CSCI 241

Scott Wehrwein

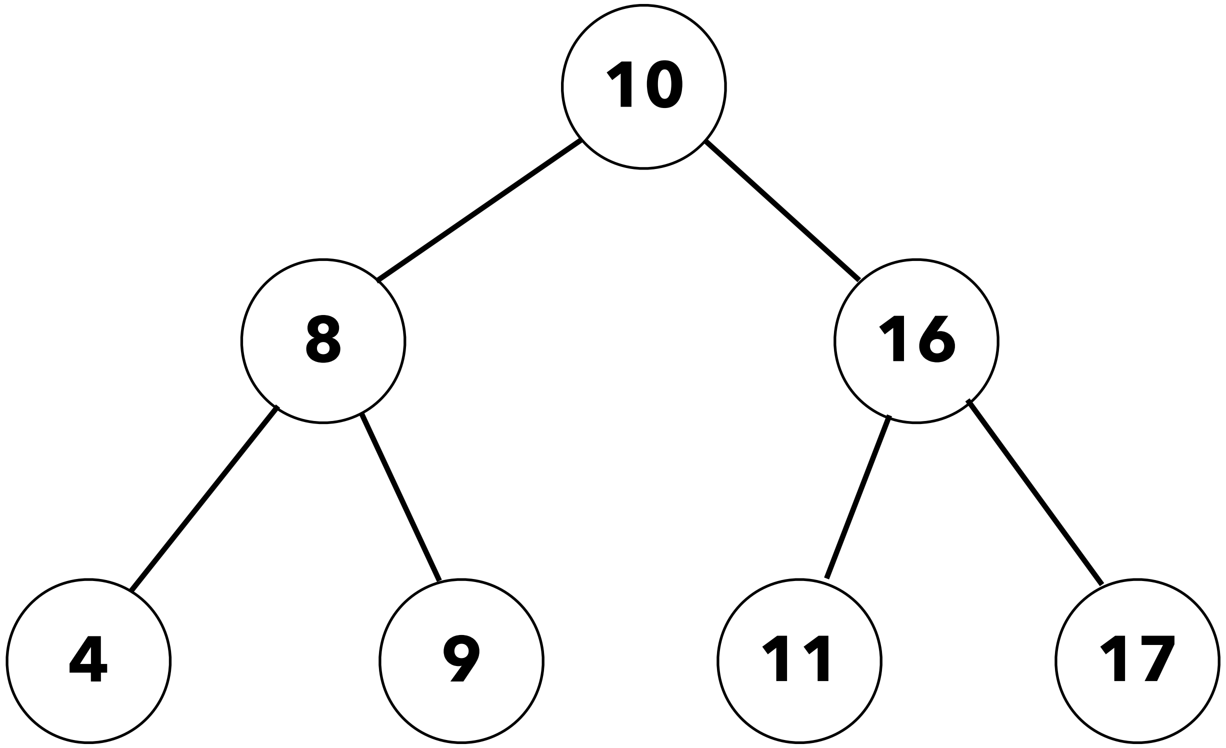
Binary Search Trees:
Runtime of BST Operations

Goals

Understand the best-case and worst-case runtime analysis of BST `add` and `contains`.

Searching a BST: What's the runtime?

```
boolean search(BST t, int v):
  if t == null:
    return false
  if t.value == v:
    return true
  if v < t.value:
    return search(t.left)
  else:
    return search(t.right)
```



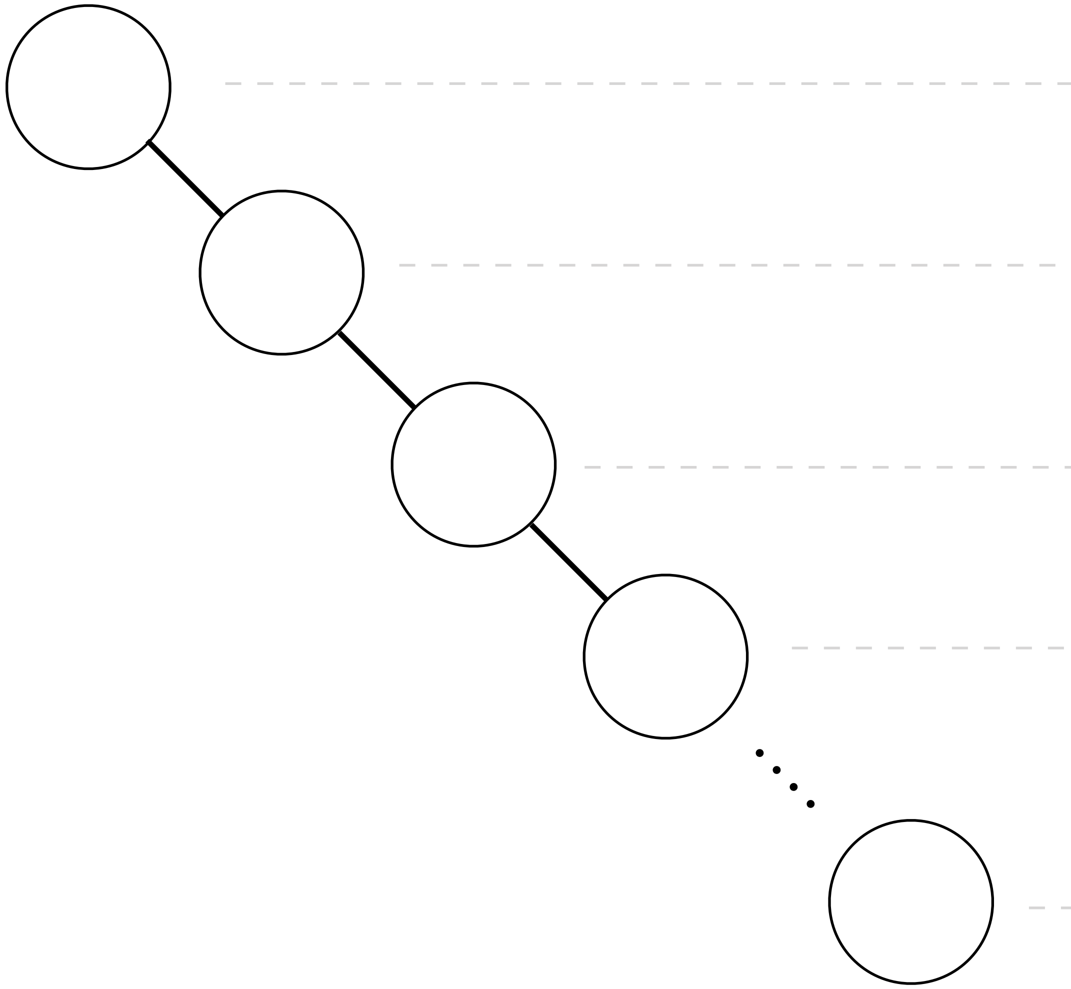
If h is the tree's **height**, search can visit at most h+1 nodes!
Runtime of search is O(**h**).

*That's great, but how does **h** relate to **n**?*

A tale of **h** and **n**

A binary search tree has height **h**.

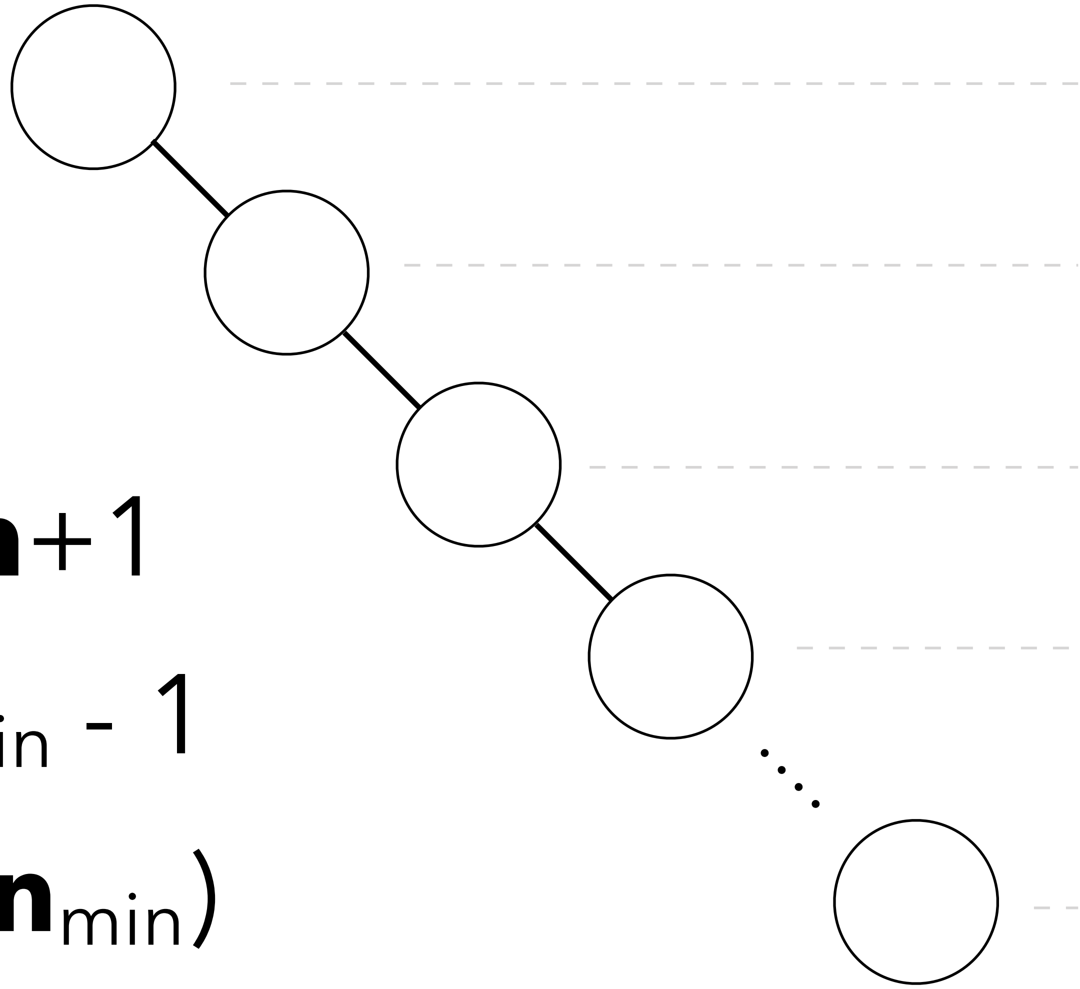
What is its *minimum* size $\mathbf{n}_{\text{min}}$?



h	$\mathbf{n}_{\text{min}}$
0	1
1	2
2	3
3	4
⋮	⋮
h	h+1

A tale of **h** and **n**

A binary search tree has height **h**.
What is its *minimum* size $\mathbf{n}_{\text{min}}$?



$\mathbf{h}$	$\mathbf{n}_{\text{min}}$
0	1
1	2
2	3
3	4
⋮	⋮
h	h+1

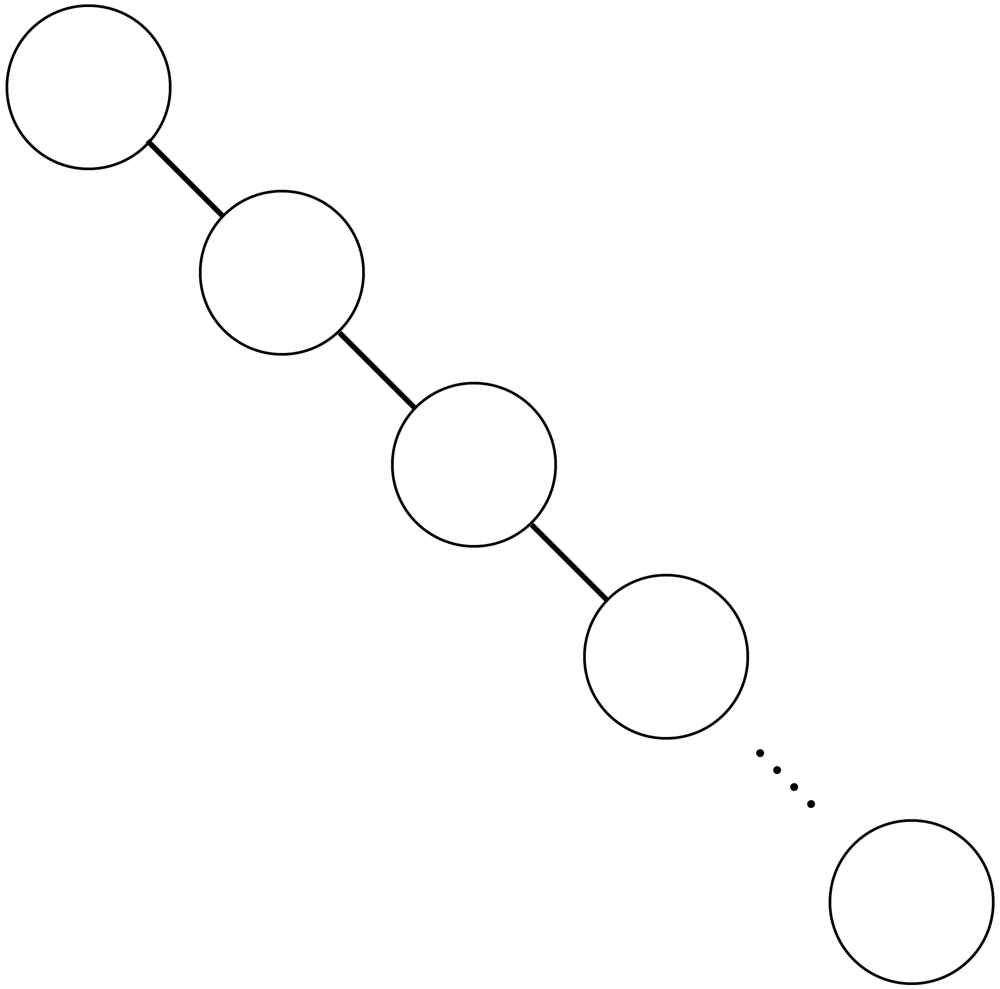
$\mathbf{n}_{\text{min}} = \mathbf{h}+1$

$\mathbf{h} = \mathbf{n}_{\text{min}} - 1$

$\mathbf{h}$ is $O(\mathbf{n}_{\text{min}})$

Searching a BST: What's the runtime?

```
boolean search(BST t, int v):
  if t == null:
    return false
  if t.value == v:
    return true
  if v < t.value:
    return search(t.left)
  else:
    return search(t.right)
```



If h is the tree's **height**, search can visit at most h+1 nodes!

Runtime of search is O(**h**).

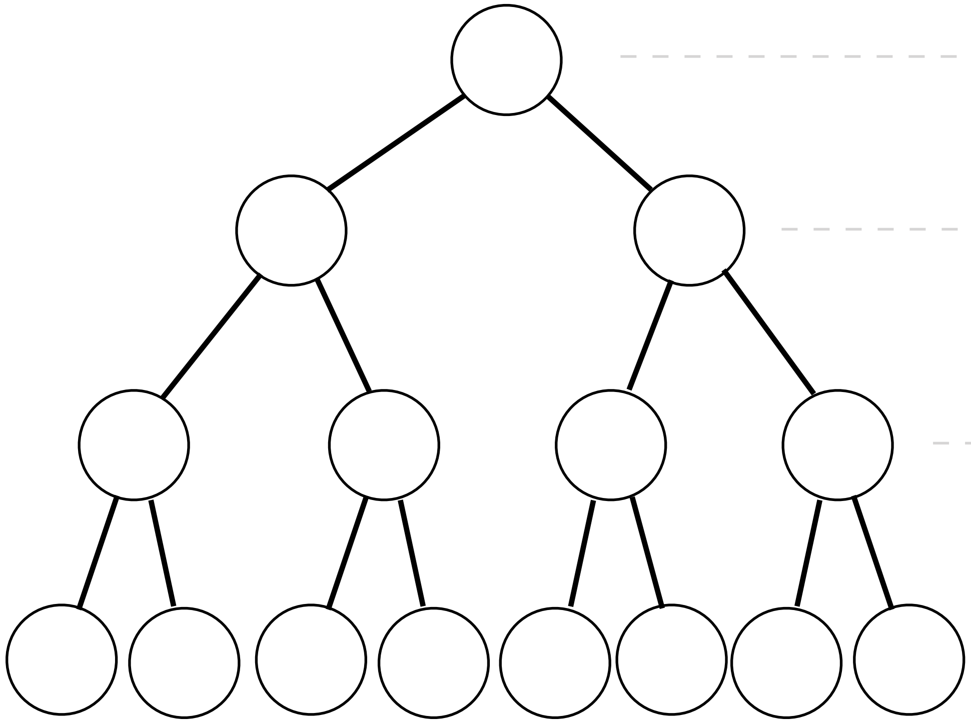
(worst-case)

*How does **h** relate to **n**?*

In a list-like tree, **h** is O(**n**), so search is O(**n**) 😢

A tale of **h** and **n**

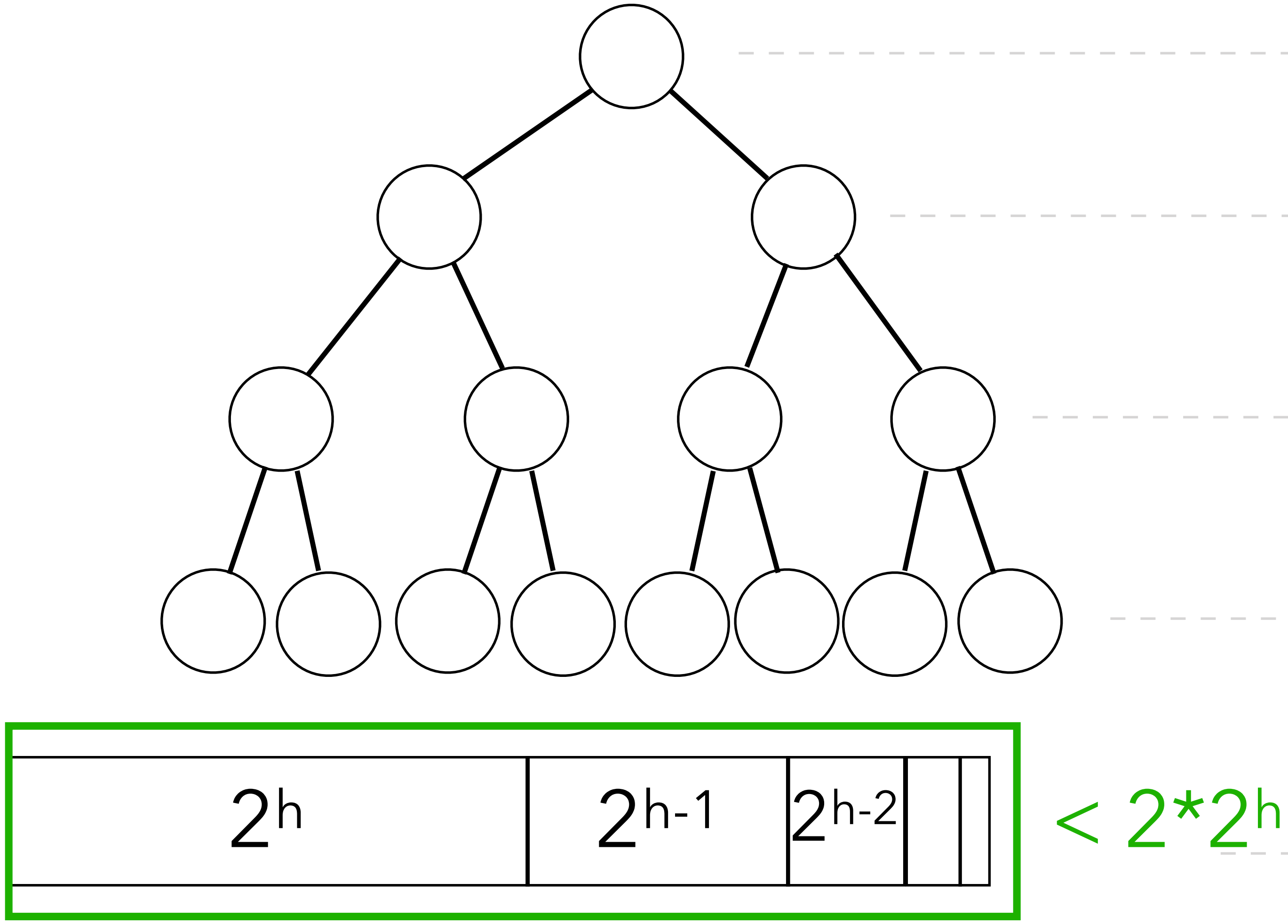
A binary search tree has height **h**.
What is its *maximum* size $\mathbf{n}_{\text{max}}$?



h	# leaves	$\mathbf{n}_{\text{max}}$
0	1	1
1	2	3
2	4	7
3	8	15
⋮	⋮	
h	2^h	2^{h+1}-1

A tale of **h** and **n**

A binary search tree has height **h**.
What is its *maximum* size $\mathbf{n}_{\text{max}}$?



h	# leaves	$\mathbf{n}_{\text{max}}$
0	1	1
1	2	3
2	4	7
3	8	15
⋮	⋮	
h	2^h	$2^{h+1}-1$

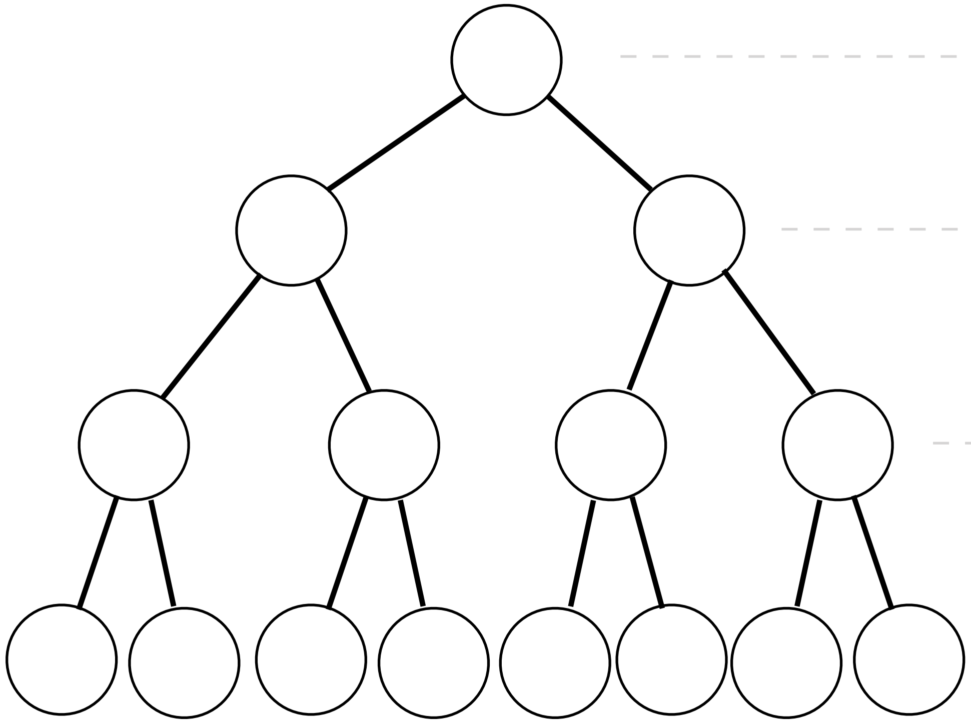
A tale of **h** and **n**

A binary search tree has height **h**.

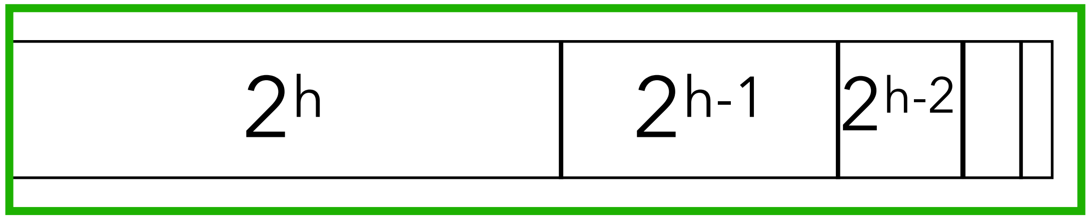
What is its *maximum* size $\mathbf{n}_{\text{max}}$?

$\mathbf{n}_{\text{max}}$ is $O(2^{\mathbf{h}})$

h is $O(\log \mathbf{n}_{\text{max}})$



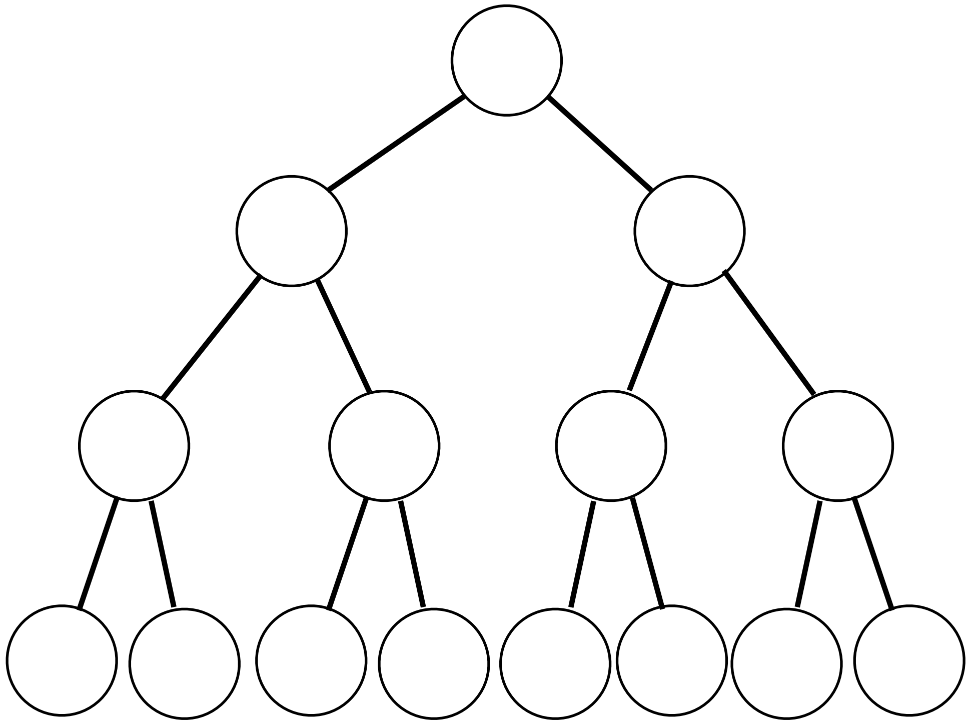
h	# leaves	$\mathbf{n}_{\text{max}}$
0	1	1
1	2	3
2	4	7
3	8	15
⋮	⋮	
h	2^h	$2^{h+1}-1$



$< 2*2^h$

Searching a BST: What’s the runtime?

```
boolean search(BST t, int v):
  if t == null:
    return false
  if t.value == v:
    return true
  if v < t.value:
    return search(t.left)
  else:
    return search(t.right)
```



If h is the tree’s **height**, search can visit at most h+1 nodes!

Runtime of search is O(**h**).

(best-case)

*How does **h** relate to **n**?*

In a *complete* tree, **h** is O(log **n**), so search is O(log **n**) 🎉

Set ADT: Possible Implementations

	contains	add	remove
LinkedList	O(n)	O(n)	O(n)
Array (sorted)	O(log n)	O(n)	O(n)
Array (unsorted)	O(n)	O(n)	O(n)
Binary Search Tree	$\boxed{O(n)}$ 😢	$\boxed{O(n)}$ 😢	??