Programming Languages and Techniques (CIS120)

Lecture 6

Jan 31, 2014

Binary Trees and Binary Search Trees

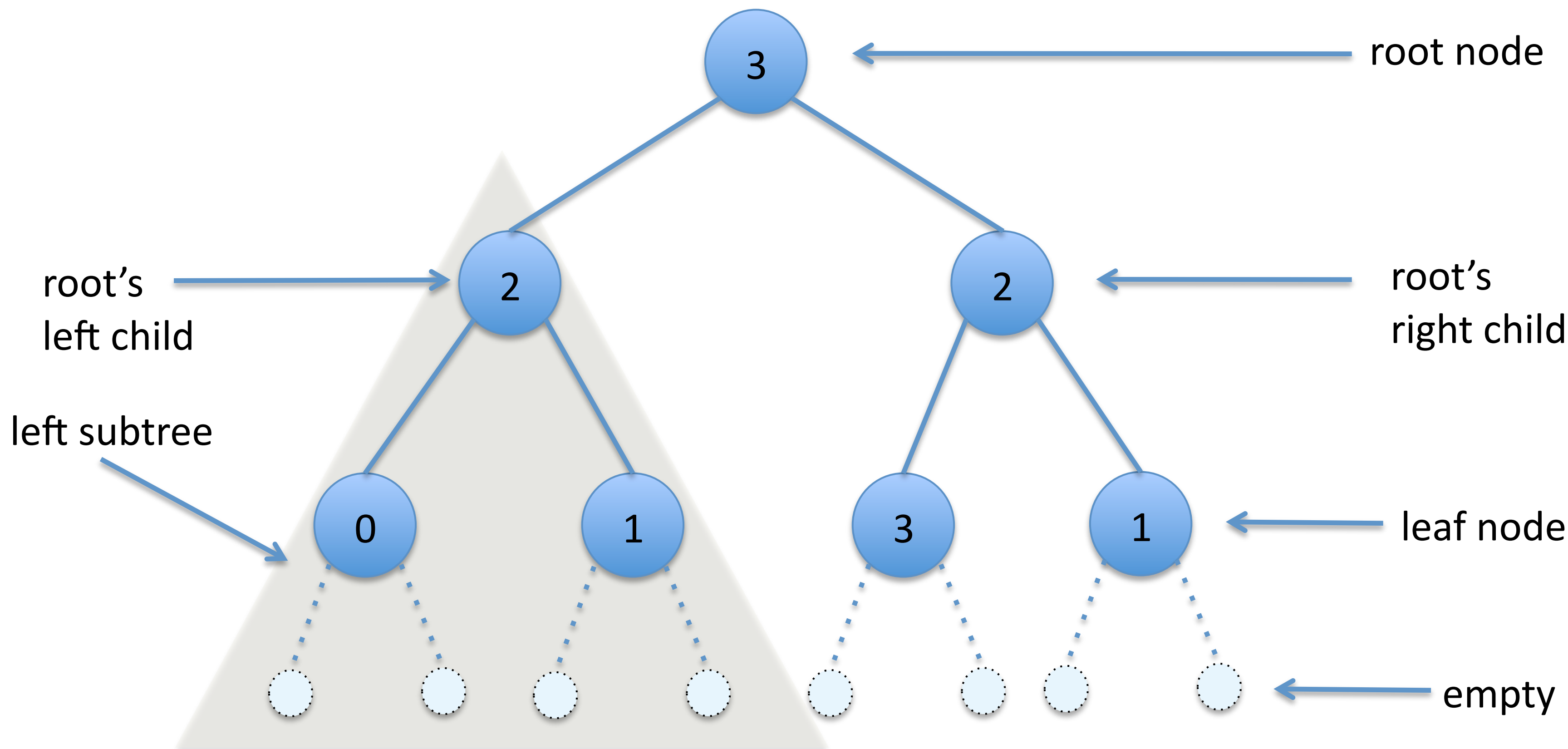
Announcements

- HW 1 stats
  - Avg: 6 ¼ hours, min: 1, max: 20, n: 123
  - Post questions about the hw to piazza, specific notes to TAs in comments. Feedback is for general comments.
- 2nd Homework assignment due Tuesday
- Lecture attendance grade (i.e. clickers)
  - Flexibility for occasional missed lectures due to minor emergencies
  - No need to inform staff (or send CAR) unless you have a major emergency

- Read Chapter 6 and 7

Binary Trees

Binary Trees



A binary tree is either *empty*, or a *node* with at most two children, both of which are also binary trees.

A *leaf* is a node whose children are both empty.

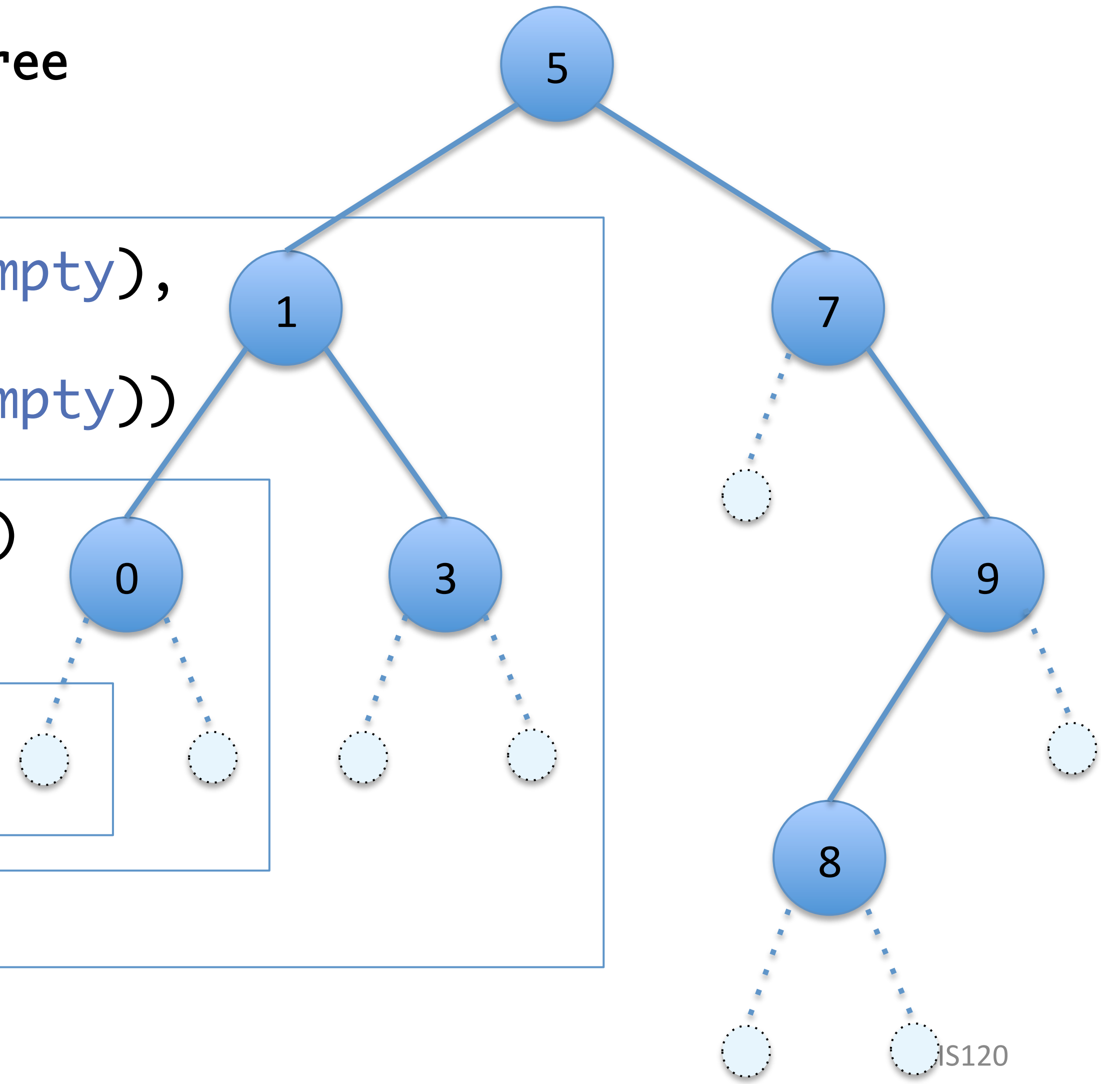
Representing trees

```
type tree =
| Empty
| Node of tree * int * tree
```

```
Node (Node (Empty, 0, Empty),
      1,
      Node (Empty, 3, Empty))
```

```
Node (Empty, 0, Empty)
```

```
Empty
```



Demo

trees.ml treeExamples.ml

Trees as Containers

- Like lists, trees aggregate ordered data
- As we did for lists, we can write a function to determine whether the data structure *contains* a particular element

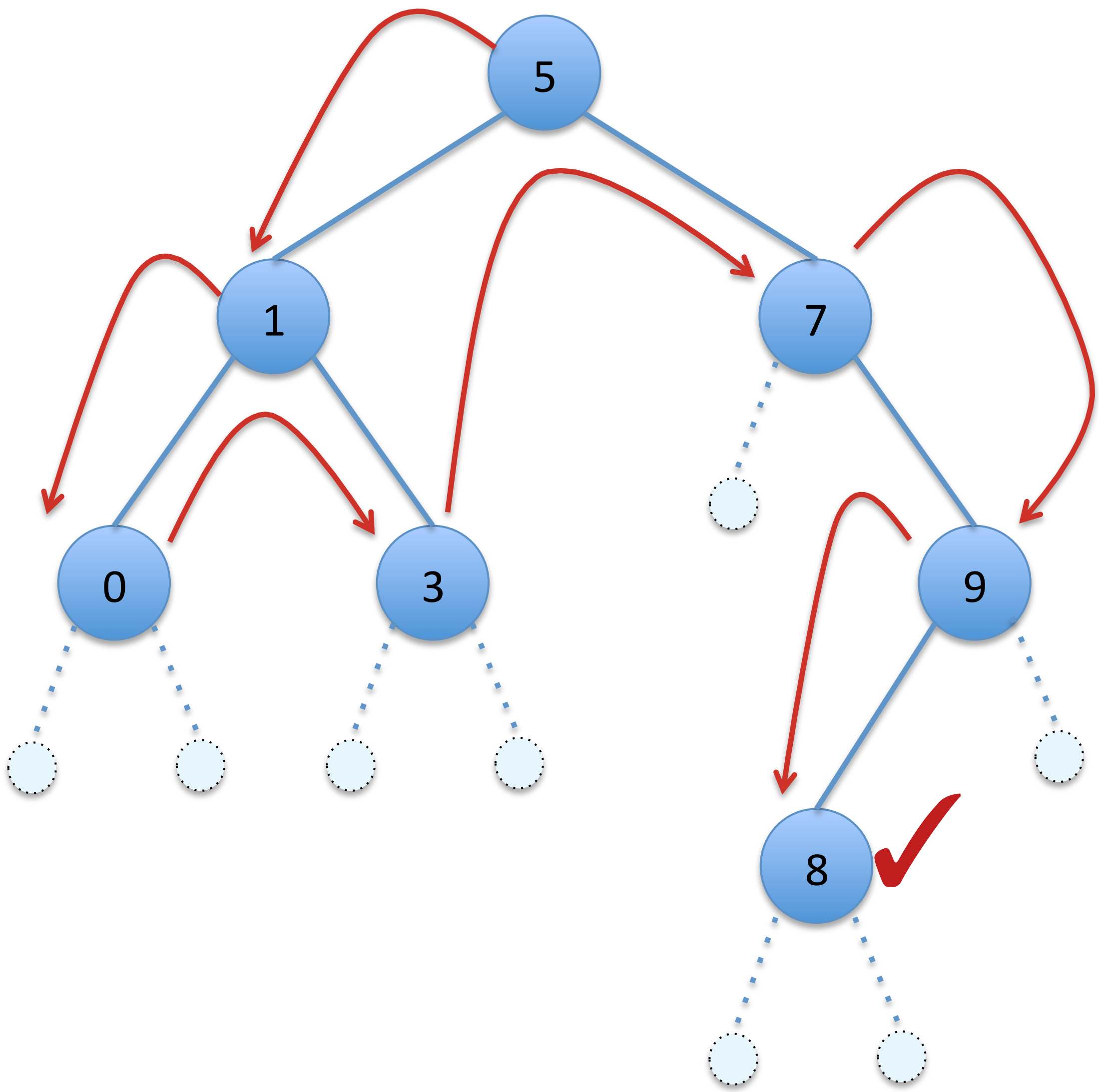
- CHALLENGE: can we use the tree structure to make this process faster?

Searching for Data in a Tree

```
let rec contains (t:tree) (n:int) : bool =
  begin match t with
  | Empty -> false
  | Node(lt,x,rt) -> x = n ||
                       (contains lt n) || (contains rt n)
  end
```

- This function searches through the tree, looking for n
- In the worst case, it might have to traverse the entire tree

Search during (contains t 8)

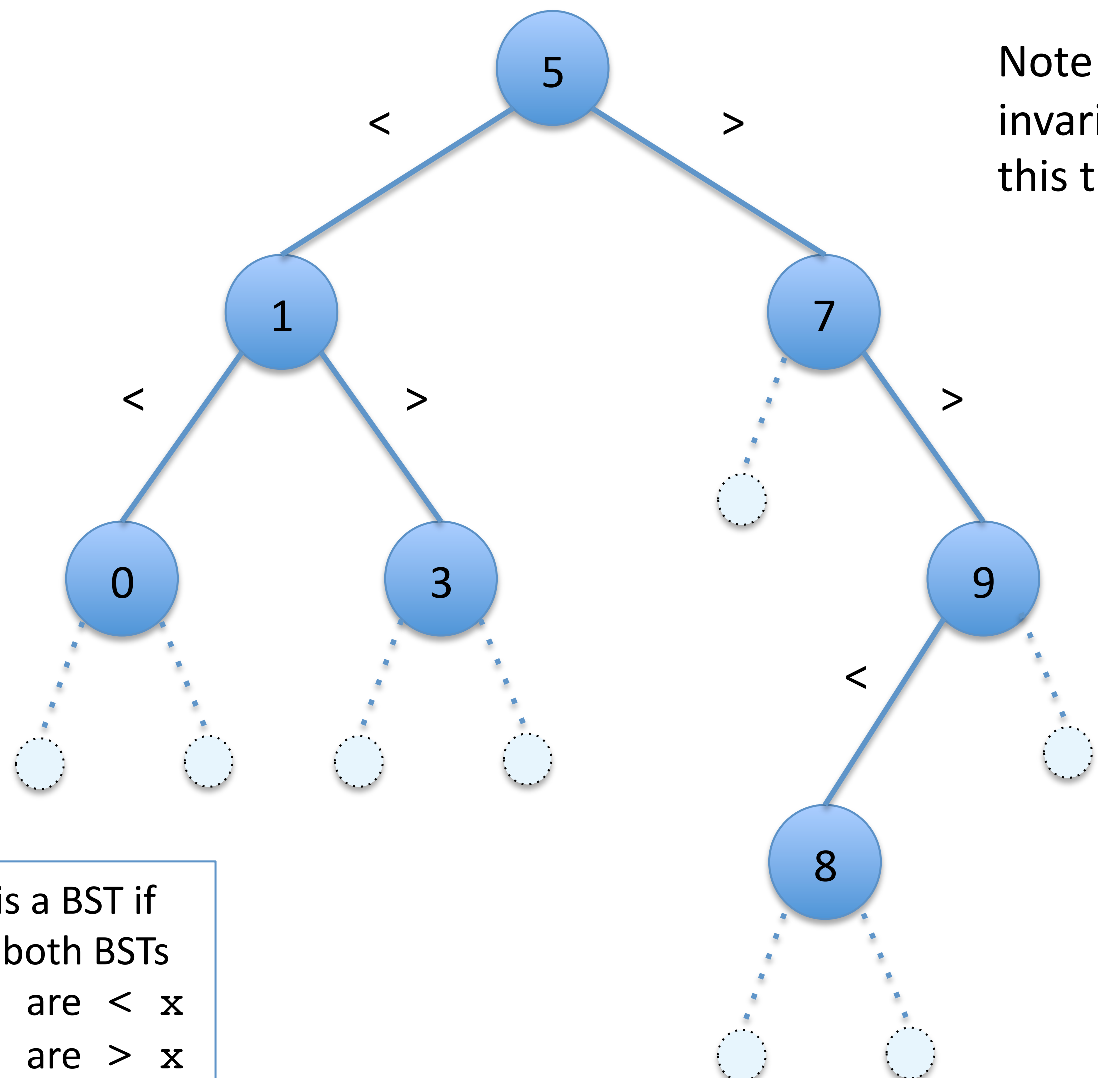


Binary Search Trees

- Key insight: *Ordered* data can be searched more quickly
  - This is why telephone books are arranged alphabetically
  - But requires the ability to focus on *half* of the current data
- A *binary search tree* (BST) is a binary tree with some additional *invariants:*

> - `Node(lt,x,rt)` is a BST if
>   - `lt` and `rt` are both BSTs
>   - all nodes of `lt` are `<` `x`
>   - all nodes of `rt` are `>` `x`
> - `Empty` is a BST

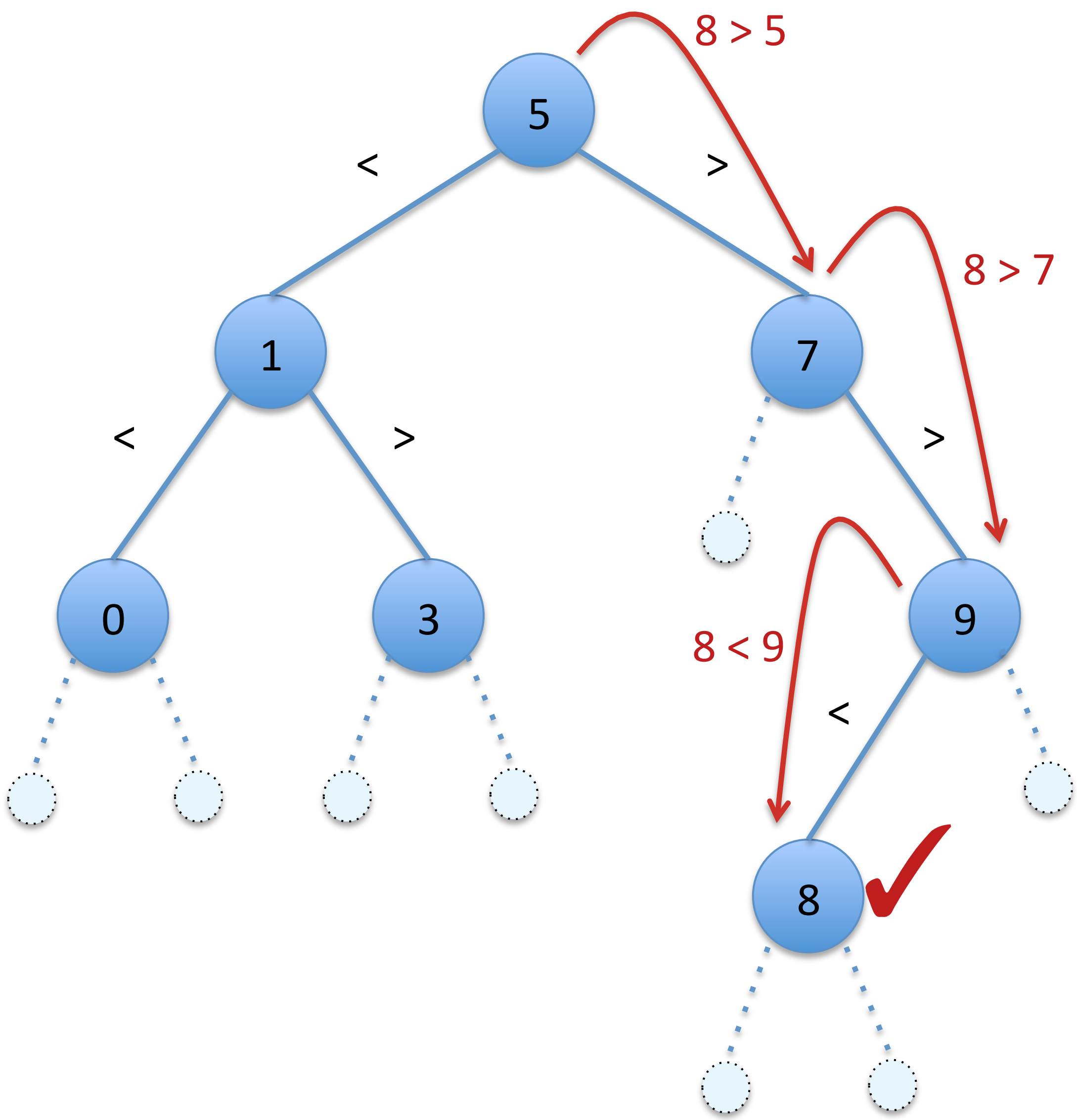
An Example Binary Search Tree



Note that the BST invariants hold for this tree.

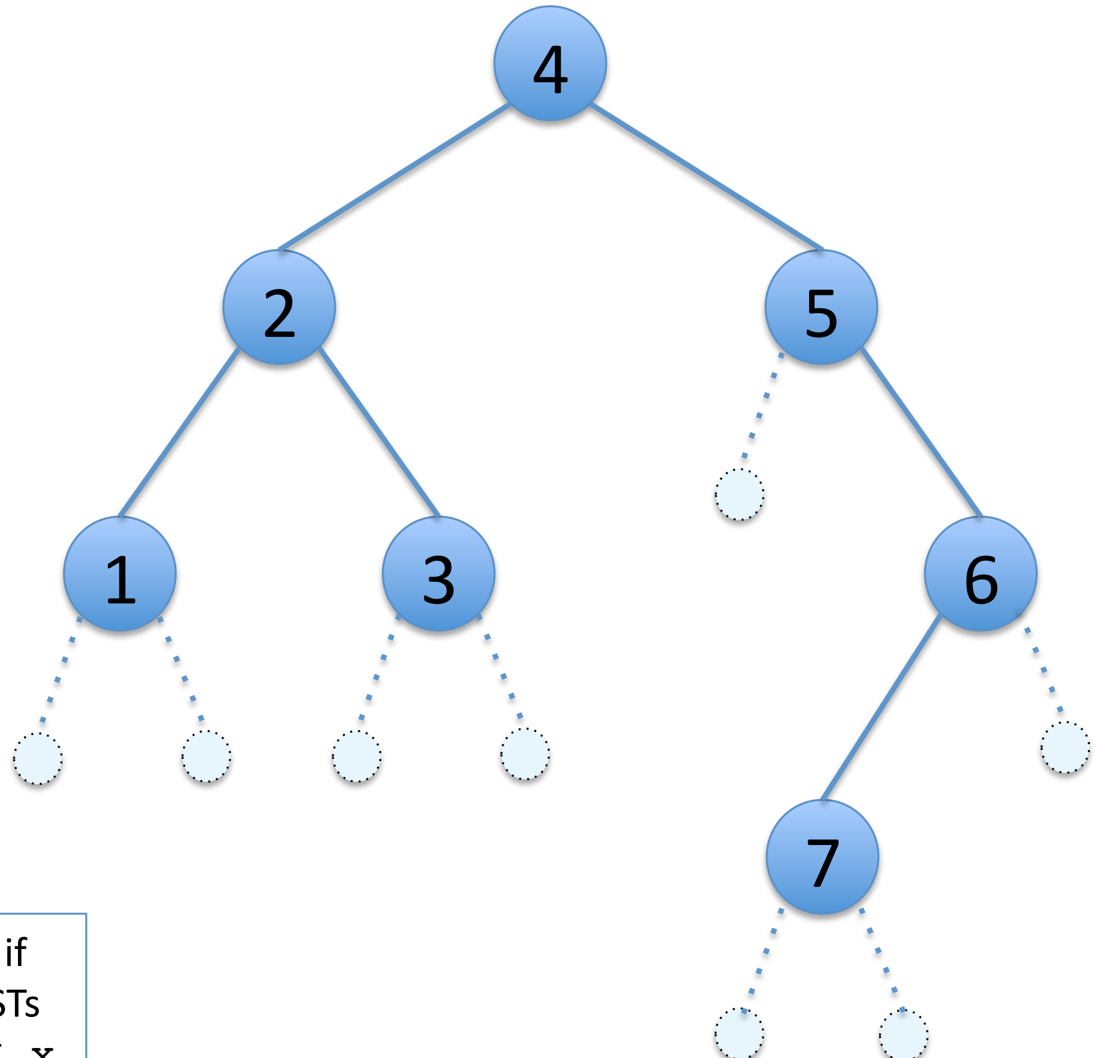
- `Node(lt,x,rt)` is a BST if
  - `lt` and `rt` are both BSTs
  - all nodes of `lt` are `<` `x`
  - all nodes of `rt` are `>` `x`
- `Empty` is a BST

Search in a BST: `(lookup t 8)`



Is this a BST??

1. yes
2. no

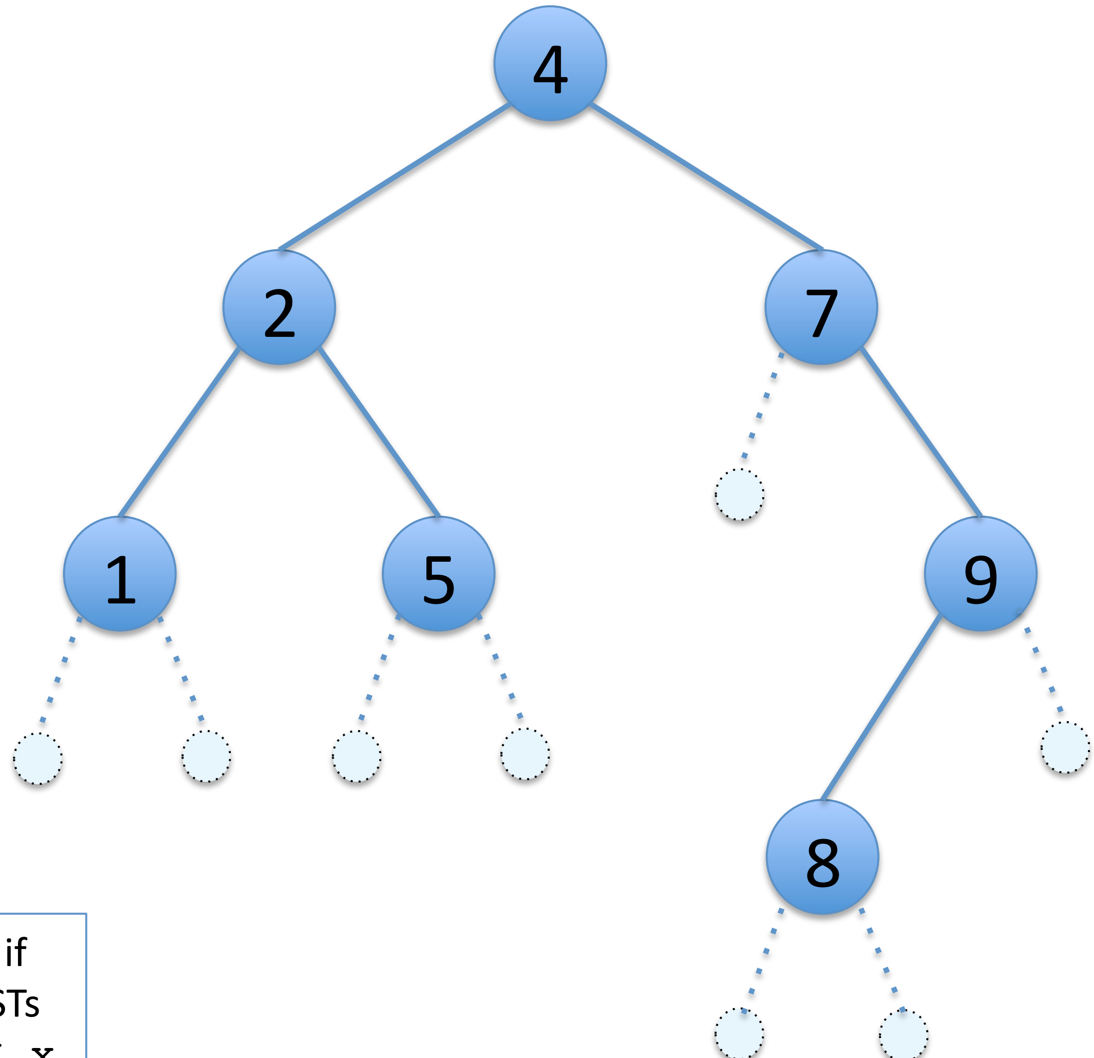


- `Node(lt,x,rt)` is a BST if
  - `lt` and `rt` are both BSTs
  - all nodes of `lt` are < `x`
  - all nodes of `rt` are > `x`
- `Empty` is a BST

Answer: no, 7 to the left of 6

Is this a BST??

1. yes
2. no

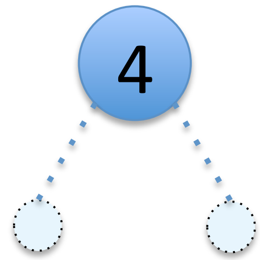


- `Node(lt,x,rt)` is a BST if
  - `lt` and `rt` are both BSTs
  - all nodes of `lt` are < `x`
  - all nodes of `rt` are > `x`
- `Empty` is a BST

Answer: no, 5 to the left of 4

Is this a BST??

1. yes
2. no



- `Node(lt,x,rt)` is a BST if
  - `lt` and `rt` are both BSTs
  - all nodes of `lt` are < `x`
  - all nodes of `rt` are > `x`
- `Empty` is a BST

Answer: yes