

External Memory II

- Searching with Fast Updates
- Searching Strings

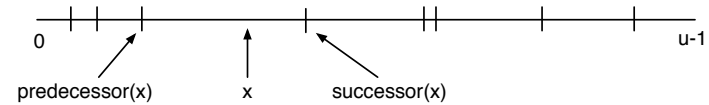
Philip Bille

External Memory II

- Searching with Fast Updates
- Searching Strings

Searching

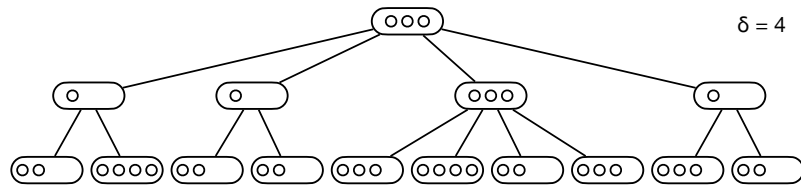
- Searching. Maintain a set $S \subseteq U = \{0, \dots, u-1\}$ supporting
 - $\text{member}(x)$: determine if $x \in S$
 - $\text{predecessor}(x)$: return largest element in $S \leq x$.
 - $\text{successor}(x)$: return smallest element in $S \geq x$.
 - $\text{insert}(x)$: set $S = S \cup \{x\}$
 - $\text{delete}(x)$: set $S = S - \{x\}$



Searching

- Applications.
 - Relational data bases.
 - File systems.

B-tree

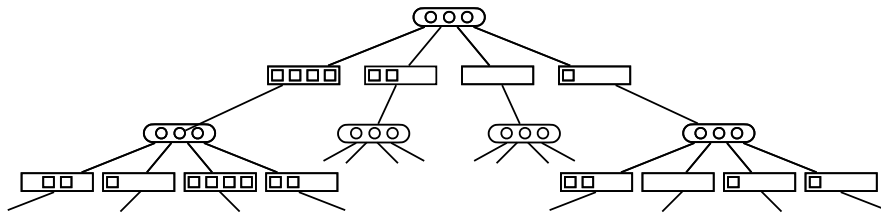


- B-tree of order $\delta = \Theta(B)$ with N keys.
 - Keys in leaves. Routing elements in internal nodes.
 - Degree between $\delta/2$ and δ .
 - Root degree between 2 and δ .
 - Leaves store between $\delta/2$ and δ keys.
 - All leaves have the same depth.
- Height. $\Theta(\log_{\delta} (N/B)) = \Theta(\log_B N)$
- Search and update. $O(\log_B N)$ I/Os.

B^ϵ -tree

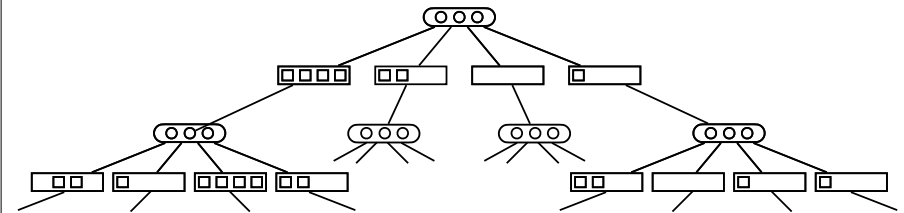
- Idea.
 - Speed up updates by buffering them at each node along the path to a leaf.
 - Move many updates together in each I/O.
 - Search (almost) as before.
 - $\epsilon \in (0, 1]$ is a parameter.
- Solution in 2 steps.
 - Focus on \sqrt{B} -tree ($\epsilon = 1/2$).
 - Searching in $O(\log_B N)$ I/Os.
 - Updates in $O((\log_B N)/\sqrt{B})$ amortized.
 - Generalize to any ϵ .

\sqrt{B} -tree



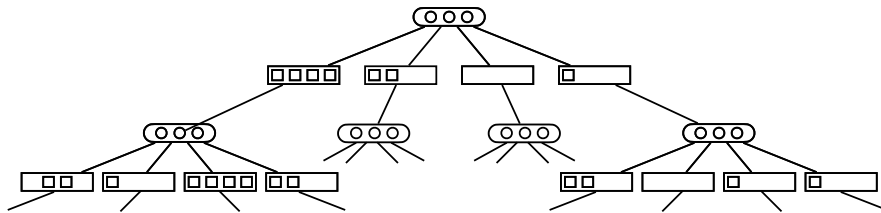
- \sqrt{B} -tree with N keys.
 - B-tree of degree $\Theta(\sqrt{B})$ with buffers of size $\Theta(\sqrt{B})$ at each edge.
 - Buffer stores delayed updates in subtree.
 - Nodes and child buffers stored together in $O(1)$ blocks.
- Height. $\Theta(\log_{\sqrt{B}} N) = \Theta(\log_B N)$

\sqrt{B} -tree



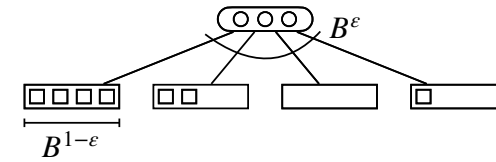
- Searching.
 - Find leaf using routing elements. Check buffers along path.
- I/Os. $O(\log_B N)$.

\sqrt{B} -tree



- Updates.
 - Insert update message into buffer at child.
 - If buffer full, flush and recurse at child.
 - If we reach leaf, rebalance tree as B-tree.
- I/O intuition. A flush moves \sqrt{B} messages together $\Rightarrow O((\log_B N)/\sqrt{B})$ amortized I/Os.
- I/Os.
 - Assign $(\log_B N)/\sqrt{B}$ credits to each update for constant $c > 1$.
 - Put c/\sqrt{B} credits each node on path.
 - \Rightarrow We can pay for buffer overflows and rebalancing.

B^ϵ -tree



- B^ϵ -tree with N keys.
 - B-tree of degree $\Theta(B^\epsilon)$ with buffers of size $\Theta(B^{1-\epsilon})$ at each edge.
- Searching. $O\left(\frac{\log_B N}{\epsilon}\right)$ I/Os.
- Updates. $O\left(\frac{\log_B N}{\epsilon B^{1-\epsilon}}\right)$ I/Os.

B^ϵ -tree

	Search	Update
B-tree	$O(\log_B N)$	$O(\log_B N)$
\sqrt{B} -tree	$O(\log_B N)$	$O\left(\frac{\log_B N}{\sqrt{B}}\right)$
B^ϵ -tree	$O\left(\frac{\log_B N}{\epsilon}\right)$	$O\left(\frac{\log_B N}{\epsilon B^{1-\epsilon}}\right)$

External Memory II

- Searching with Fast Updates
- Searching Strings

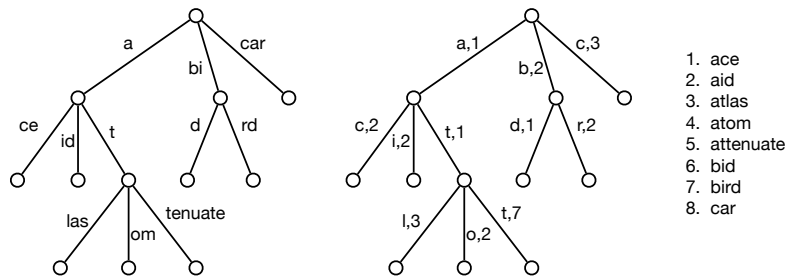
String Searching

- String searching. Maintain a set $S = \{S_1, S_2, \dots, S_K\}$ of K strings of total length N supporting the following operations:
 - search(P): return string in S with longest common prefix with P .
 - insert(P): set $S = S \cup \{P\}$
 - delete(P): set $S = S - \{P\}$

String Searching

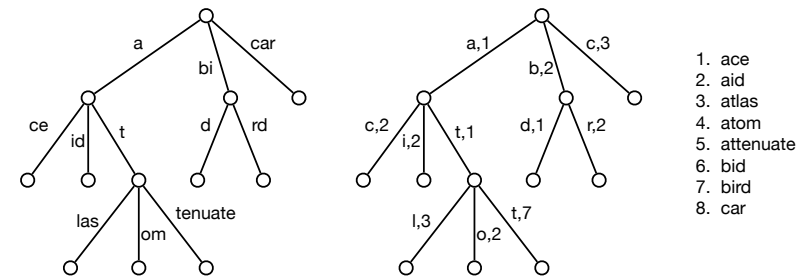
- Goal.
 - Searching in $O(\log_B N + |P|/B)$ I/Os.
 - Ignore insert and delete.
- Solution in 3 steps.
 - Blind tries.
 - String B-trees.
 - String B-trees with fast searches.

Blind Trie



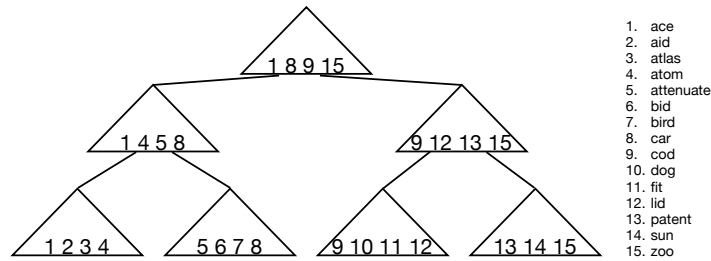
- Data structure.
 - Sorted set of strings.
 - Compact trie for S . Edges store first char + string length.
- Space. Strings + trie: $O(N + K) = O(N)$.

Blind Trie



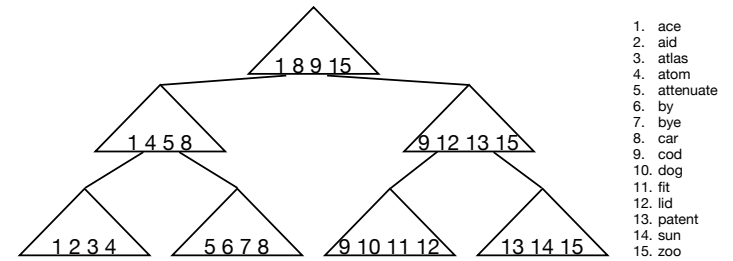
- Search. Traverse and verify candidate.
- Time. $O(|P|)$

String B-tree



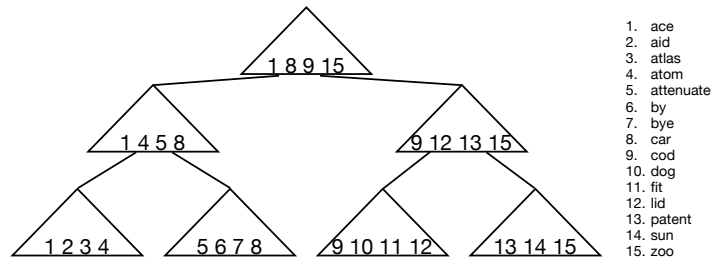
- Data structure. Combination of B-tree and blind tries.
 - Sorted set of strings.
 - Nodes store blind trie over B strings.
 - Leftmost and rightmost string in subtree stored for each child.
- Space. Strings + $O(B)$ per node: $O(N + N) = O(N)$

String B-tree



- Searching.
 - Traverse and verify at each node.
- I/Os.
 - $O(|P|/B + 1)$ I/Os at each node.
 - $\Rightarrow O((|P|/B) \log_B N)$ I/Os total.

String B-tree



- Fast searching.
 - Traverse and verify at each node.
 - But: remember longest prefix matched at each node.
- I/Os.
 - $O(|P|/B)$ I/Os in total for string verification. $O(1)$ I/Os at each node.
 - $\Rightarrow O((|P|/B) + \log_B N)$ I/Os.

External Memory II

- Searching with Fast Updates
- Searching Strings