BST RUNNING TIME ANALYSIS

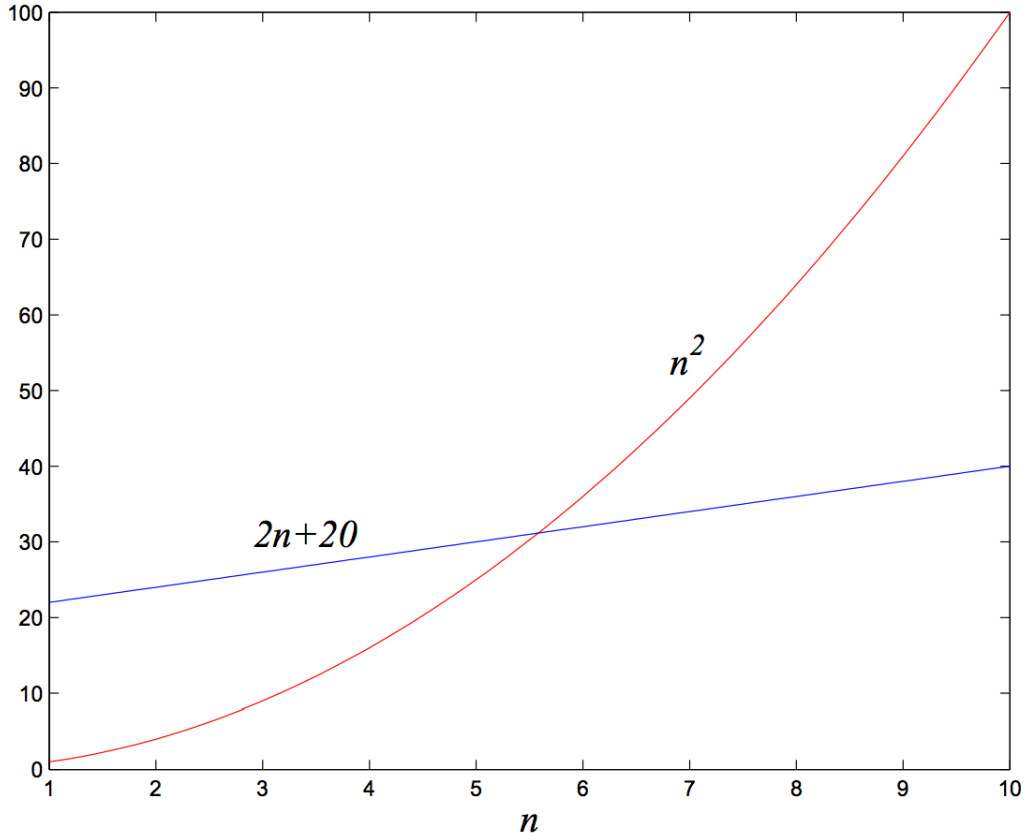
Problem Solving with Computers-II

Big-Omega

- f(n) and g(n) map positive integer inputs to positive reals.

We say $f = \Omega(g)$ if there are constants $c > 0, k>0$ such that
$c \cdot g(n) \leq f(n)$ for $n >= k$

$f = \Omega(g)$
means that "f grows at least as fast as g"

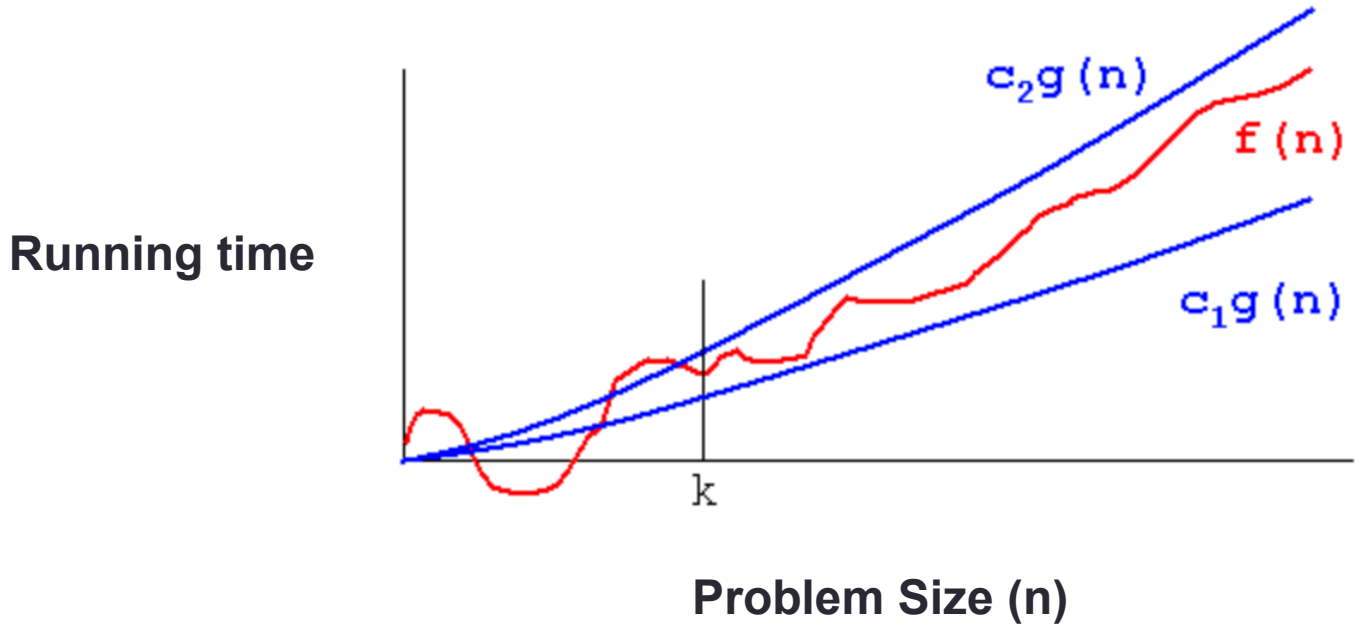


Big-Theta

- f(n) and g(n) map positive integer inputs to positive reals.

We say $f = \Theta(g)$ if there are constants c_1, c_2, k such that

$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$, for $n >= k$



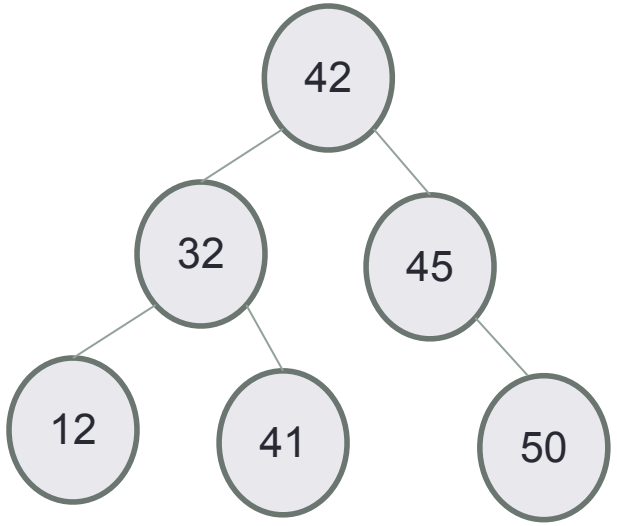
Binary Search Trees

- WHAT are the operations supported?
- HOW do we implement them?
- WHAT are the (worst case) running times of each operation?

- Path – a sequence of nodes and edges connecting a node with another node.
- A path starts from a node and ends at another node or a leaf
- Height of node – The height of a node is the number of edges on the longest downward path between that node and a leaf.

BSTs of different heights are possible with the same set of keys
Examples for keys: 12, 32, 41, 42, 45

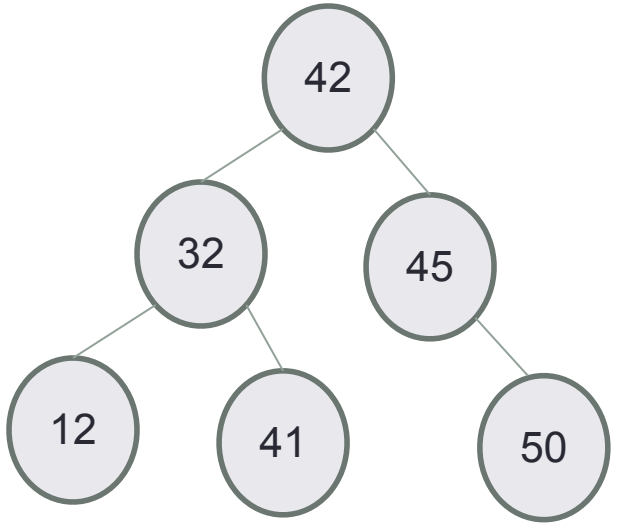
Worst case Big-O of search, insert, min, max



Given a BST of height H with N nodes, what is the worst case complexity of searching for a key?

A. O(1)
B. O(log H)
C. O(H)
D. O(H*log H)
E. O(N)

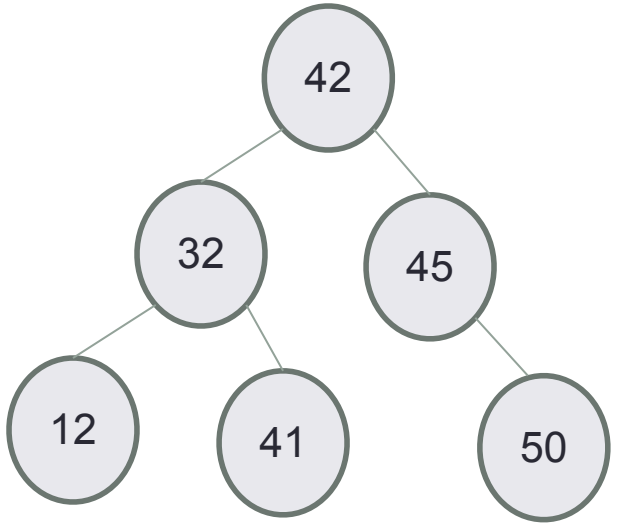
Worst case Big-O of predecessor / successor



Given a BST of height H and N nodes, what is the worst case complexity of finding the predecessor or successor key?

A. O(1)
B. O(log H)
C. O(H)
D. O(H*log H)
E. O(N)

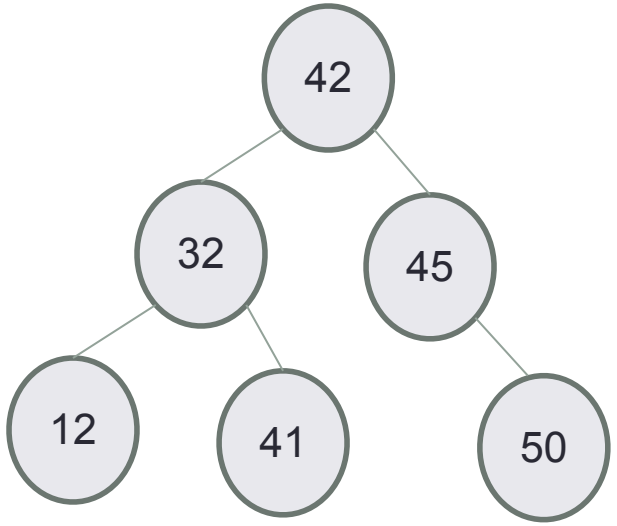
Worst case Big-O of delete



Given a BST of height H and N nodes, what is the worst case complexity of deleting a node?

A. O(1)
B. O(log H)
C. O(H)
D. O(H*log H)
E. O(N)

Big O of traversals

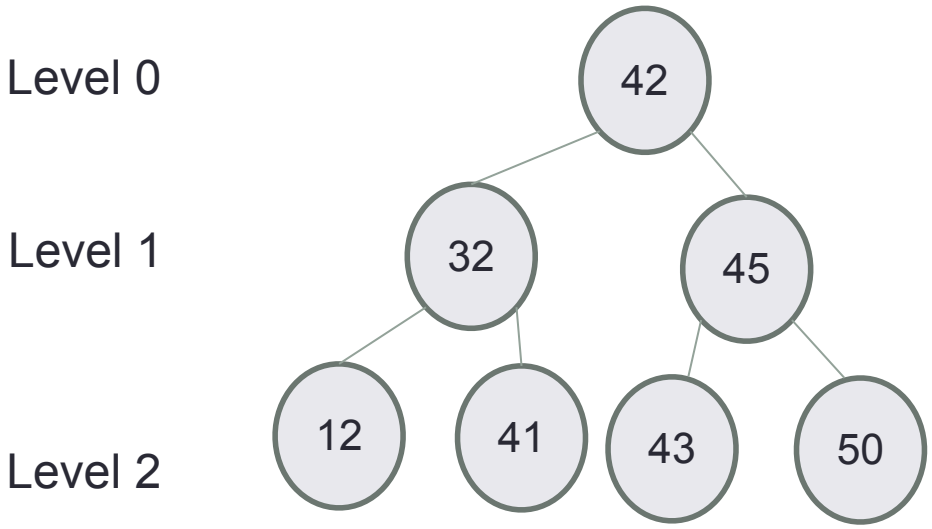


In Order:

Pre Order:

Post Order:

Types of BSTs

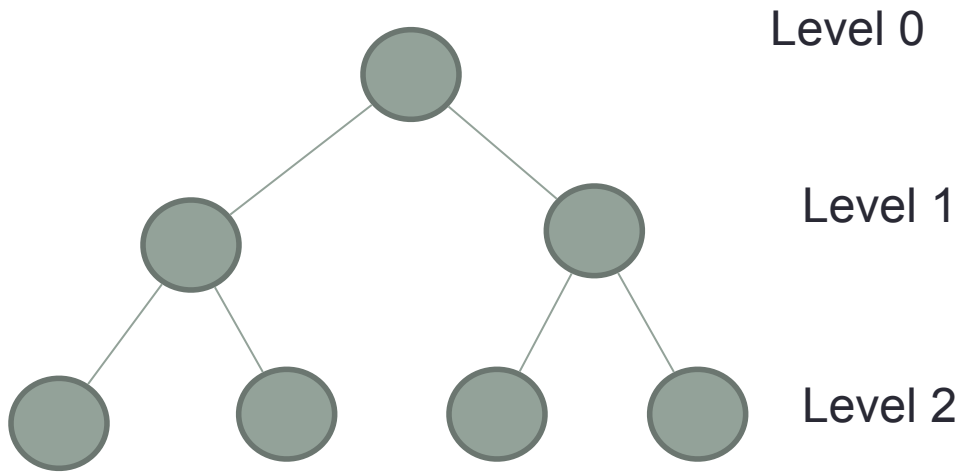


Balanced BST:

Full Binary Tree: Every node other than the leaves has two children.

Complete Binary Tree: Every level, except possibly the last, is completely filled, and all nodes are as far left as possible

Relating H (height) and N (#nodes)



What is the height (exactly) of a full binary tree in terms of N?

Balanced trees

- Balanced trees by definition have a height of O(log N)
- A completely filled tree is one example of a balanced tree
- Other Balanced BSTs include AVL trees, red black trees and so on
- Visualize operations on an AVL tree: https://visualgo.net/bn/bst

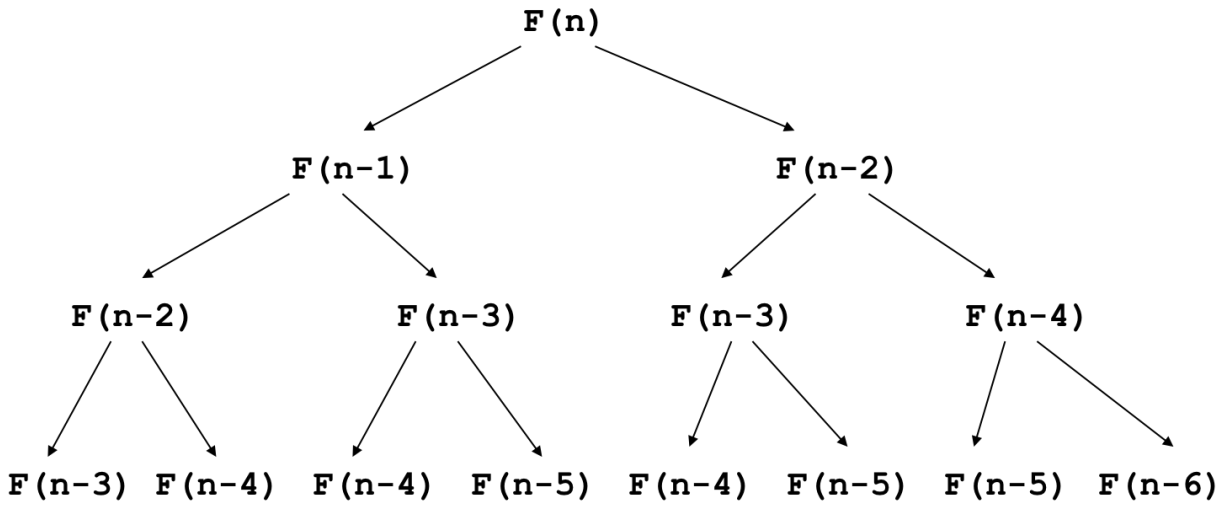
Big-O analysis of iterative Fibonacci

```
function F(n){
 Create an array fib[1..n]
 fib[1] = 1
 fib[2] = 1
 for i = 3 to n:
    fib[i] = fib[i-1] + fib[i-2]
 return fib[n]
}
```

Big-O analysis of recursive Fibonacci

```
function F(n){
    if(n == 1) return 1
    if(n == 2) return 1
return F(n-1) + F(n-2)
}
```

What takes so long? Let's unravel the recursion…



The same subproblems get solved over and over again!