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 child than key.
4. Root nodes can be a leaf or have $[2, m]$ children.
5. All non-root, internal nodes have $[\lceil m/2 \rceil, m]$ children.
6. 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 our structure:

Therefore, the number of seeks is no more than: __________.

...suppose we want to prove this!

BTree Proof #1
In our AVL Analysis, we saw finding an upper bound on the height ($h$ given $n$, aka $h = f(n)$) is the same as finding a lower bound on the keys ($n$ given $h$, aka $f^{-1}(h)$).

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

BTree Strategy:
1. Define a function that counts the minimum number of nodes in a BTree of a given order.
   a. Account for the minimum number of keys per node.
2. Proving a minimum number of nodes provides us with an upper-bound for the maximum possible height.

Proof:
1a. The minimum number of nodes for a BTree of order $m$ at each level is as follows:

   root:
   level 1:
   level 2:
   level 3:
   ...
   level h:

1b. The minimum total number of nodes is the sum of all levels:

2. The minimum number of keys:

3. Finally, we show an upper-bound on height:
So, how good are BTrees?
Given a BTree of order 101, how much can we store in a tree of height=4?

Minimum:

Maximum:

---

Hashing

<table>
<thead>
<tr>
<th>Locker Number</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td></td>
</tr>
<tr>
<td>330</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td></td>
</tr>
<tr>
<td>124</td>
<td></td>
</tr>
</tbody>
</table>

...how might we create this today?

---

Goals for Understanding Hashing:
1. We will define a **keyspace**, a (mathematical) description of the keys for a set of data.
2. We will define a function used to map the **keyspace** into a small set of integers.

---

All hash tables consist of three things:
1. 
2. 
3. 

---

A Perfect Hash Function

- (Angrave, CS 241)
- (Beckman, CS 421)
- (Cunningham, CS 210)
- (Davis, CS 101)
- (Evans, CS 126)
- (Fagen-Ulmschneider, CS 225)
- (Gunter, CS 422)
- (Herman, CS 233)

...characteristics of this function?

A Second Hash Function

---

CS 225 – Things To Be Doing:
1. Programming Exam B starts Thursday
2. MP4 is due tonight by 11:59pm; MP5 released Tuesday
3. lab_ltree released on Wednesday
4. Daily POTDs are ongoing!