

CSE 100: BSTS, BIG O, AND C++

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Welcome to CSE100" << endl;
    return 0;
}
```

Announcements

- PA0 due Tuesday @10am. PA1 coming Tuesday (noon). Will be due a week from the following Friday.
 - Concentrate on the reading assignment until then

CLICKERS OUT

Frequency BD

Q1: Write a line of code in C++ that declares an object, named **myNode** and initializes its data members to be the same value as those of another object named **otherNode**. Both objects are of type struct **Node**

A. `Node otherNode (myNode) ;`

B. `Node myNode (otherNode) ;`

C. `Node *myNode=otherNode ;`

D. `Node &myNode= otherNode ;`

copy constructor

Q2: True or false: In C++ you must explicitly call delete on every object that gets created to free up the memory used by that object.

☒ A. False

☐ B. True

Q3: If `myClass` is the name of a class, which of the following is true about the variable `x` in C++ statement: `myClass *x;`

A. It is an *object* of type `myClass`

☒ B. It is a *pointer* to objects of type `myClass`

C. It is a *constructor* of `myClass`

D. It is a *destructor* of `myClass`

Q4: Which of the following is true about the C++ statement: `p = new int[5]`

A. Memory is allocated to store 5 integers

B. Memory is allocated to store one integer with value '5'

C. Memory is allocated to store one integer using 5 bytes

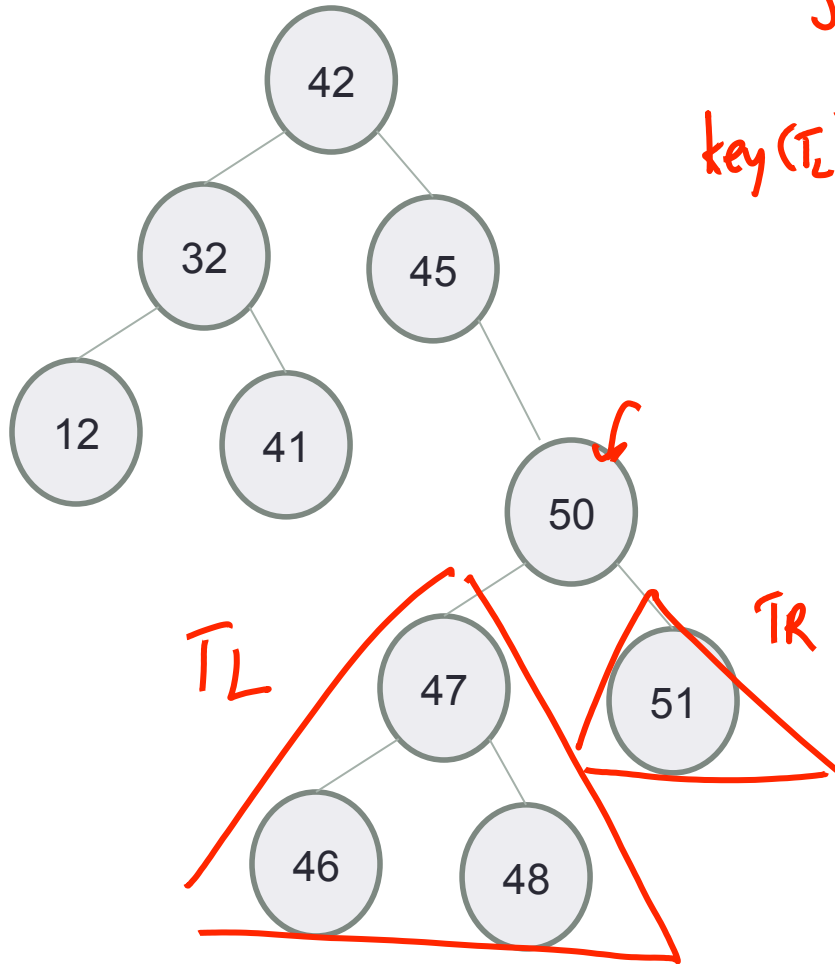
Q5: What is a C++ template?

- A. It is like a Java abstract class in that it is a class that may not specify implementations for all of its functions
- ☒ B. It is like a Java generic in that it allows you to make a class work with different types of data
- C. It is like a Java interface in that it specifies the functions which a class must implement

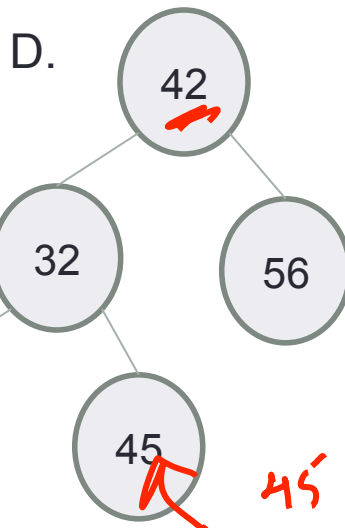
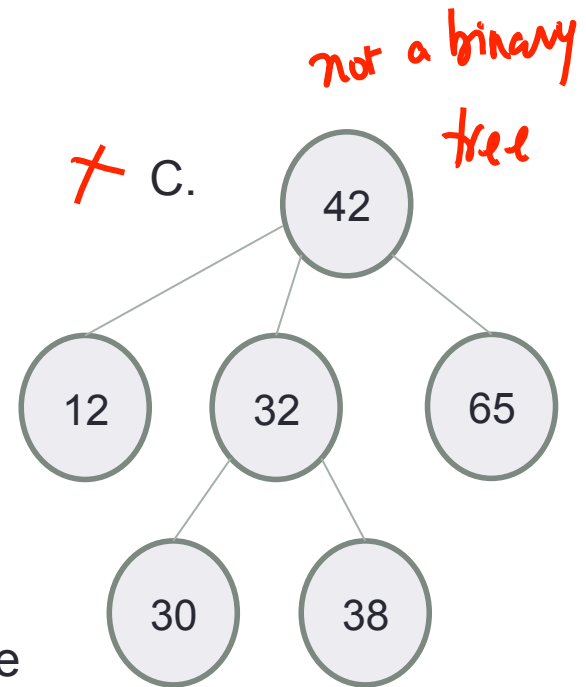
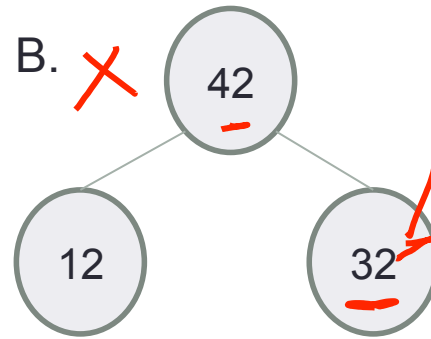
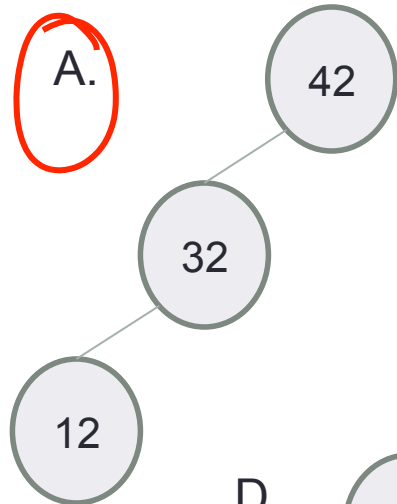
Binary Search Tree – What is it?

Search property

$$\text{key}(T_L) < x \leq \text{key}(T_R)$$



Which of the following is/are a binary search tree?

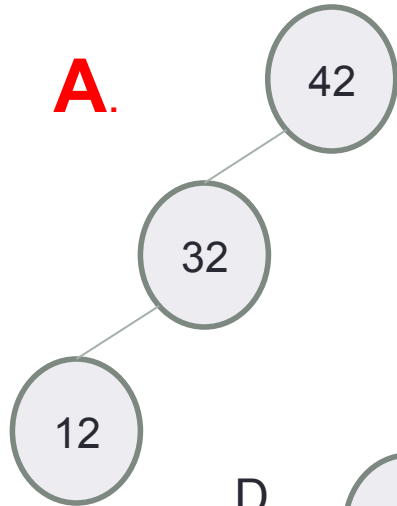


E. More than one of these

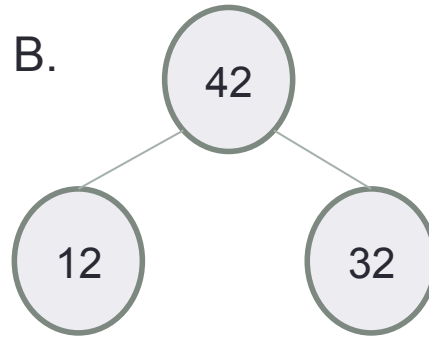
45 should not be in the left subtree of 42

Which of the following is/are a binary search tree?

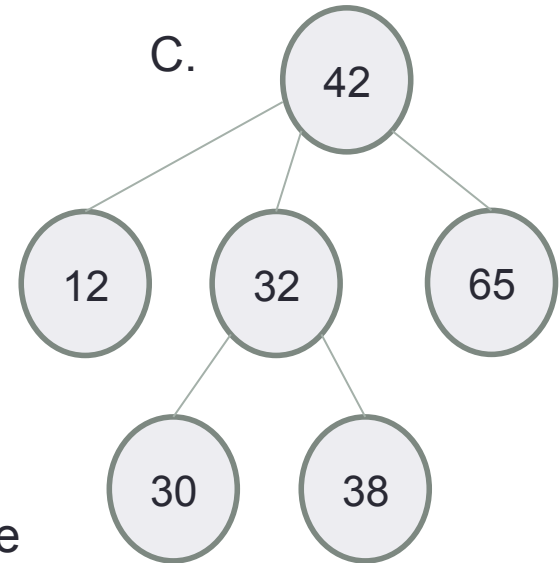
A.



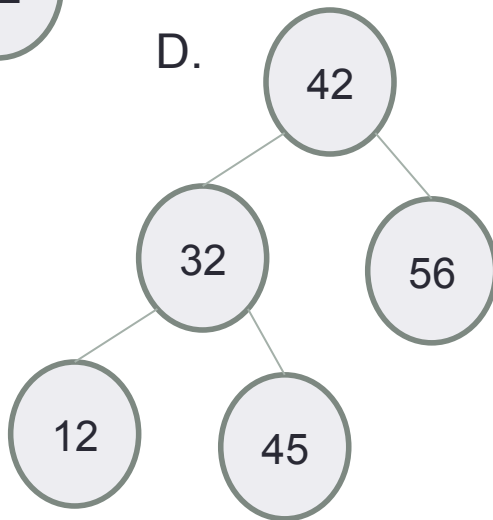
B.



C.



D.



E. More than one of these

Binary Search Trees

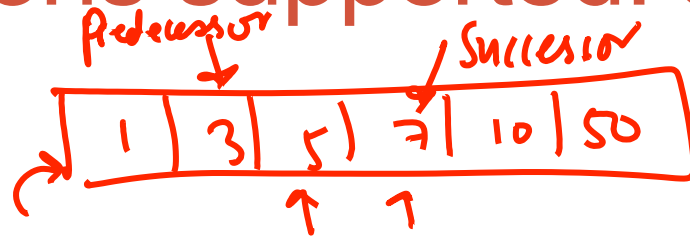
- What are the operations supported and their running times? *↑ today*
- Why do we get these running times? *Analysis (Later lectures)*
- How do you implement the data structure (and operations supported by it)? *PA1*

Binary Search Trees

- What is it good for?
 - If it satisfies a special property i.e. Balanced, you can think of it as a dynamic version of the sorted array

i.e. Supports all operations that are supported by the sorted array + Fast Insert & Delete!

Operations supported: sorted array



Running Time

(sorted array)

- | | |
|--|-----------------------------|
| 1. Search | $O(\log_2 N)$ Binary Search |
| 2. Selection — the i^{th} largest element | $O(1)$ |
| 3. Min/Max | $O(1)$ |
| 4. Predecessor/ Successor | $O(1)$ |
| 5. Rank: <u>given key</u> : how many elements smaller than key | $O(\log_2 N)$ |
| 6. Output in sorted order | $O(N)$ |
| 7. Insert | $O(N)$ |
| 8. Delete | $O(N)$ |
- } ← slow here

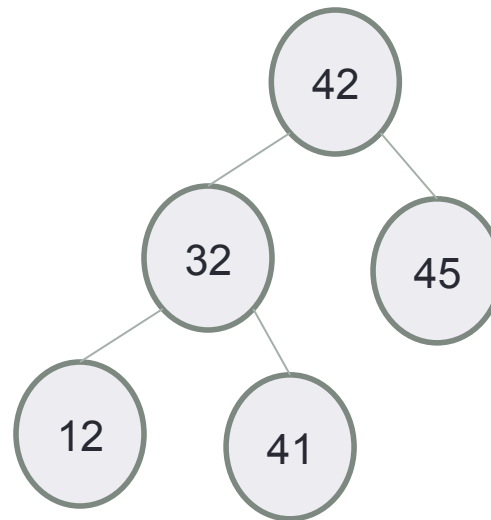
Operations supported: Balanced Binary Search Trees vs. Sorted Array

Operations	Running Time	
	(Balanced-BST)	(sorted array)
Search	$O(\log_2 n)$	$O(\log_2 n)$
Selection	$O(\log_2 n)$	$O(1)$
Min/Max	$O(\log_2 n)$	$O(1)$
Predecessor/ Successor	$O(\log_2 n)$	$O(1)$
Rank	$O(\log_2 n)$	$O(\log_2 n)$
Output in sorted order	$O(n)$	$O(n)$
Insert	$O(\log_2 n)$	$O(n)$
Delete	$O(\log_2 n)$	$O(n)$

Under the hood of the BST: Searching an element in the tree

To search for element with key k

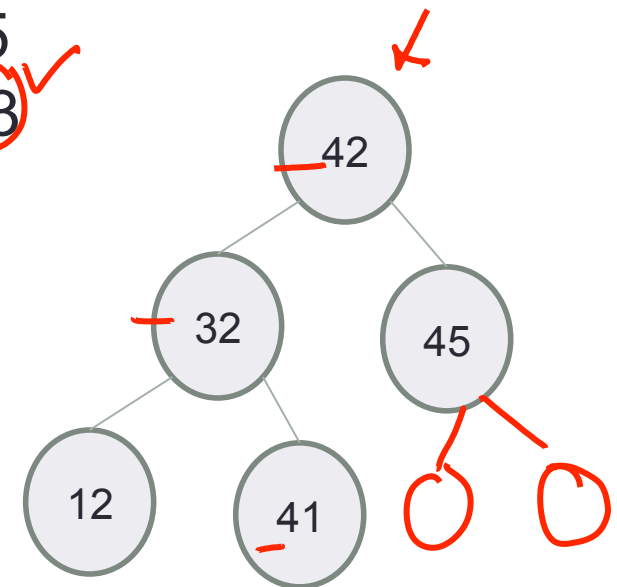
1. Start at the root
2. If $k < \text{key}(\text{root})$, recursively search T_L
Else recursively search T_R



Under the hood of the BST: Searching an element in the tree

Q: In the worst case how many comparisons do we need to find an element in any ^{BST} BST and in particular the given ^{BST} BST

- A. The number of nodes in the tree, 5
- B. The number of leaves in the tree, 3 ✓
- C. The number of roots in the tree, 1
- ☒ D. None of the above

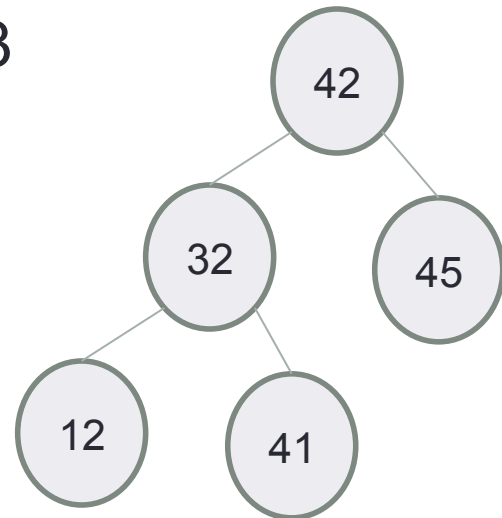


if 45 had 2 children, BST would have 4 leaves but no. of compares to find 41 would be 3 ←

Under the hood of the BST: Searching an element in the tree

Q: In the worst case how many comparisons do we need to find an element in any tree and in particular the given tree

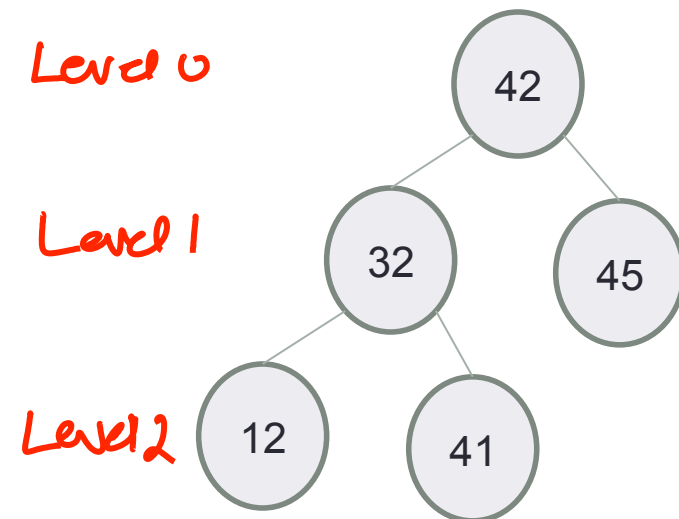
- A. The number of nodes in the tree, 5
- B. The number of leaves in the tree, 3
- C. The number of roots in the tree, 1
- D. None of the above



Towards describing the running time of the search operation: Height of the tree

Height of a node: Longest path from node to any leaf node plus one
Height of the tree: Height of the root

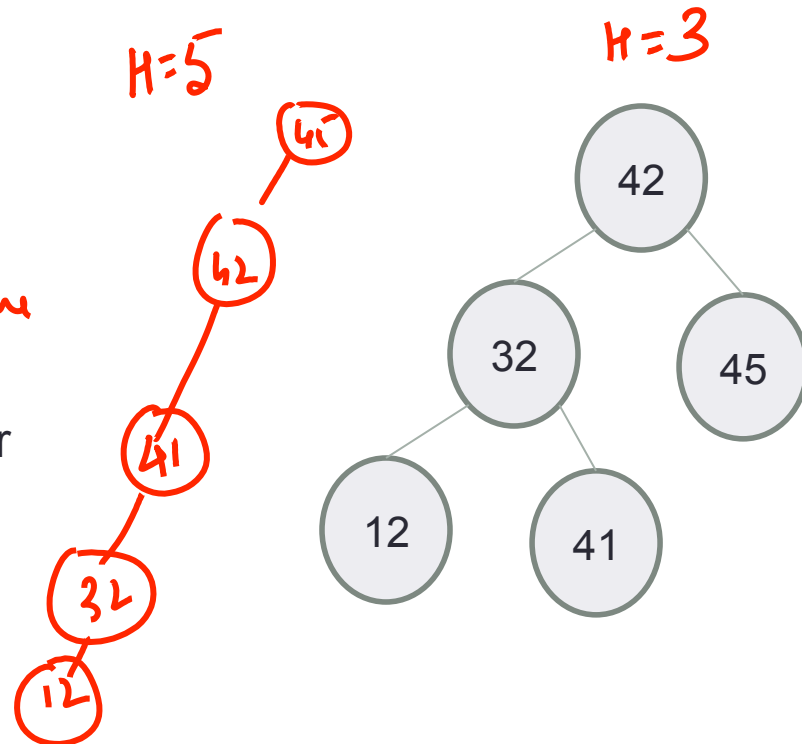
What is the height of the given tree? 3



What about the running time of Search?

How long does it take to find an element in a tree with N keys and height H in the worst case?

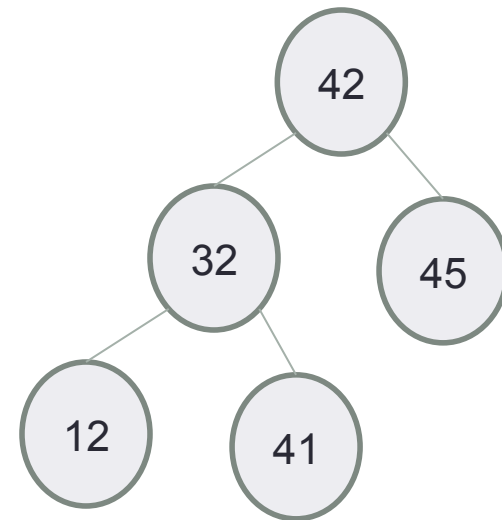
- A. $O(1)$
- B. $O(\log_2 N)$
- ☒ C. $O(H)$ *but H may vary, in the worst case $H = N$*
- D. $O(N)$
- E. More than one correct answer



What about the running time of Search?

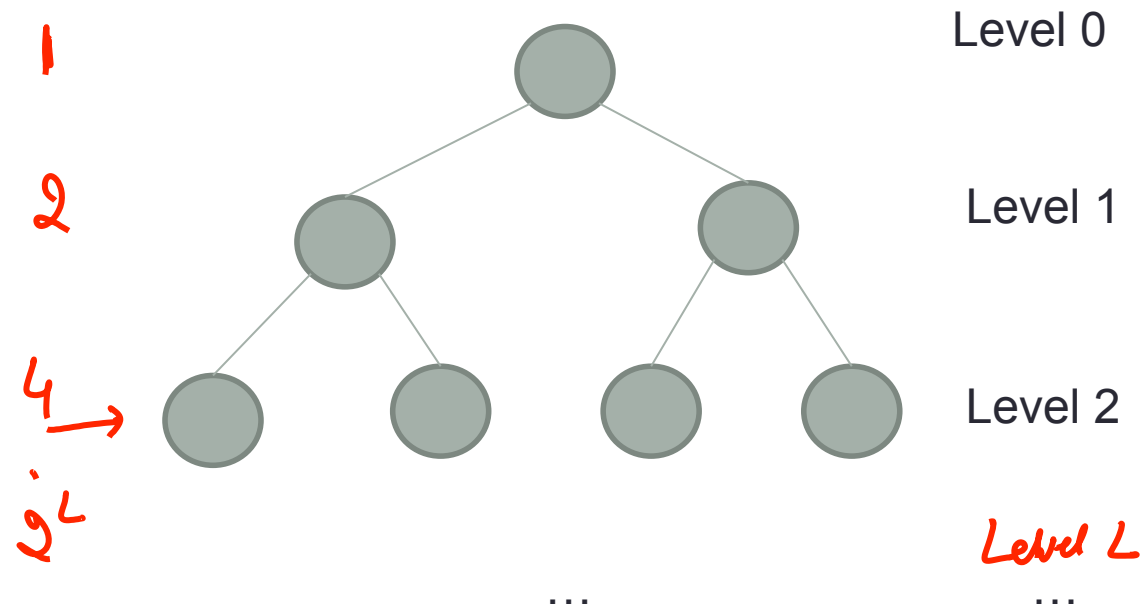
How long does it take to find an element in a tree with N keys and height H in the worst case?

- A. $O(1)$
- B. $O(\log_2 N)$
- C. $O(H)$
- D. $O(N)$
- E. More than one correct answer



Is the height of the tree (H) and the number of keys (N) related?

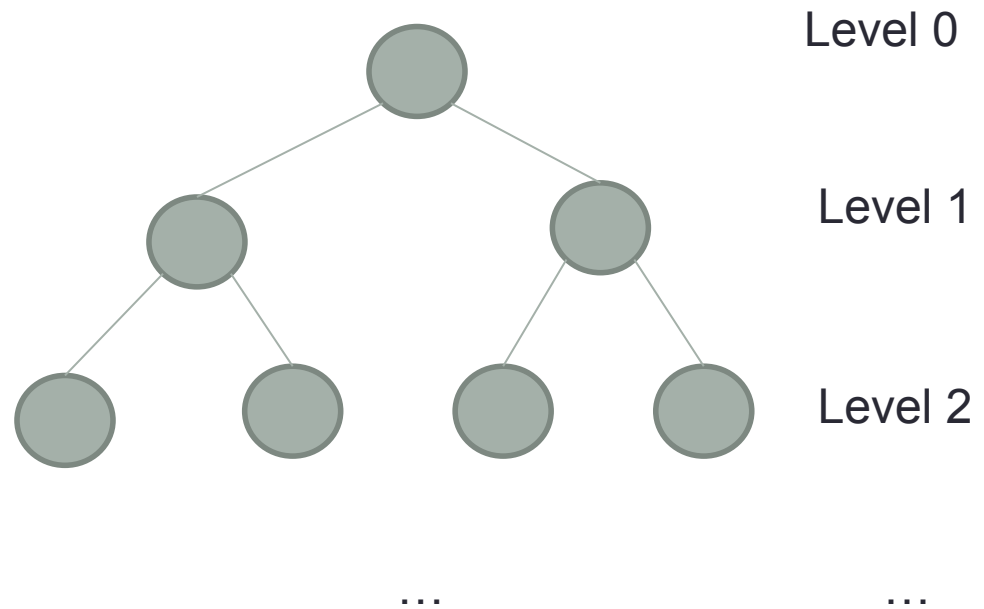
Relating H (height) and N (#nodes) for a Balanced BST



How many nodes are on level L in a completely filled binary search tree?

- A. 2
- B. L
- C. $2 * L$
- ☒ D. 2^L

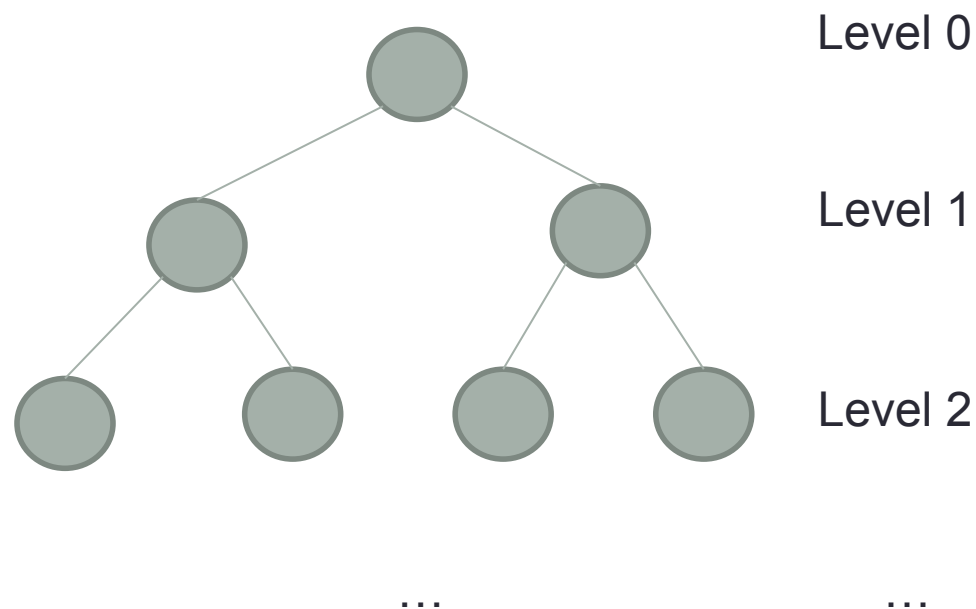
Relating H (height) and N (#nodes) for a Balanced BST



How many nodes are on level L in a completely filled binary search tree?

- A. 2
- B. L
- C. $2 * L$
- D. 2^L

Relating H (height) and N (#nodes)



How many nodes are in a completely filled BST?

A. $N = \sum_{L=0}^{H-1} 2^L$

B. $N = 2^L$

C. $N = \sum_{H=0}^N 2^H$

D. $N = 2^{H+L} - 1$