B-Trees

Motivation for B-Trees
Disk access time is relatively slow, but disk transfer rate is relatively fast.
Efficiency can be gained by moving many records per disk transfer.
A node in a B-tree is intended to correspond to one disk transfer.

Definition of B-Trees
B-trees are search trees with the properties:
- $t =$ minimum degree, each internal node (except for root) has $\geq t$ children.
- The root node contains 1 to $2t - 1$ keys.
- Other nodes contain $t - 1$ to $2t - 1$ keys.
- Keys in a node are in sorted order.
- An internal node with $k$ keys has $k + 1$ children.
- $x.\text{key}[i - 1] \leq \text{keys in } i\text{th subtree} \leq x.\text{key}[i]$
- Every leaf node has the same depth.
The Height of B-Trees

\( n \) keys and minimum degree \( t \) imply

\[
h \leq \log_t \frac{n + 1}{2}
\]

Let \( T(d) \) = minimum number of nodes at level \( d \) assuming \( d \leq h \). First prove \( T(d) = 2t^{d-1} \)

**Basis:** \( T(1) = 2 = 2t^0 \)

**Induction:** Assume: \( T(d - 1) = 2t^{d-2} \) for \( 1 < d \leq h \)

**Show:** \( T(d) = 2t^{d-1} \) if \( 1 < d \leq h \)

---

**Height Part 2**

Proof: Each node at depth \( d - 1 \) has \( \geq t \) children, so \( T(d) = t(2t^{d-2}) = 2t^{d-1} \)

Root node has \( \geq 1 \) key.
Other nodes have \( \geq t - 1 \) keys.

\[
n \geq 1 + (t - 1) \sum_{d=1}^{h} 2t^{d-1} = 2t^h - 1
\]

This is equivalent to: \( t^h \leq \frac{n + 1}{2} \)

which implies: \( h \leq \log_t \frac{n + 1}{2} \)
Inserting into a B-Tree

\[
\text{B-Tree-Insert}(T, k) \\
x \leftarrow T.\text{root} \\
\text{loop} \\
\quad \text{if } x \text{ is full} \\
\quad \quad \text{then split } x \\
\quad \quad \quad x \leftarrow \text{left or right node of split} \\
\quad \quad \text{if } x \text{ is a leaf node then exit loop} \\
\quad \quad x \leftarrow \text{next node down} \\
\text{end loop} \\
\text{insert } k \text{ into } x
\]

Deleting from a B-Tree

\[
\text{B-Tree-Delete}(T, k) \\
x \leftarrow T.\text{root} \\
\text{loop} \\
\quad \text{if } x \text{ has } t - 1 \text{ keys and } x \neq T.\text{root} \\
\quad \quad \text{then shift from or merge with sibling of } x \\
\quad \quad \text{if } x \text{ contains } k \text{ then exit loop} \\
\quad \quad x \leftarrow \text{next node down} \\
\text{end loop} \\
\text{if } x \text{ is a leaf node} \\
\quad \text{then delete } k \text{ from } x \\
\text{else } k' \leftarrow \text{successor of } k \\
\quad \text{delete } k' \text{ starting search from } x \\
\quad \text{replace } k \text{ in } x \text{ with } k'
\]