CS 225
Data Structures

March 12 – BTrees
Wade Fagen-Ulmschneider
B-Tree Motivation

Big-O assumes uniform time for all operations, but this isn’t always true.

However, seeking data from the cloud may take 100ms+. ...an O(lg(n)) AVL tree no longer looks great:
Real Application

Imagine storing Facebook profiles for everyone in the US:

How many records?

How much data in total?

How deep is the AVL tree?
BTree Motivations

Knowing that we have large seek times for data, we want to:
BTree (of order m)

\[-3 \quad 8 \quad 23 \quad 25 \quad 31 \quad 42 \quad 43 \quad 55\]

\[m=9\]

**Goal:** Minimize the number of reads!

Build a tree that uses ______________________ / node

[1 network packet]

[1 disk block]
BTree Insertion

A BTree of order $m$ is an $m$-way tree:
- All keys within a node are ordered
- All leaves contain hold no more than $m-1$ nodes.
BTree Insertion

When a BTree node reaches $m$ keys:

$m=5$
BTree Recursive Insert
BTree Recursive Insert

m=3

-3 8 25 31 43 55

23 42
BTree Visualization/Tool

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

A **BTrees** of order $m$ is an $m$-way tree:
- All keys within a node are ordered
- All leaves contain hold no more than $m-1$ nodes.
- All internal nodes have exactly one more key than children
- Root nodes can be a leaf or have $[2, m]$ children.
- All non-root, internal nodes have $[\text{ceil}(m/2), m]$ children.
- All leaves are on the same level
BTree
BTree Search
BTree Search

```cpp
bool Btree::_exists(BTreeNode & node, const K & key) {
    unsigned i;
    for (i = 0; i < node.keys_ct_ && key < node.keys_[i]; i++) {}
    if (i < node.keys_ct_ && key == node.keys_[i]) {
        return true;
    }
    if (node.isLeaf()) {
        return false;
    } else {
        BTreeNode nextChild = node._fetchChild(i);
        return _exists(nextChild, key);
    }
}
```
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: