BST RUNNING TIME ANALYSIS

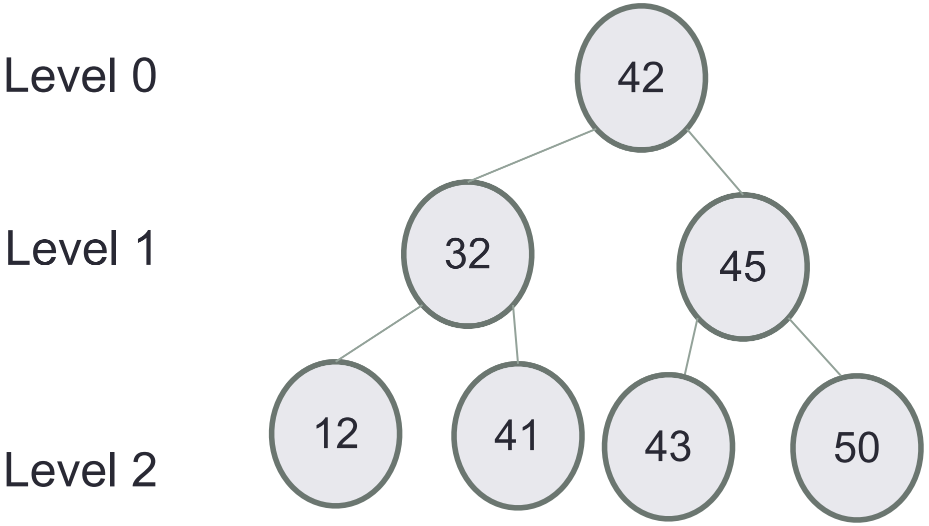
Problem Solving with Computers-II

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;

}
```

Types of BSTs

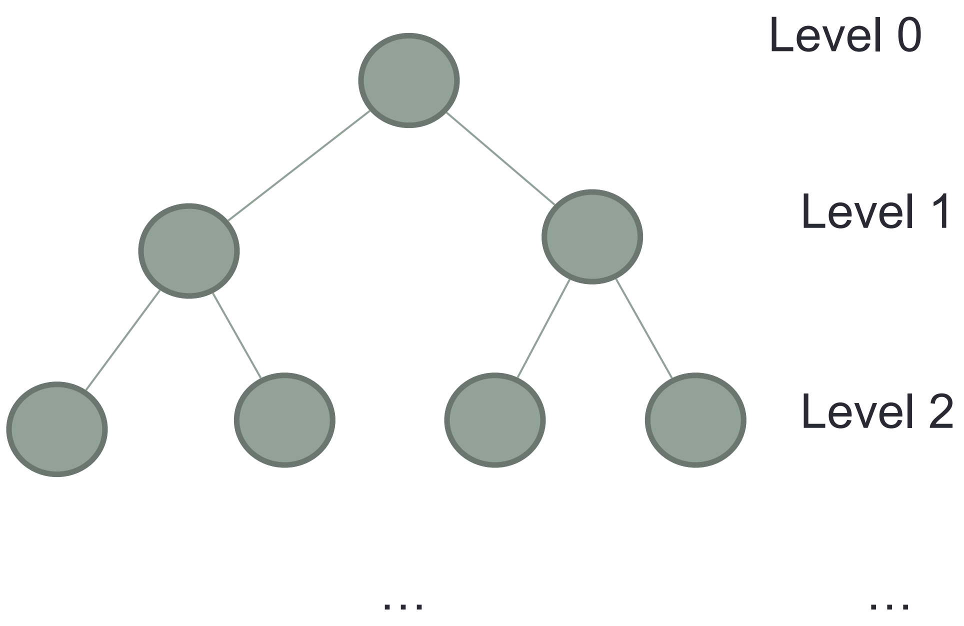


Balanced BST:

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) for a full binary tree

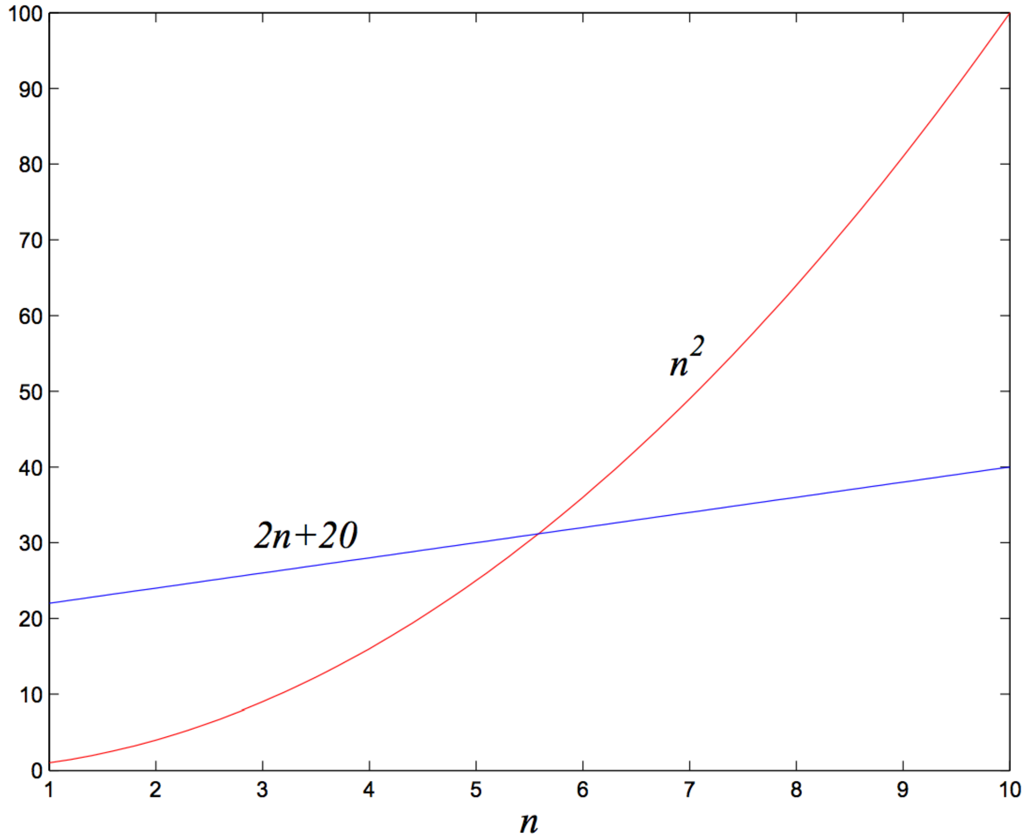


Big-Omega

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

We say f = Ω(g) if there are constants c > 0, k>0 such that
c · g(n) ≤ f(n) for n >= k

f = Ω(g)
means that "f grows at least as fast as g"

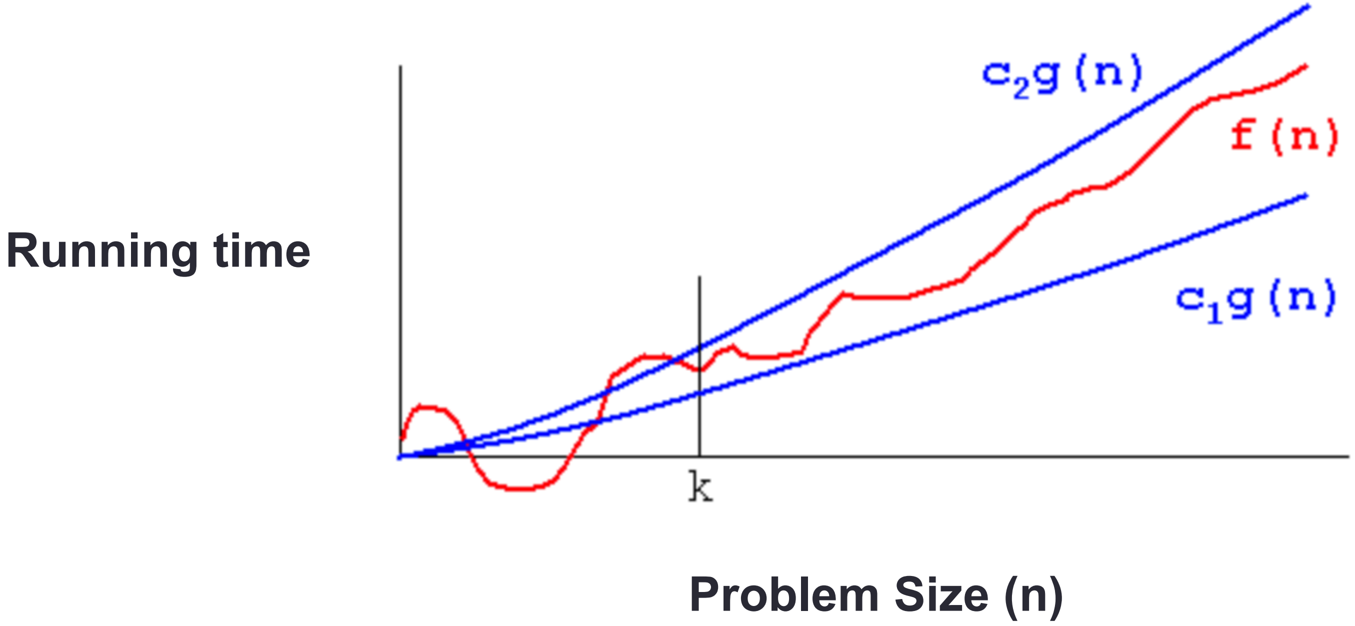


Big-Theta

- 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 $n >= k$



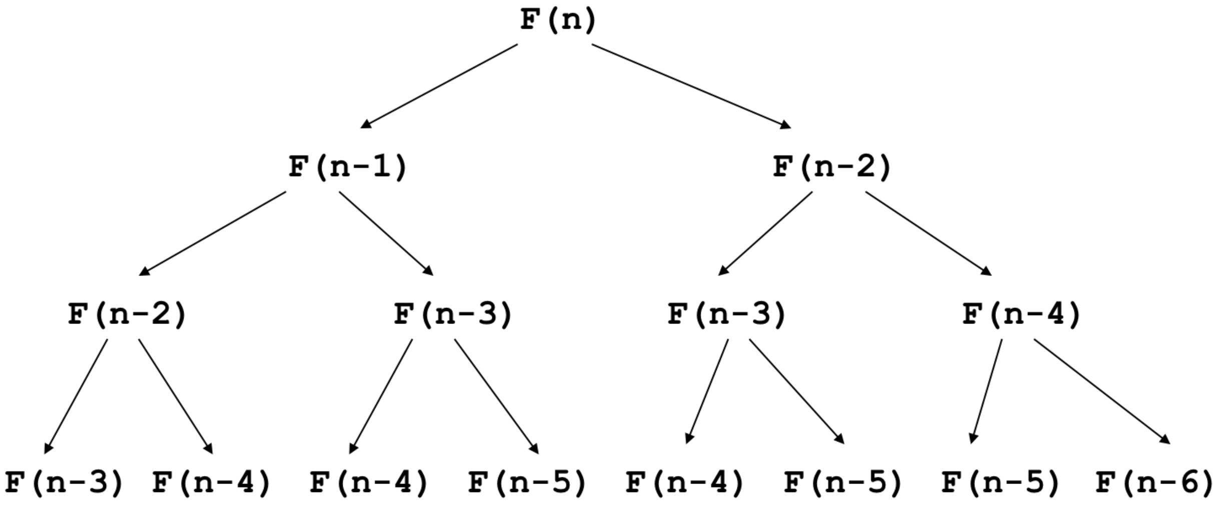
Big-O analysis of iterative fibonacci

```
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

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

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



The same subproblems get solved over and over again!

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

Summary of operations

Operation	Sorted Array	Binary Search Tree	Linked List
Min			
Max			
Median			
Successor			
Predecessor			
Search			
Insert			
Delete			