LECTURE 20

BINARY SEARCH TREES

MCS 275 Spring 2022
Emily Dumas

LECTURE 20: BINARY SEARCH TREES

Course bulletins:

- Project 2 due 6pm today

SAMPLE CODE

I've created a new directory trees in the course sample code repository.

Live coding examples from the next couple of lectures will be added there.

GOAL

Learn about **search** and **insert** operations on binary search trees.

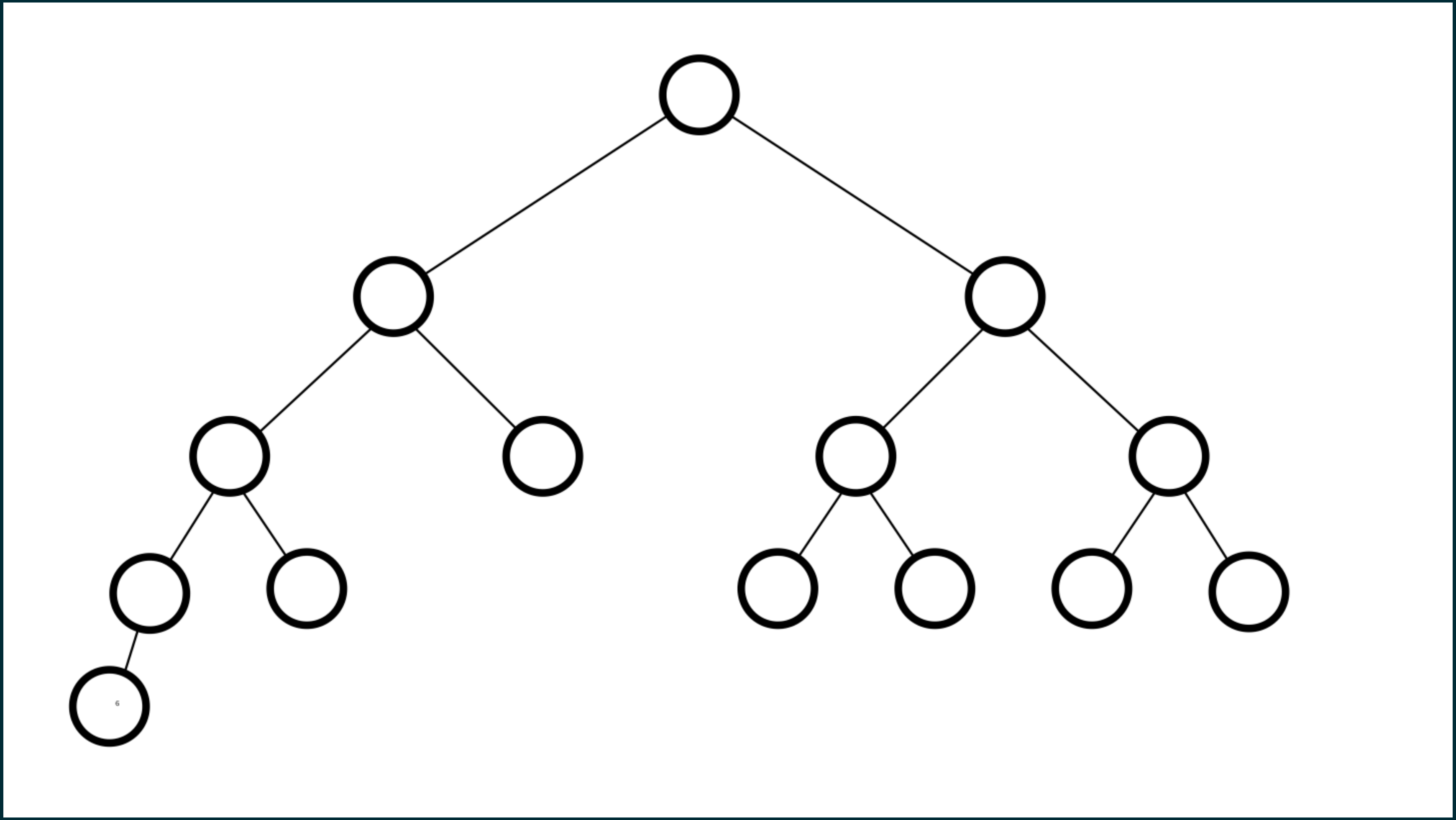
Explore an application to a fast data structure for storing a set of integers.

BINARY SEARCH TREE (BST)

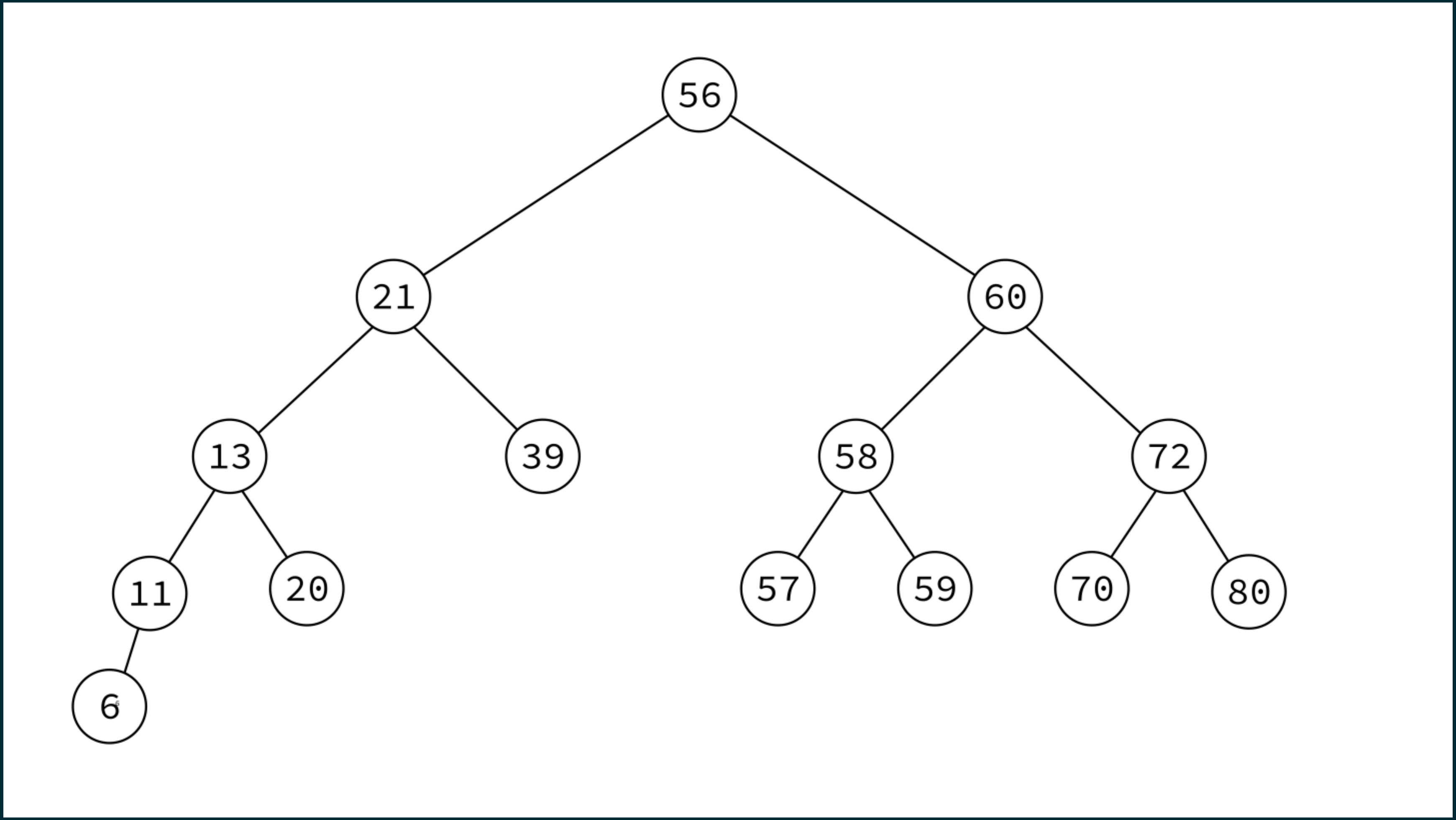
A binary tree in which:

- Nodes have **keys** that can be compared
- The key of a node is greater than or equal to any key in its left subtree.
- The key of a node is less than or equal to any key in its right subtree.

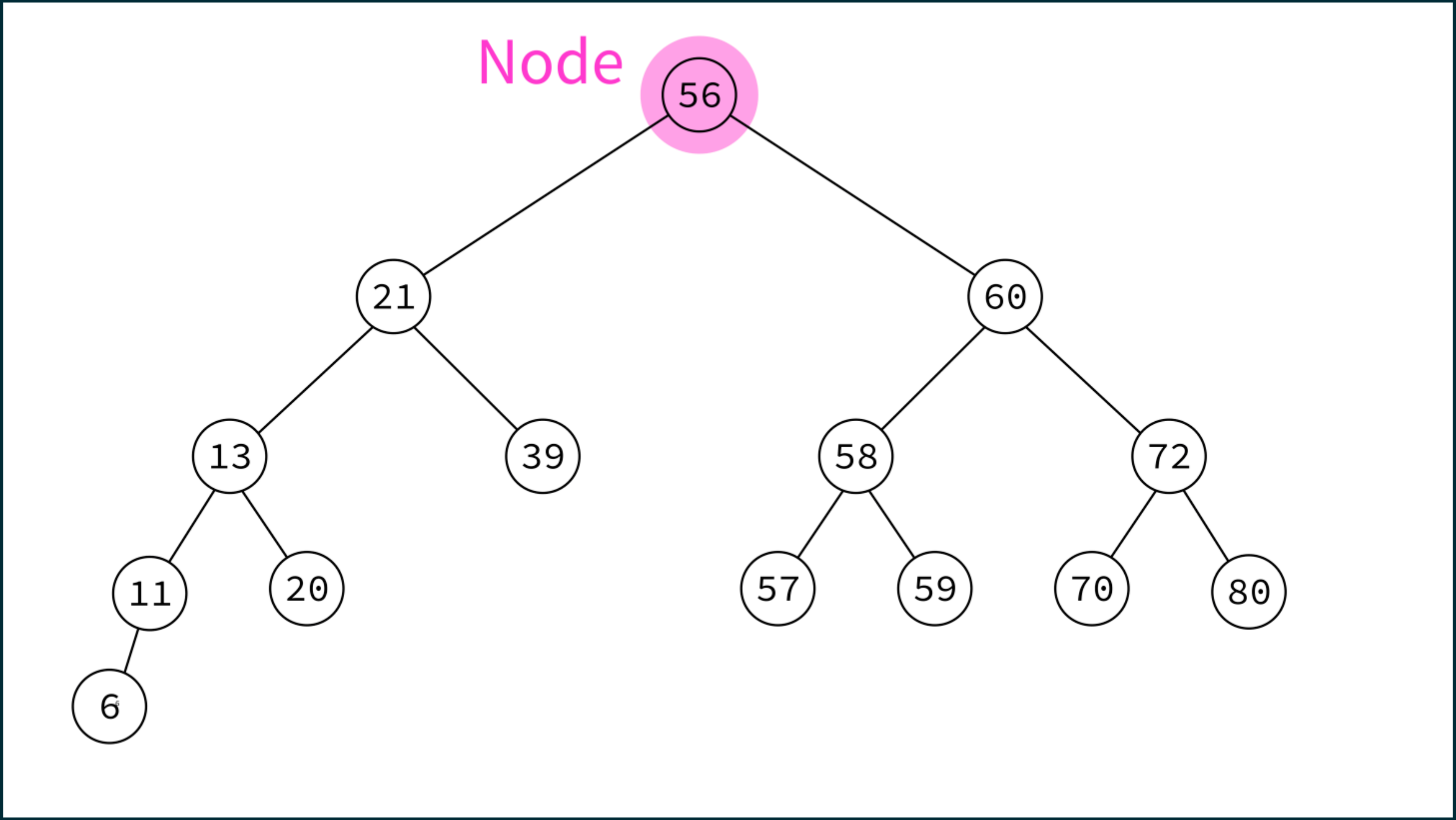
BINARY TREE



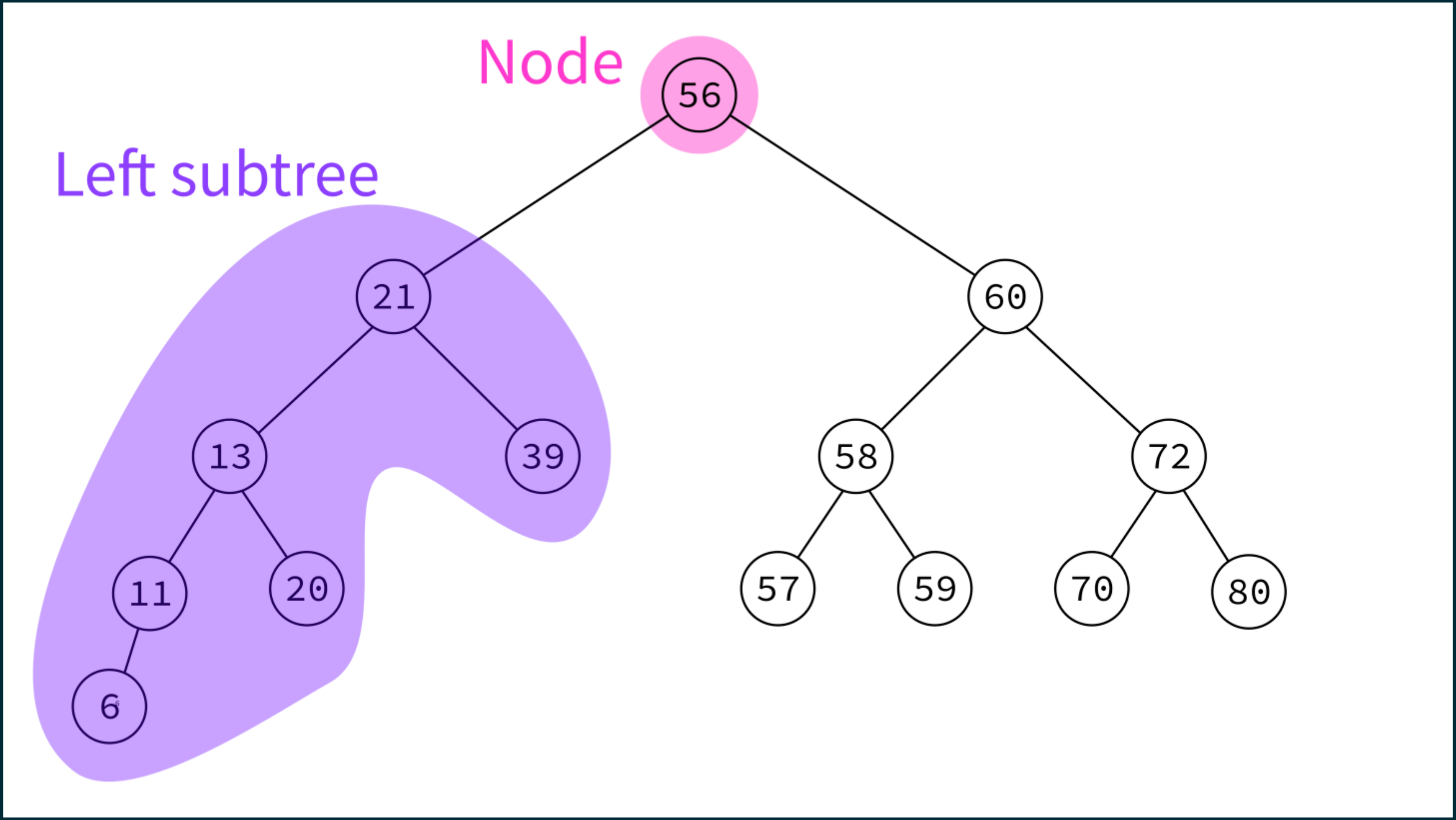
BST



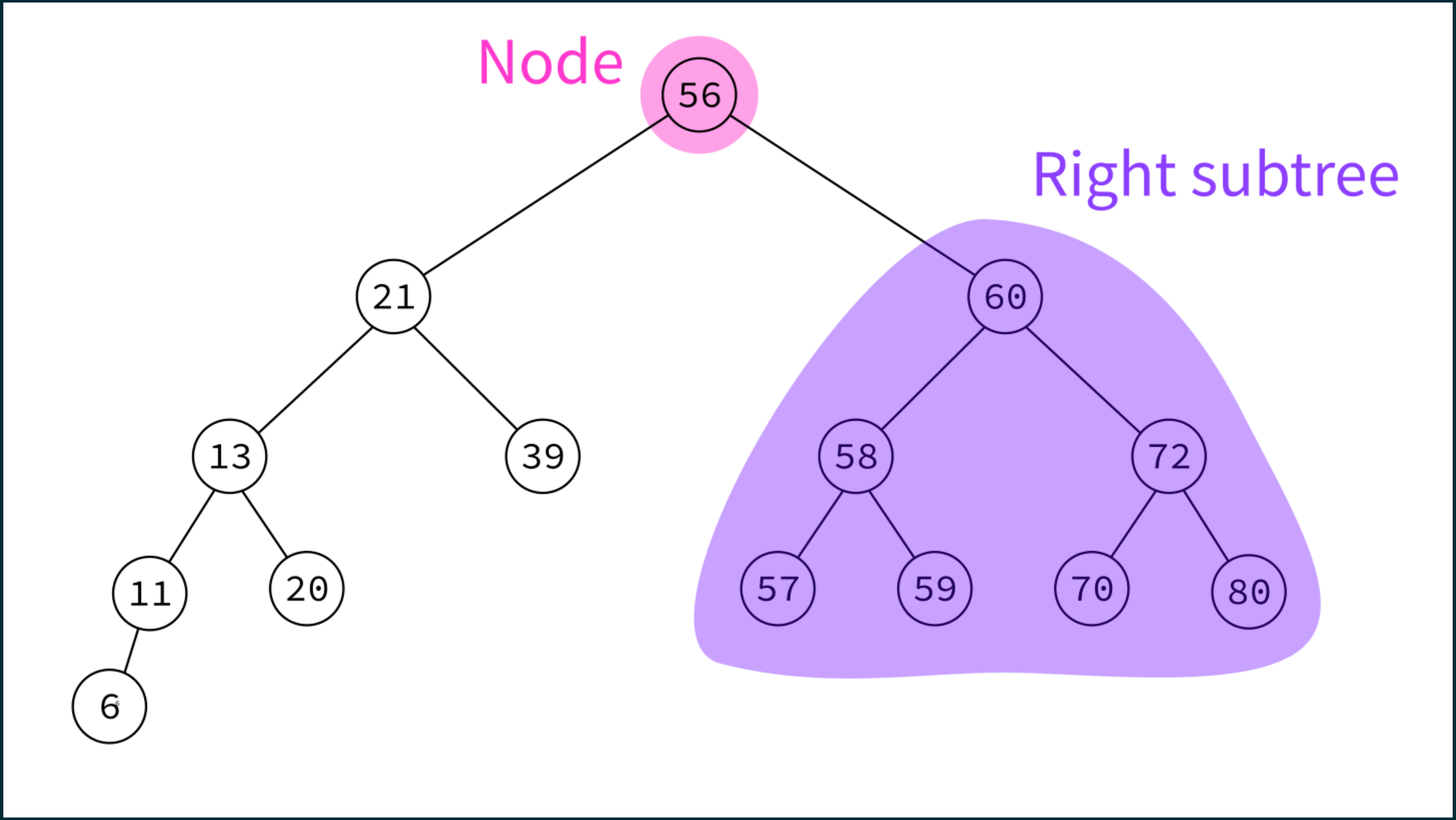
BST



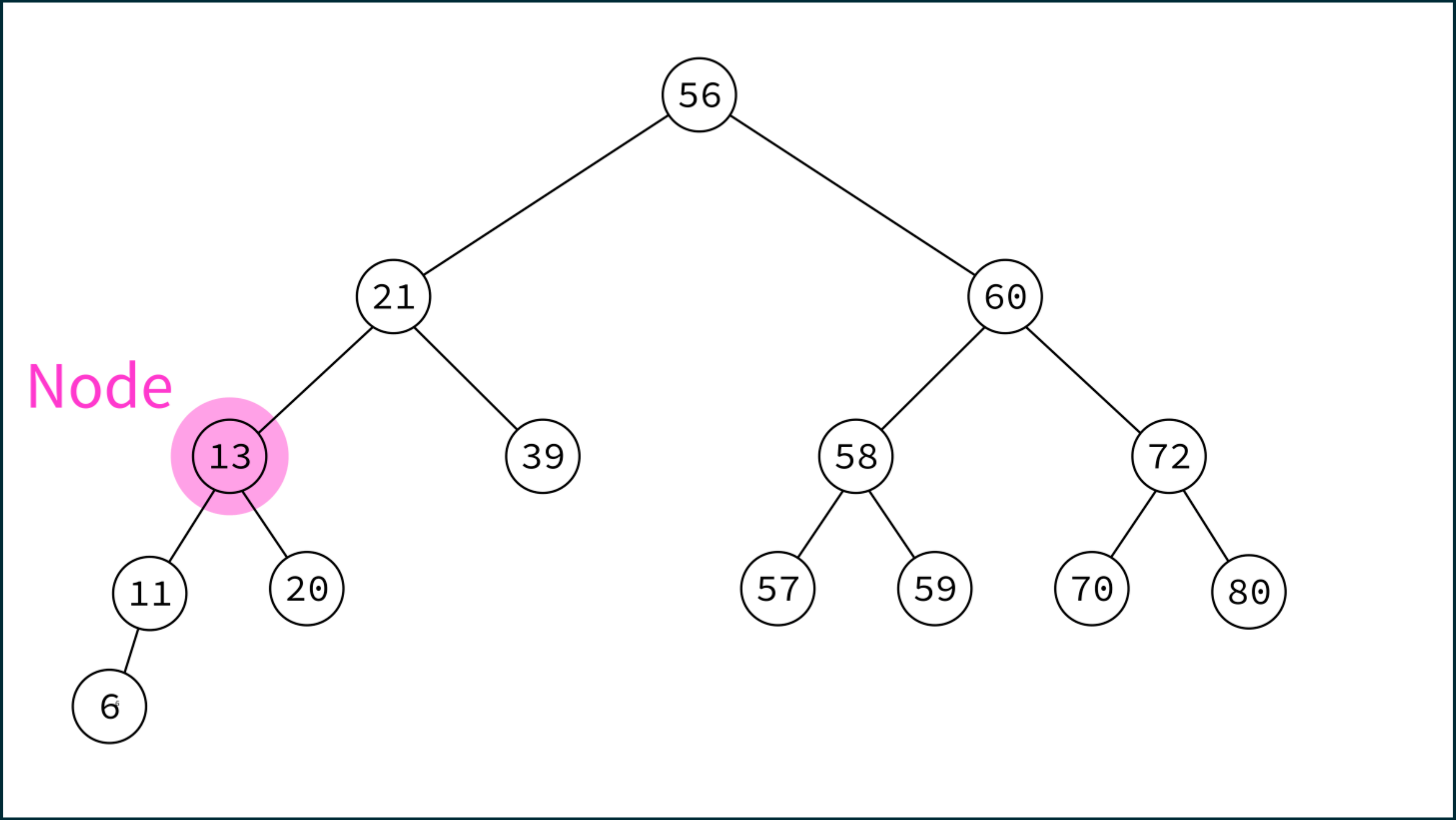
BST



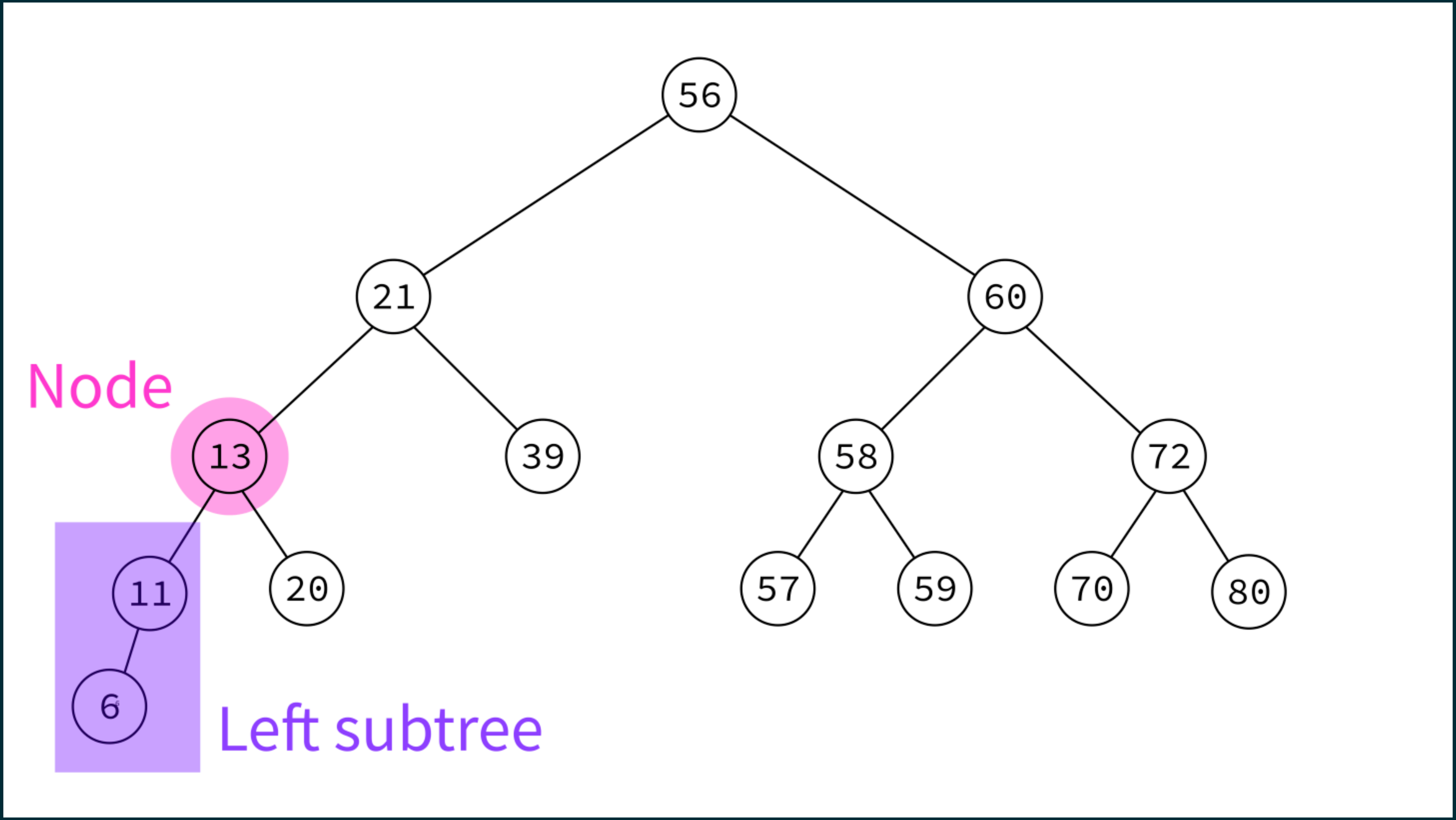
BST



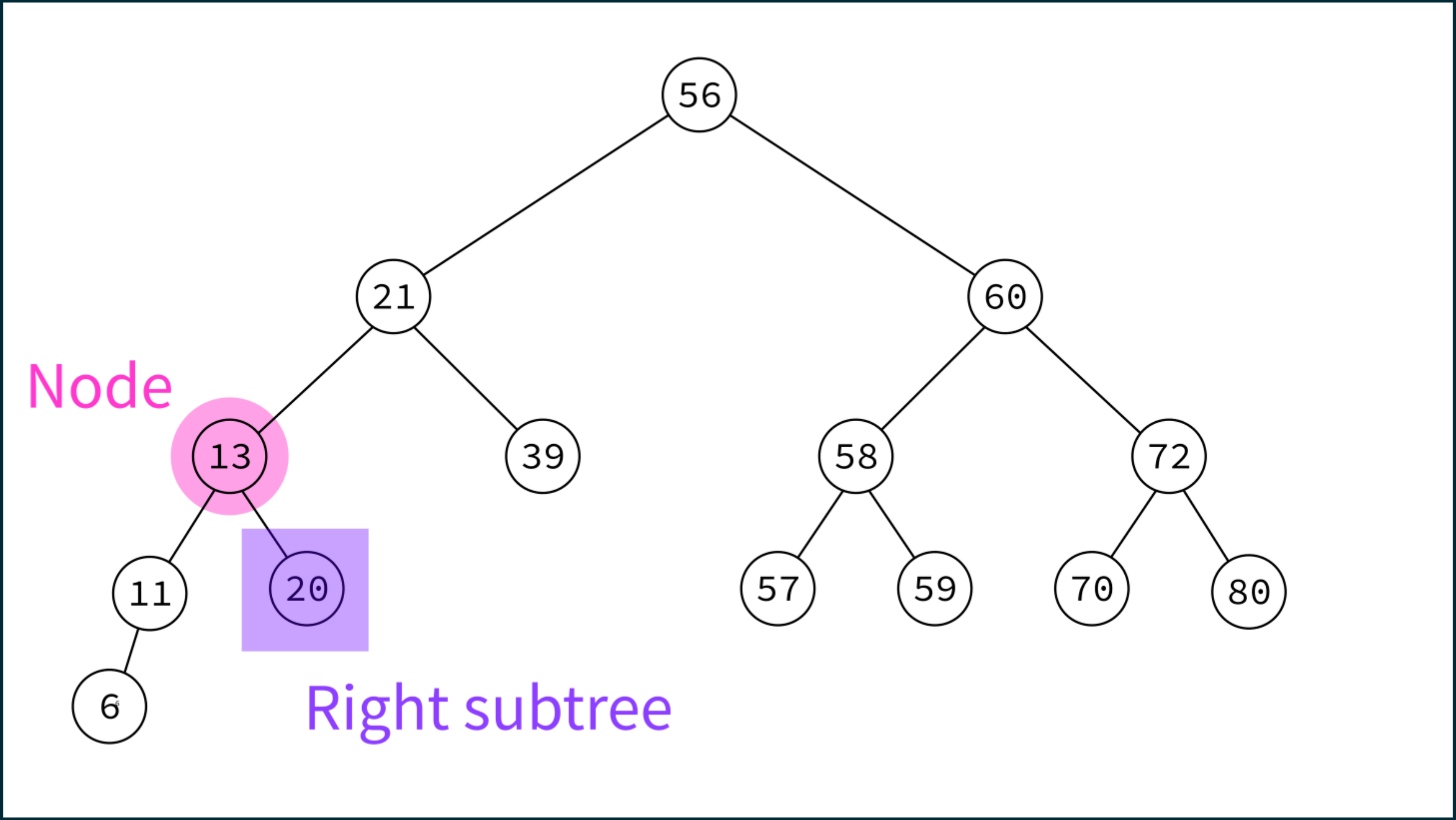
BST



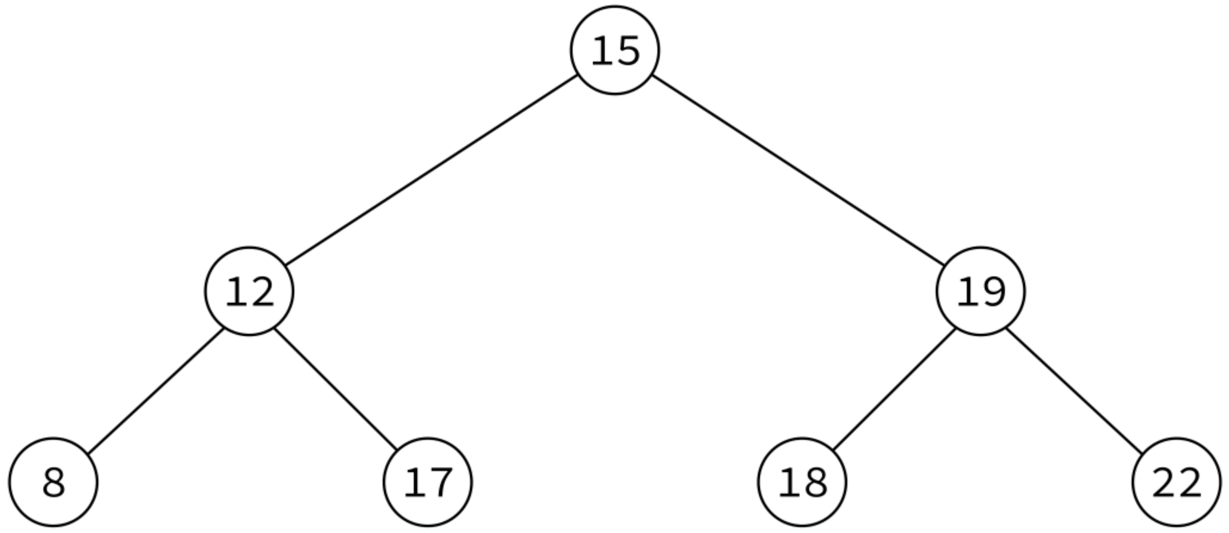
BST



BST

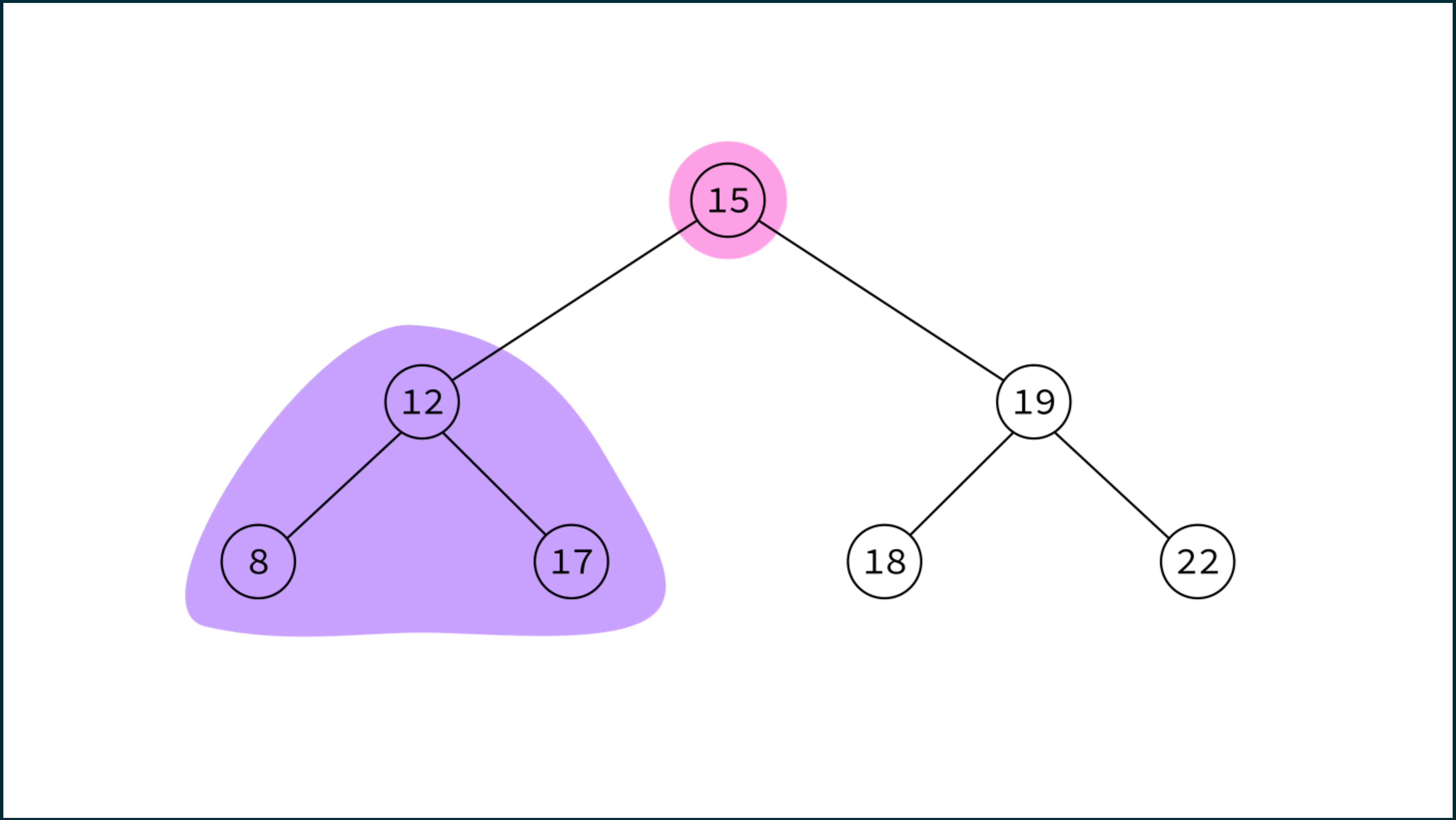


NOT A BST



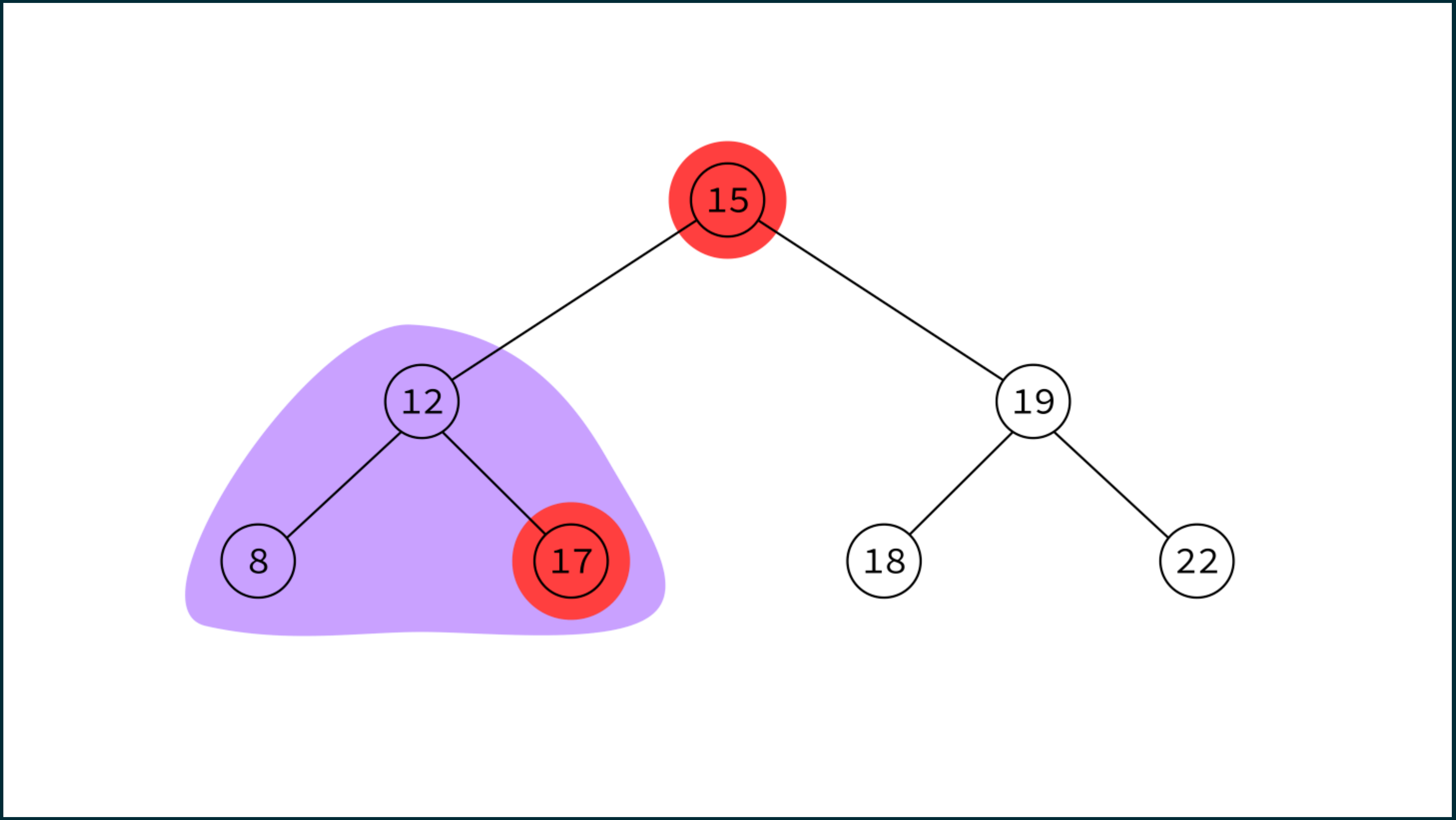
This "just" is a binary tree with keys.

NOT A BST



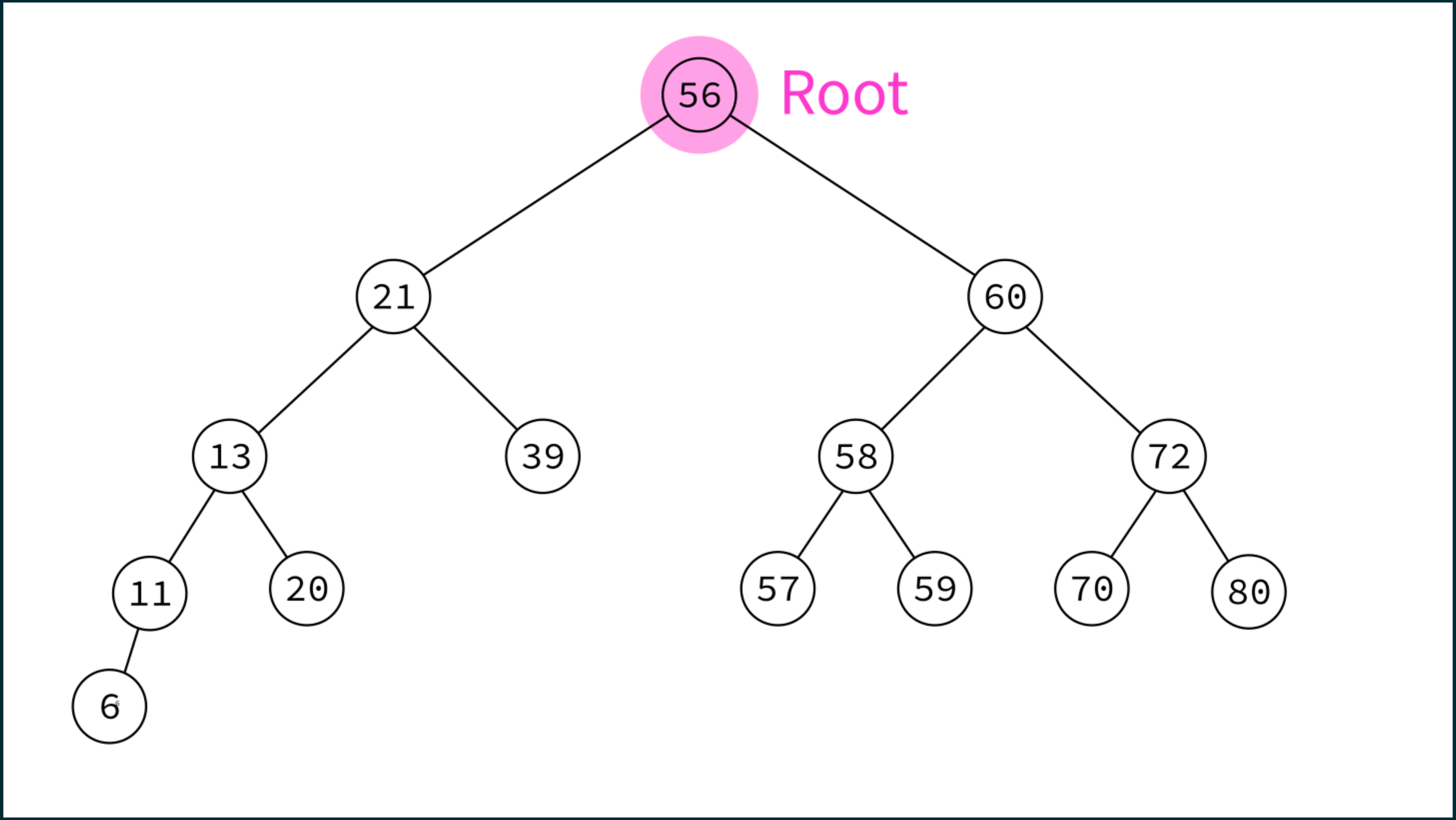
This "just" is a binary tree with keys.

NOT A BST

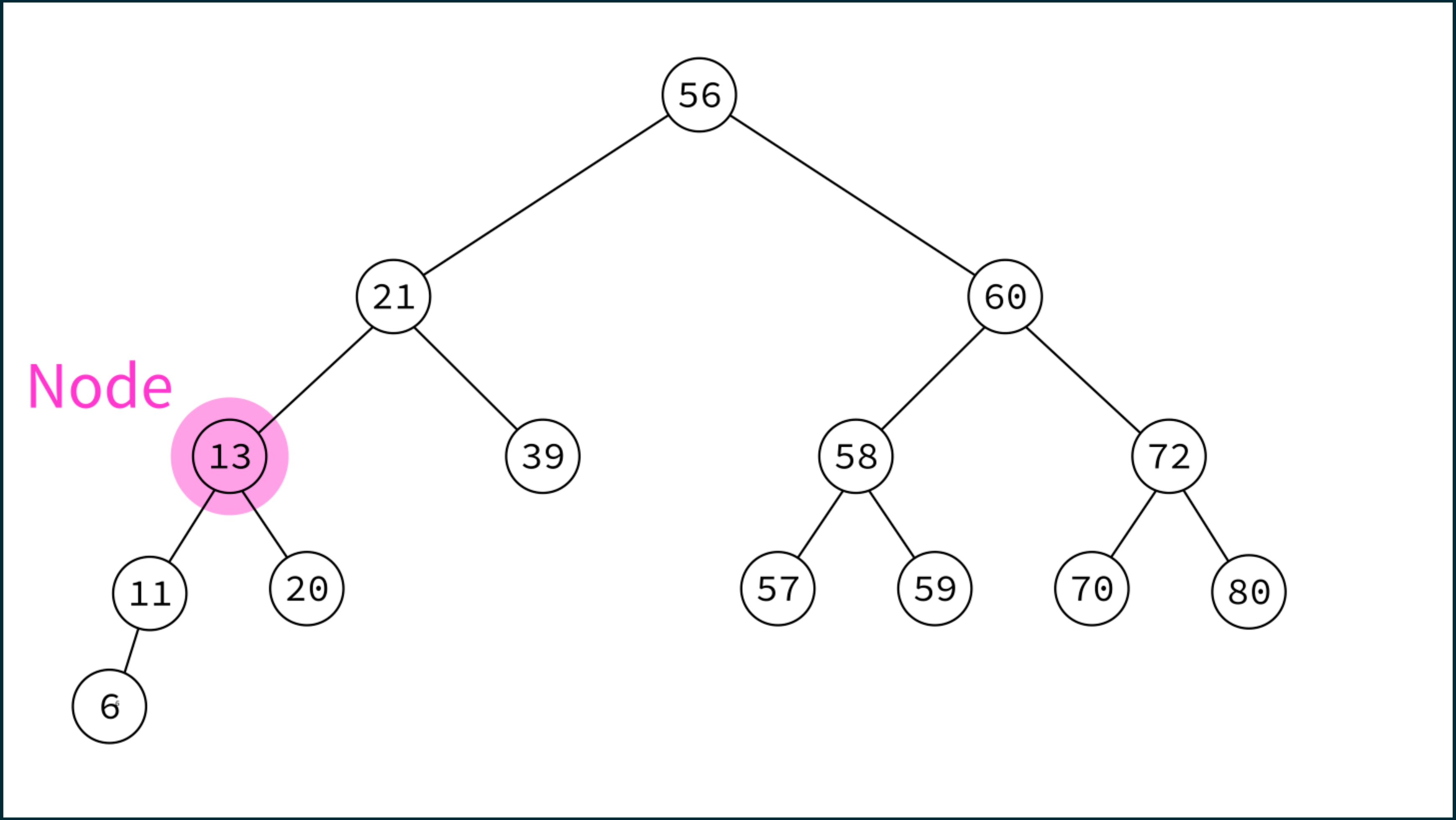


This "just" is a binary tree with keys.

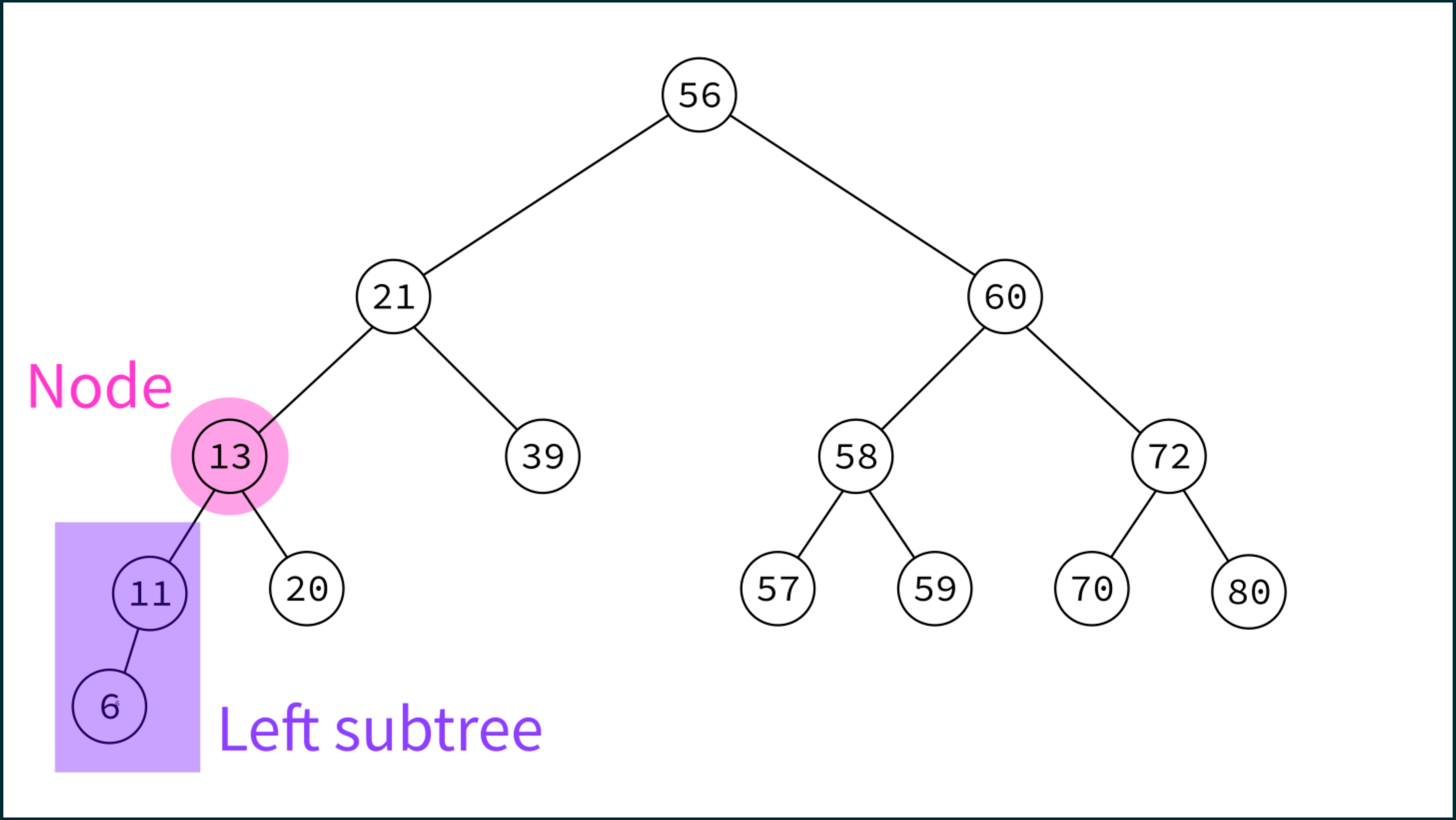
TREE TERMS



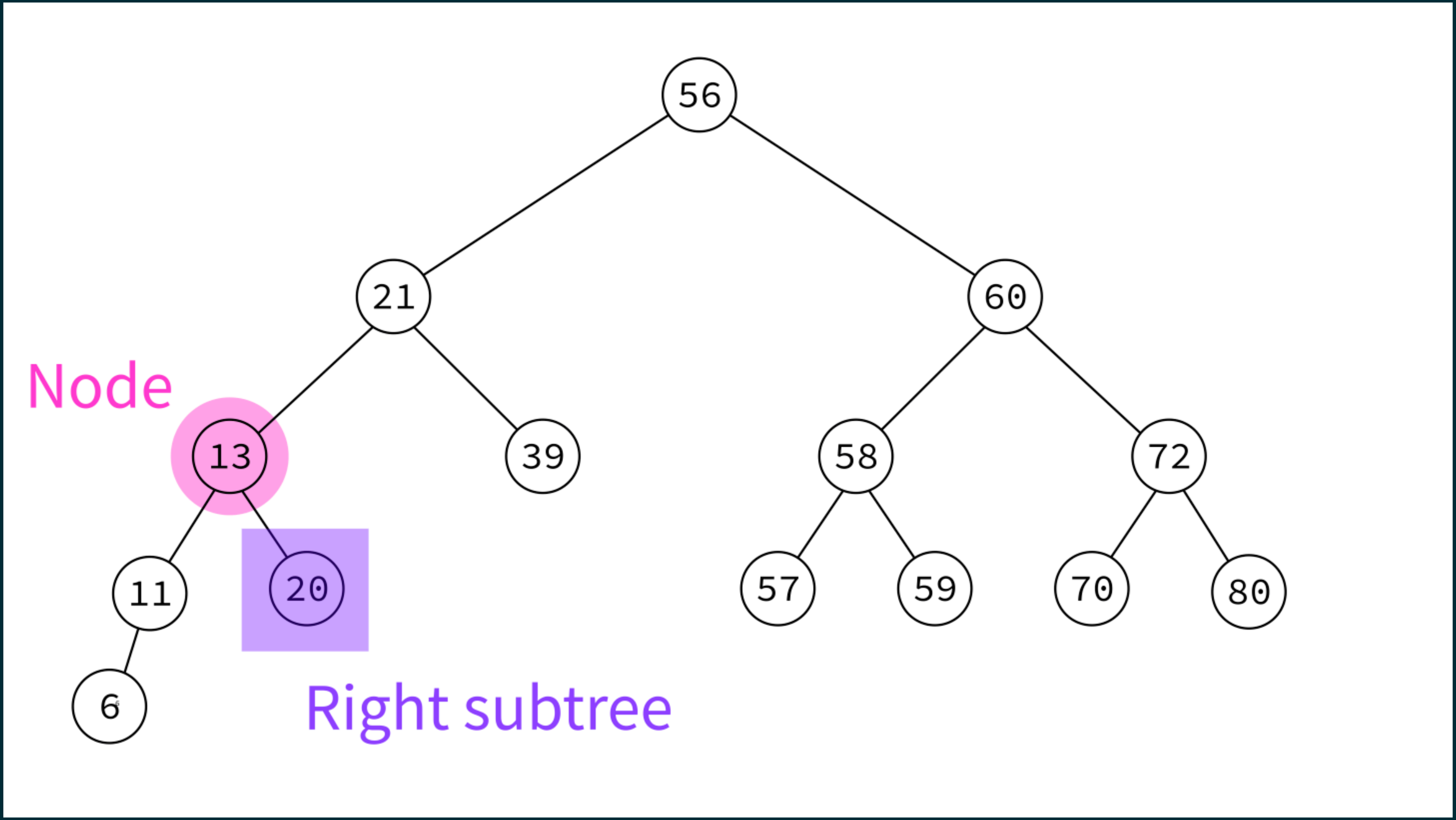
TREE TERMS



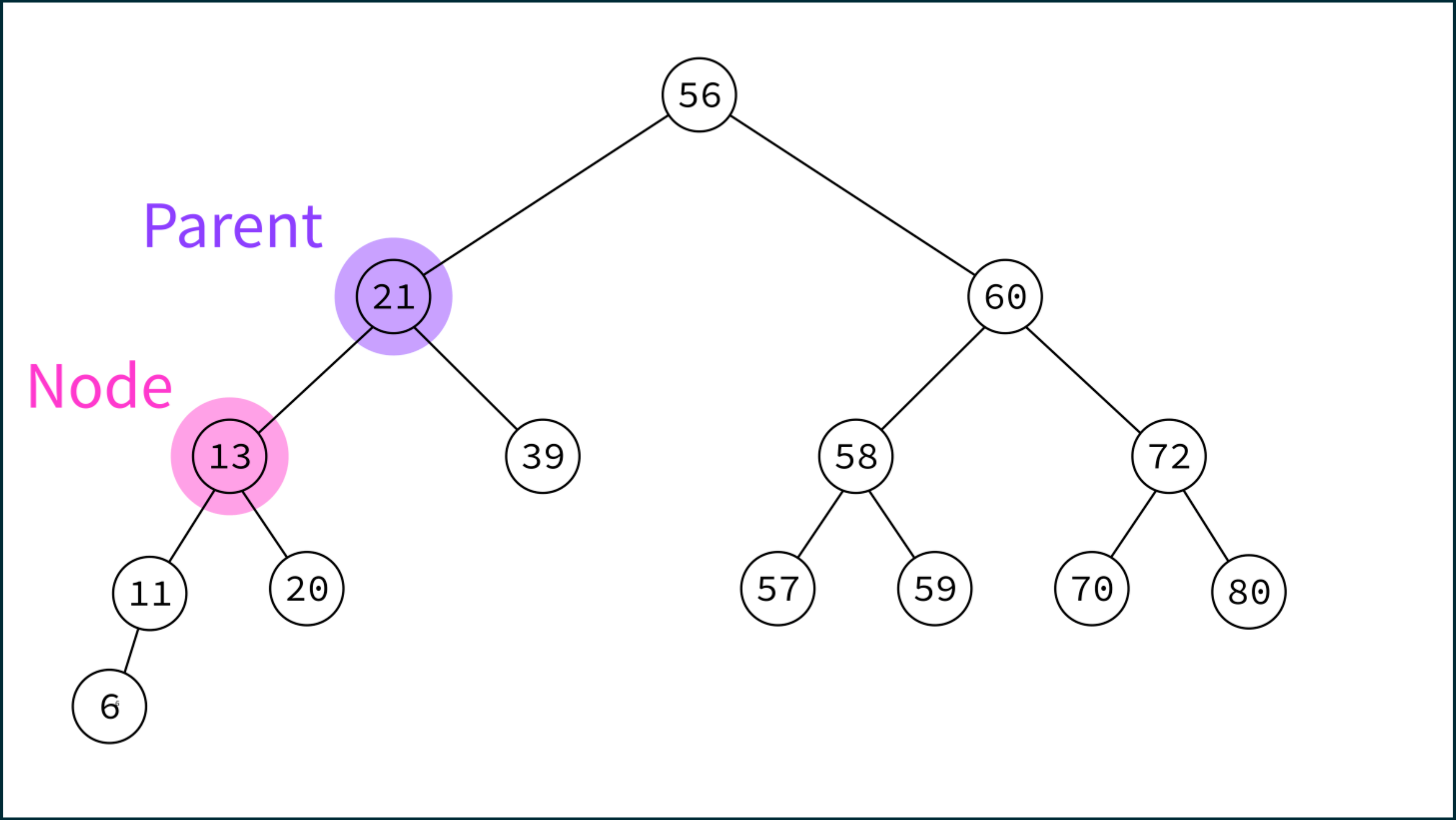
TREE TERMS



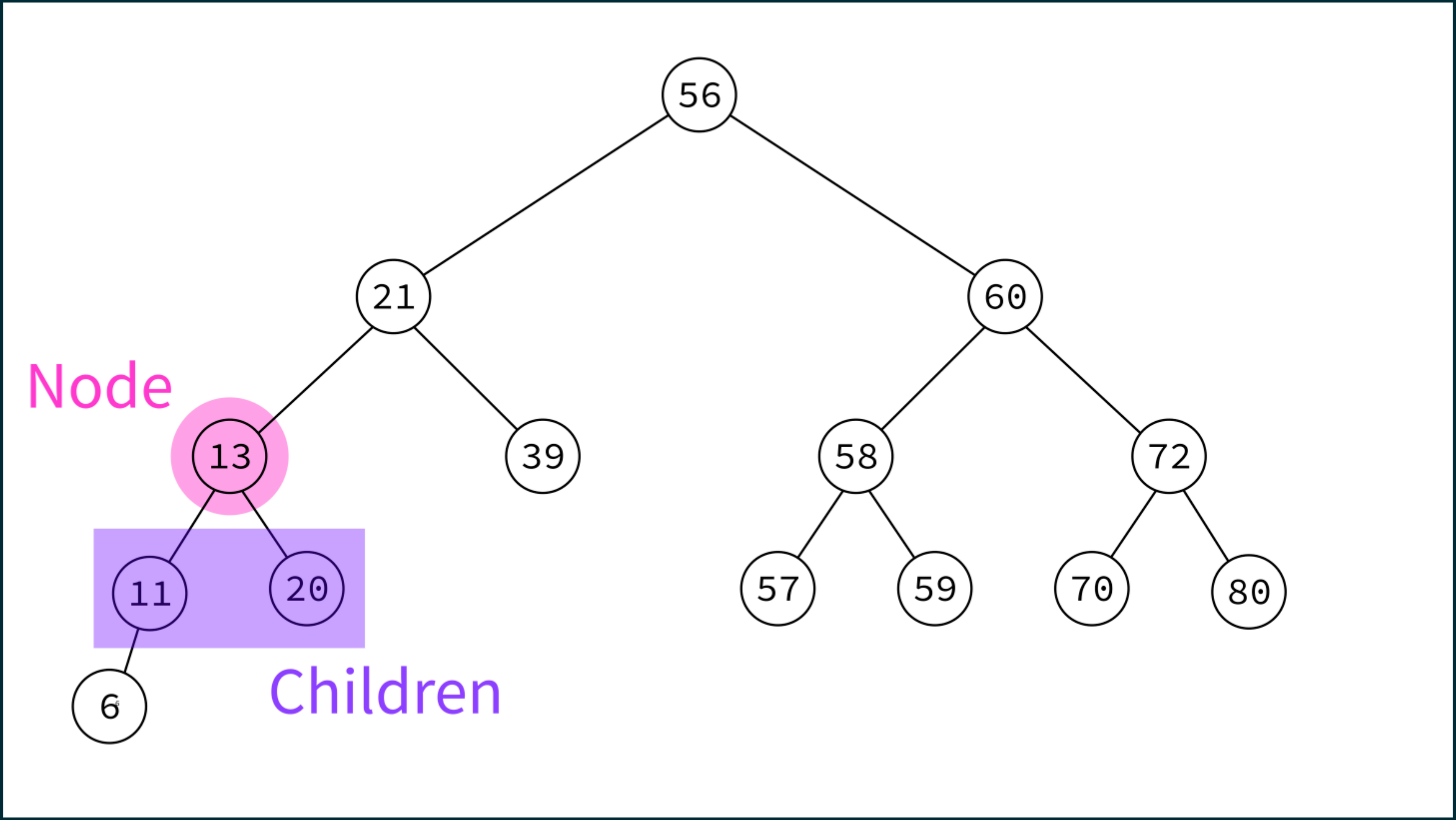
TREE TERMS



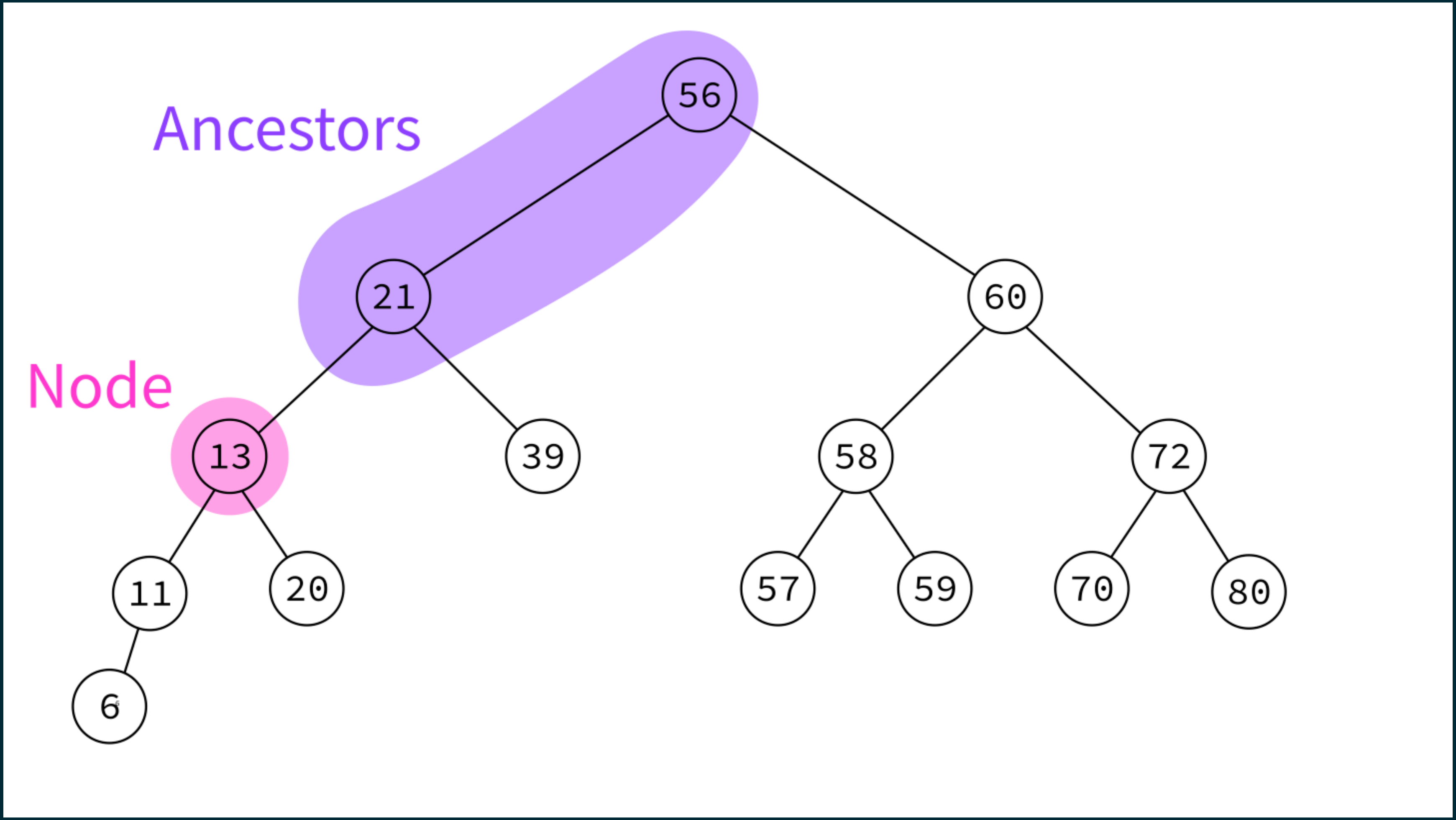
TREE TERMS



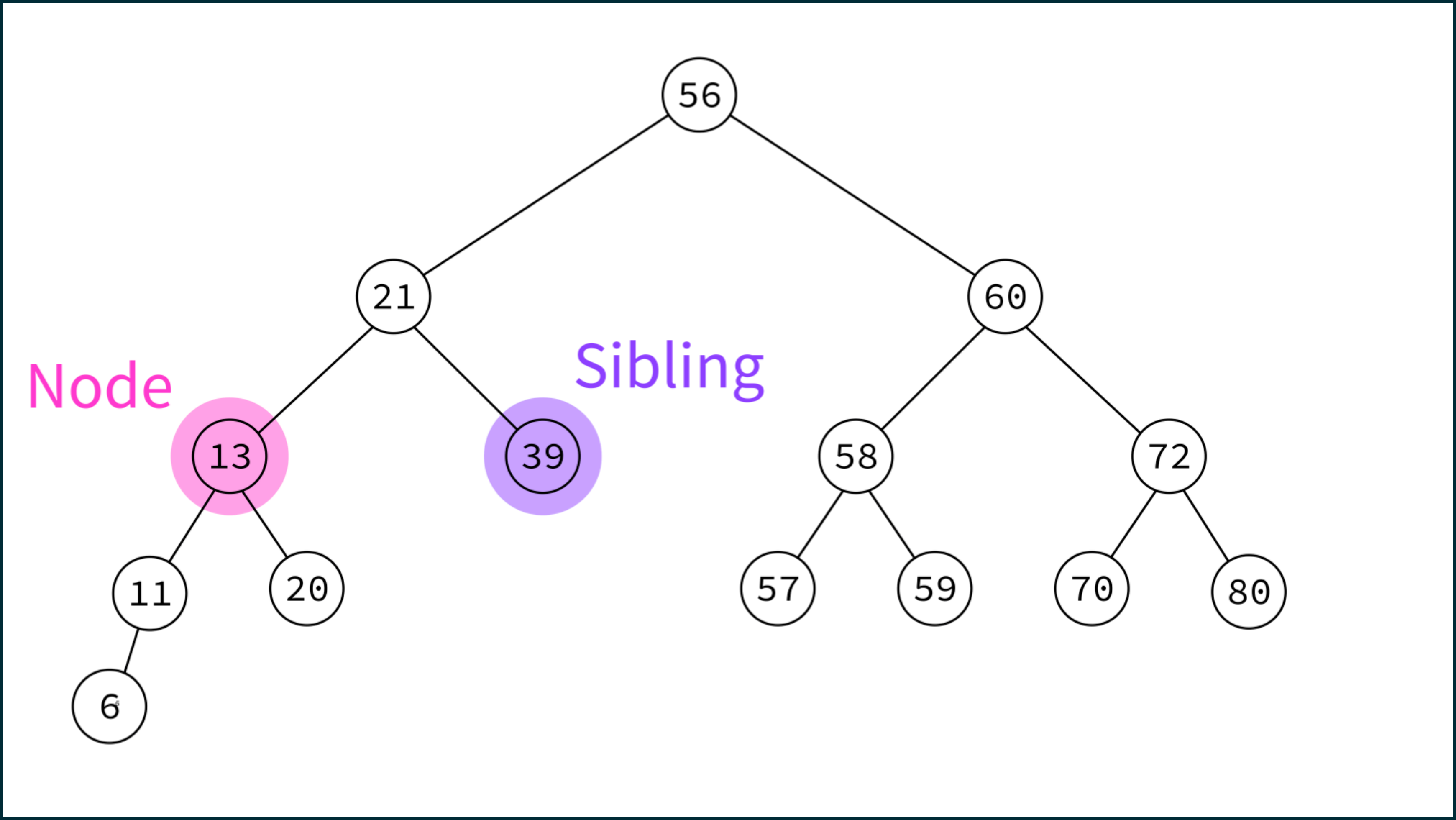
TREE TERMS



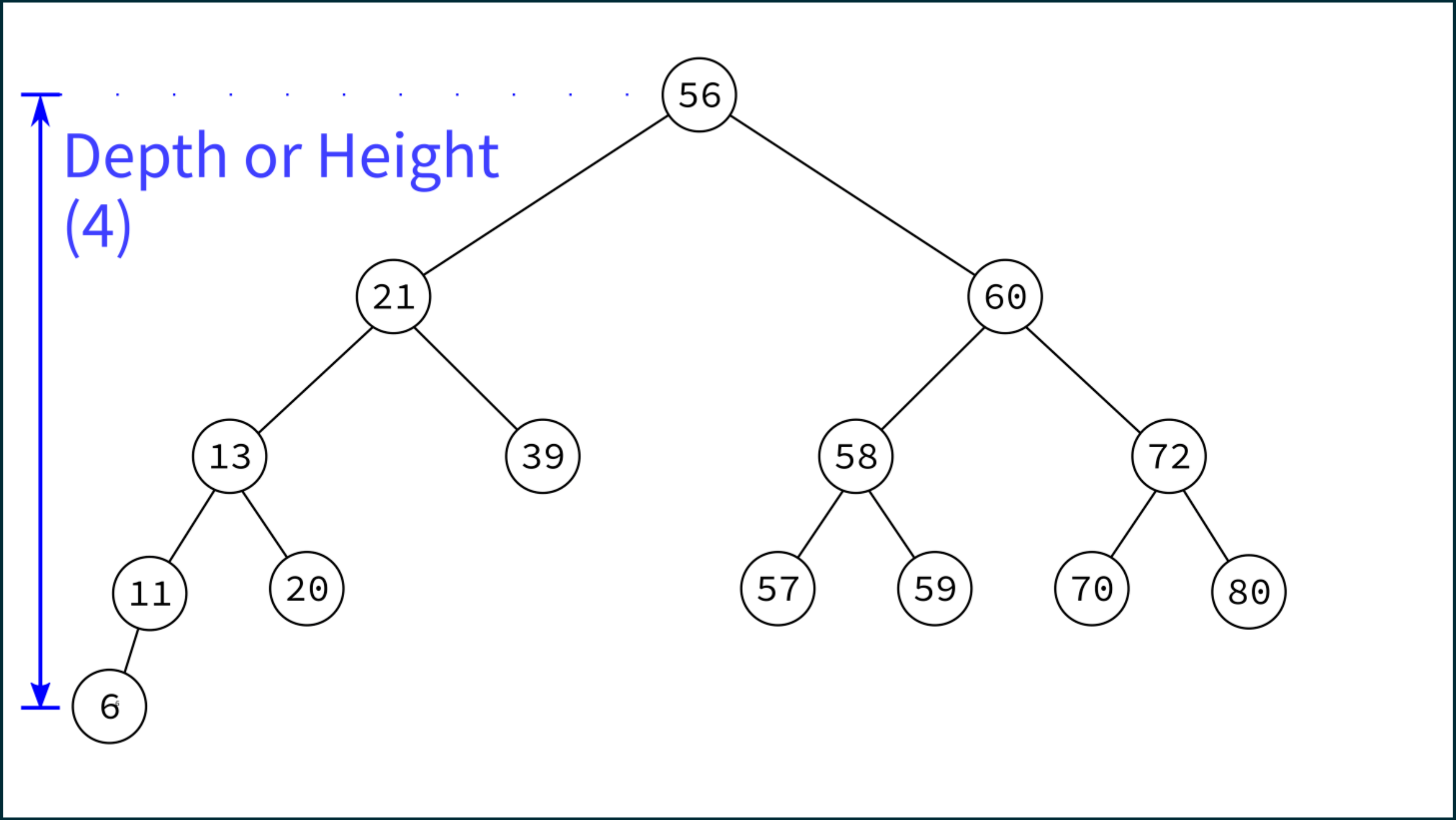
TREE TERMS



TREE TERMS



TREE TERMS



CODING

Let's build a class to represent nodes of a binary tree that also store keys.

TREEVIS

I provide a module `treevis` in the sample code repository that can "pretty print" a tree with the function `treeprint(root)`.

Challenge: Read the source of `treevis` and figure out how it works!

FROM TREE TO BST

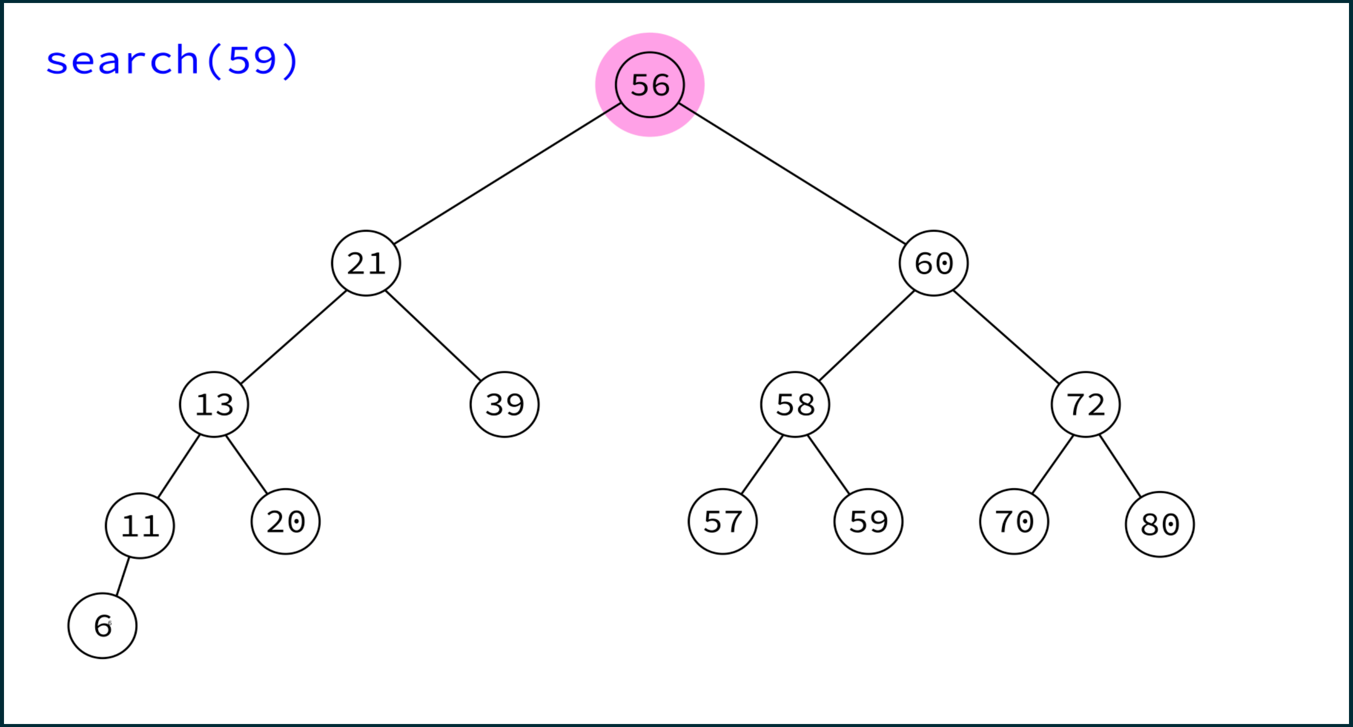
Now let's build a subclass of `Node` to represent a BST.

Desired features:

- Insert nodes (maintaining BST property)
- Search for nodes by key

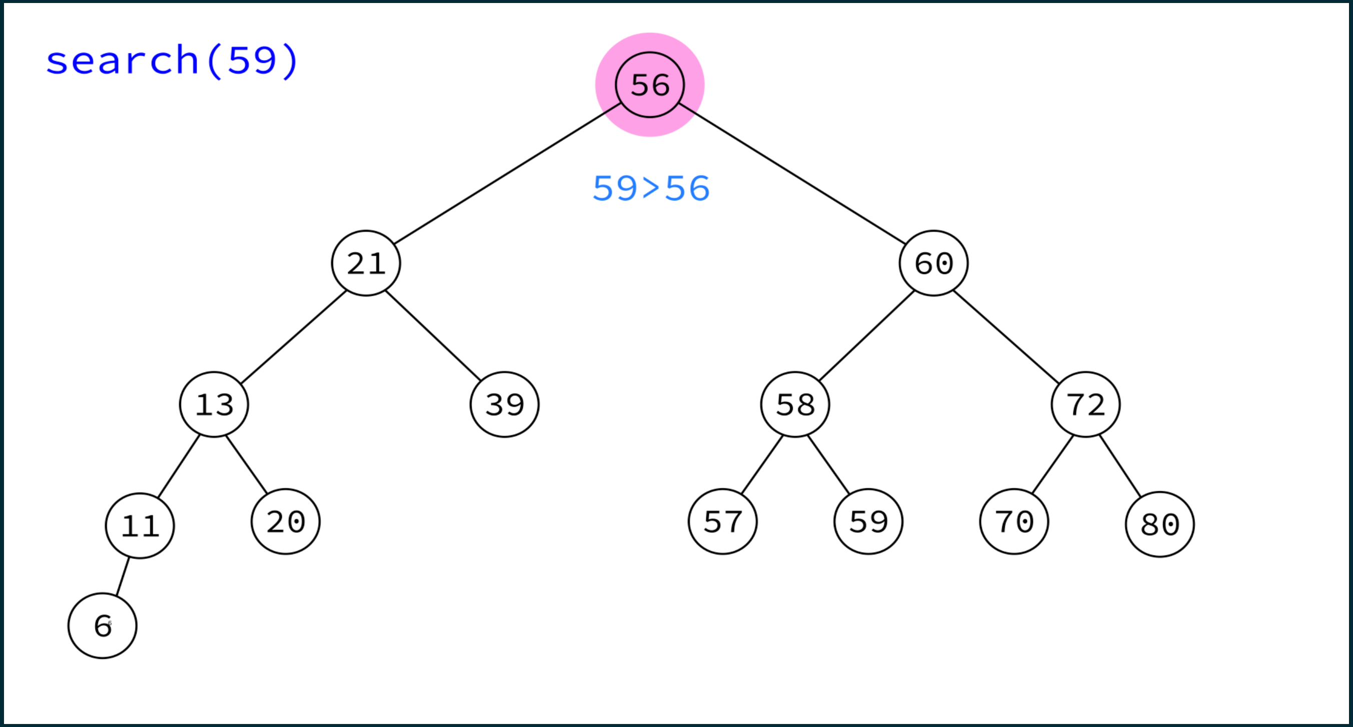
SEARCH

Given `x`, find and return a node with key `x`. Return `None` if no such node exists.



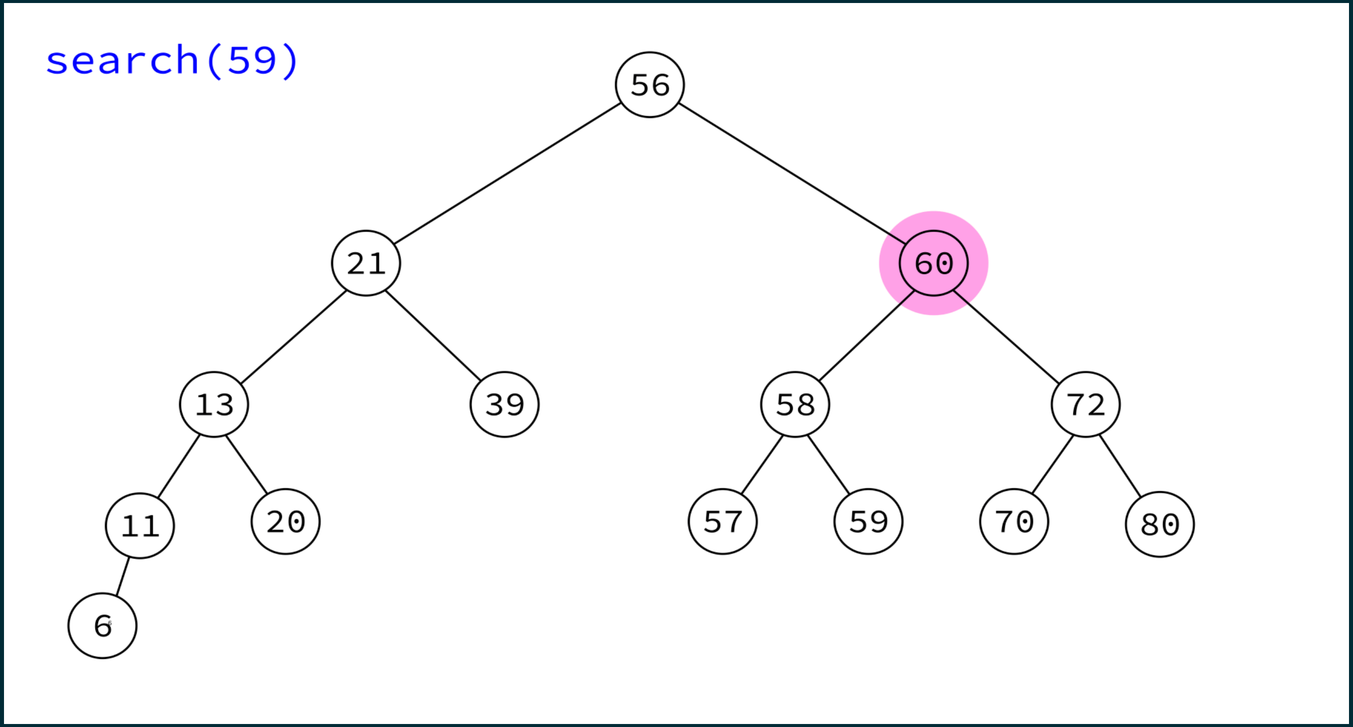
SEARCH

Given `x`, find and return a node with key `x`. Return `None` if no such node exists.



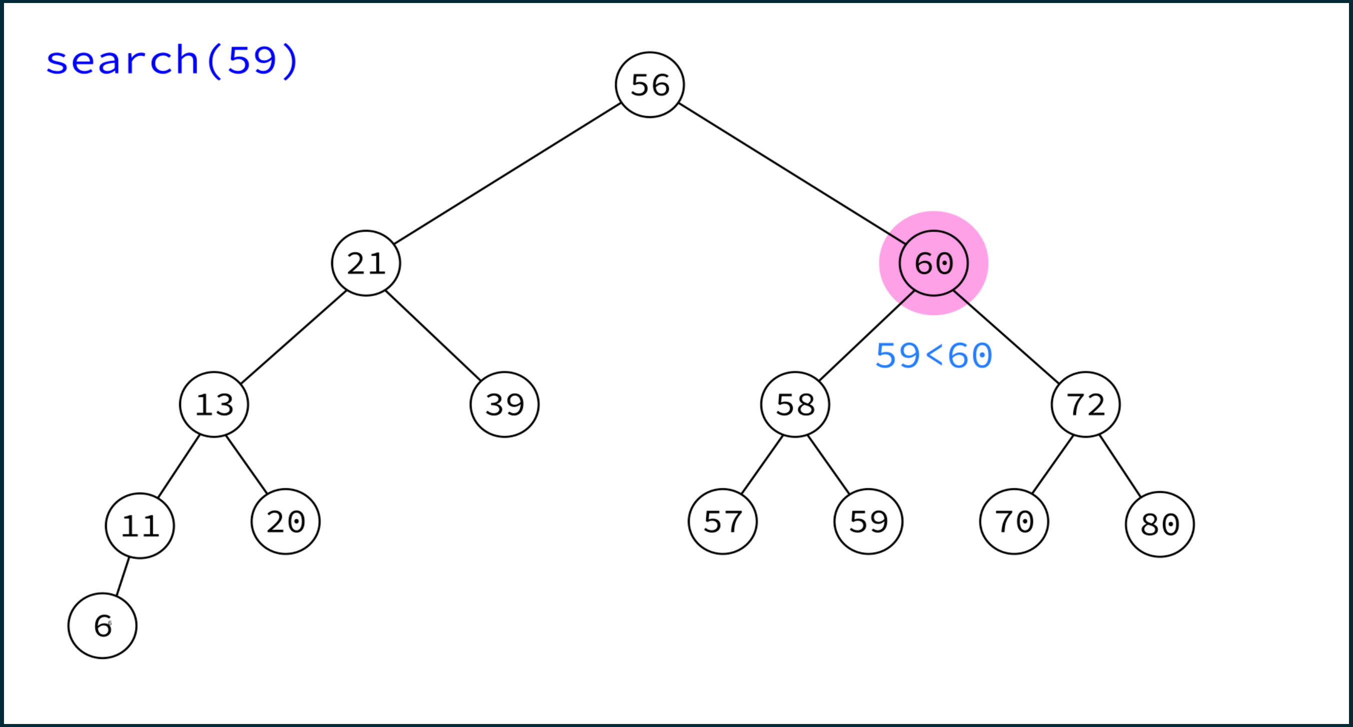
SEARCH

Given `x`, find and return a node with key `x`. Return `None` if no such node exists.



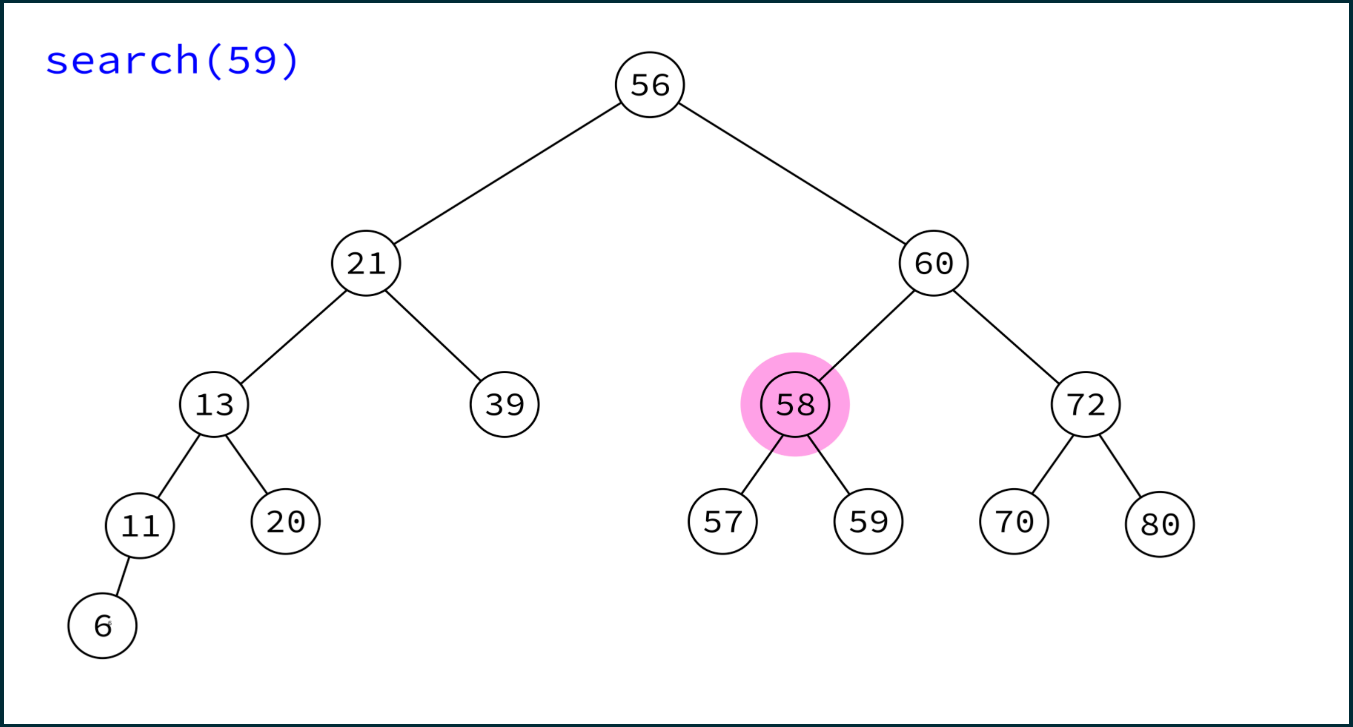
SEARCH

Given `x`, find and return a node with key `x`. Return `None` if no such node exists.



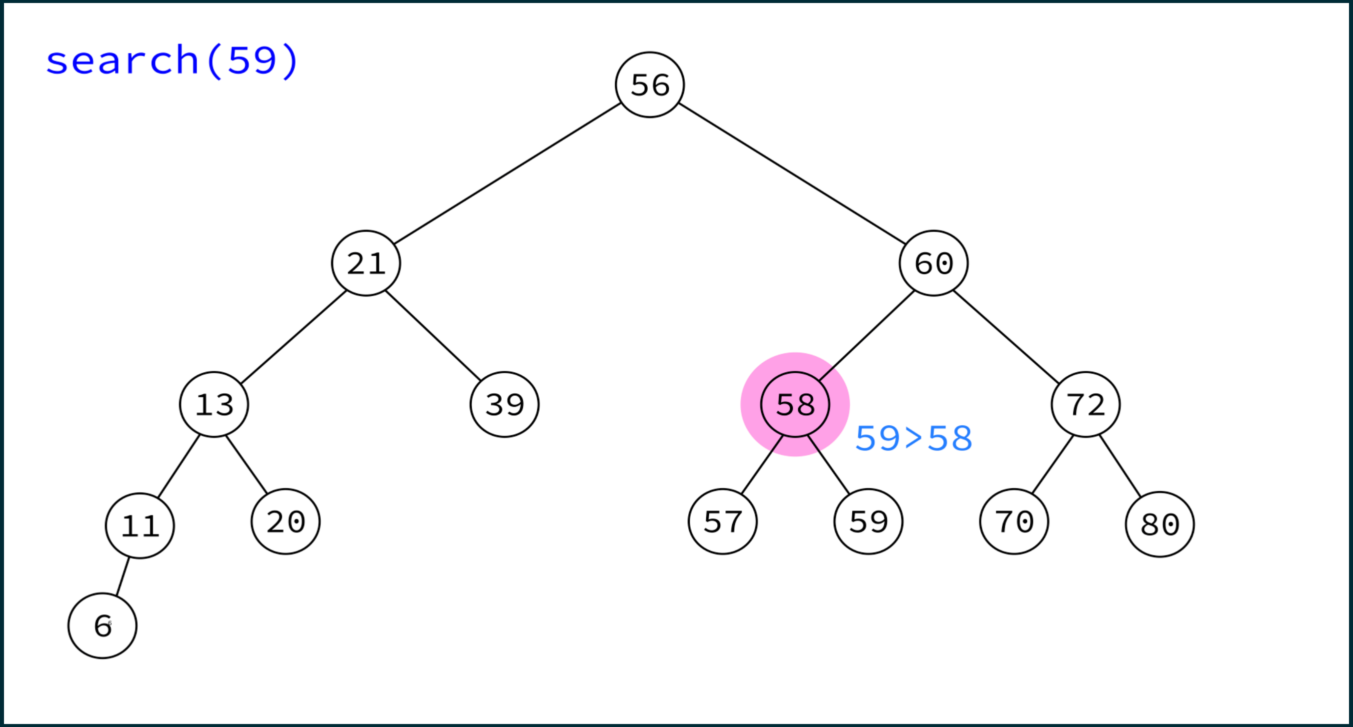
SEARCH

Given `x`, find and return a node with key `x`. Return `None` if no such node exists.



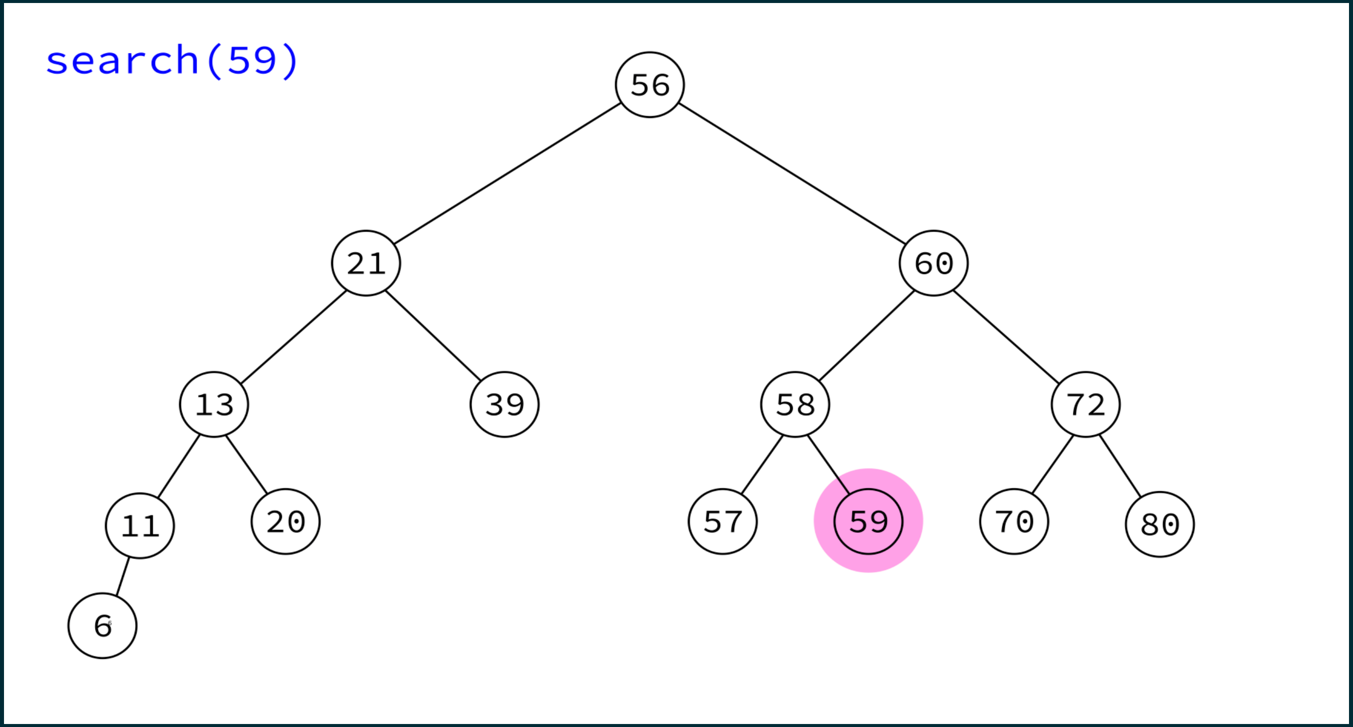
SEARCH

Given `x`, find and return a node with key `x`. Return `None` if no such node exists.



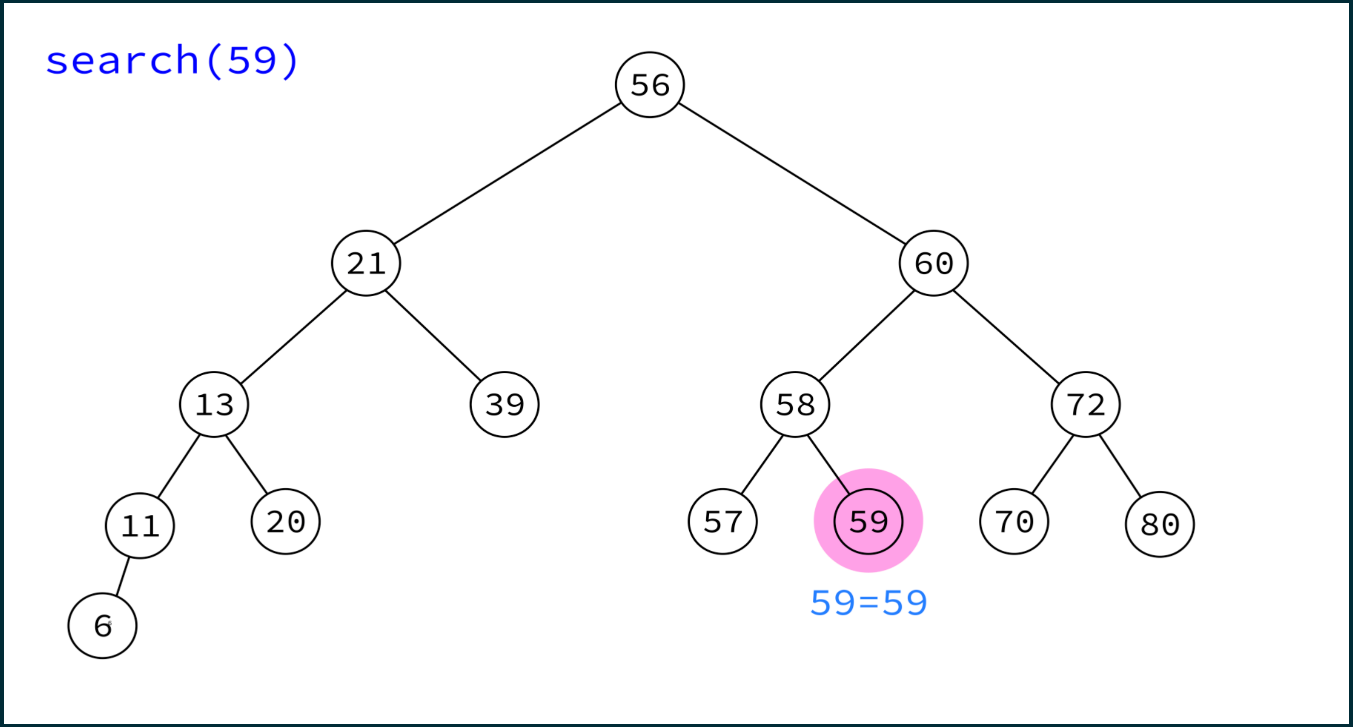
SEARCH

Given `x`, find and return a node with key `x`. Return `None` if no such node exists.



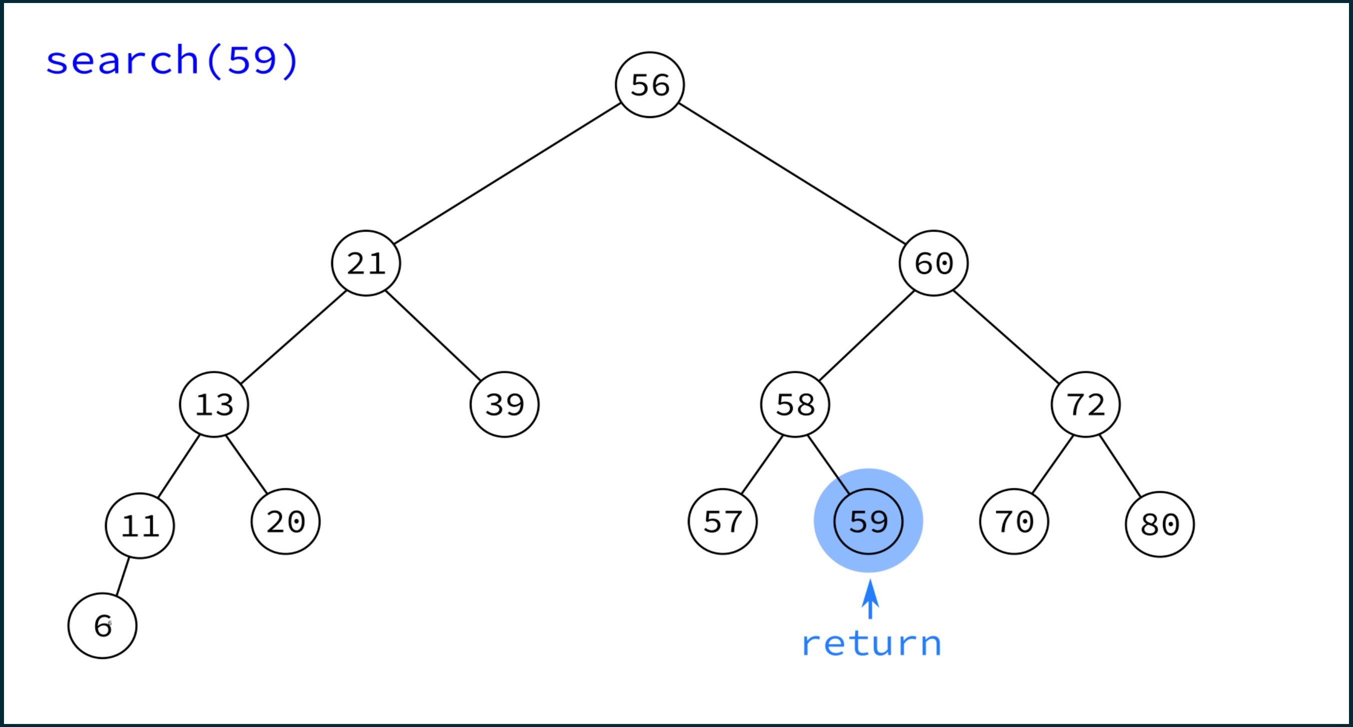
SEARCH

Given `x`, find and return a node with key `x`. Return `None` if no such node exists.



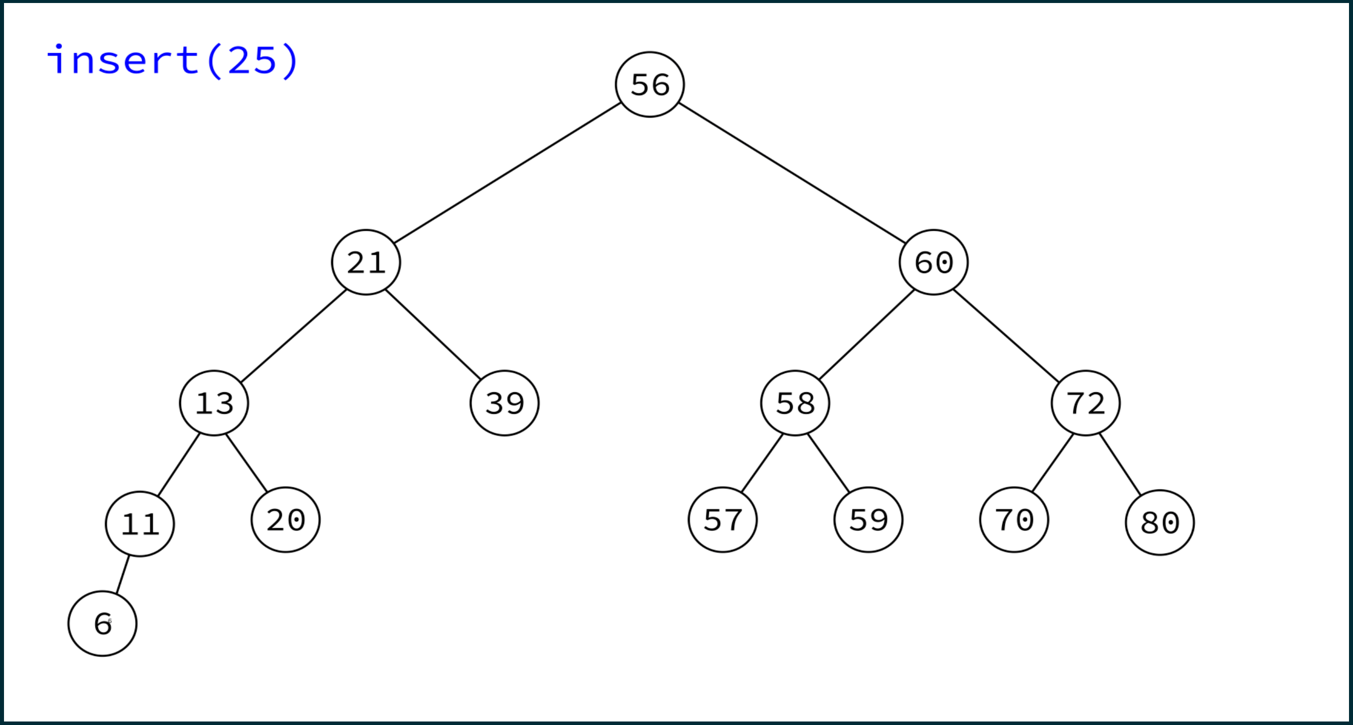
SEARCH

Given `x`, find and return a node with key `x`. Return `None` if no such node exists.



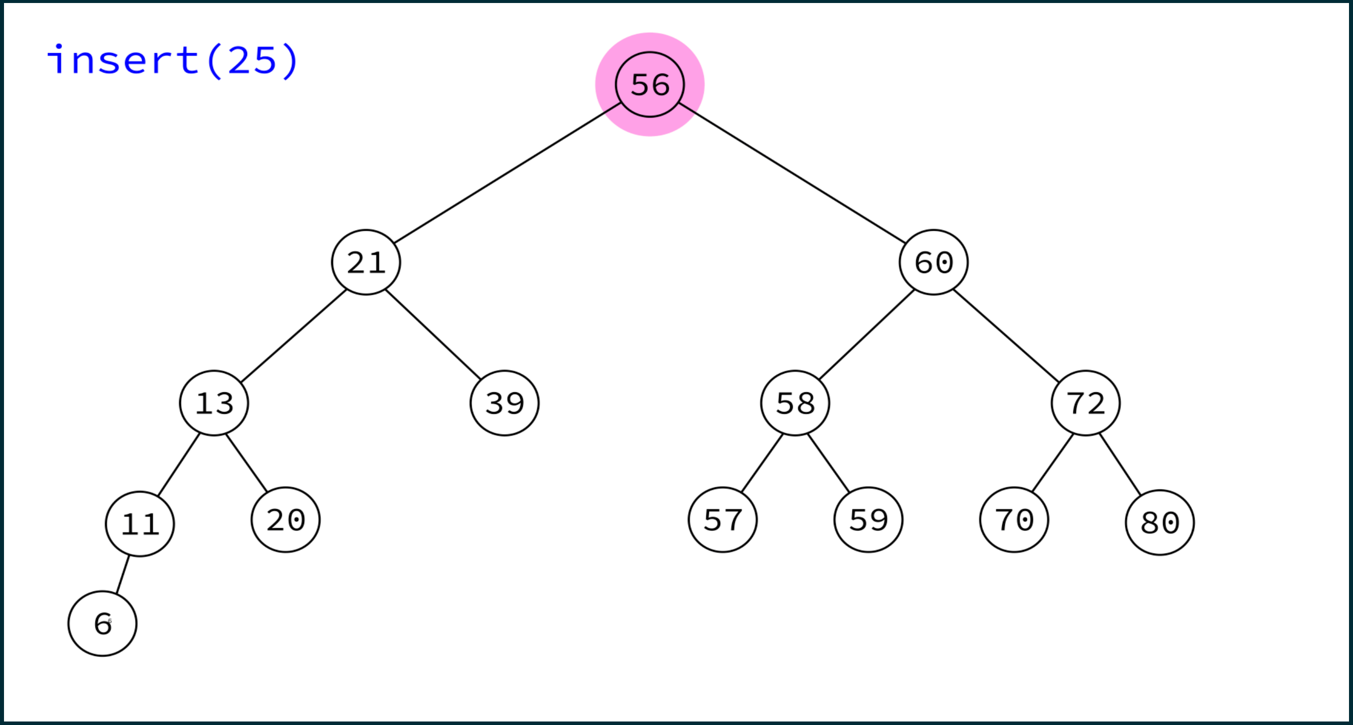
INSERT

Given a key, add a node to the tree with that key, maintaining the BST property.



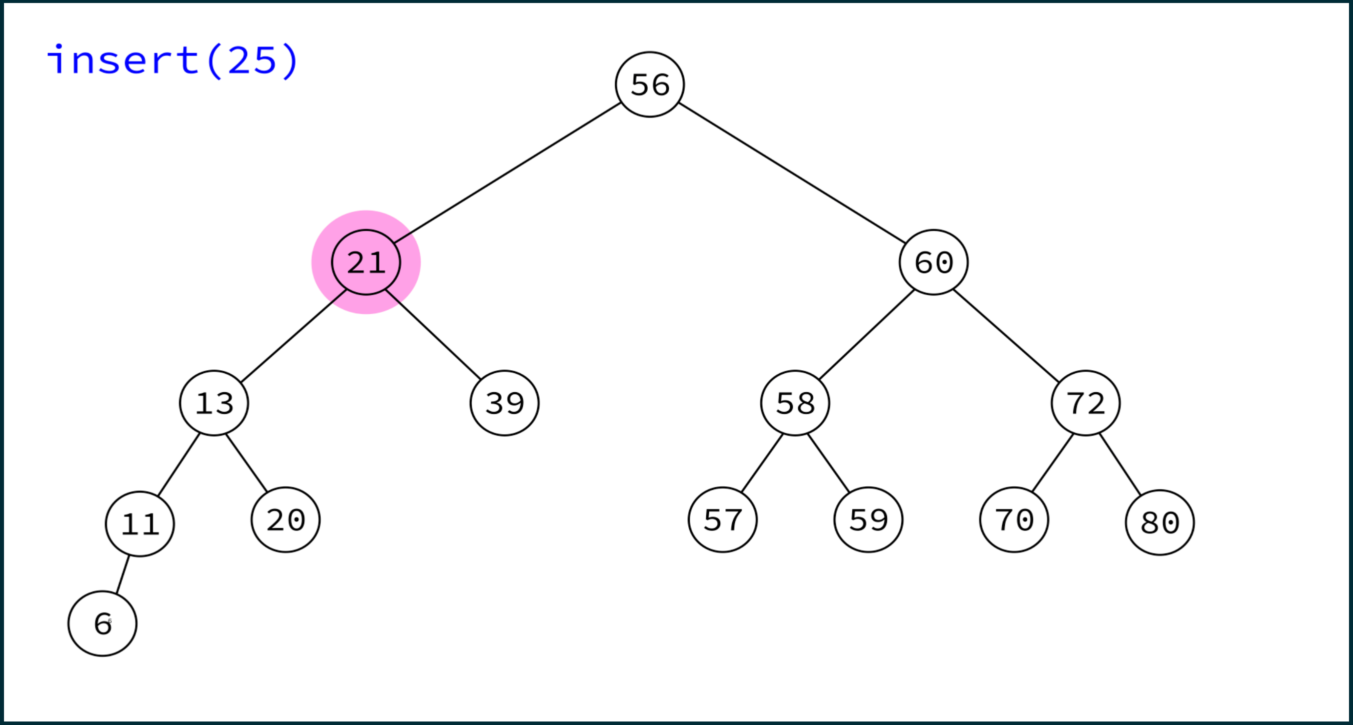
INSERT

Given a key, add a node to the tree with that key, maintaining the BST property.



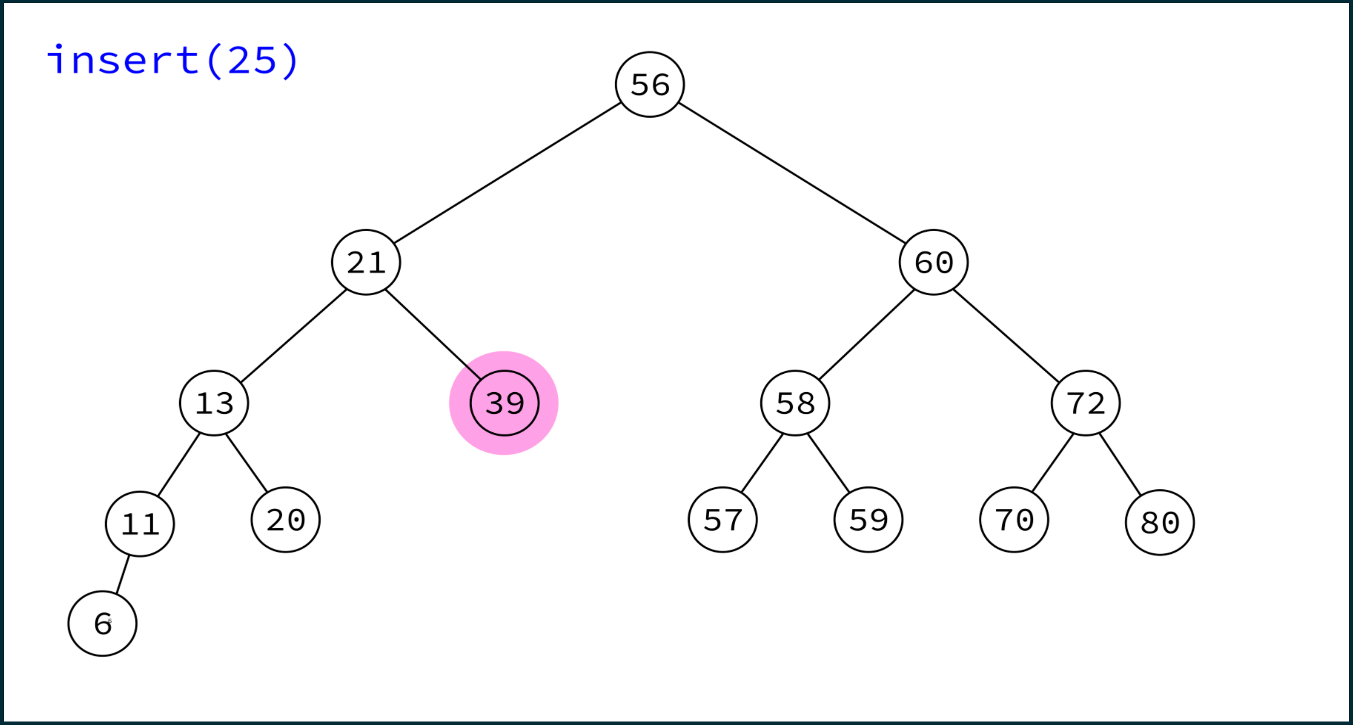
INSERT

Given a key, add a node to the tree with that key, maintaining the BST property.



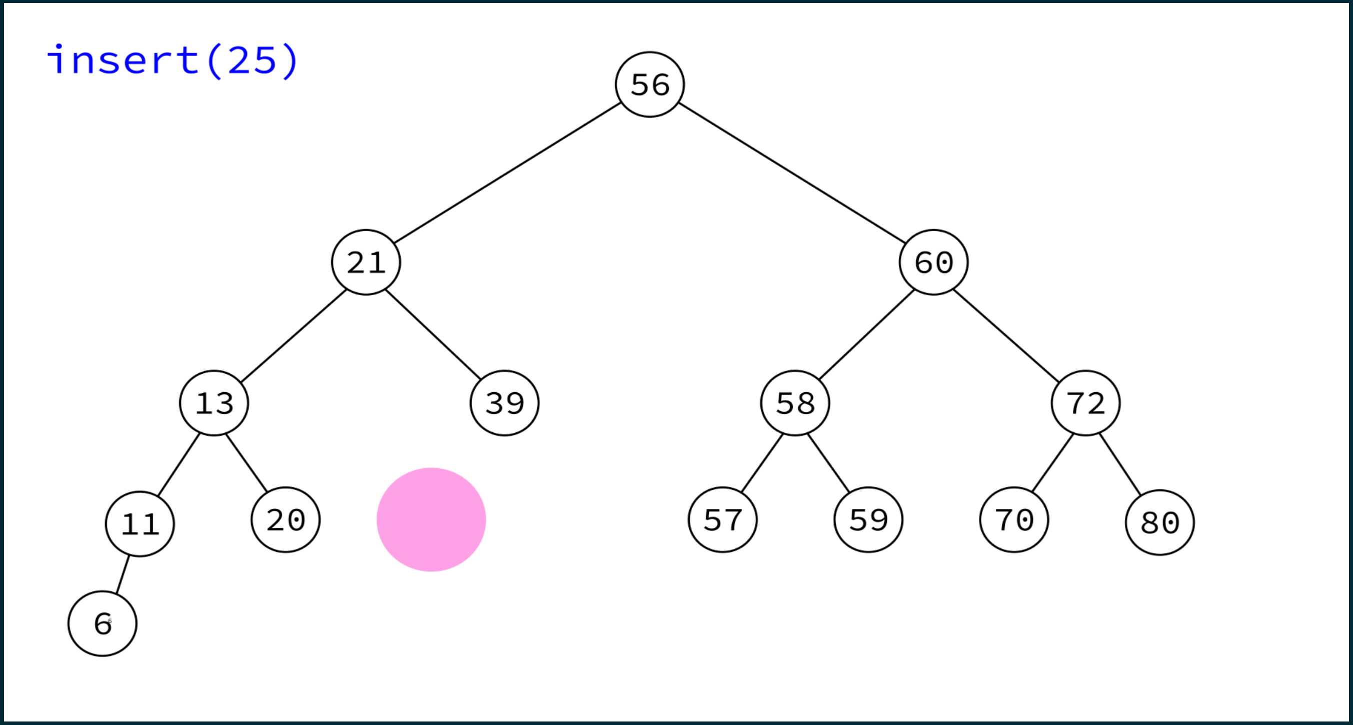
INSERT

Given a key, add a node to the tree with that key, maintaining the BST property.



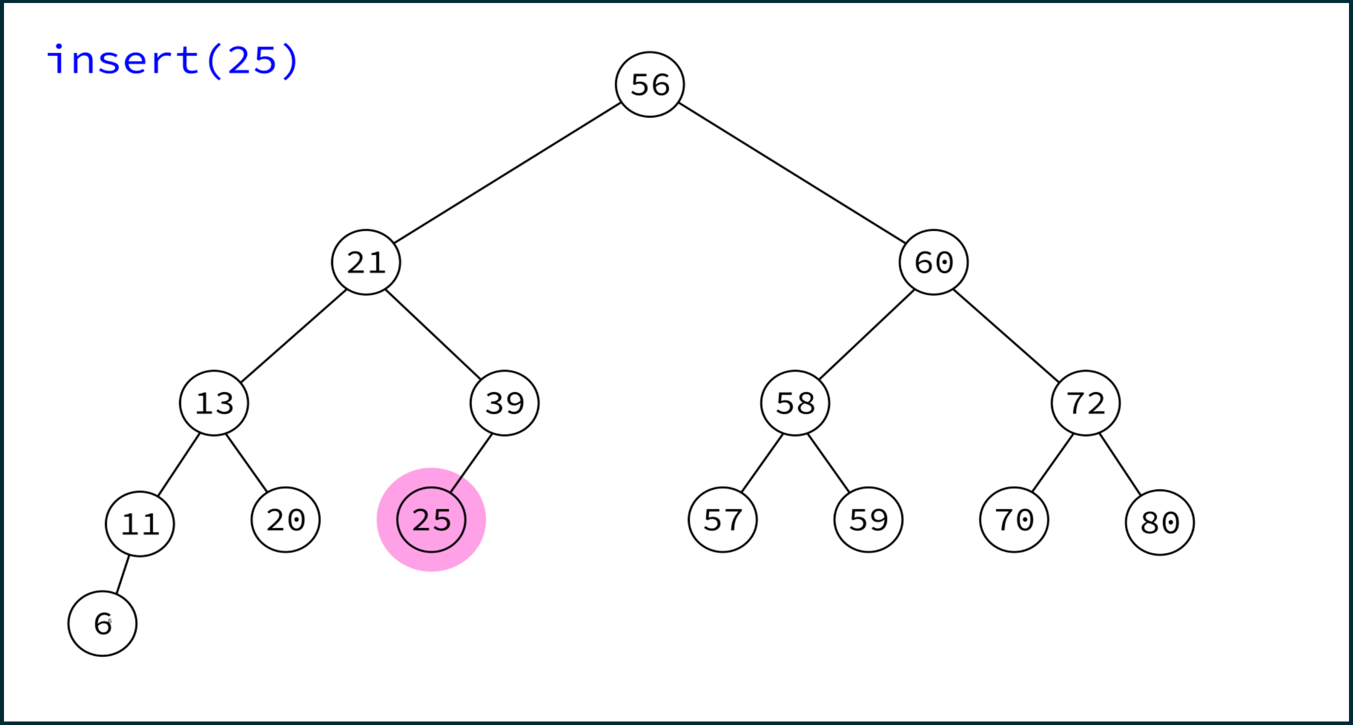
INSERT

Given a key, add a node to the tree with that key, maintaining the BST property.



INSERT

Given a key, add a node to the tree with that key, maintaining the BST property.



INTEGERSET

Let's use this to build a class to store a collection of integers that supports fast insertion and membership testing.

IMPLEMENTATION HIDING

`IntegerSet` has many possible implementations (e.g. a list, a tree, ...), and a user of the class doesn't need to know about which one it uses.

REFERENCES

- In optional course texts:
  - *Problem Solving with Algorithms and Data Structures using Python* by Miller and Ranum, discusses binary trees in Chapter 7.
- Elsewhere:
  - Cormen, Leiserson, Rivest, and Stein discusses graph theory and trees in Appendices B.4 and B.5, and binary search trees in Chapter 12.

REVISION HISTORY

- 2022-02-24 Initial publication