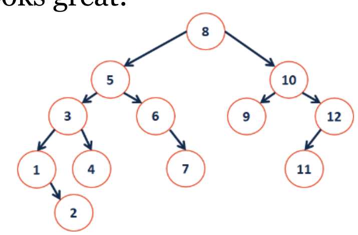
CS 225

#23: BTrees
October 19, 2018 · *Wade Fagen-Ulmschneider*

BTree 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:



Consider Instagram profile data:

How many profiles?		
How much data /profile?		
	AVL Tree	BTree
Tree Height		

BTree Motivations
Knowing that we have long seek times for data, we want to build a data structure with two (related) properties:

1.

2.

BTree$_m$

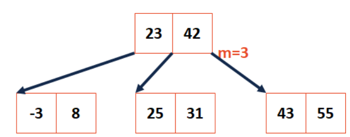
Goal: Build a tree that uses ____________________ /node!
…optimize the algorithm for your platform!

A **BTree of order m** is an m-way tree where:
1. All keys within a node are ordered.

BTree Insert, using m=5

…when a BTree node reaches **m** keys:

BTree Insert, m=3:



Great interactive visualization of BTrees:
https://www.cs.usfca.edu/~galles/visualization/BTree.html

BTree Properties

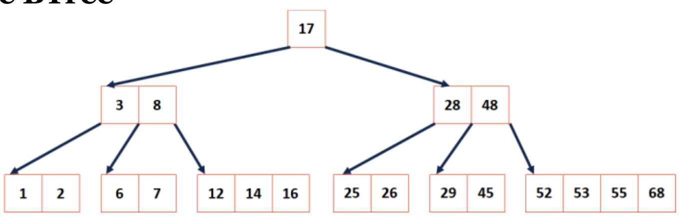
For a BTree of order **m**:

1. All keys within a node are ordered.
2. All leaves contain no more than **m-1** nodes.

3. All internal nodes have exactly **one more key than children**.
4. Root nodes can be a leaf or have **[2, m]** children.
5. All non-root, internal nodes have **[ceil(m/2), m]** children.

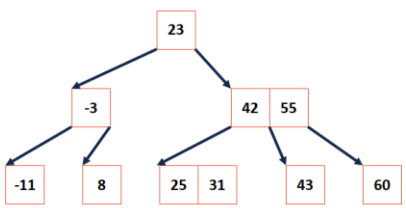
6. All leaves are on the same level.

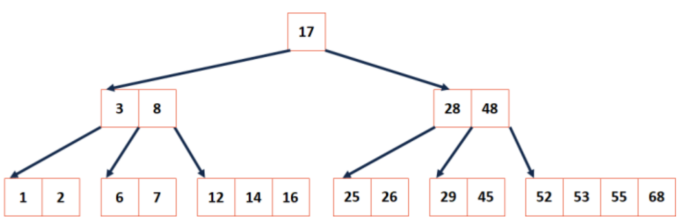
Example BTree



What properties do we know about this BTree?

BTree Search





BTree.hpp

```
bool Btree<K, V>::_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 our structure:

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**).

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

CS 225 – Things To Be Doing:

1. Programming Exam B starts next Thursday
2. MP4 due next Monday (Oct. 22)
3. lab_avl due Sunday
4. Daily POTDs are ongoing!