

External Memory II

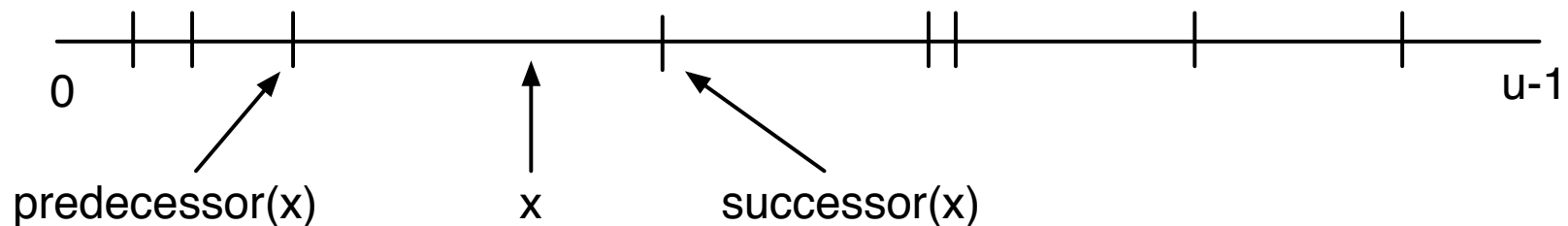
- Searching with Fast Updates
- Searching Strings

External Memory II

- Searching with Fast Updates
- Searching Strings

Searching

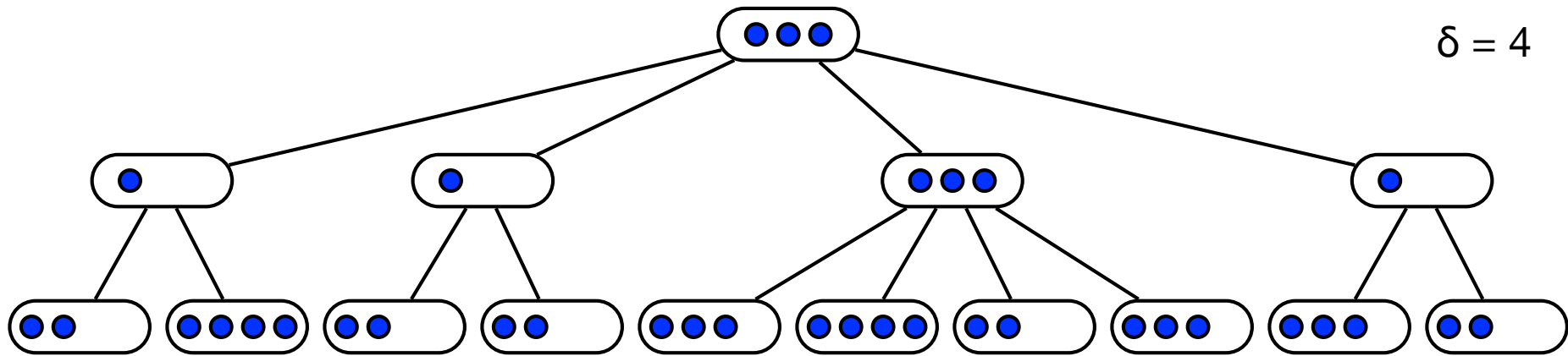
- **Searching.** Maintain a set $S \subseteq U = \{0, \dots, u-1\}$ supporting
 - $\text{member}(x)$: determine if $x \in S$
 - $\text{predecessor}(x)$: return largest element in $S \leq x$.
 - $\text{successor}(x)$: return smallest element in $S \geq x$.
 - $\text{insert}(x)$: set $S = S \cup \{x\}$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



Searching

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

B-tree

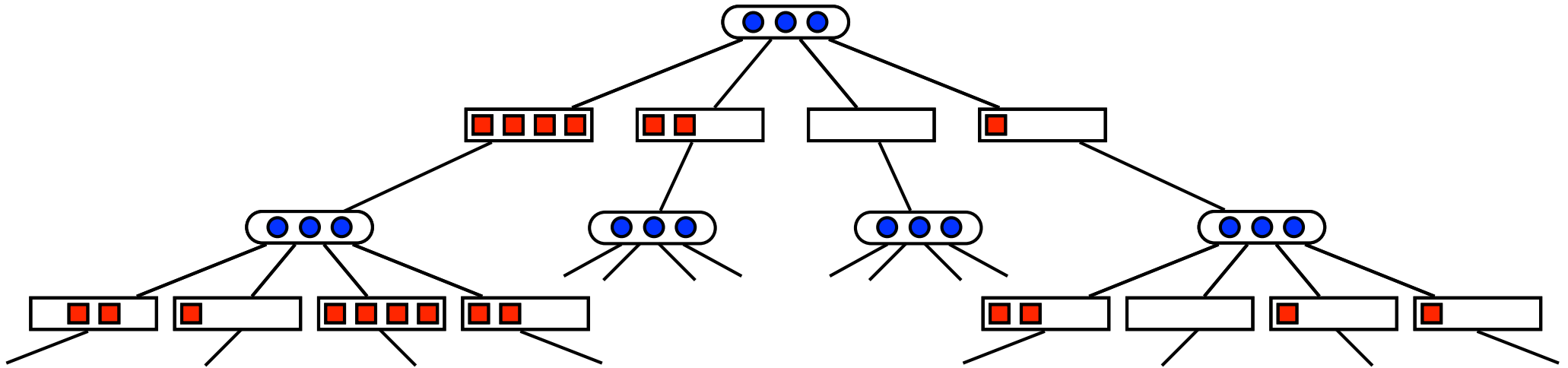


- B-tree of order $\delta = \Theta(B)$ with N keys.
 - Keys in leaves. **Routing** elements in internal nodes.
 - Degree between $\delta/2$ and δ .
 - Root degree between 2 and δ .
 - Leaves store between $\delta/2$ and δ keys.
 - All leaves have the same depth.
- **Height.** $\Theta(\log_{\delta} (N/B))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

B^ε -tree

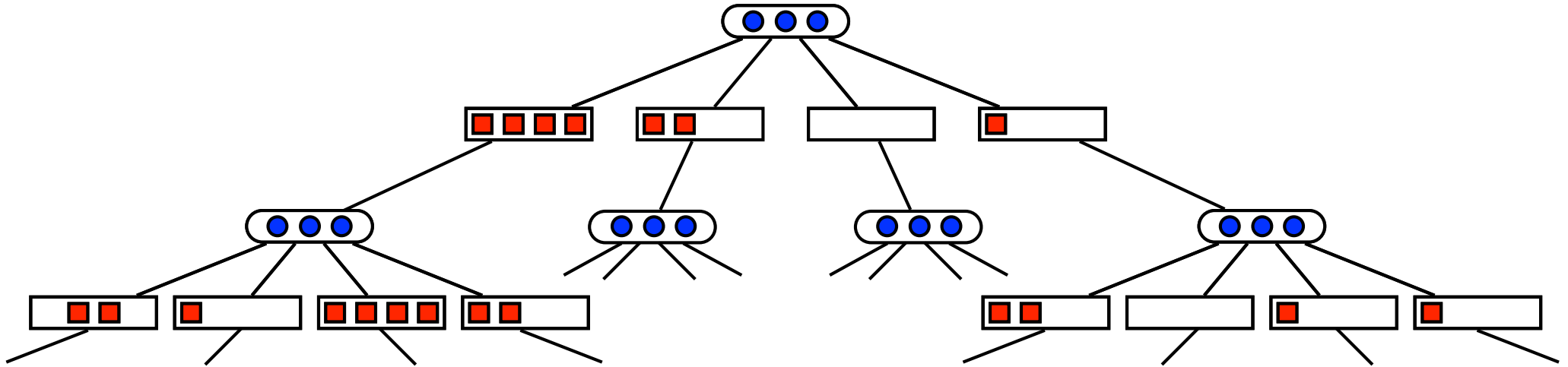
- Idea.
 - Speed up updates by **buffering** them at each node along the path to a leaf.
 - Move many updates together in each I/O.
 - Search (almost) as before.
 - $\varepsilon \in (0, 1]$ is a parameter.
- Solution in 2 steps.
 - Focus on \sqrt{B} -tree ($\varepsilon = 1/2$).
 - Searching in $O(\log_B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

\sqrt{B} -tree



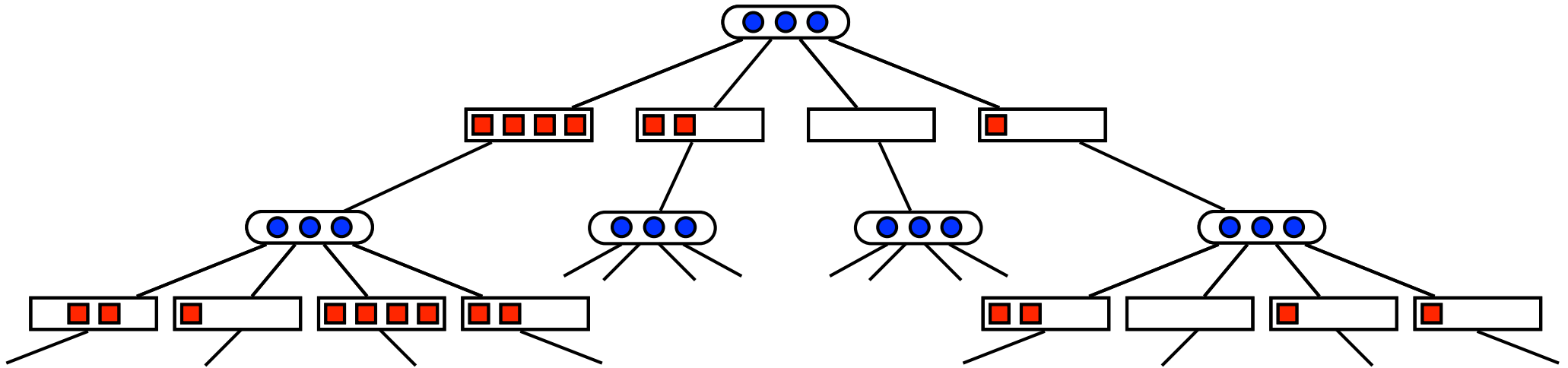
- \sqrt{B} -tree with N keys.
 - B-tree of degree $\Theta(\sqrt{B})$ with **buffers** of size $\Theta(\sqrt{B})$ at each edge.
 - Buffer stores **delayed updates** in subtree.
 - Nodes and child buffers stored together in $O(1)$ blocks.
- **Height.** $\Theta(\log_{\sqrt{B}} N) = \Theta(\log_B N)$

\sqrt{B} -tree



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

\sqrt{B} -tree



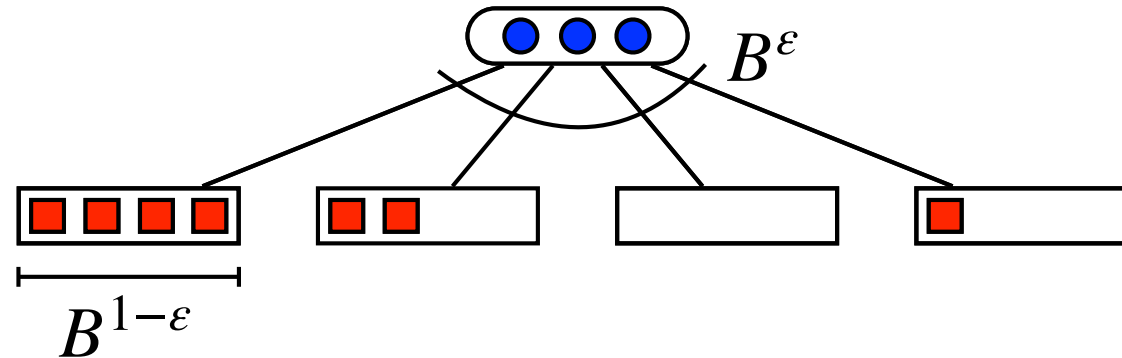
- Updates.

- 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/Os. Intuition: A flush moves \sqrt{B} messages $\Rightarrow O((\log_B N)/\sqrt{B})$ amortized I/Os.

- Assign $(c \log_B N)/\sqrt{B}$ 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. $O\left(\frac{\log_B N}{\epsilon B^\epsilon}\right)$ 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	$O\left(\frac{\log_B N}{\varepsilon}\right)$	$O\left(\frac{\log_B N}{\varepsilon B^\varepsilon}\right)$

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:
 - $\text{search}(P)$: return string in S with longest common prefix with P .
 - $\text{insert}(P)$: set $S = S \cup \{P\}$
 - $\text{delete}(P)$: set $S = S - \{P\}$

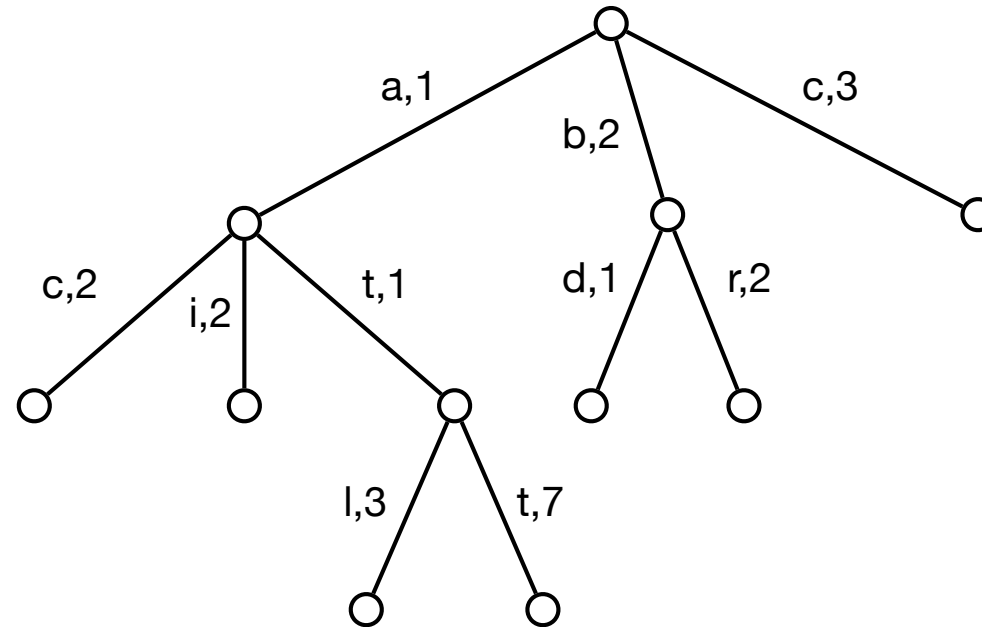
String Searching

- Which solutions do we know (on the RAM model)?

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.

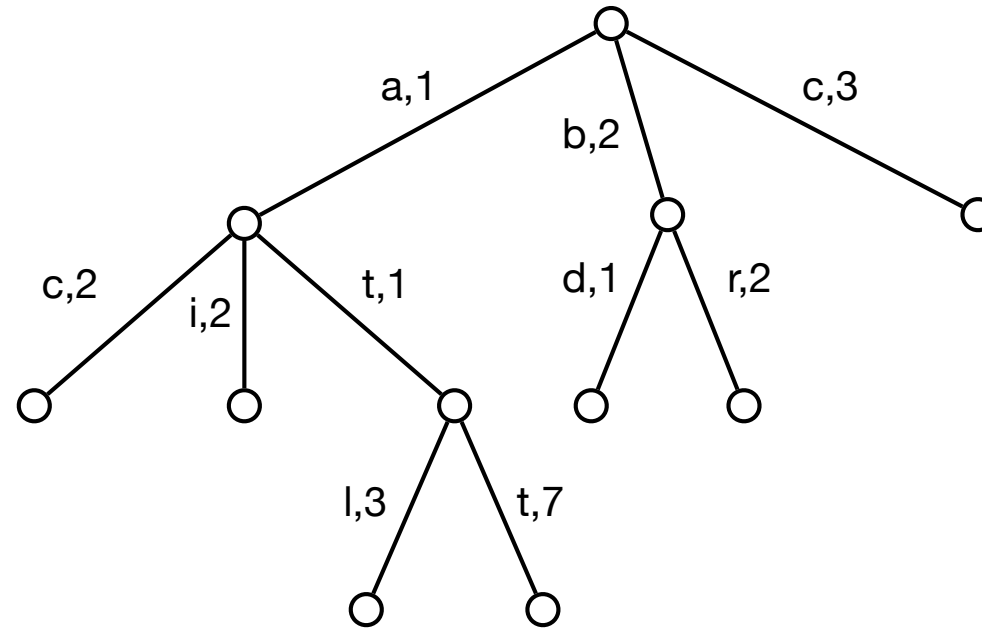
Blind Trie



1. ace
2. aid
3. atlas
4. atom
5. attenuate
6. bid
7. bird
8. car

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

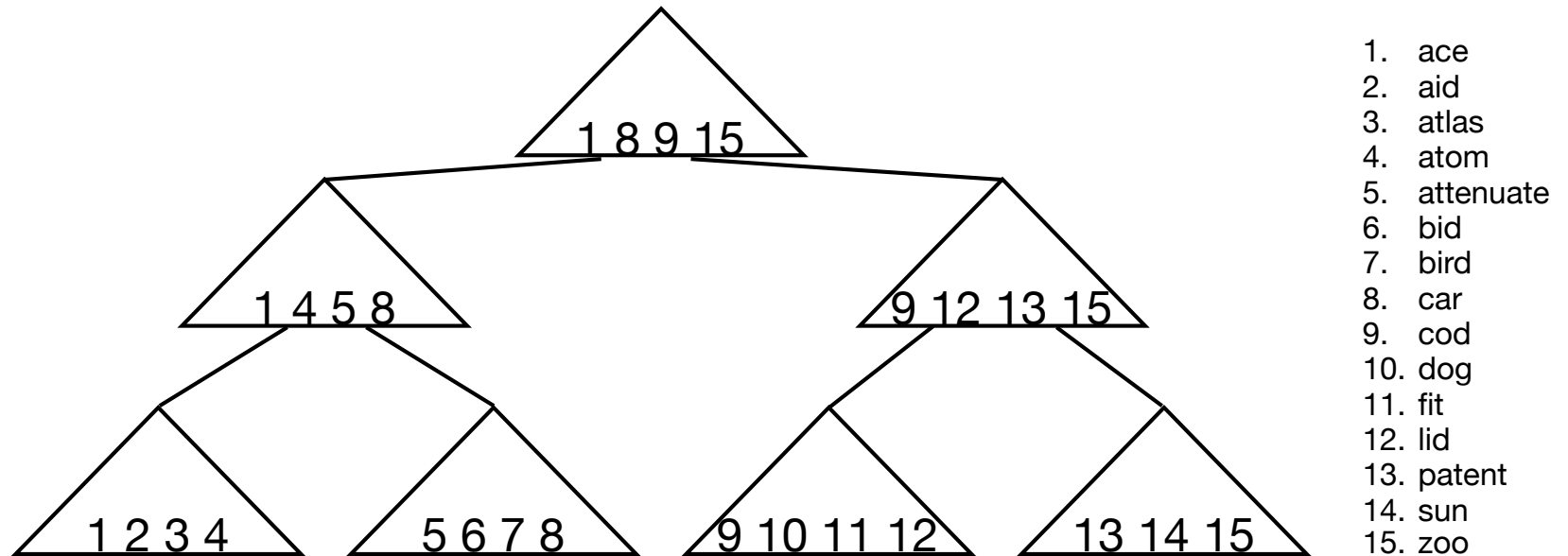
Blind Trie



1. ace
2. aid
3. atlas
4. atom
5. attenuate
6. bid
7. bird
8. car

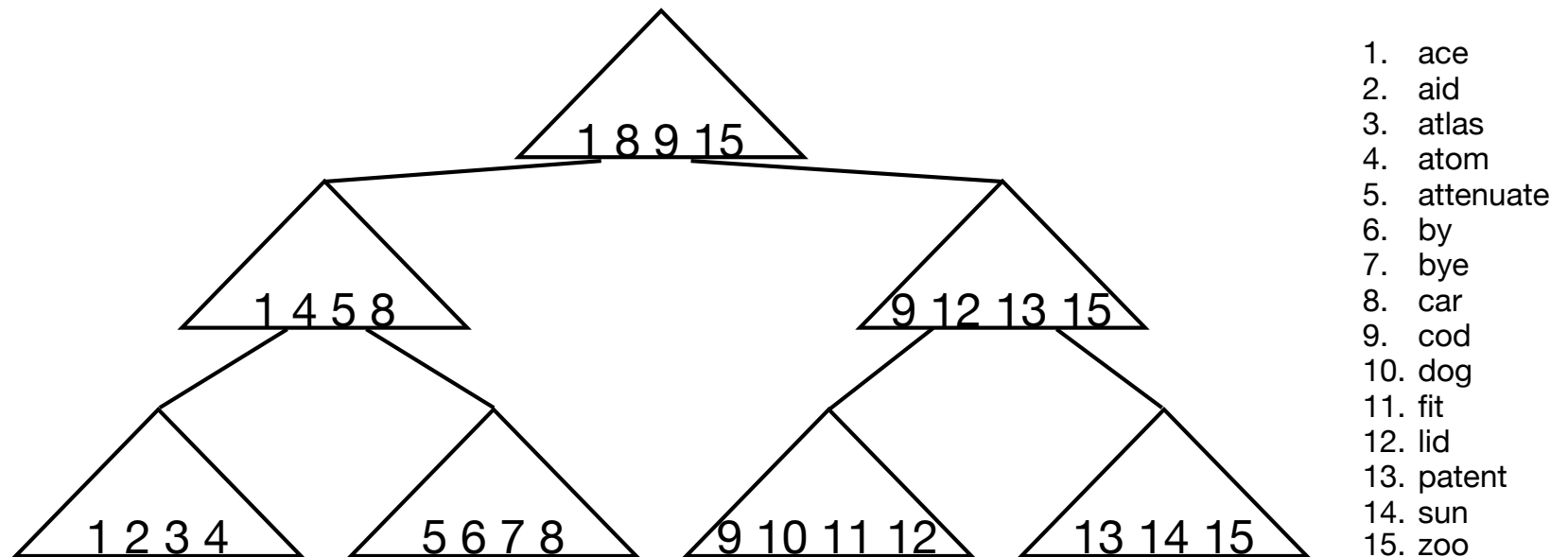
- **Search.** Traverse and verify candidate.
- **Time.** $O(|P|)$

String B-tree



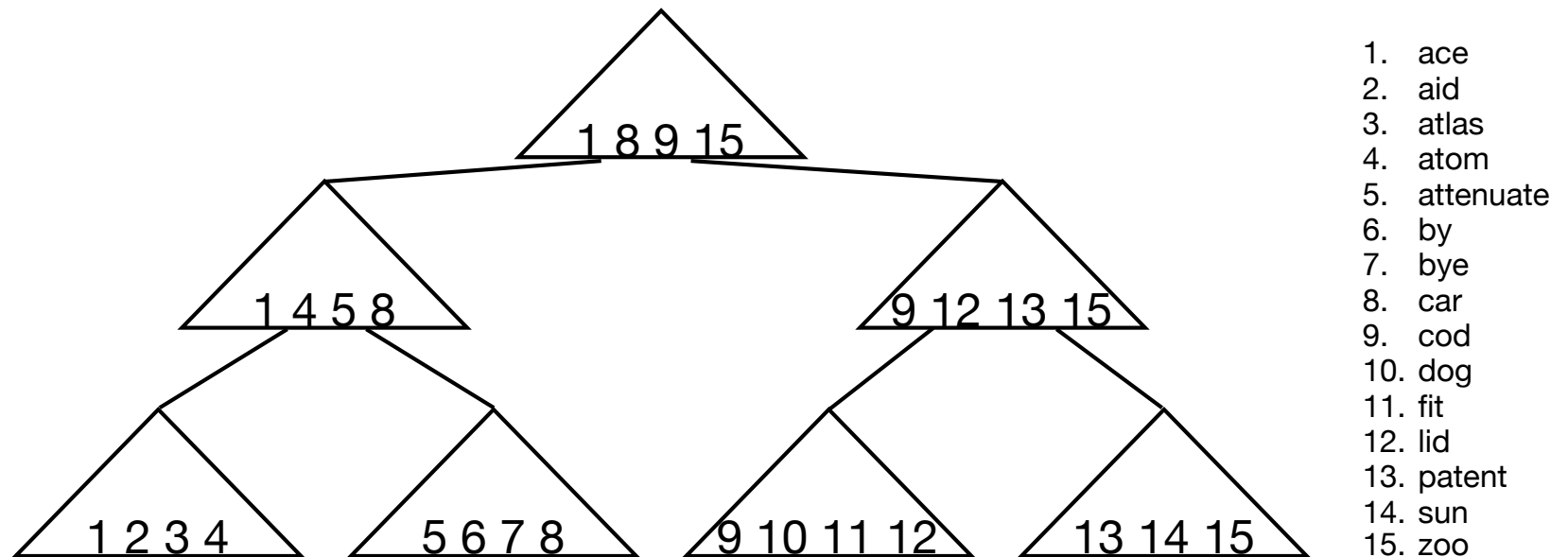
- **Data structure.** Combination of B-tree and blind tries.
 - Sorted set of strings.
 - Nodes store blind trie over B strings.
 - Leftmost and rightmost string in subtree stored for each child.
- **Space.**
 - Strings: $O(N)$
 - B-tree: $O(B)$ per node.
 - $\Rightarrow O(N)$.

String B-tree



- Searching.
 - Find leaf using routing with blind tries.
 - Verify leaf.
- I/Os.
 - Routing at node: $O(|P|/B)$ I/Os.
 - $\Rightarrow O((|P|/B)\log_B N)$ I/Os.

String B-tree



- **Fast searching.**
 - Remember longest prefix at each node.
 - Verify leaf.
- **I/Os.**
 - Routing in total: $O(|P|/B)$ I/Os.
 - $\Rightarrow O((|P|/B) + \log_B N)$ I/Os.

External Memory II

- Searching with Fast Updates
- Searching Strings