Advanced Data Structures

CSCI 1030U - Intro to Computer Science
@IntroCS

Randy J. Fortier
@randy_fortier

OntarioTech
UNIVERSITY

Outline

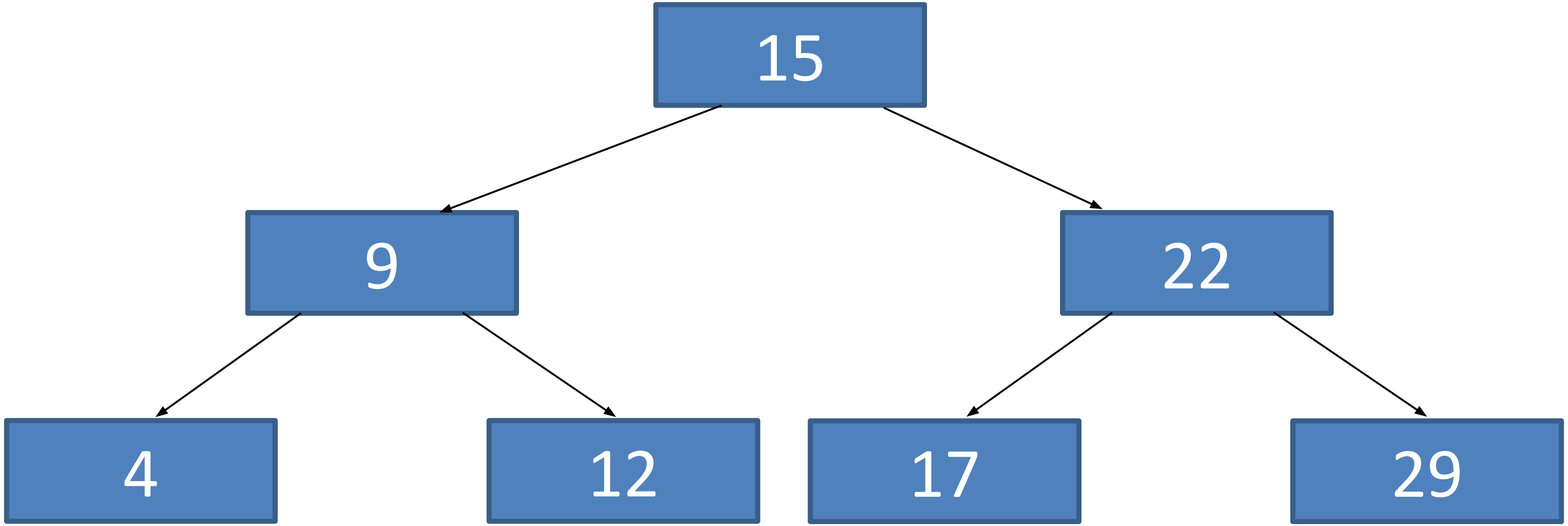
- Advanced data structures:
    - Binary trees
        - Binary tree array implementation
    - Binary search trees
        - Print
        - Search
        - Insert

Trees

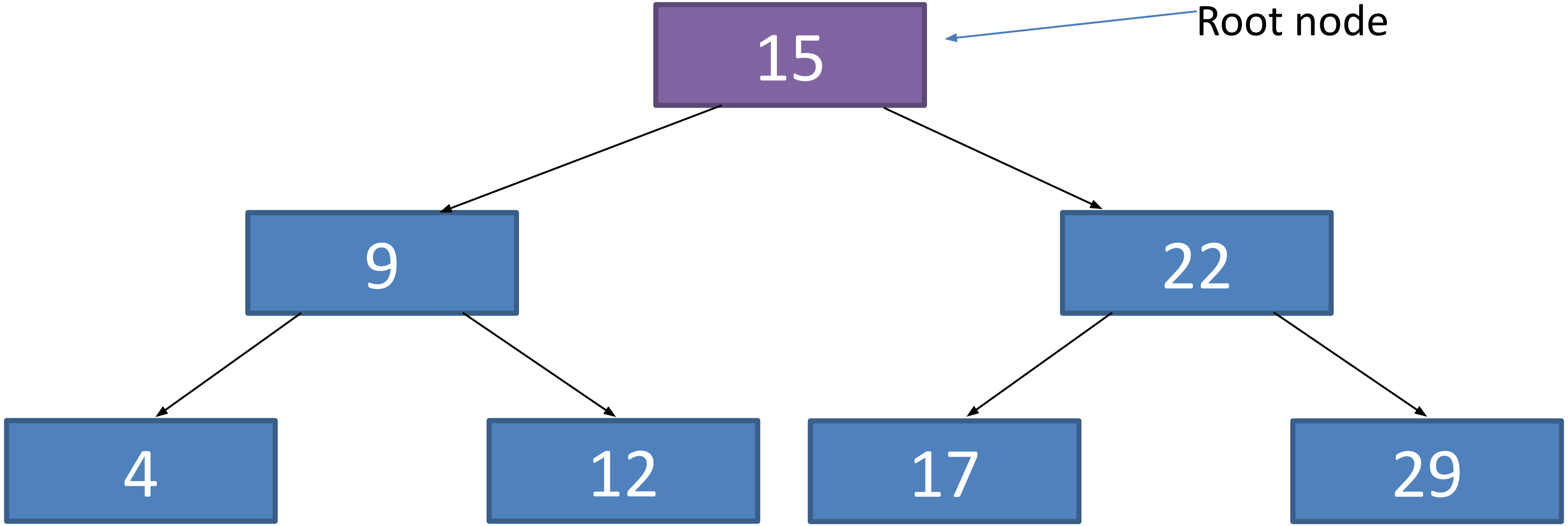
Trees

- A tree is a non-linear data structure
- Binary tree – a tree where branch nodes have (at most) two children
- Binary search tree - a binary tree where all elements are ordered from left to right

Trees - Terminology

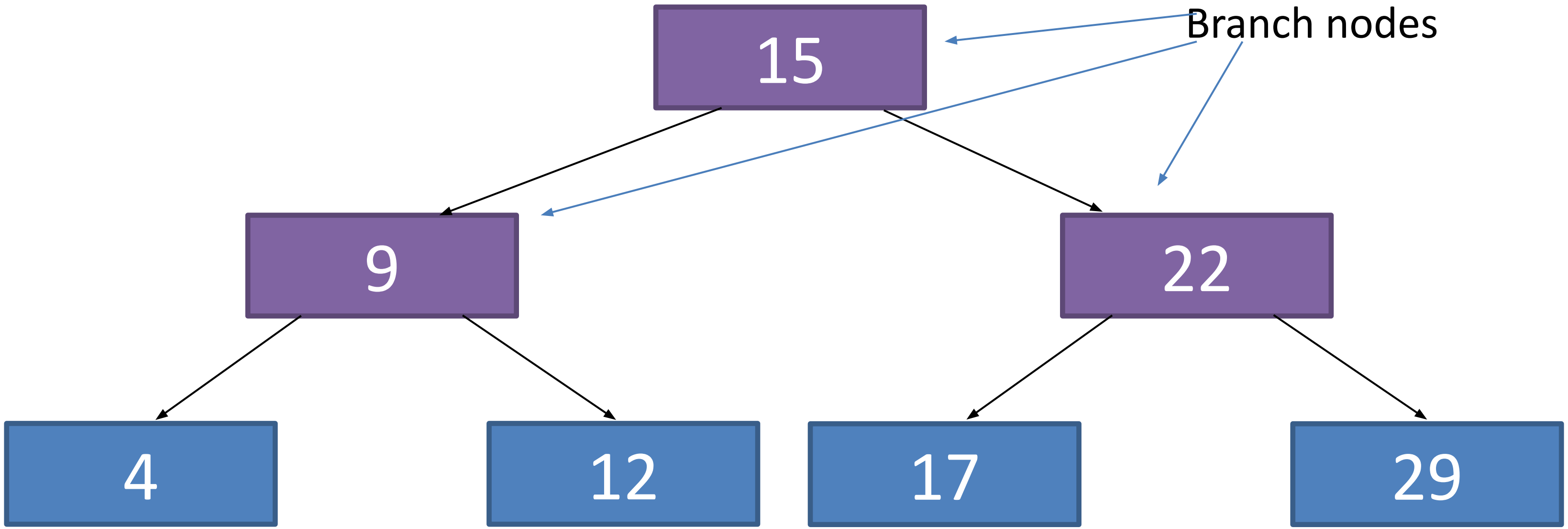


Trees - Terminology

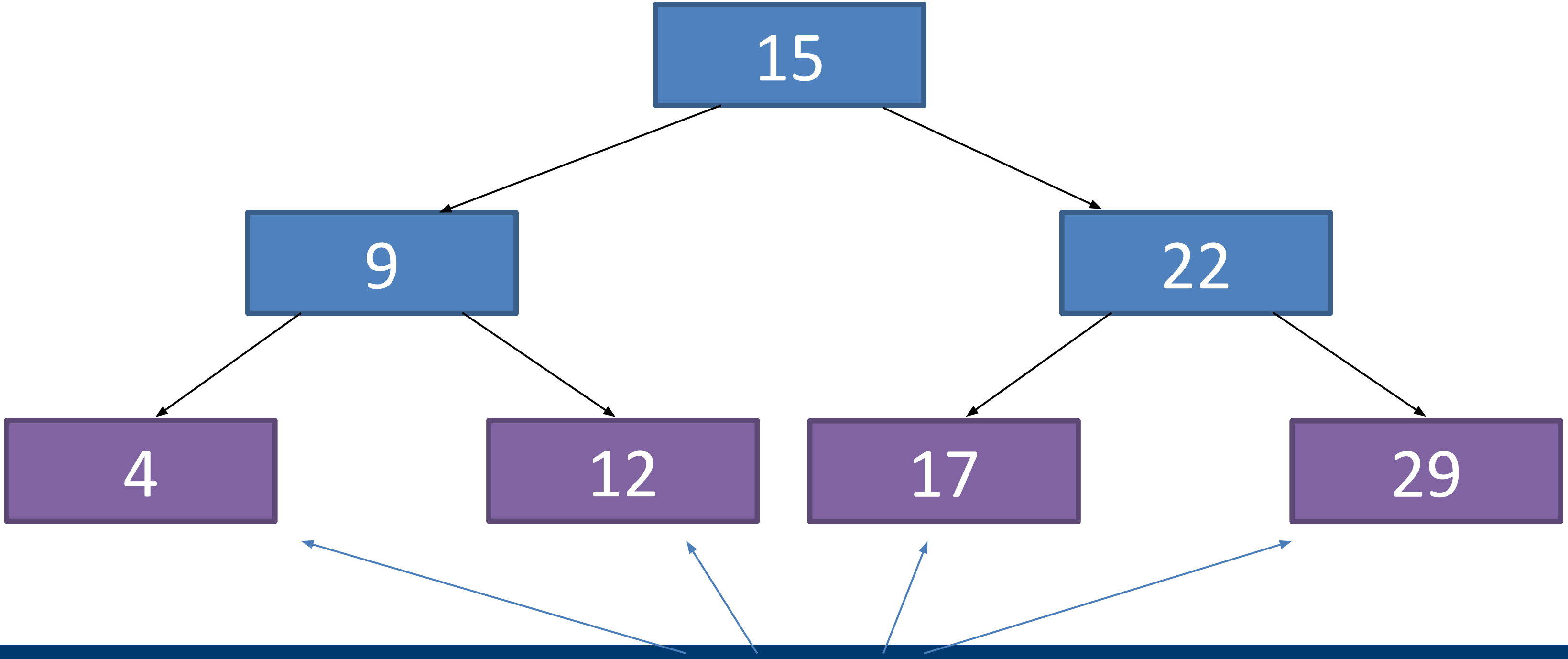


Root node

Trees - Terminology



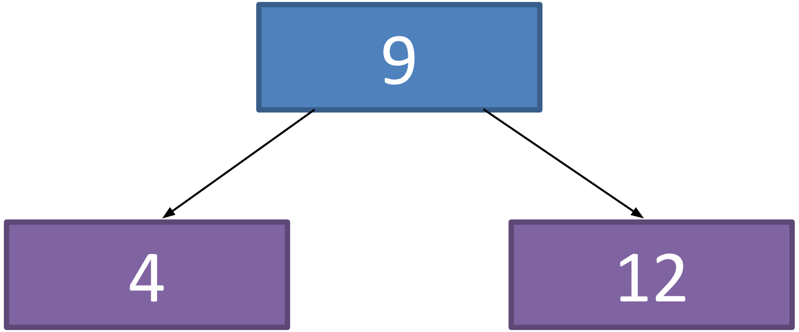
Trees - Terminology



Leaf nodes

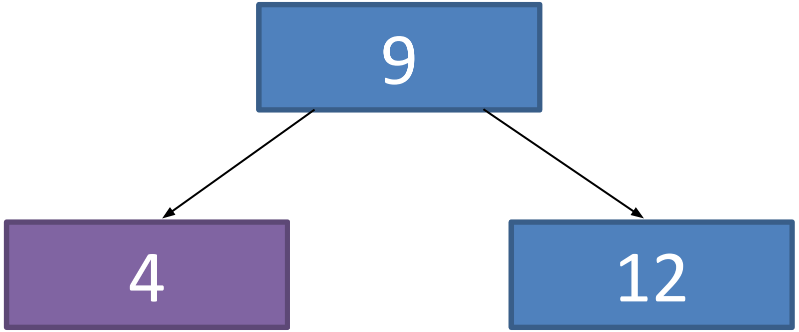
Trees - Terminology

- A branch node's links are called its children



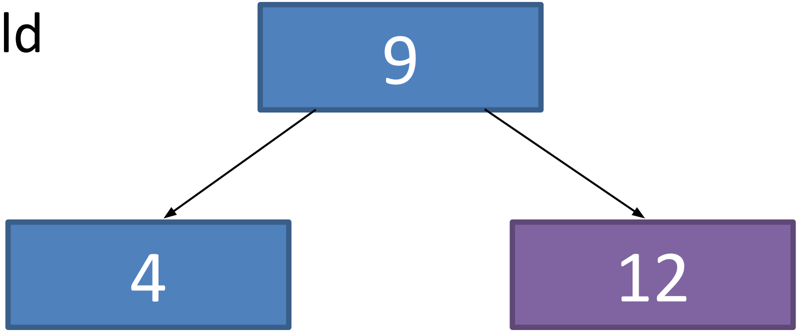
Trees - Terminology

- A branch node's links are called its children
- Binary trees have two children:
  - Left child



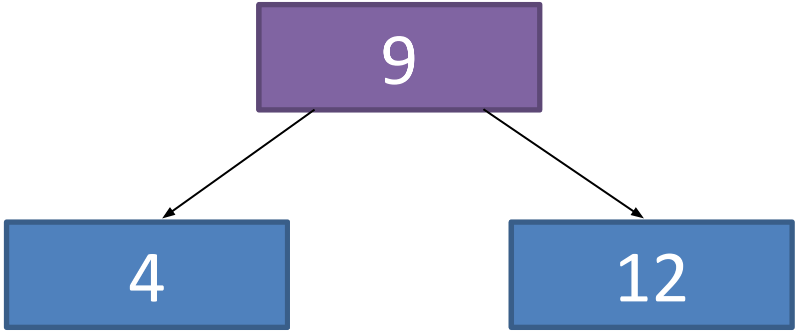
Trees - Terminology

- A branch node's links are called its children
- Binary trees have two children:
  - Left child
  - Right child



Trees - Terminology

- A child node's branch node is called its parent



Trees

- Operations:
  - `insert()`: Inserts a new value into the tree
  - `delete()`: Deletes a value from the tree
  - `root()`: Returns the root node of the tree

Binary Trees

Binary Trees

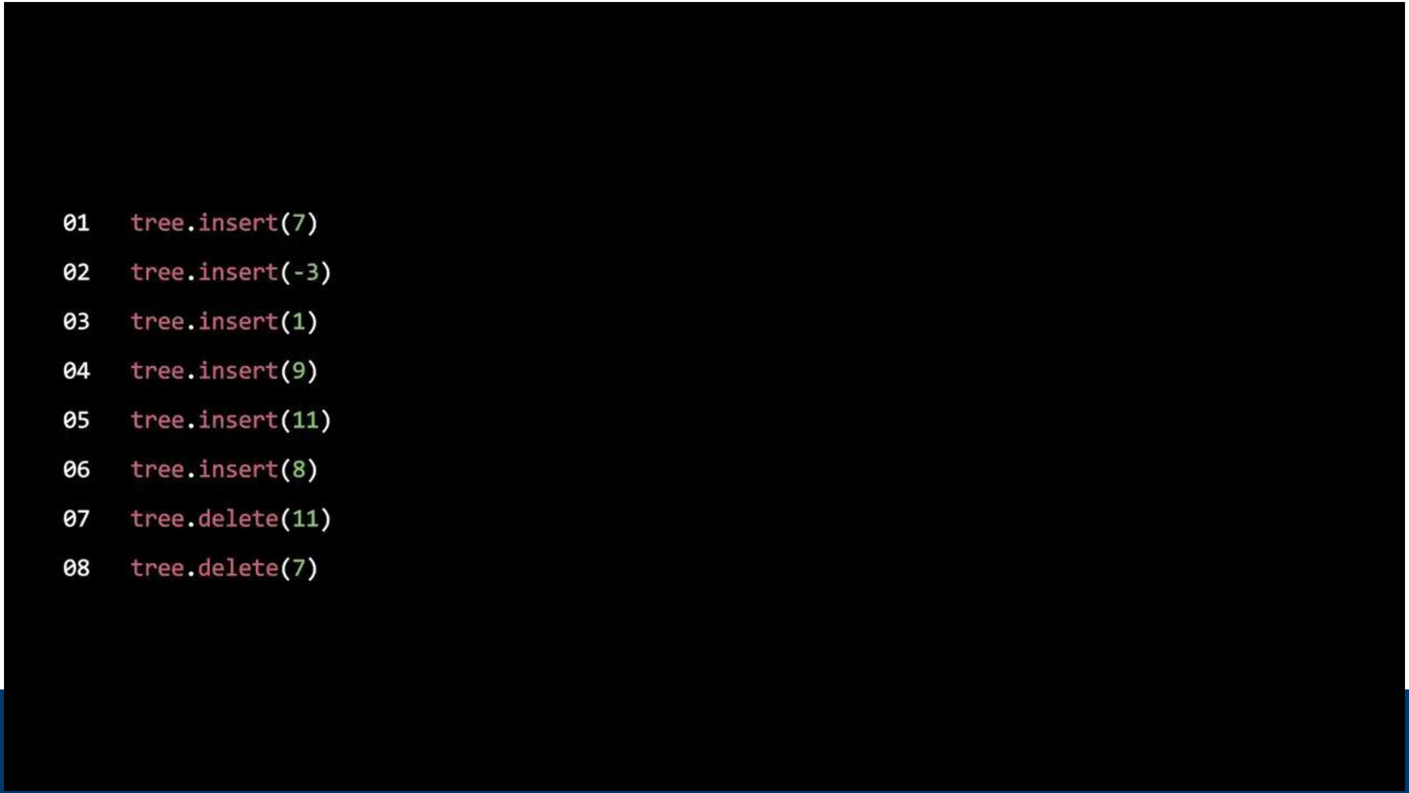
- Binary trees are trees where nodes have at most 2 children
- Binary search trees are (generally) more efficient for searching, due to their sprawling nature
  - The height of a *balanced* binary tree is approximately $\log_2 n$

Binary Search Trees

Binary Search Trees

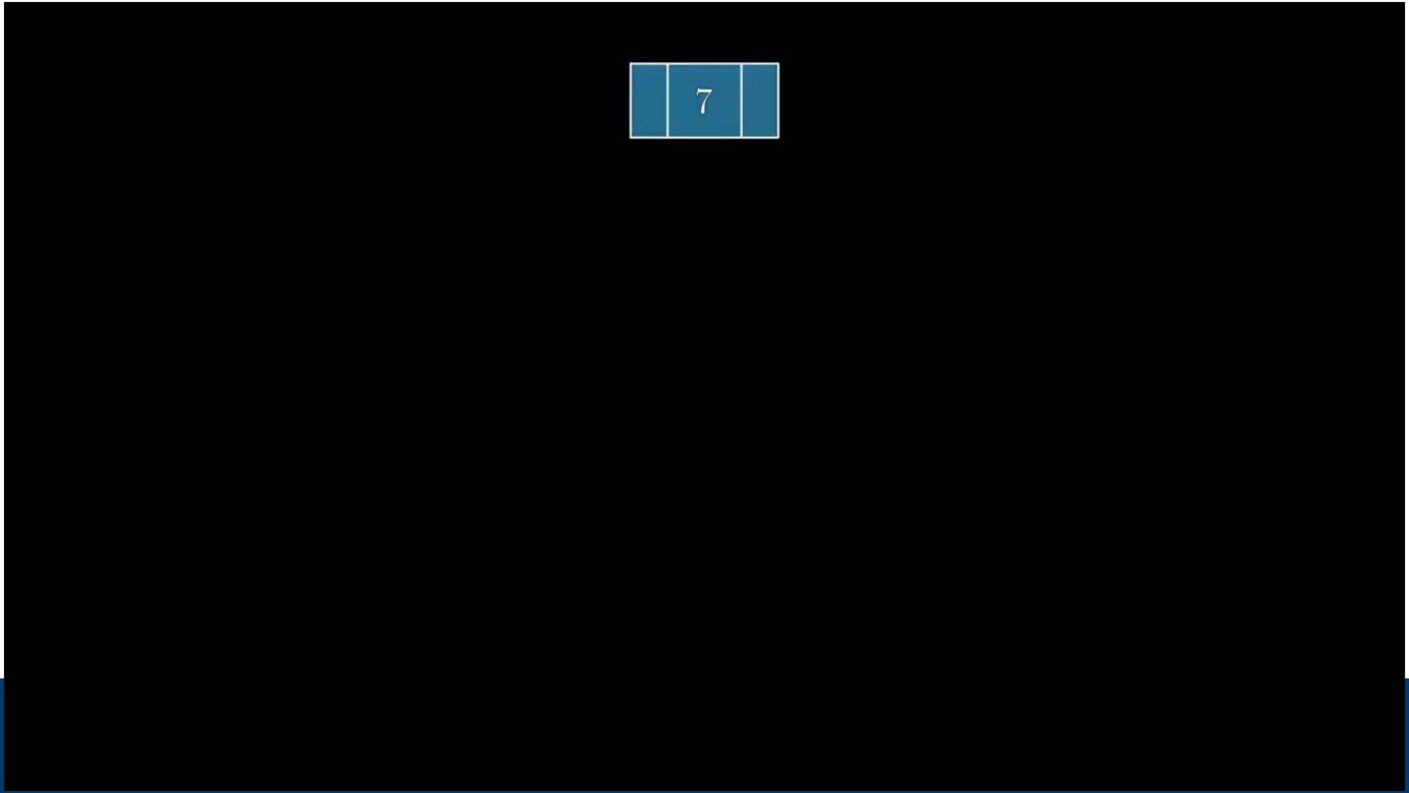
- Binary search trees are binary trees whose elements are ordered from left to right (binary search tree property)
  - Their name comes from the fact that searching a binary search tree is very similar to the binary search algorithm

Binary Search Trees

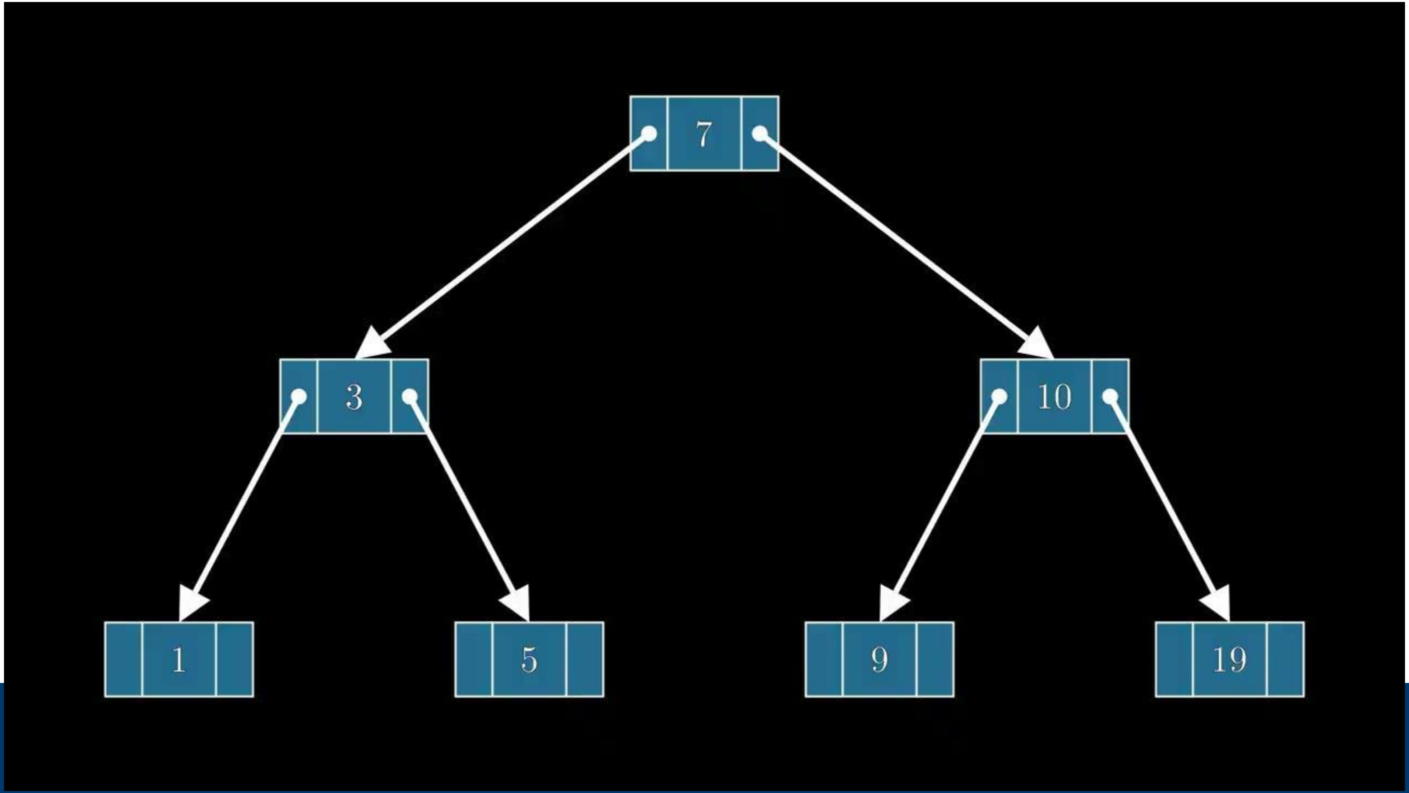


```
tree.insert(7)
tree.insert(-3)
tree.insert(1)
tree.insert(9)
tree.insert(11)
tree.insert(8)
tree.delete(11)
tree.delete(7)
```

BSTs - Insertion



BSTs - Deletion



Tree Implementations

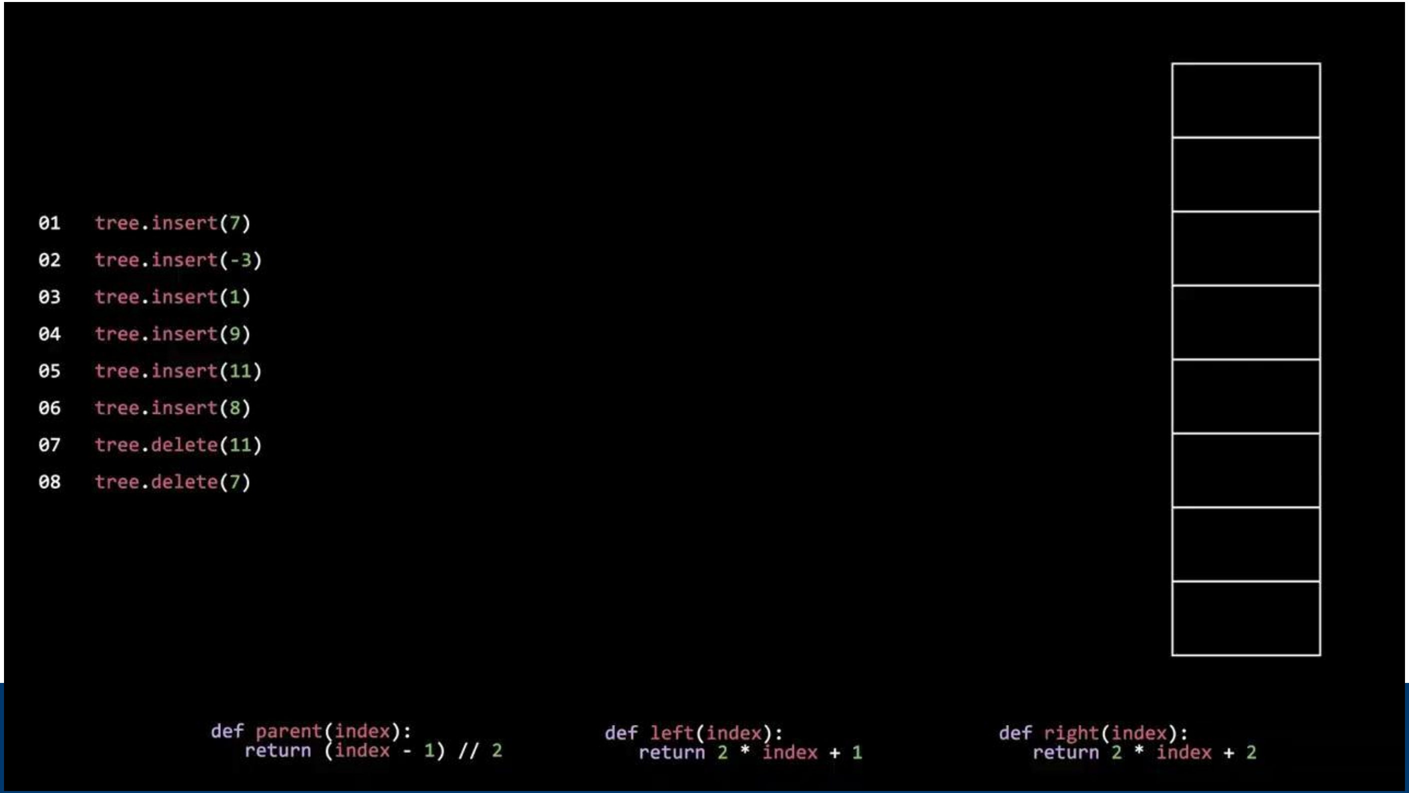
Binary Trees - Implementations

- Array
  - Root element is inserted at index 0
  - For element `i`
    - Its left child is found at `2i + 1` (if any)
    - Its right child is found at `2i + 2` (if any)
    - Its parent is found at floor `(i - 1) // 2`
- Linked structure
  - Similar to linked list nodes, each tree node has two pointers to other nodes
  - For leaf nodes, these pointers are simply null values

Binary Search Trees - Implementations

- The binary tree implementations also work well for binary search trees
- However, insertion and deletion is more constrained, since we need to maintain the binary search tree property

BSTs - Array Implementation



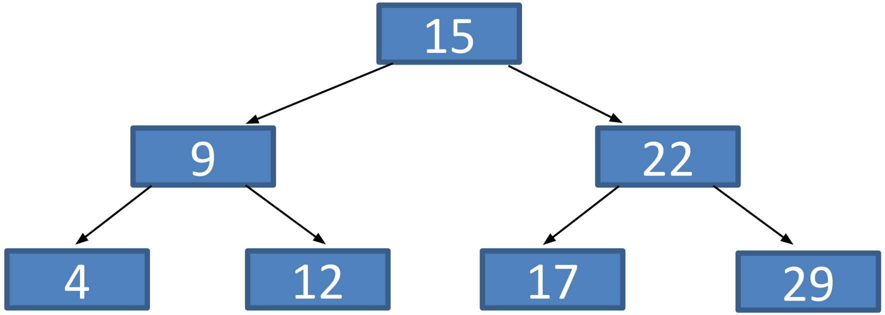
```
tree.insert(7)
tree.insert(-3)
tree.insert(1)
tree.insert(9)
tree.insert(11)
tree.insert(8)
tree.delete(11)
tree.delete(7)
```

```
def parent(index):
    return (index - 1) // 2
```

```
def left(index):
    return 2 * index + 1
```

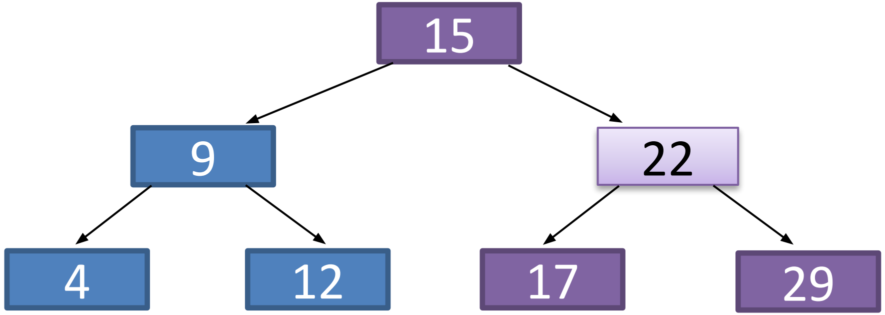
```
def right(index):
    return 2 * index + 2
```

Binary Trees - Array Implementation



15	9	22	4	12	17	29

Binary Trees - Array Implementation



0	1	2	3	4	5	6
15	**9**	**22**	**4**	**12**	**17**	**29**

(i-1) // 2

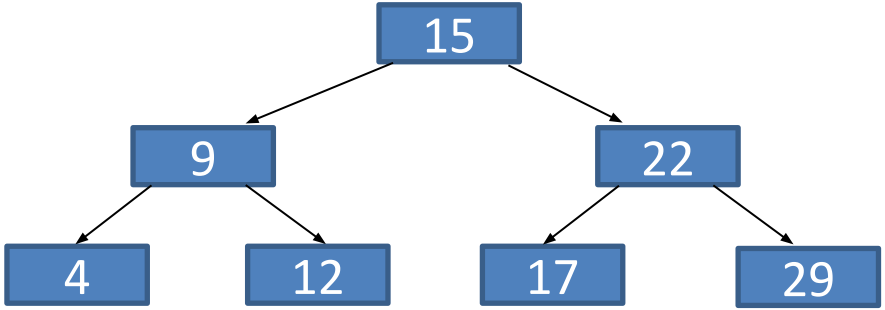
i

2i + 1

2i + 2

Binary Search Trees - Array Implementation

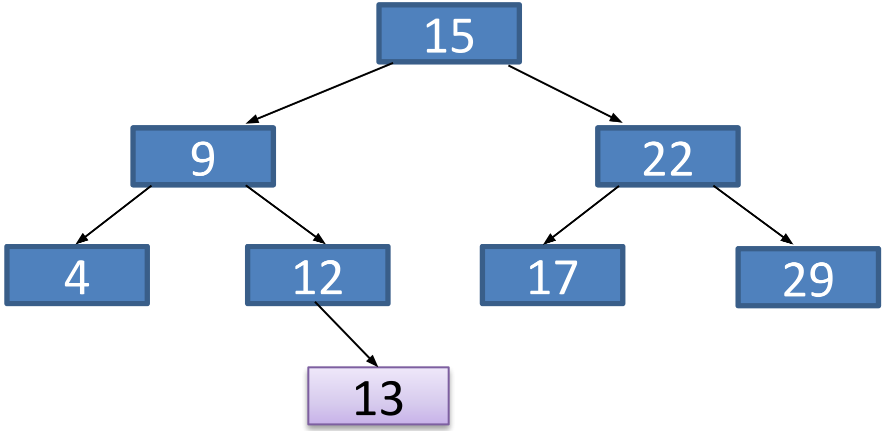
- Insert a new element (13)



15	9	22	4	12	17	29

Binary Search Trees - Array Implementation

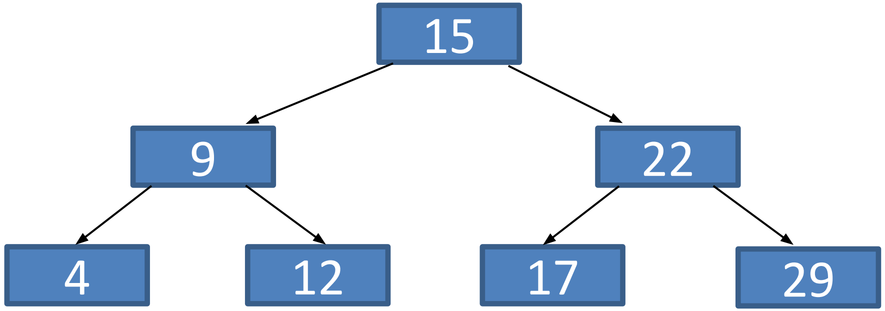
- Insert a new element (13)



15	9	22	4	12	17	29				13

Binary Search Trees - Array Implementation

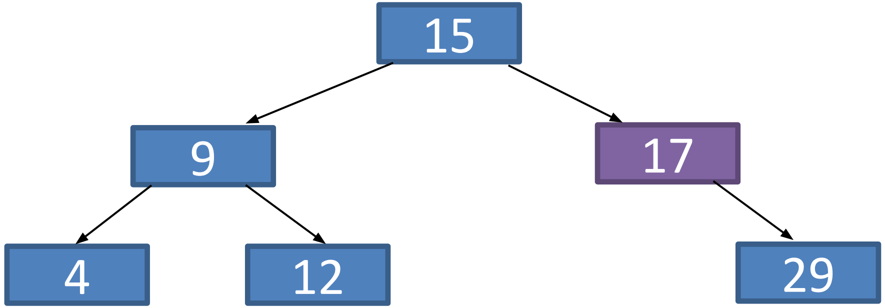
- Delete an element (22):



15	9	22	4	12	17	29

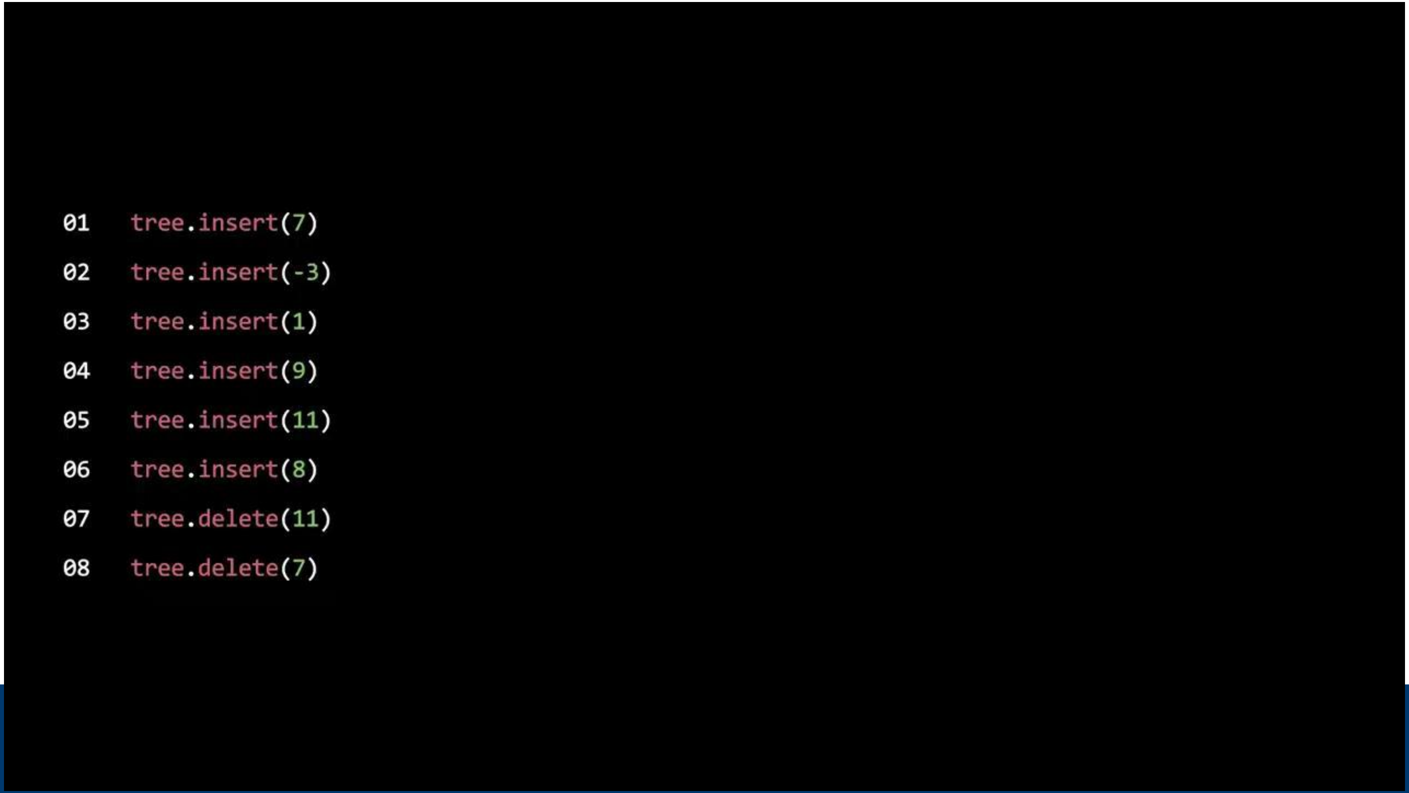
Binary Search Trees - Array Implementation

- Delete an element (22):



15	9	17	4	12		29

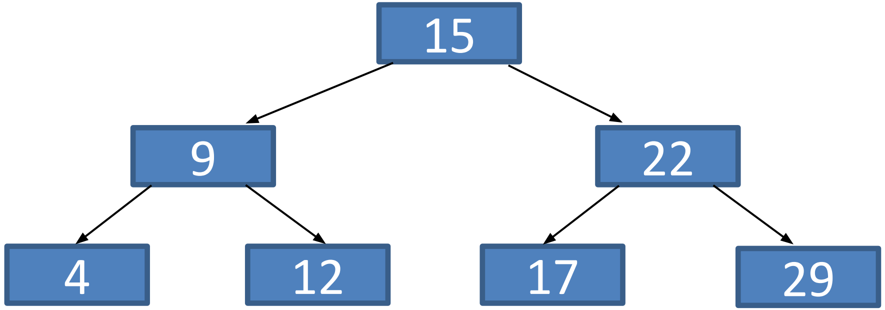
BSTs - Linked List Implementation



```
tree.insert(7)
tree.insert(-3)
tree.insert(1)
tree.insert(9)
tree.insert(11)
tree.insert(8)
tree.delete(11)
tree.delete(7)
```

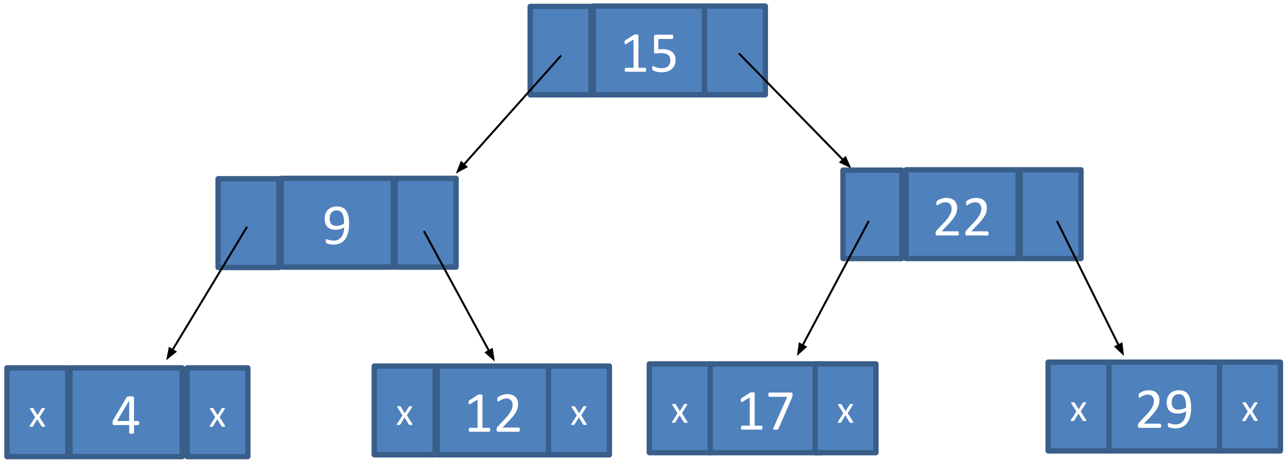
Binary Search Trees - Exercise

- Delete an element (15)
- Insert an element (13)
- Delete an element (9)
- Insert an element (33)



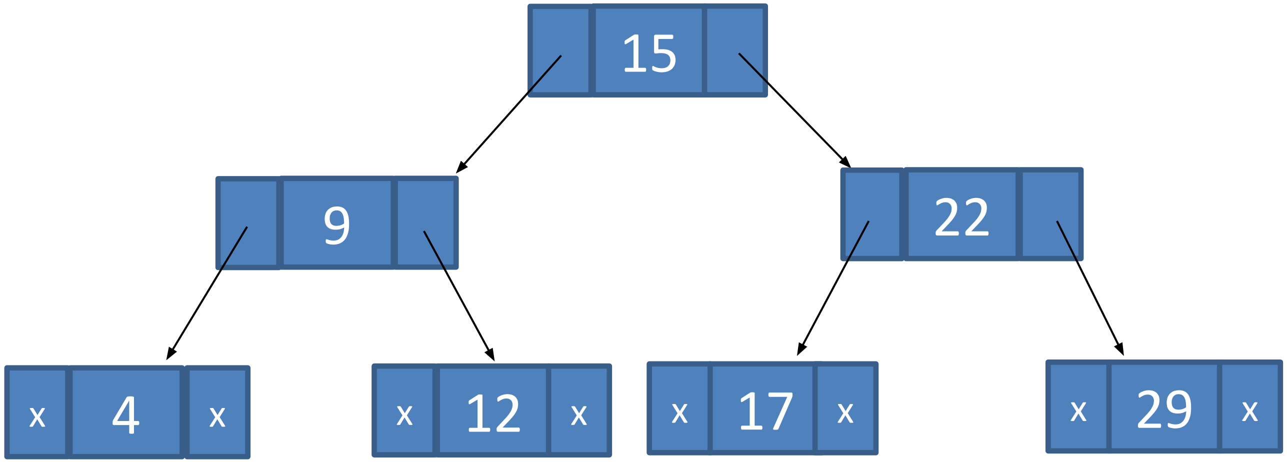
15	9	22	4	12	17	29

BSTs - Linked Structure Implementation



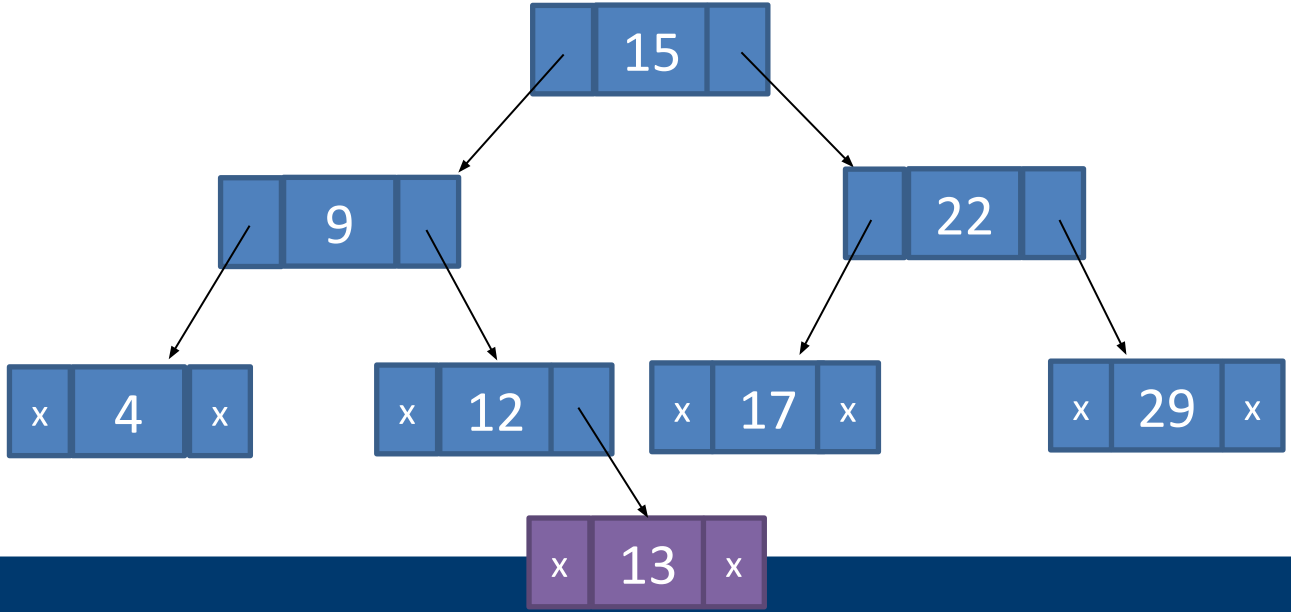
BSTs - Linked Structure Implementation

- Insert a new element (13):



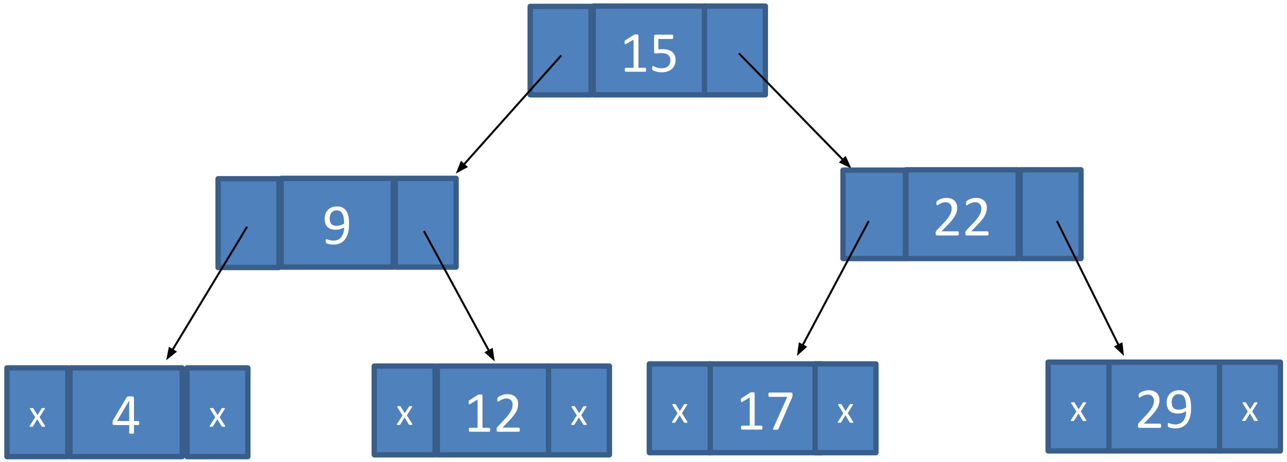
BSTs - Linked Structure Implementation

- Insert a new element (13):



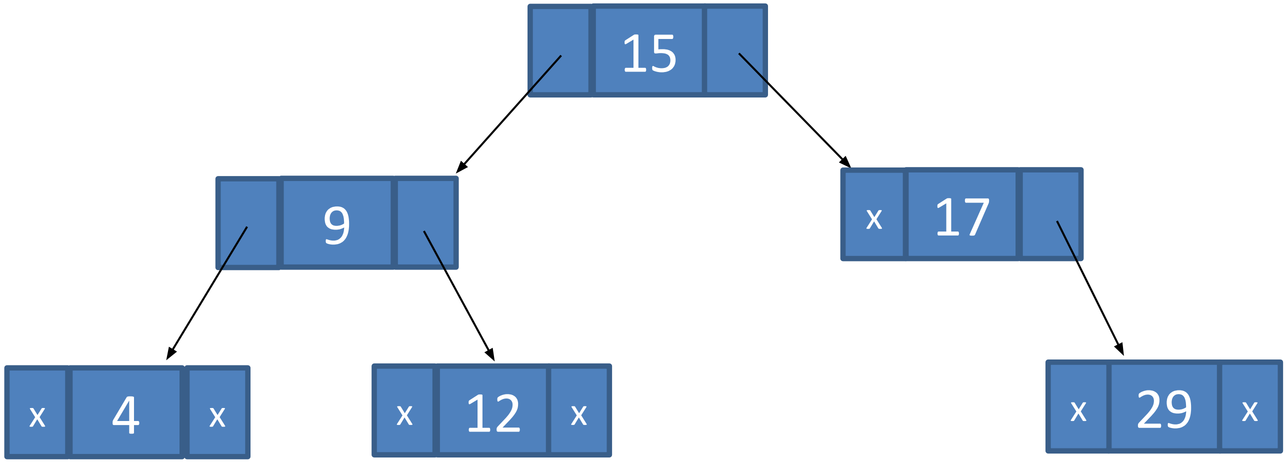
BSTs - Linked Structure Implementation

- Delete an element (22):



BSTs - Linked Structure Implementation

- Delete an element (22):



Programming Exercise 10b.1

- Write an array implementation of a binary search tree in Python
  - Class name: `Binary_Search_Tree`
  - Initialize with an existing array
  - No insert or delete functionality needs to be implemented
  - Create these methods:
    - `left_child_index(parent_index)`
    - `right_child_index(parent_index)`
    - `parent_index(child_index)`

Programming Exercise 10b.1 (cont'd)

- Test code:

```
values = [7, 3, 12, 1, 5, 9, 14]
bst = Binary_Search_Tree(values=values)
index = 1
pindex = bst.parent_index(index)
lindex = bst.left_child_index(index)
rindex = bst.right_child_index(index)
print(f'parent is at index {pindex}, value: {values[pindex]}')
print(f'left child is at index {lindex}, value: {values[lindex]}')
print(f'right child is at index {rindex}, value: {values[rindex]}')
```

Programming Exercise 10b.1 (cont'd)

- Let's modify our array implementation of a binary search tree in Python
  - Create this method:
    - `print()`
    - `search()`

Wrap-up

- Advanced data structures:
  - Binary trees
    - Binary tree array implementation
  - Binary search trees
    - Print
    - Search
    - Insert

Coming Up

- Learning
  - Unsupervised
  - Supervised
- Neural networks
- Genetic algorithms
- Bayesian networks