



CS 225

Data Structures

September 30 – BTrees

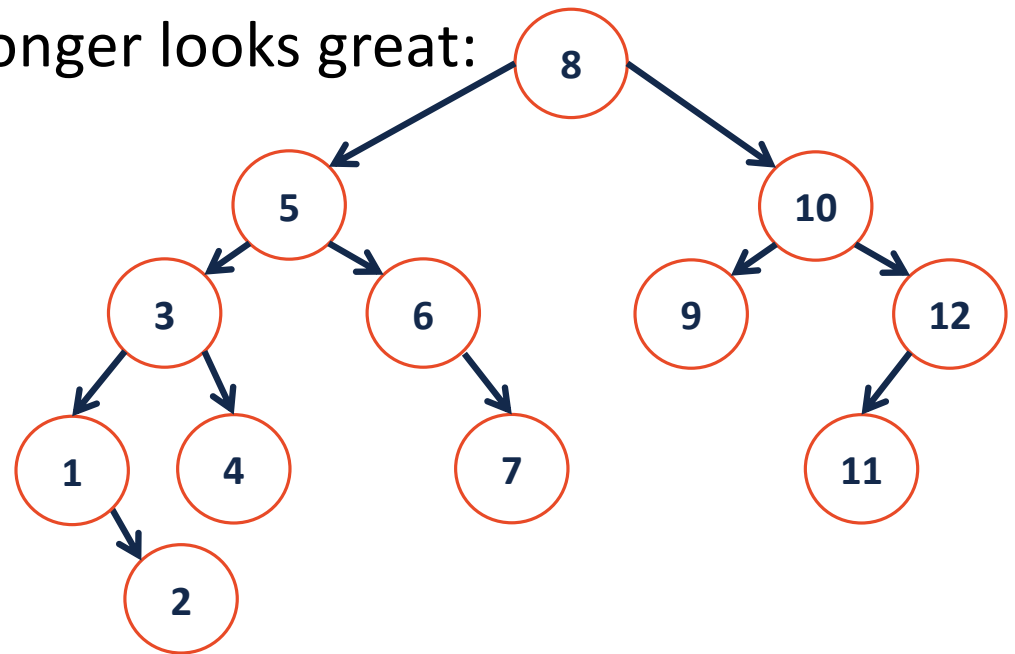
G Carl Evans

B-Tree Motivation

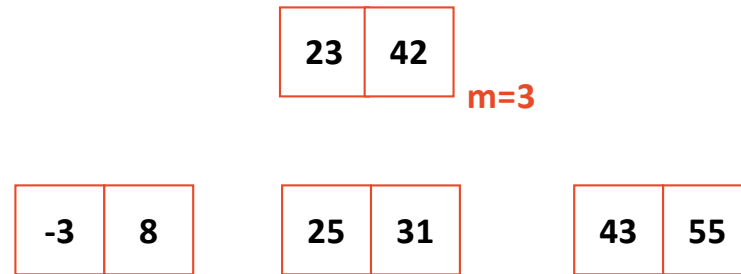
In Big-O we have assumed uniform time for all operations, but this isn't always true.

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

...an $O(\lg(n))$ AVL tree no longer looks great:



BTree Recursive Insert



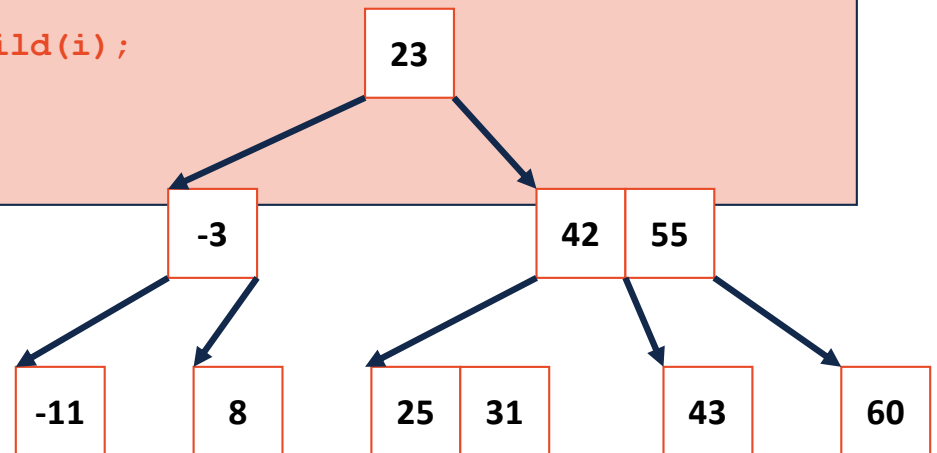


BTree Visualization/Tool

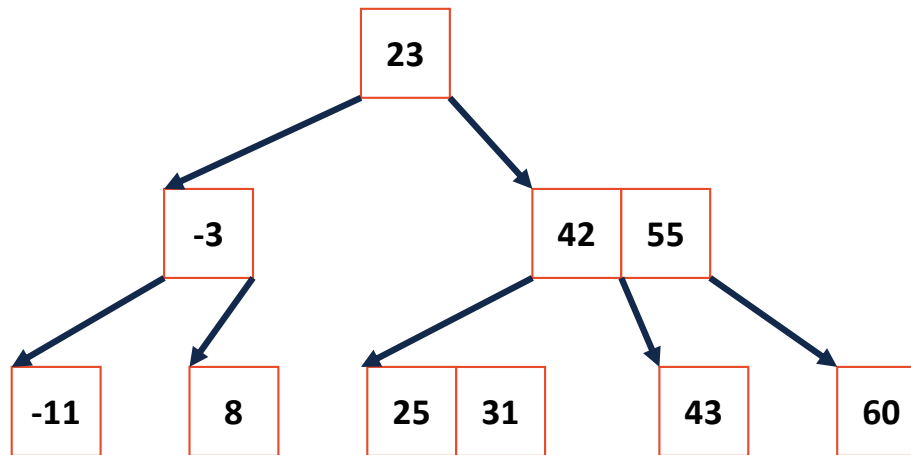
<https://www.cs.usfca.edu/~galles/visualization/BTree.html>

BTree Search

```
1 bool Btree::_exists(BTreeNode & node, const K & key) {
2
3     unsigned i;
4     for ( i = 0; i < node.keys_ct_ && key > node.keys_[i]; i++) { }
5
6     if ( i < node.keys_ct_ && key == node.keys_[i] ) {
7         return true;
8     }
9
10    if ( node.isLeaf() ) {
11        return false;
12    } else {
13        BTreeNode nextChild = node._fetchChild(i);
14        return _exists(nextChild, key);
15    }
16 }
```



BTree Search





Btree Properties

A **BTree** of order **m** is an m-way tree:

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

- All leaves are on the same level



BTree Analysis

The height of the BTree determines maximum number of _____ possible in search data.

...and the height of the structure is: _____.

Therefore: The number of seeks is no more than _____.

...suppose we want to prove this!



BTree Analysis

In our AVL Analysis, we saw finding an upper bound on the height (given n) is the same as finding a lower bound on the nodes (given h).

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.



BTree Analysis

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

root:

level 1:

level 2:

level 3:

...

level h :



BTree Analysis

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



BTree Analysis

The **total number of keys:**



BTree Analysis

The **smallest total number of keys** is:

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

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



BTree Analysis

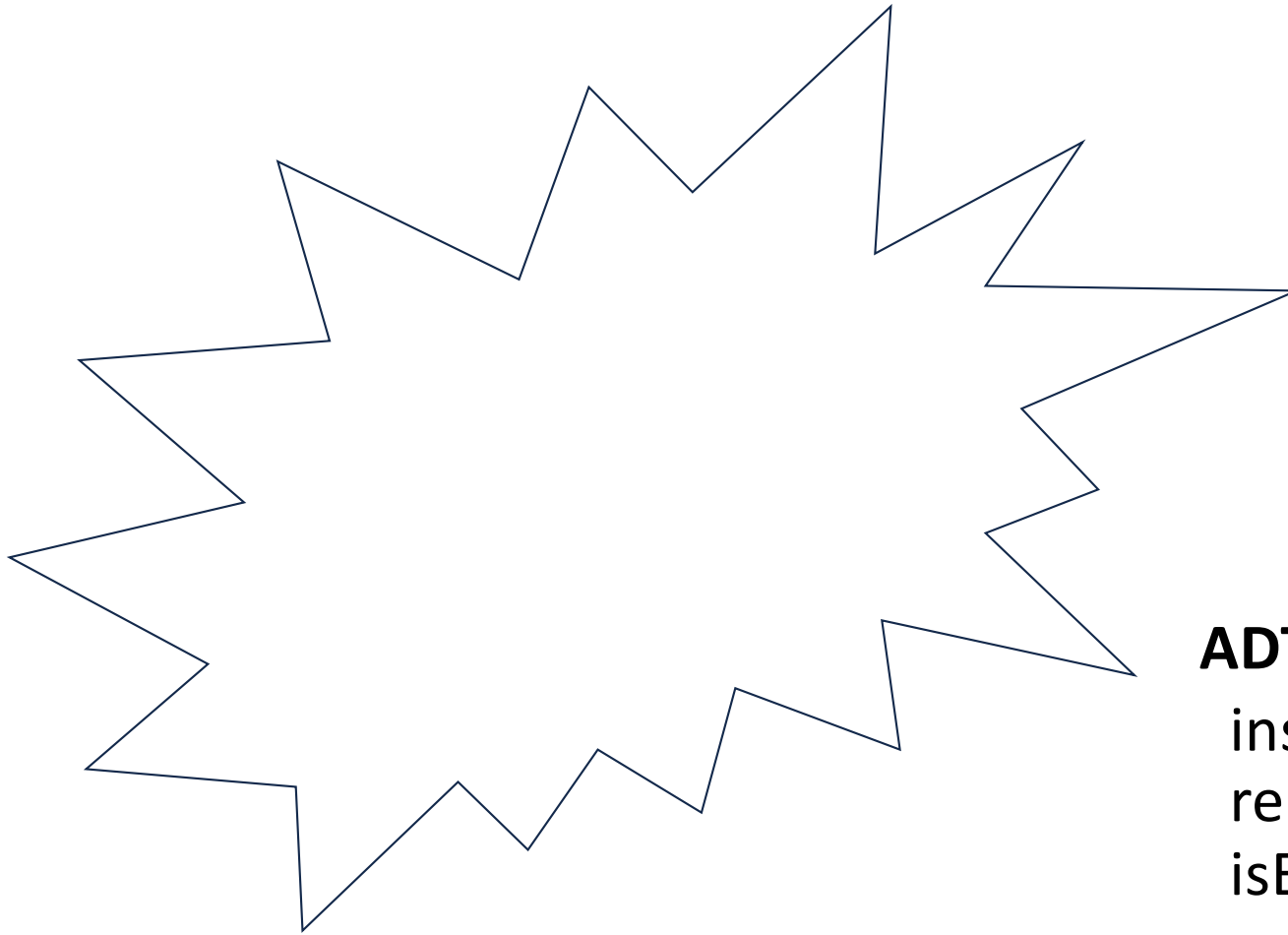
Given $m=101$, a tree of height $h=4$ has:

Minimum Keys:

Maximum Keys:



Secret, Mystery Data Structure

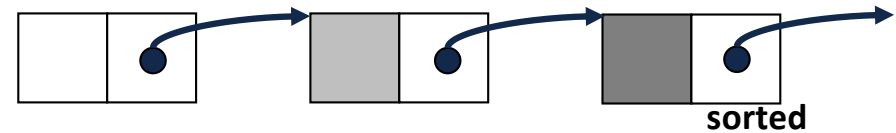
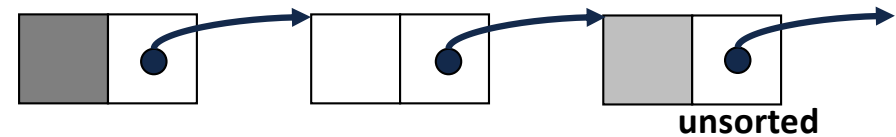


ADT:

insert
remove
isEmpty

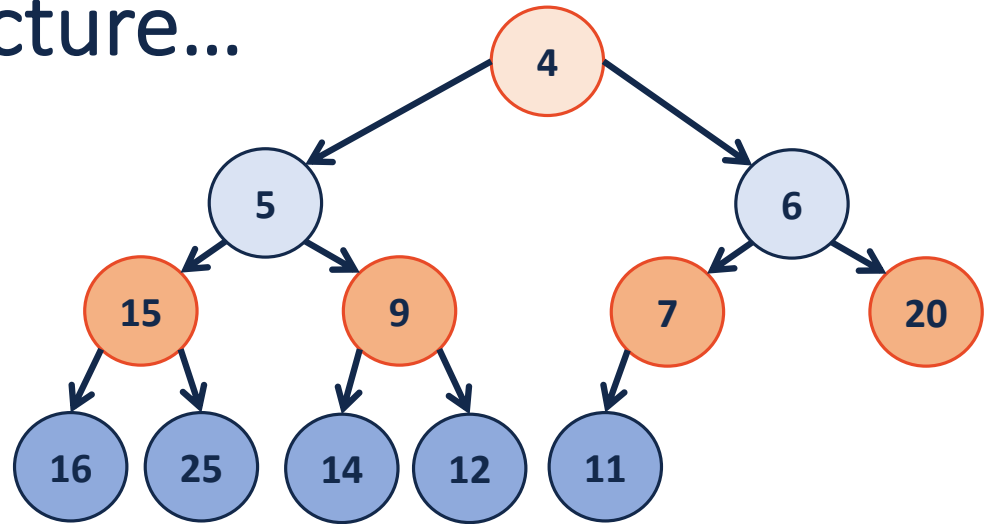
Priority Queue Implementation

insert	removeMin
$O(n)$	$O(n)$
$O(1)$	$O(n)$
$O(\lg(n))$	$O(1)$
$O(\lg(n))$	$O(1)$



What all is wrong in this table?

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.

