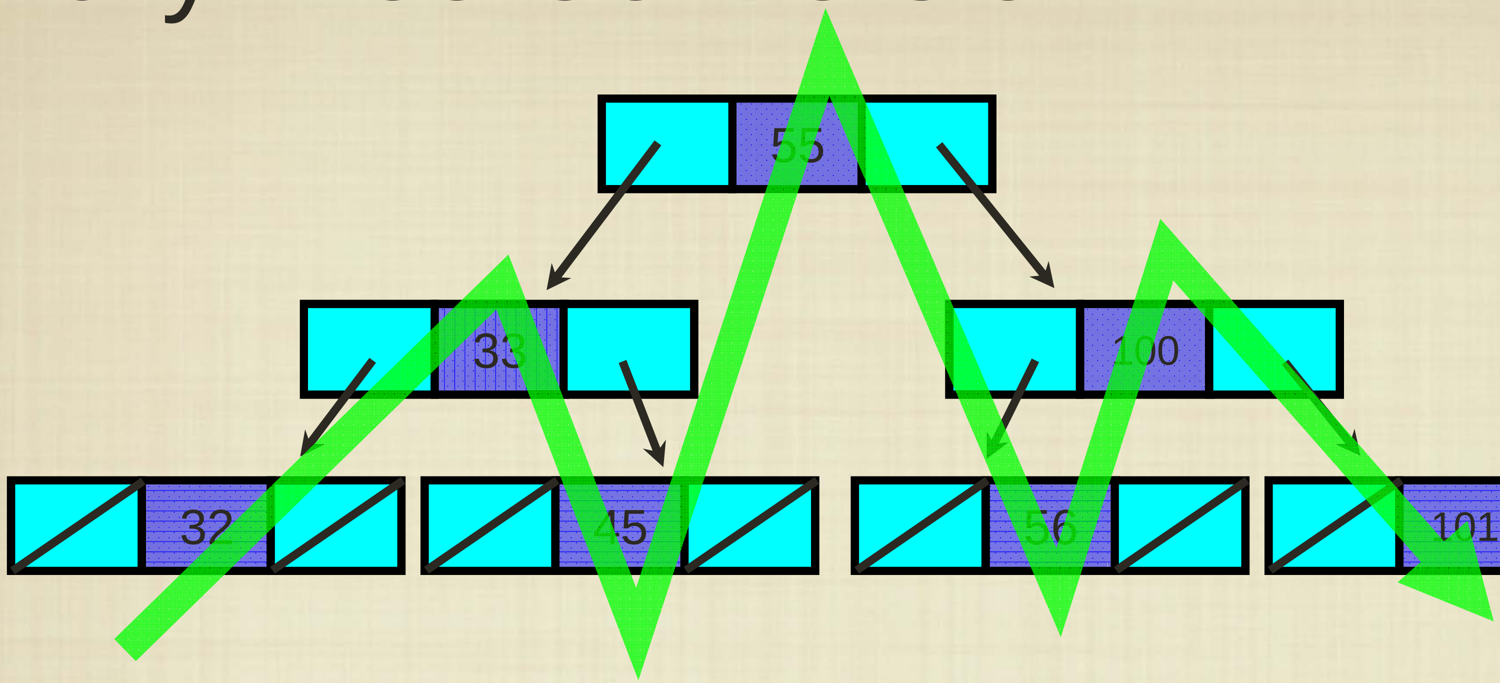
Algorithm: Replace the item to be deleted with the smallest item larger than it. (The minimum element in the right subtree.)

Binary Tree Conversion



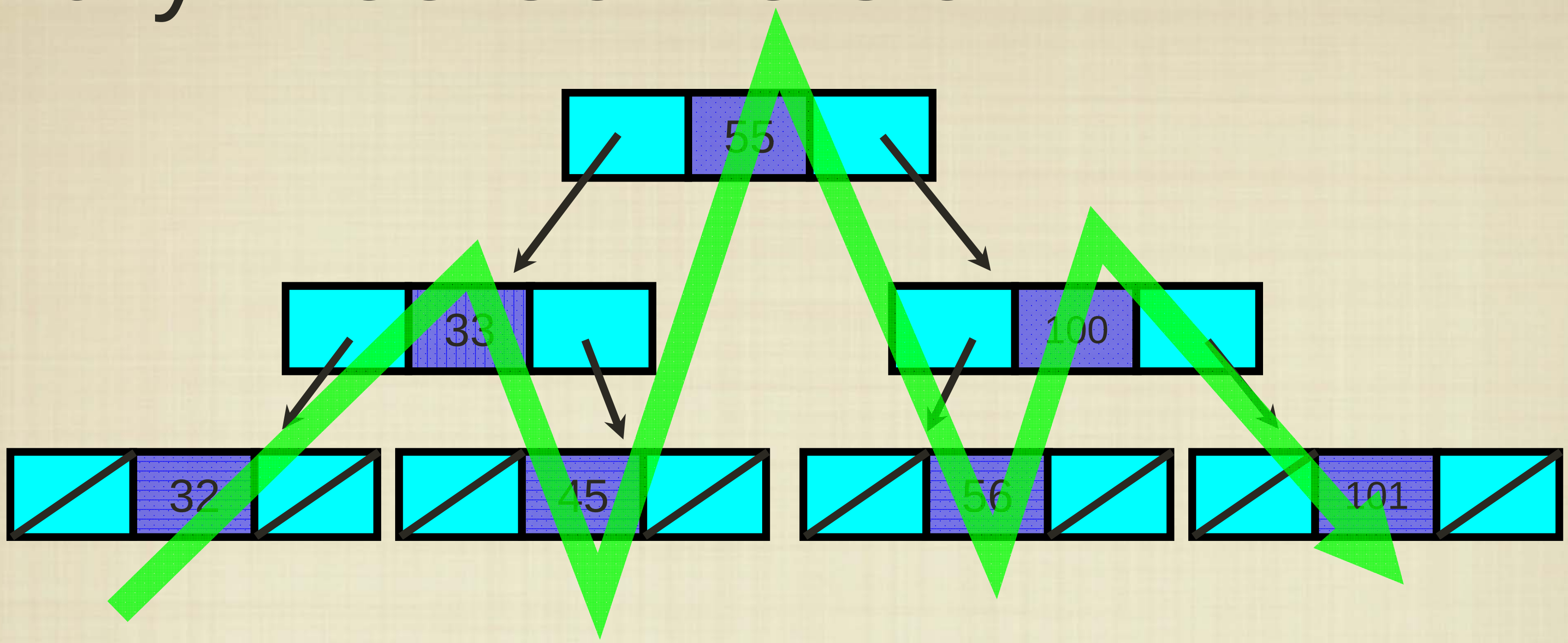
- Suppose I gave you a tree, how would you convert it into a sorted array?
- What if I wanted to save the tree to a file and reconstruct it exactly?

Binary Tree Conversion



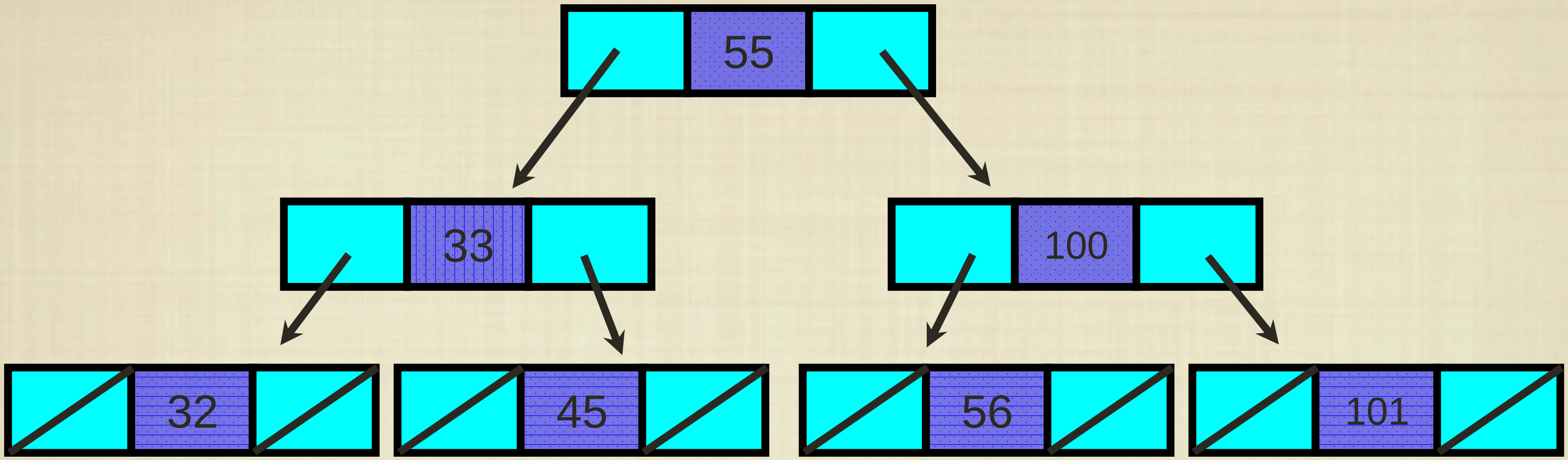
- Suppose I gave you a tree, how would you convert it into a sorted array?
- What if I wanted to save the tree to a file and reconstruct it exactly?

Binary Tree Conversion



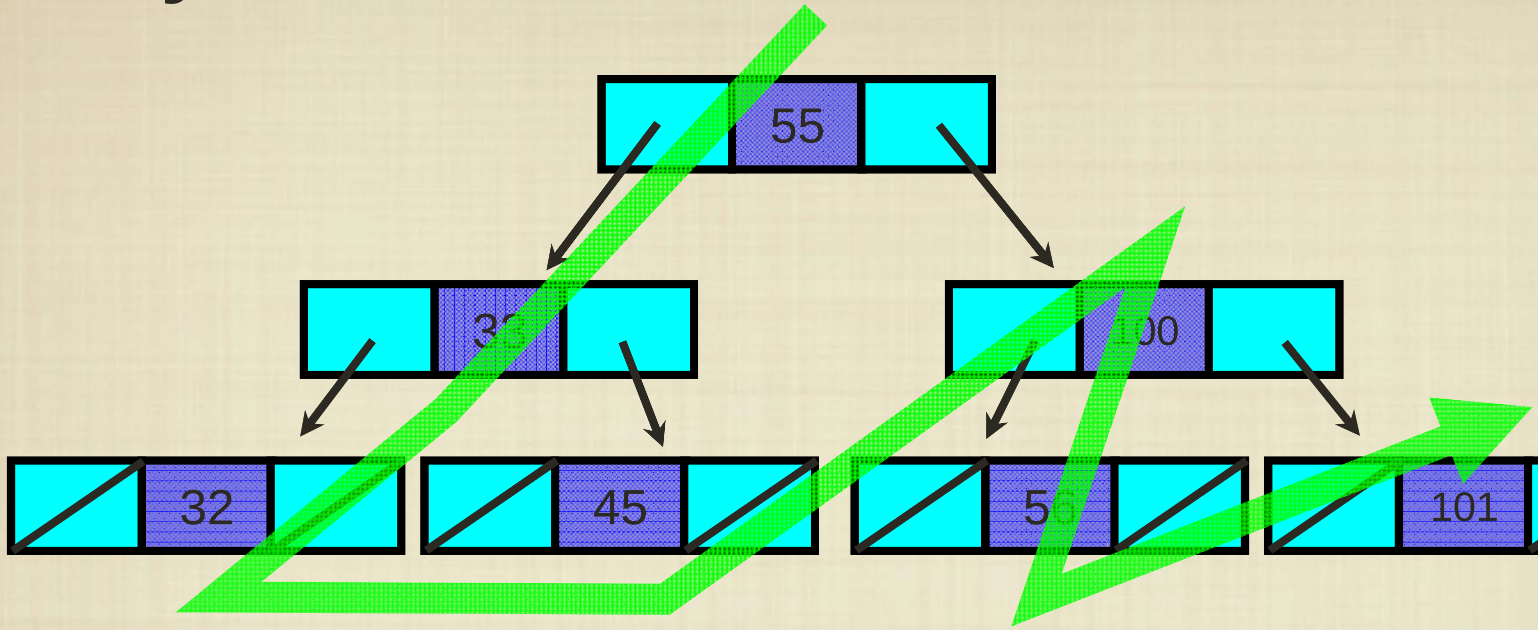
- An <u>in-order traversal</u> works by “visiting” the left subtree, then the current item, and then visiting the right subtree.

Binary Tree Conversion



An in-order traversal works by "visiting" the left subtree, then the current item, and then visiting the right subtree.

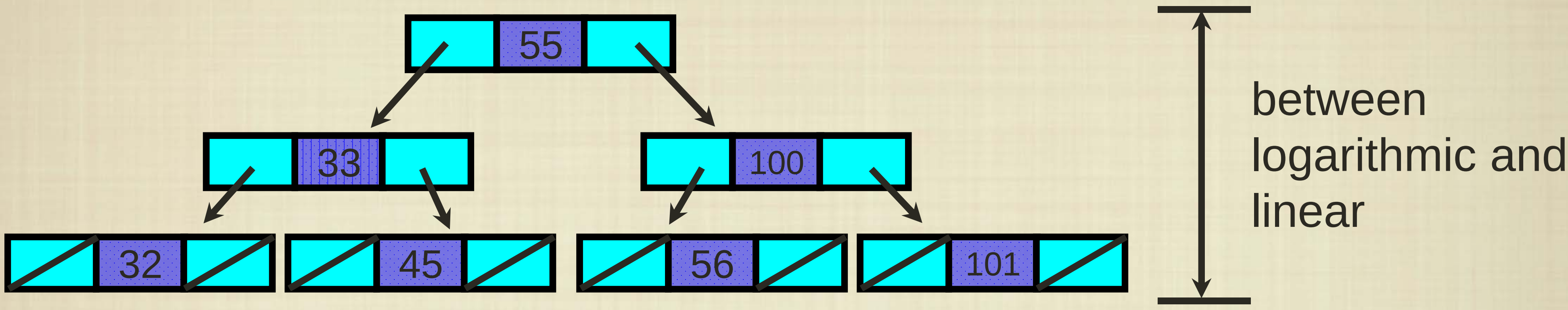
Binary Tree Conversion



An <u>in-order traversal</u> works by “visiting” the left subtree, then the current item, and then visiting the right subtree.

A <u>pre-order traversal</u> works by “visiting” the item we are at, then the left subtree, and then the right subtree.

Summary of Binary Search Trees



- The time to perform operations in binary search trees is highly dependent on how they are built.
- The best-case height of a binary tree is logarithmic in the number of elements; there are sophisticated techniques (AVL, red-black) for ensuring this height is logarithmic in the number of elements in the worst-case.
- Do tree data structures always have to be binary? Are they always used to add/remove/find elements in a collection?