Announcements

Join Code: **225**

Thanks for being a great class!

EC Survey is closed

↳ Everyone gets 2pts EC!

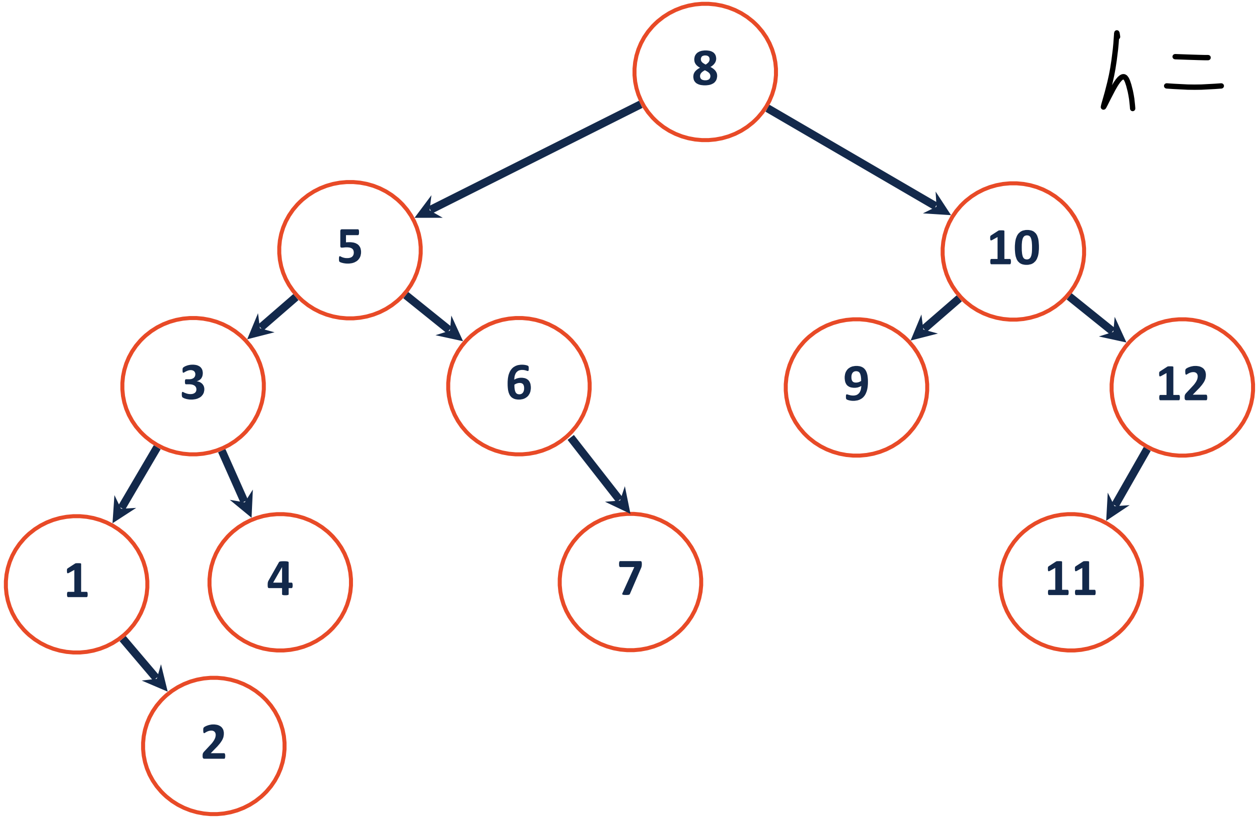
ILLINOIS

BTree Introduction

Learning Objectives

1. Understand the motivation behind BTrees
2. Know all the BTree Properties
3. Understand how to insert elements into a BTree
4. Implement Search in a BTree

AVL Trees



$h = O(\log n)$

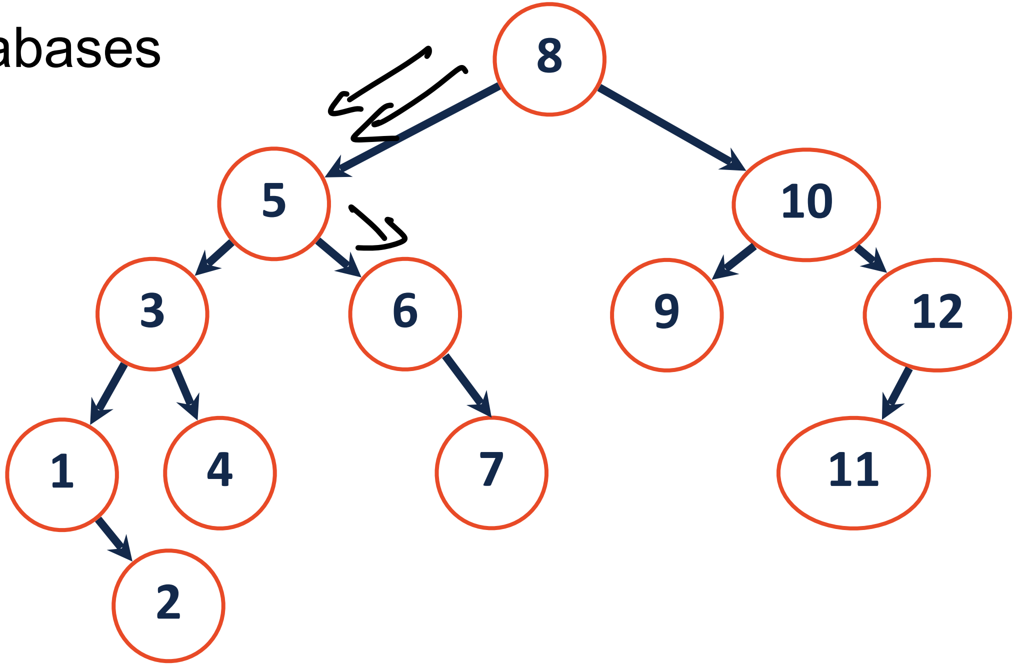
Rate Limiting Factor - Seeks

Distributed Databases

Hard Disks

Goal

- Shorter Tree
- More Data at each node



System Design

Goal: Minimize the number of reads! seeks

Build a tree that uses ________________________ / node

1. Distributed Database - [1 network packet]
   a. 1500 Bytes (1 MTU) (Maximum Transmissible Unit)
2. Hard Disk - [1 disk block]
   a. Old Systems 512 Bytes
   b. New Systems 4096 Bytes

Ex. 1 billion integers

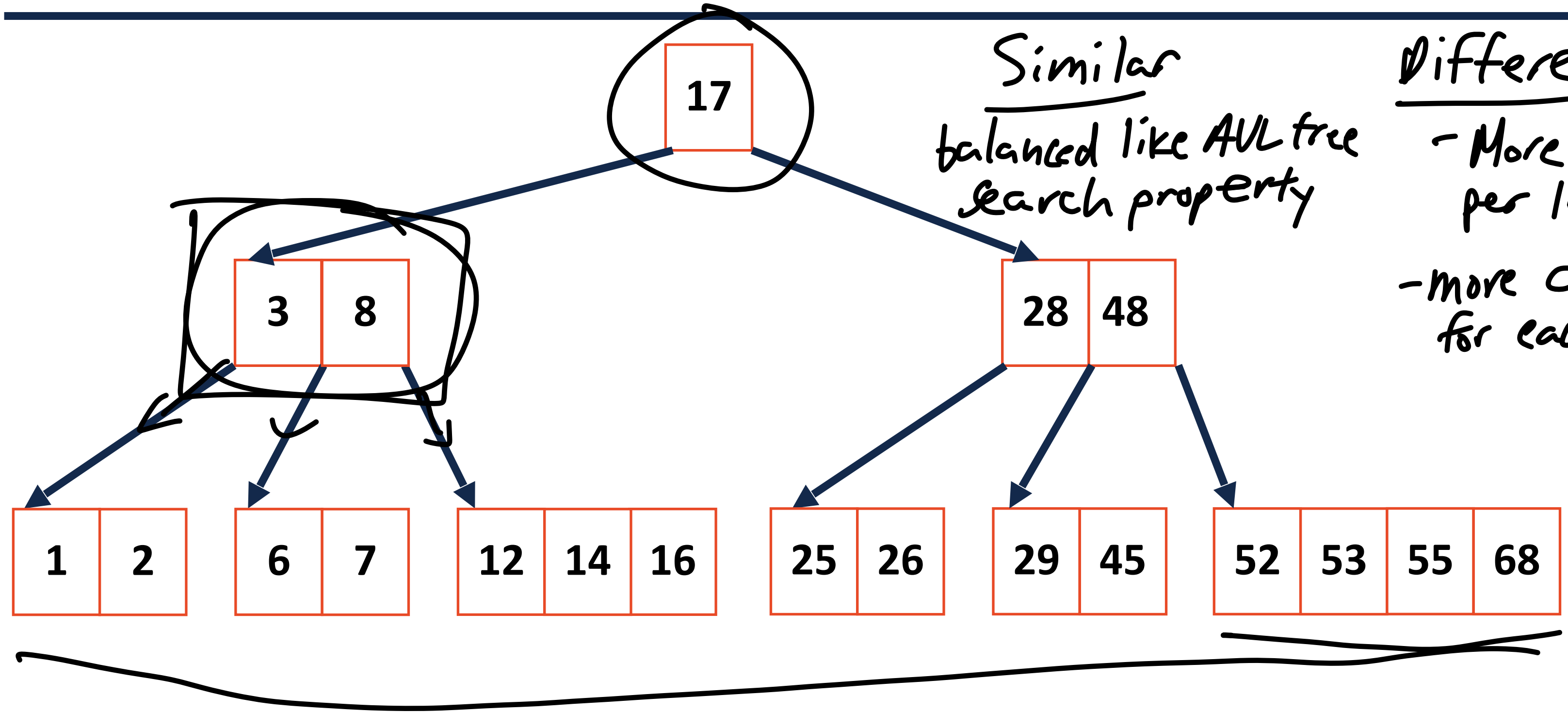
integer – 4 bytes

$$\frac{4096 \text{ bytes/node}}{4 \text{ bytes/integer}} = 1024 \frac{\text{int}}{\text{node}}$$

$$\log_{1024}(10^9) \sim 3$$

$$\log_2(10^9) \sim 30$$

What are some observations about B-Trees?



Similar

balanced like AVL tree
search property

Different

- More nodes per level

- more children for each nodes

BTree Properties

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

Definition

- All keys within a node are ordered
- All nodes contain no more than **m-1** keys
- All internal nodes have **one more child than keys**
- All leaves are on the same level

B-Tree Operations

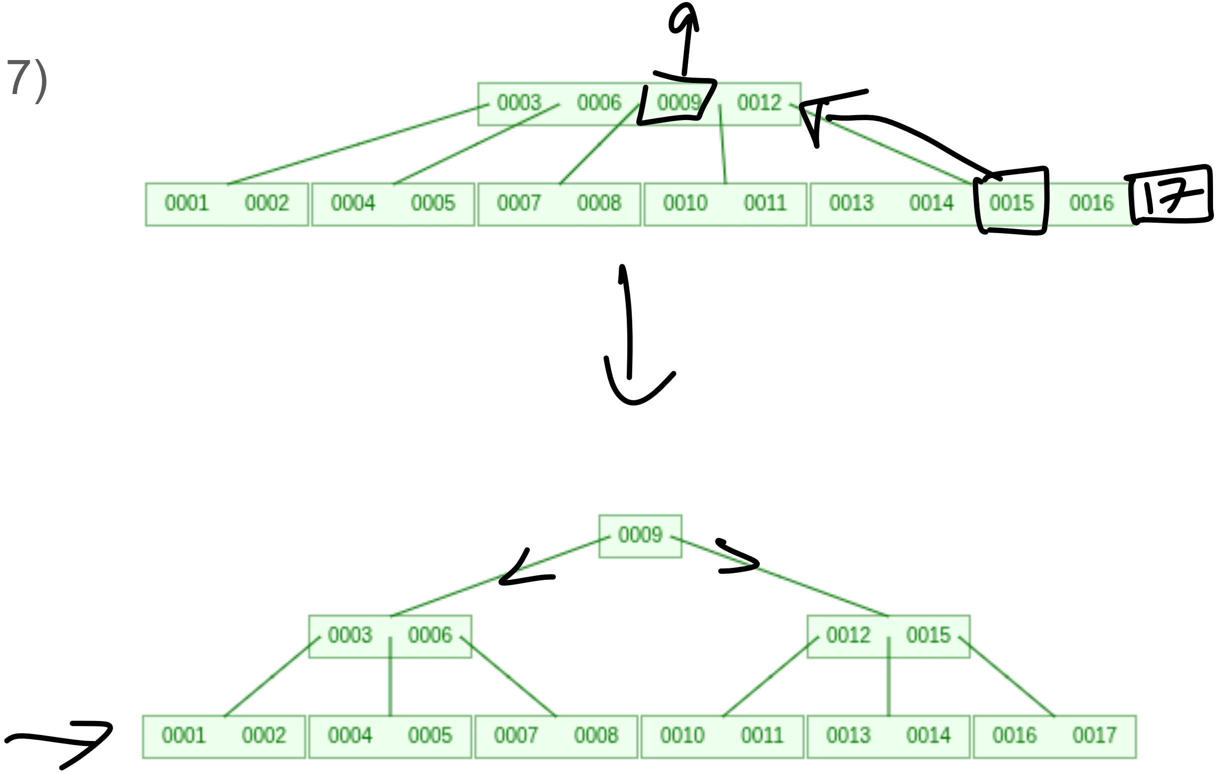
Insert

Find

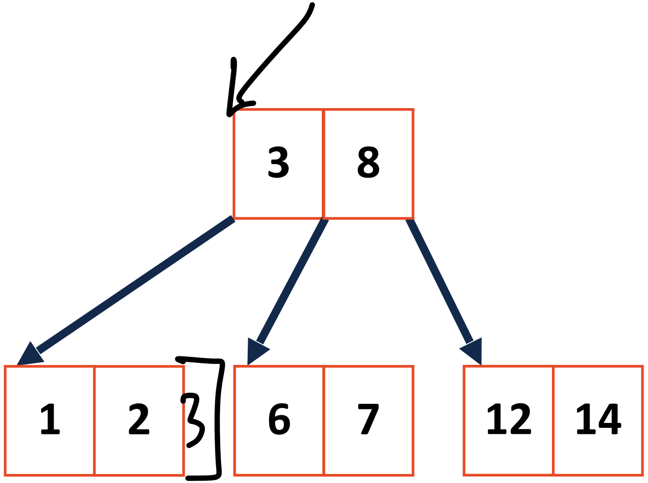
Remove - More complicated, there is a POTD about this, not discussed in this class

Insert - Recursive Split

Insert(17)

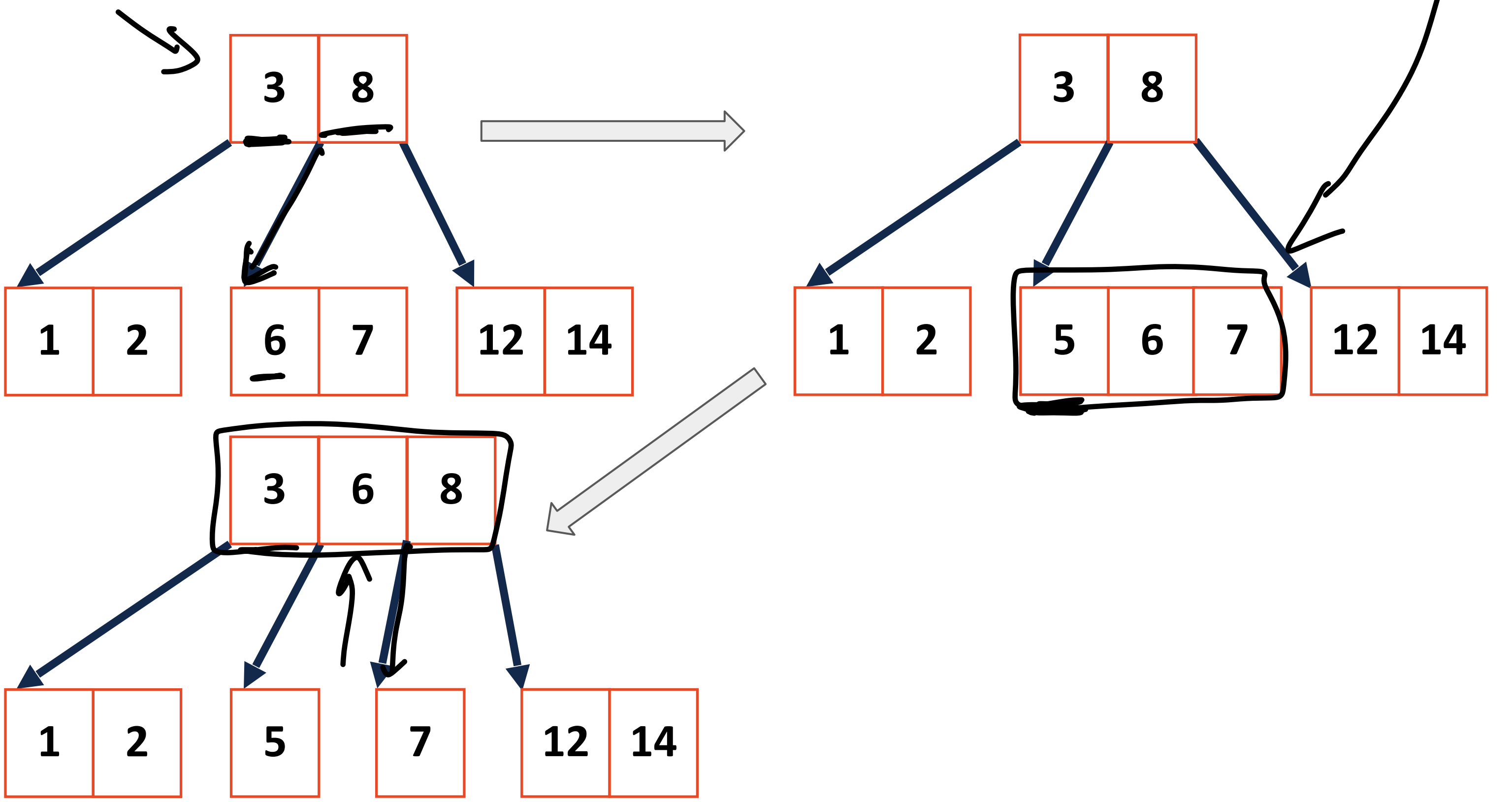


What will be the root after insert(5)? Assume m = 3

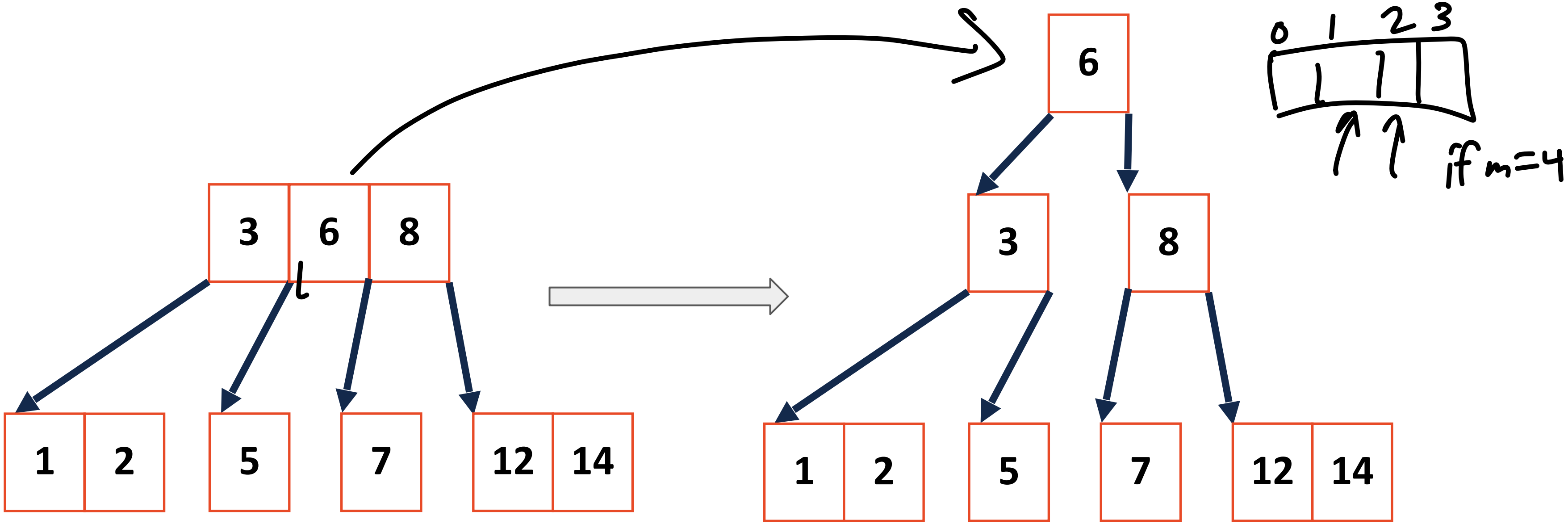


Join Code: **225**

What will be the root after insert(5)? Assume m = 3



What will be the root after insert(5)? Assume m = 3



B-Tree Insert

- Always inserts at leaf level
- Split if m ~~nodes~~ keys are reached in a node
- Split could propagate up the tree recursively

Split by "throwing up" the median

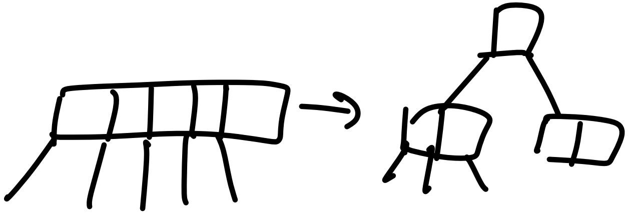
BTree Properties

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

Definition

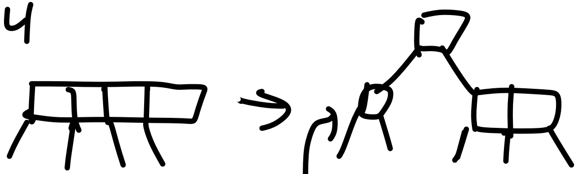
- All keys within a node are ordered
- All nodes contain no more than **m-1** keys
- All internal nodes have **one more child than keys**
- All leaves are on the same level
- Root node is a leaf or has **[2, m]** children.
- All non-root, internal nodes have **[ceil(m/2), m]** children

$m=5$



$\text{ceil}\left(\frac{5}{2}\right)=3$

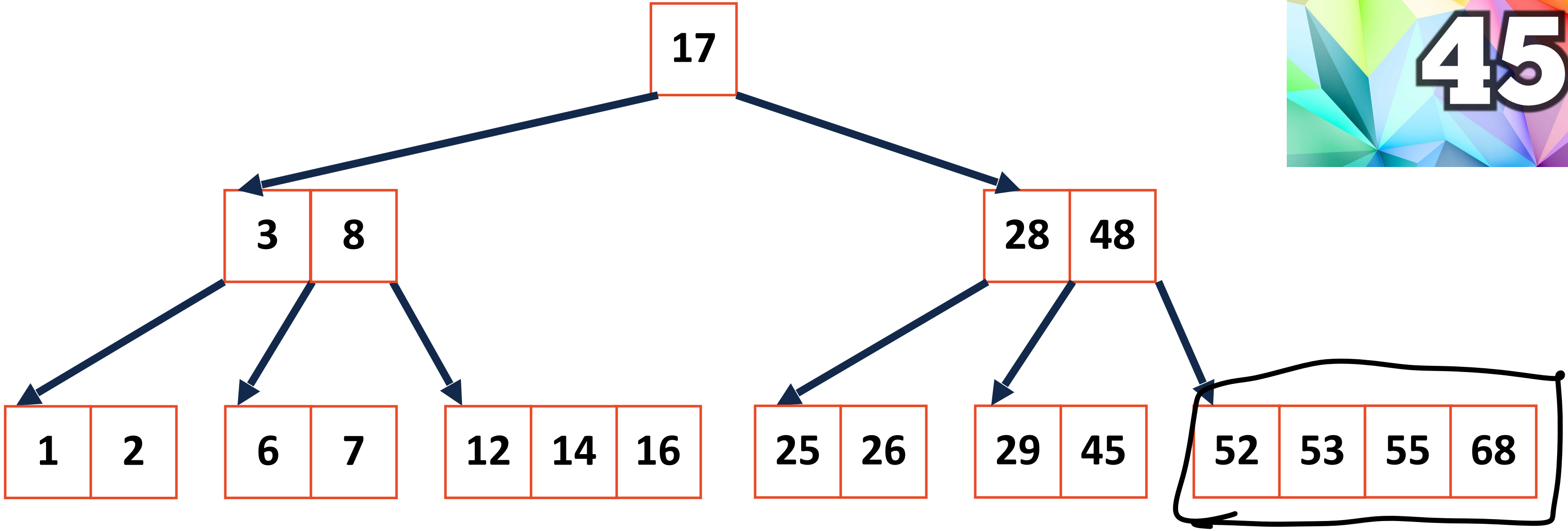
$m=4$



$\text{ceil}\left(\frac{4}{2}\right)=2$

If this is a valid BTree, what is the lower bound for m?

Join Code: **225**



m = 5

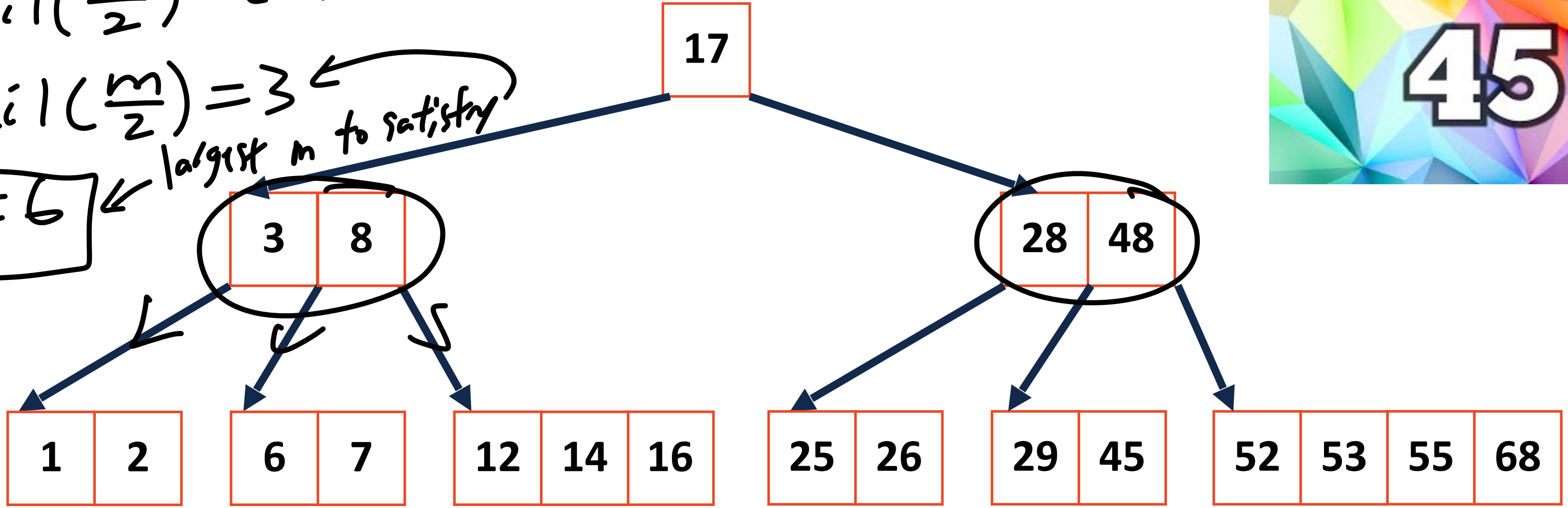
If this is a valid BTree, what is the upper bound for m?

$\text{ceil}\left(\frac{m}{2}\right)$ children at least

$\text{ceil}\left(\frac{m}{2}\right) = 3$

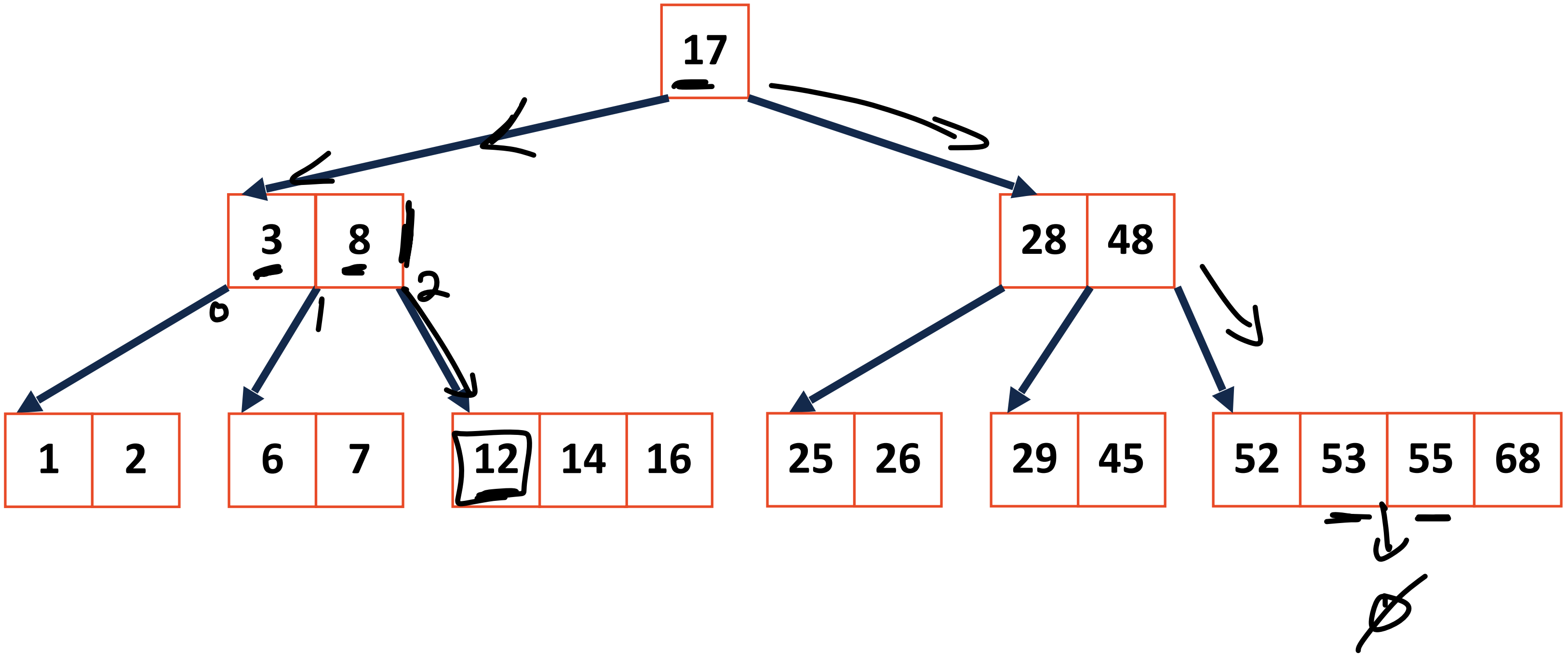
largest m to satisfy

$\boxed{m = 6}$



If $m=7$, $\text{ceil}\left(\frac{m}{2}\right)=4$. non root internal nodes only have 3 children, so $m \neq 7$

B-Tree Find: How would find(12) work? find(54)?



BTree Search

```
template <typename K, typename V>
V BTree<K, V>::_find(BTreeNode * node, const K & key) const {
  unsigned i;
  for (i = 0; i < node.keys_ct_ && key > node.keys_[i]; i++) { }

  if (i < node.keys_ct_ && key == node.keys_[i]) {
    return node.values_[i];
  }

  if (node.isLeaf()) {
    return V();
  }

  BTreeNode nextChild = node._fetchChild(i);
  return _find(nextChild, key);
}
```

stopping at end

node equal or larger

found key

return not found

recursively call on child

seek

BTree Properties

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

Definition

- All keys within a node are ordered
- All nodes contain no more than **m-1** keys
- All internal nodes have **one more child than keys**
- All leaves are on the same level
- Root node is a leaf or has **[2, m]** children.
- All non-root, internal nodes have **[ceil(m/2), m]** children