B-Trees

Definition
Height
Operations
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.
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.
- $x.key[i - 1] \leq \text{keys in } i\text{th subtree} \leq x.key[i]$
- 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 $1 < d \leq h$
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: $h \leq \log_t \left(\frac{n + 1}{2}\right)$
Splitting and merging the root:

- **Splitting**
  - **H**
    - **C**
      - **T1**
    - **D**
      - **T2**
    - **H**
      - **T3**
    - **I**
      - **T4**
    - **L**
      - **T5**
    - **merge**

- **Merging**
  - **C**
    - **T1**
  - **D**
    - **T2**
  - **merge**
    - **T3**
  - **I**
    - **T4**
  - **L**
    - **T5**
  - **T6**
Splitting and merging everything else:

\[ \begin{array}{c}
\text{... A O ...} \\
C \quad D \quad H \quad I \quad L
\end{array} \quad \xrightarrow{\text{split}} \quad \begin{array}{c}
\text{... A H O ...} \\
C \quad D\quad \quad \quad I \quad L
\end{array} \]

\[ \begin{array}{c}
T1 \quad T2 \quad T3 \quad T4 \quad T5 \quad T6
\end{array} \quad \xrightarrow{\text{merge}} \quad \begin{array}{c}
T1 \quad T2 \quad T3 \quad T4 \quad T5 \quad T6
\end{array} \]
Shifting Between Siblings

Motivation

Definition

Height

Height 2

Split and merge

Split and merge 2

▷ Shift

Insert

Delete

... A H O ...

C D G

I L

C D

H I L

T1 T2 T3 T4 T5 T6 T7

T1 T2 T3 T4 T5 T6 T7
Inserting into a B-Tree

**B-Tree-Insert**($T, k$)

1. $x \leftarrow T.root$
2. **loop**
   - **if** $x$ is full
     - **then** split $x$
       - $x \leftarrow$ left or right node of split
3. **if** $x$ is a leaf node **then** exit loop
4. $x \leftarrow$ next node down
5. **end loop**
6. insert $k$ into $x$
Deleting from a B-Tree

**B-Tree-Delete**\((T, k)\)

\[ x \leftarrow T.\text{root} \]

**loop**

\[ \text{if } x \text{ has } t - 1 \text{ keys and } x \neq T.\text{root} \]

\[ \text{then shift from or merge with sibling of } x \]

\[ \text{if } x \text{ contains } k \text{ then exit loop} \]

\[ x \leftarrow \text{next node down} \]

**end loop**

**if** \( x \) **is a leaf node**

\[ \text{then delete } k \text{ from } x \]

**else** \( k' \leftarrow \text{successor of } k \)

\[ \text{delete } k' \text{ starting search from } x \]

\[ \text{replace } k \text{ in } x \text{ with } k' \]