I/O data structures

B-trees and B^ε-trees

Inge Li Gørtz

B-trees

- Operations
  - insert(k, v)
  - delete(k)
  - v = search(k)
  - \([v_1, v_2, \ldots] = \text{range-query}(k_1, k_2)\)

• Range query \([q_1, q_2]:\)
  - search down \(T\) for \(q_1\) and \(q_2\) (or successor and predecessor).
  - report the elements in the leaves between the leaves containing (successor of) \(q_1\) and (predecessor of) \(q_2\).
  - \(\#\text{I/Os} = O(\log_B N + \text{occ}/B)\).

- A node/leaf can be stored in \(O(1)\) blocks.
- Search uses \(O(\log_B N)\) I/Os.

- Elements in leaves in sorted order: between \(B/2\) and \(B\) in each leaf.
- Height \(\log_B N\)
- Branching factor of root: \([2:B]\)
- Node:
  - \(\leq 10\)
  - \(> 10\) and \(\leq 65\)
  - \(> 65\) and \(\leq 83\)
  - \(> 83\)

- Branching factor of root: \([B/2:B]\)

- Elements in leaves in sorted order: between \(B/2\) and \(B\) in each leaf.
Insertion in B-tree

- Insert(k, v)
  - search for relevant leaf u and insert (k,v) in u.
  - If u now contains B+1 elements:
    - split it into two leaves u' and u''.
    - update parent(u)
    - If parent(u) now has degree B+1 recursively split it.
    - If root split: add a new root node with degree 2 (height of tree grows)
  - Example. B= 5. Insert(24, v)

- #I/Os = O(log_b N)

Deletion in B-tree

- Delete(k)
  - search for relevant leaf u and delete element with key k in u.
  - If u now contains B/2 - 1 elements:
    - merge u with its sibling u'. If this results in u containing more than B elements split it into two leaves.
    - update parent(u)
    - If parent(u) now has degree B/2 - 1 recursively merge it.
    - If root has degree 1: delete root (height decreases)
  - Example. B= 6. Delete(24)
Deletion in B-tree

- \textbf{Delete}(k)
  - search for relevant leaf \( u \) and delete element with key \( k \) in \( u \).
  - If \( u \) now contains \( \frac{B}{2} - 1 \) elements:
    - merge \( u \) with its sibling \( u' \). If this results in \( u \) containing more than \( B \) elements split it into two leaves.
    - update parent(\( u \))
    - If parent(\( u \)) now has degree \( \frac{B}{2} - 1 \) recursively merge it.
  - If root has degree 1: delete root (height decreases)

- **Example**: \( B = 6 \). \textbf{Delete}(18)

\begin{align*}
\text{Node:} & \leq 10 & > 10 \text{ and } \leq 65 & > 65 \text{ and } \leq 83 & > 83 \\
\text{between a-1 and b-1 pivot elements} & \text{a B-tree is an (a,b)-tree with a,b=Θ(B)}
\end{align*}

(a,b)-trees

- Operations
  - insert(\( k, v \))
  - delete(\( k \))
  - search(\( k \))
  - \( v = \text{range-query}(k) \)
  - \( \{v_1, v_2, \ldots\} = \text{range-query}(k_1, k_2) \)

\begin{align*}
\text{Elements in leaves in sorted order: between a and b in each leaf} & \text{Height } \log_b N \\
\text{branching factor of root: } [2,b] & \text{branching factor: } [a,b]
\end{align*}
Amortized updates in (a,b)-trees

- If $b \geq 2a$ then the number of rebalancing operations caused by an update $O(1/a)$ amortized

$B^\varepsilon$ trees

- For $0 \leq \varepsilon \leq 1$:
  - Updates: $O((\log_{1+B^\varepsilon} N)/B^{1-\varepsilon})$
  - Point query: $O(\log_{1+B^\varepsilon} N)$
  - Range query:
    - $O((\log_{1+B^\varepsilon} N) + \text{occ}/B)$

- $\varepsilon = 1/2$:
  - Updates: $O((\log_2 N)/\sqrt{B})$
  - Point query: $O(\log_2 N)$
  - Range query:
    - $O((\log_2 N) + \text{occ}/B)$