Data Structures

BTree Analysis (and Heaps)

CS 225
Brad Solomon & G Carl Evans

October 8, 2023

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science

Exam 3 (10/16 — 10/18)

Sign up now on Prairietest!

Cumulative content through end of BTrees (today)

Coding question based on trees (know your tree labs!)

Learning Objectives

Analyze the performance of the BTree

Introduce a specialized data structure (discuss tradeoffs)

BTree Properties

A **BTrees** of order **m** is an m-ary tree and by definition:
- All keys within a node are ordered
- All nodes 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 **[2, m]** children.

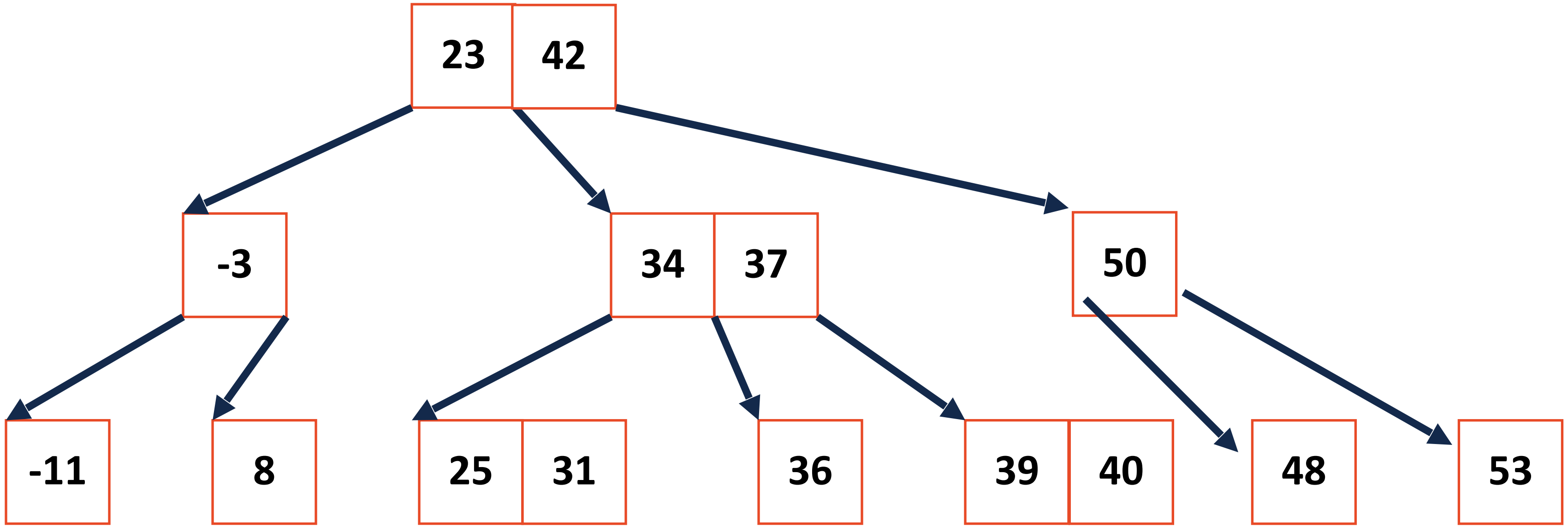
All non-root, internal nodes have **[ceil(m/2), m]** children.

All leaves in the tree are at the same level.

BTree Analysis

Let **n** be the number of keys in a BTree of order **m**.

What is our best approximation for the runtime for find? For insert?



BTree Analysis

Like the BST, BTree height determines the runtime of our operations!

Claim: The BTree structure limits our height to $O(log_m(n))$

Proof: We want to find a relationship for BTrees between the number of keys (**n**) and the height (**h**).

BTree Analysis

Strategy:

We will first count the number of nodes, level by level.

Then, we will add the minimum number of keys per node (**n**).

The minimum number of nodes will tell us the largest possible height (**h**), allowing us to find an upper-bound on height.

Key Facts:

Root nodes can be a leaf or have **[2, m]** children.

All non-root, internal nodes have **[ceil(m/2), m]** children.

BTree Analysis

Minimum number of **nodes** for a BTree of order m **at each level:**

Root:

Level 1:

Level 2:

Level 3:

Level h:

BTree Analysis

$$t = \lceil \frac{m}{2} \rceil$$

The **total number of nodes** is the sum of all the levels:

$$1 + 2\sum_{k=0}^{h-1} t^k$$

$$\sum_{i=0}^{n-1} x^i = \frac{x^n - 1}{x - 1}$$

BTree Analysis

$$t = \lceil \frac{m}{2} \rceil$$

The **total number of nodes**:

$$1 + 2\frac{t^h - 1}{t - 1}$$

The **total number of keys**:

BTree Analysis

$t = \lceil \frac{m}{2} \rceil$

The **smallest total number of keys** is: $2t^h - 1$

So an inequality about **n**, the total number of keys:

Solving for **h**, since **h** is the max number of seek operations:

BTree Analysis

Given **m=101**, a tree of height **h=4** has:

Minimum Keys:

Maximum Keys:

BTree

The BTree is still used heavily today!

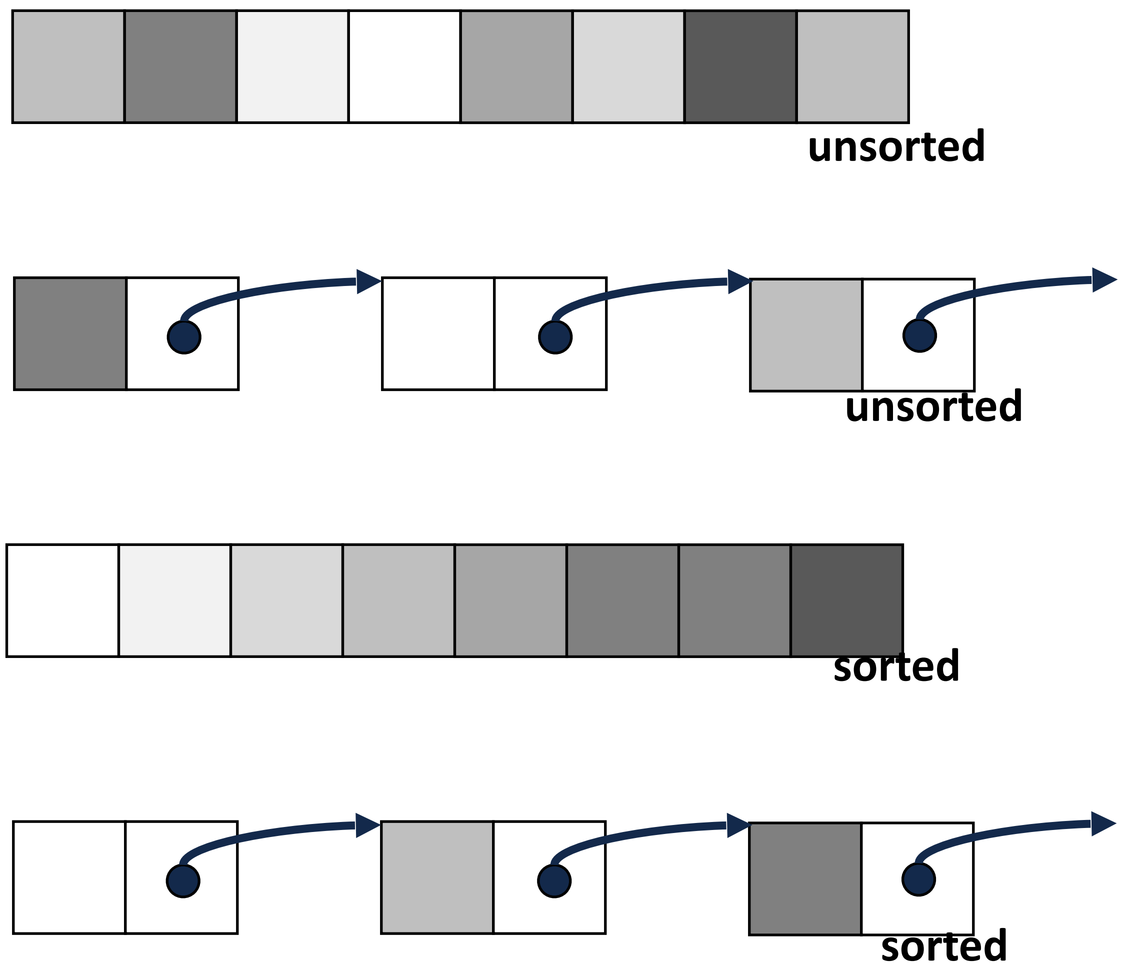
Improvements such as B+Tree and B*Tree exist far outside class scope

Thinking conceptually: Sorting a queue

How might we build a ‘queue’ in which our front element is the min?

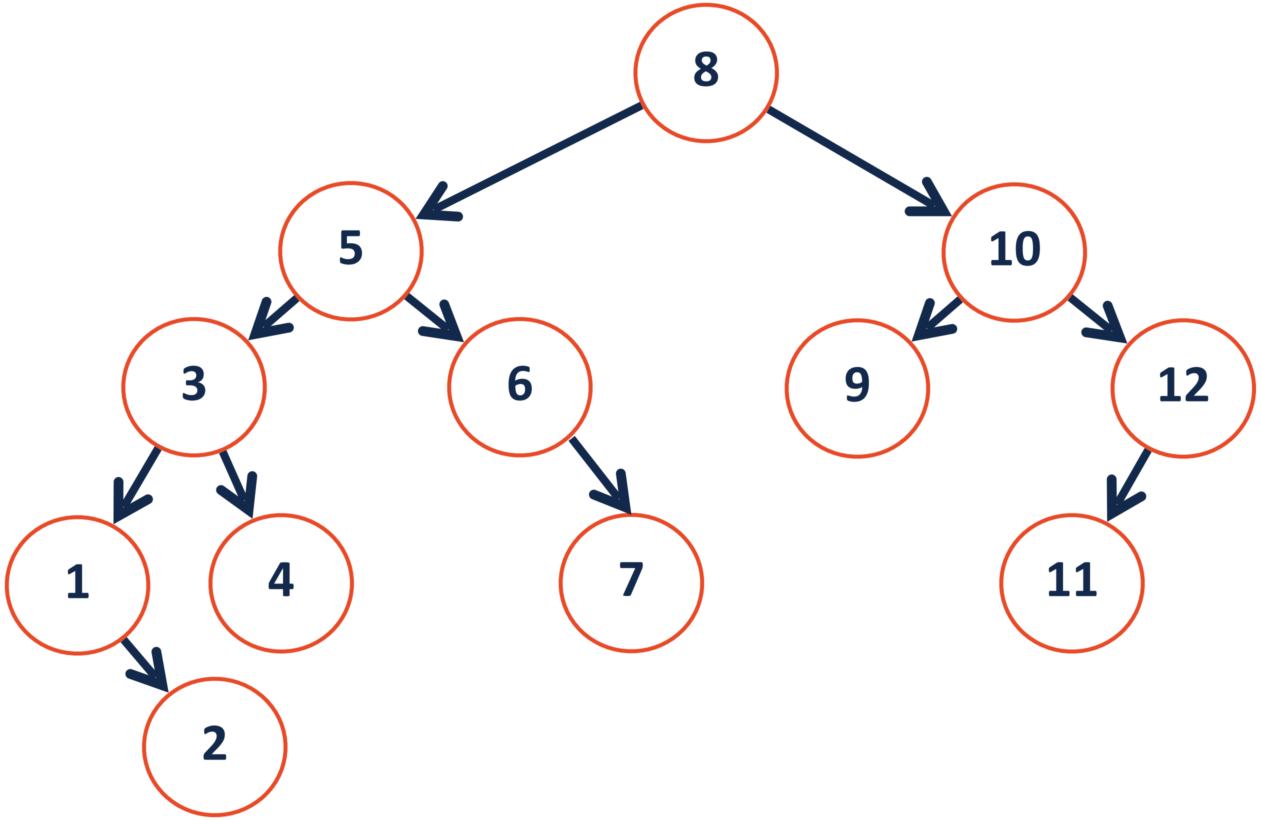
Priority Queue Implementation

insert	removeMin
O(n)*	O(n)
O(1)	O(n)
O(n)	O(1)
O(n)	O(1)



Priority Queue Implementation

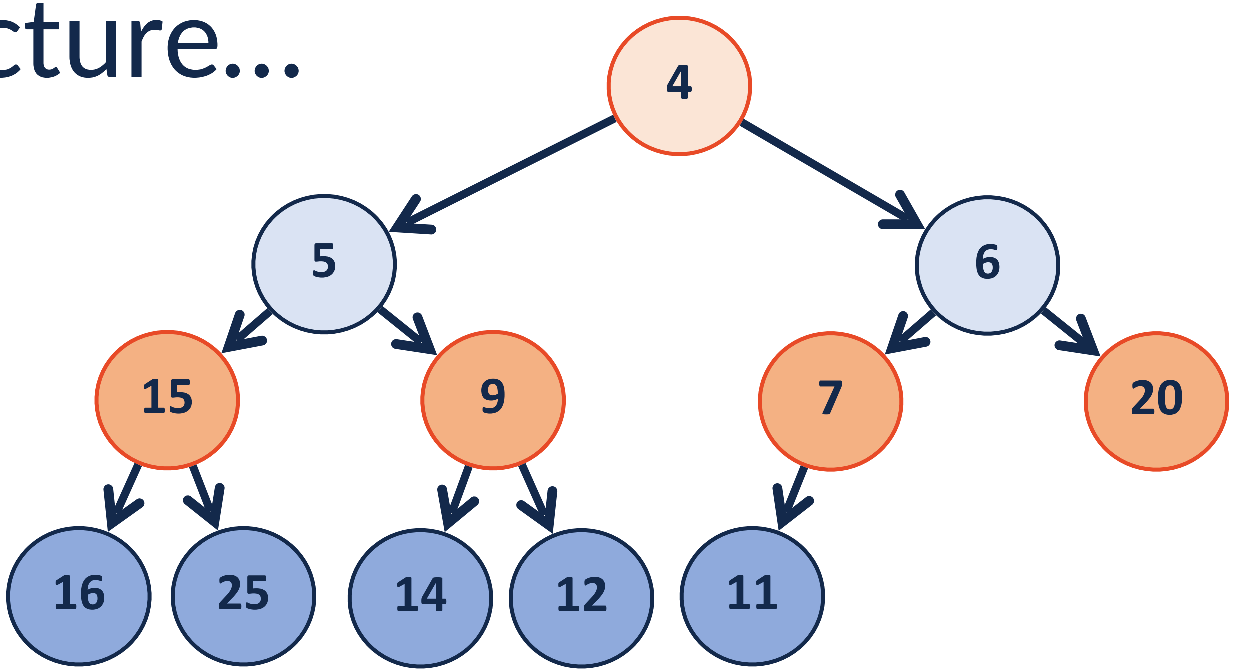
insert	removeMin



Thinking conceptually: A tree without pointers

What class of (non-trivial) trees can we describe without pointers?

Another possibly structure…



(min)Heap

A complete binary tree T is a min-heap if:

- **T = {}** or
- **T = {r, T_L, T_R}**, where **r** is less than the roots of {**T_L, T_R**} and {**T_L, T_R**} are min-heaps.