Data Structures

AVL Tree Proof (and BTree)

CS 225
Brad Solomon & G Carl Evans

October 2, 2023

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science

Learning Objectives

Review and finish AVL proof

Discuss alternatives to BSTs

AVL Tree Analysis

For an AVL tree of height h:

Find runs in: ___O(h)___.

Insert runs in: ___O(h)___.

Remove runs in: ___O(h)___.

Claim: The height of the AVL tree with n nodes is: ___O(log(n))___.

Plan of Action

Since our goal is to find the lower bound on **n** given **h**, we can begin by defining a function given **h** which describes the smallest number of nodes in an AVL tree of height **h**:

$N(h)$ = minimum number of nodes in an AVL tree of height h

Simplify the Recurrence

$$N(h) = 1 + N(h-1) + N(h-2)$$

State a Theorem

Theorem: An AVL tree of height h has at least ___________.

Proof by Induction:

I. Consider an AVL tree and let **h** denote its height.

II. Base Case: ________________

An AVL tree of height ____ has at least ____ nodes.

Prove a Theorem

III. Base Case: ________________

An AVL tree of height ____ has at least ____ nodes.

Prove a Theorem

IV. Induction Case: ________________

Assume for all heights $i < h$, $N(i) \geq 2^{i/2}$. Prove that $N(h) \geq 2^{h/2}$

Prove a Theorem

V. Using a proof by induction, we have shown that:

…and inverting:

AVL Runtime Proof

An upper-bound on the height of an AVL tree is **O(lg(n))**:

N(h) := Minimum # of nodes in an AVL tree of height h

N(h) = 1 + **N(h-1)** + **N(h-2)**

$> 1 + 2^{(h-1)/2} + 2^{(h-2)/2}$

$> 2 \times 2^{(h-2)/2} = 2^{(h-2)/2+1} = 2^{h/2}$

Theorem #1:

Every AVL tree of height h has at least $2^{h/2}$ nodes.

Summary of Balanced BST

AVL Trees

- Max height: 1.44 * lg(n)
- Rotations:

    Zero rotations on find

    One rotation on insert

    O(**h**) == O(**lg(n)**) rotations on remove

Red-Black Trees

- Max height: 2 * lg(n)
- Constant number of rotations on insert (max 2), remove (max 3).

Summary of Balanced BST

Pros:

- Running Time:
  - Improvement Over:
- Great for specific applications:

Summary of Balanced BST

Cons:

- Running Time:

- In-memory Requirement:

Considering hardware limitations

Can we always fit our data in main memory?

Where else can we keep our data?

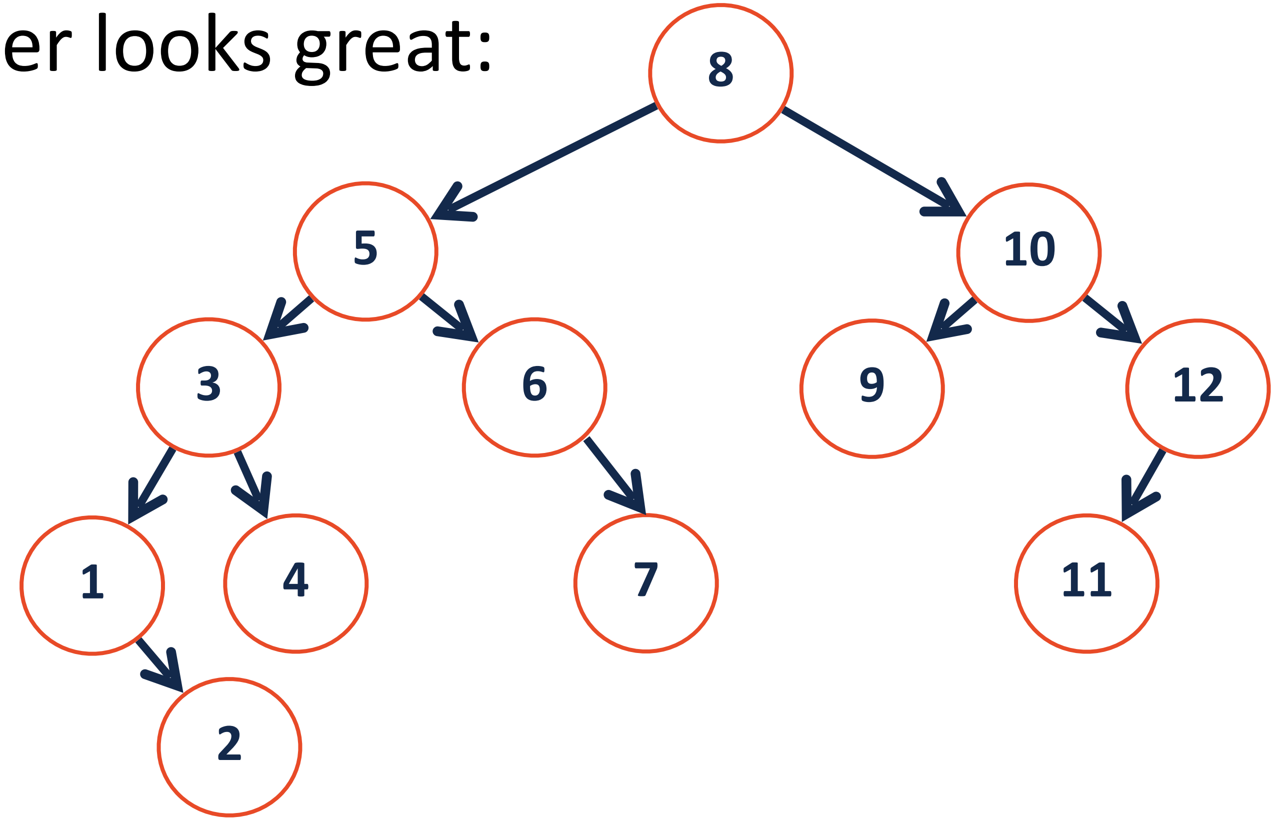
Does this match our assumption that all memory lookups are O(1)?

B-Tree Motivation

In Big-O we have assumed uniform time for all operations, but this isn't always true.

However, seeking data from the cloud may take 40ms+.

…an O(lg(n)) AVL tree no longer looks great:



BTree Design Motivations

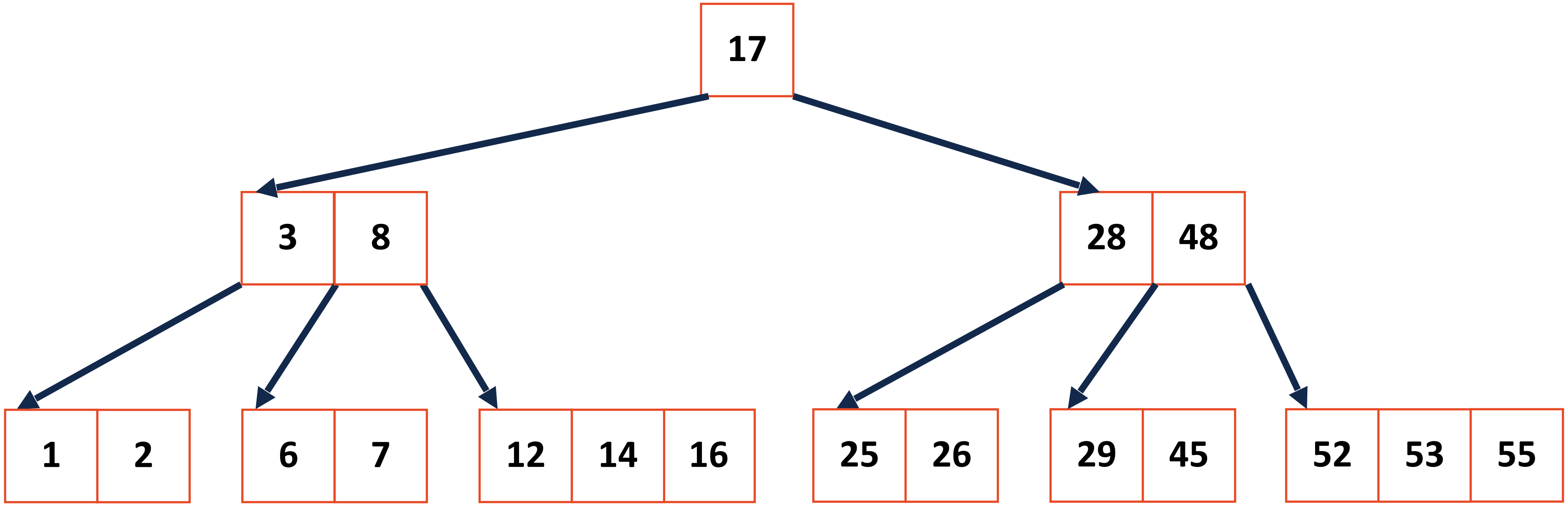
When large seek times become an issue, we address this by:

BTree

A BTree (of order m) is a m-ary tree

Nodes contain up to **m-1** keys and have **|keys|+1** children

All leaves in a BTree are on the same level



BTree Node (of order m)

$\mathbf{M} \geq \mathbf{5}$

-3	5	8	13

BTree Node (of order m)

What value of **m** should we be using?

BTree Insertion

M = 5

All keys within a BTree are ordered

Insert(10)

Insert(5)

Insert(7)

Insert(9)

Insert(2)

BTree Insertion

M = 5

When a BTree node reaches **m** keys (or when you try to insert):

2	5	7	9	10

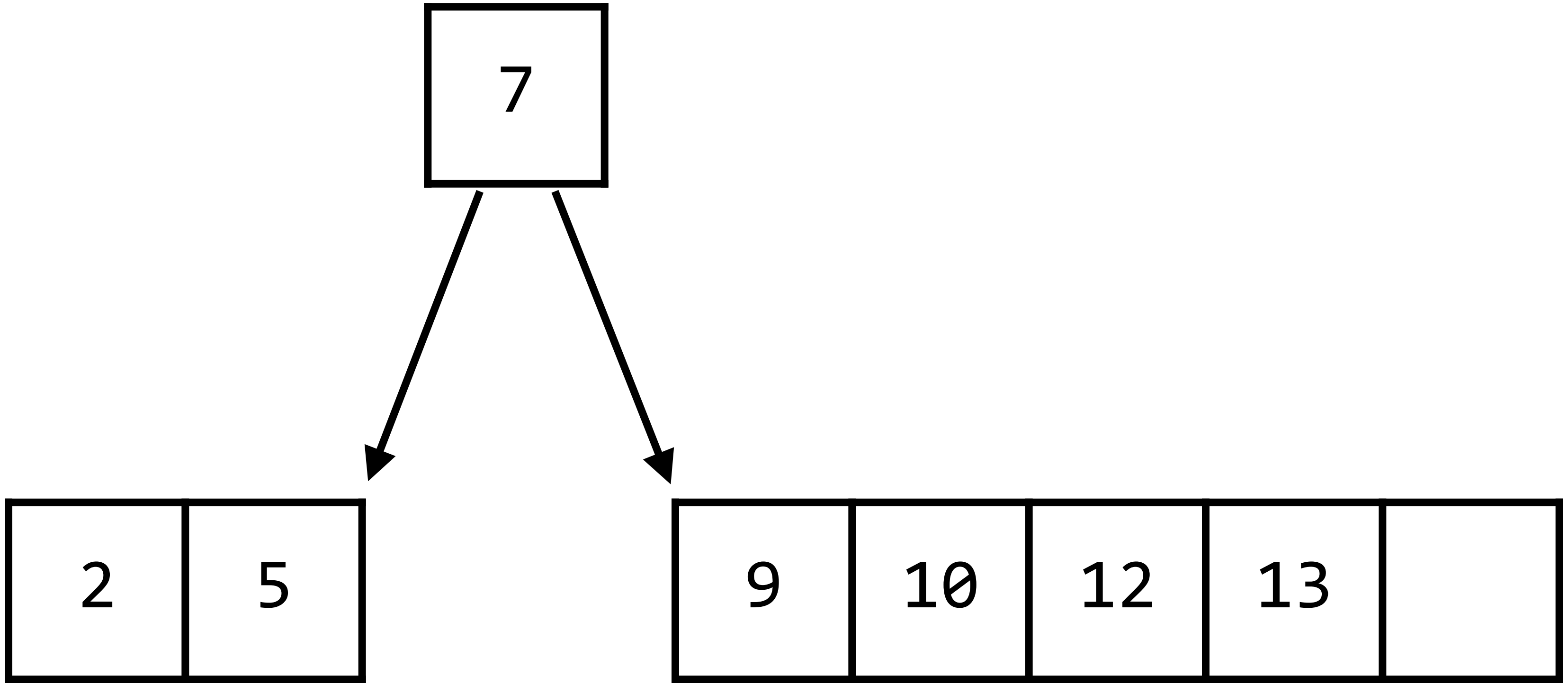
Insert(2)

5	7	9	10

BTree Insertion

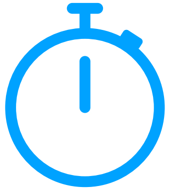
M = 5

When a BTree node reaches **m** keys, **split and make a new parent.**

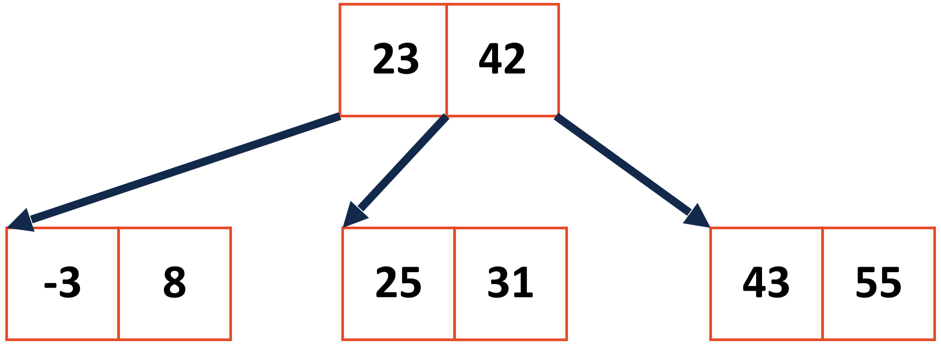


BTree Recursive Insert

Insert(56), M = 3



Insert always starts at a leaf but can propagate up repeatedly.



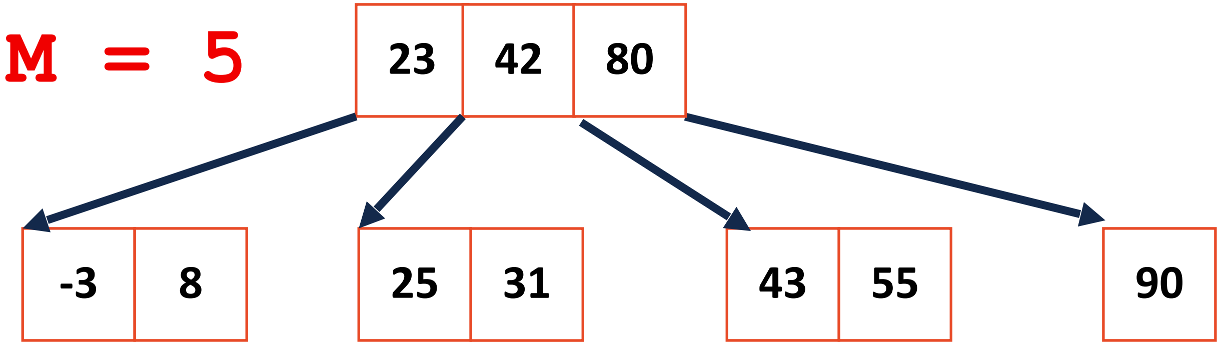
BTree Visualization/Tool

https://www.cs.usfca.edu/~galles/visualization/BTree.html

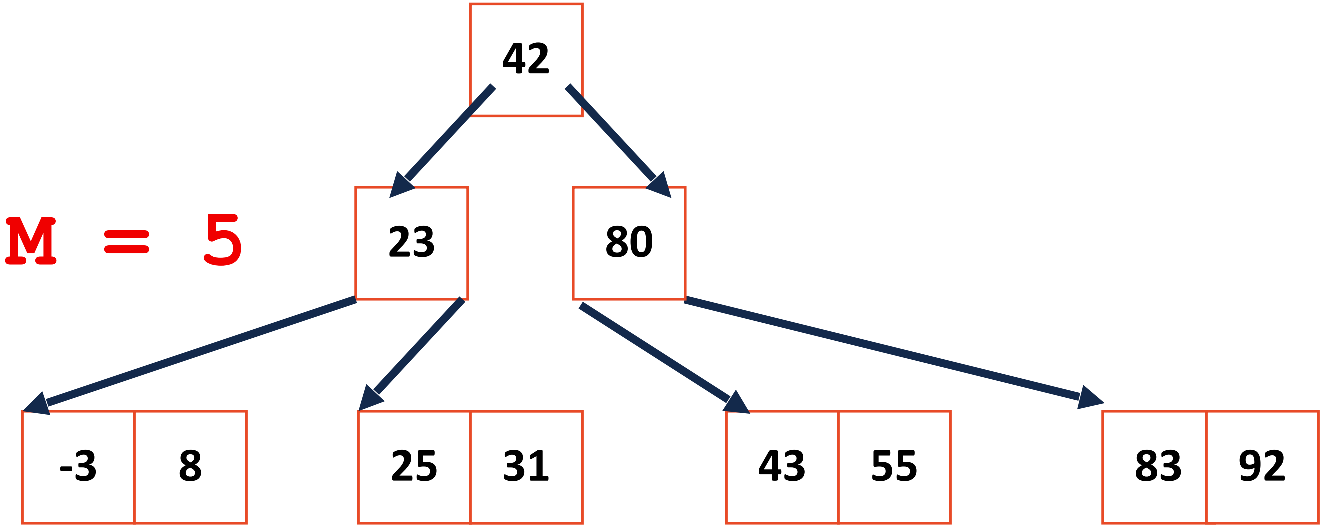
BTree Size Restrictions

By definition we have max, but do we have min? Are these trees valid?

M = 5



M = 5



BTree Properties

A **BTrees** of order **m** is an m-ary tree and by definition:

- All keys within a node are ordered
- All leaves contain no more than **m-1** keys.
- All internal nodes have exactly **one more child than keys**

Root nodes can be a leaf or have _____________ children.

All non-root, internal nodes have ___________________ children.

All leaves in the tree are at the same level.

BTree

If I tell you this is a valid BTree, what is the value of m?