S(b)-Trees: an Optimal Balancing of Variable Length Keys

Konstantin V. Shvachko

2
Dynamic Dictionaries

Let K be a set of dictionary elements, called keys.
For any finite subset D of K and for any key k three operations of search, insertion, and deletion are defined as follows

$$\begin{aligned} Search(D,k) &= k \in D \\ Insert(D,k) &= D \cup \{k\} \\ Delete(D,k) &= D \setminus \{k\} \end{aligned}$$

The problem is to provide space-efficient way of storing keys, and time-efficient algorithms for performing the operations.

3
Linearly Ordered Key Sets

- For linearly ordered key sets searching can be performed in logarithmic on the number of keys time.
- Otherwise, only the exhaustive search algorithm is applicable.
- A logarithmic lower bound is proven for searching in a finite linearly ordered set.
- $log\ n$ is the optimum for searching in linearly ordered sets.
- $log\ n$ is also the optimum for insertions and deletions, since in order to insert or delete a key it is particularly necessary to check whether the key is contained in the input set.

4 Trees

- Balanced trees are considered to be a standard solution for the problem.
- Trees store keys chosen from a finite linearly ordered key set K.
- A node $S = \langle S_0, k_1, S_1, \dots, k_m, S_m \rangle$ of the tree contains a sequence of keys k_i from K separating references to child nodes S_i, such that if the number of keys is m then the number of references is $m+1$.
- For the leaf nodes all the references are empty.
- Keys are placed into the tree according to the ordering

$$0 \le i \le m \Rightarrow k_i < S_{i+1} < k_{i+1}$$

- All paths in the tree from the root to the leaves have equal length.
- Structured trees.

5
History of Balanced Trees

- 1962 *AVL-tree* G.M.Adelson-Velskii and E.M.Landis
- 1970 *2-3-tree* J.Hopcroft
- 1972 *B-tree* R.Bayer
- *B*-tree, B+tree, (a,b)-tree, red-black-tree*
- 1992 *S(1)-tree* utilization $\frac{1}{2} - \varepsilon$
- 1994 *S(2)-tree* utilization $\frac{2}{3} - \varepsilon$
- 1995 *S(b)-tree* utilization $1 - \varepsilon$

6
B-trees

- B-tree *T* of order *q* is a structured tree such that for any node *S* except for the root the number of keys in it is

$$q \le |k(S)| \le 2q$$

- Utilization $\delta(T) = \frac{|K(T)|}{2qn}$
- Lower bound $\delta(T) > \frac{1}{2} - \frac{1}{2q}$
- Search, insertion, and deletion can be performed in time

$$O(\log n)$$

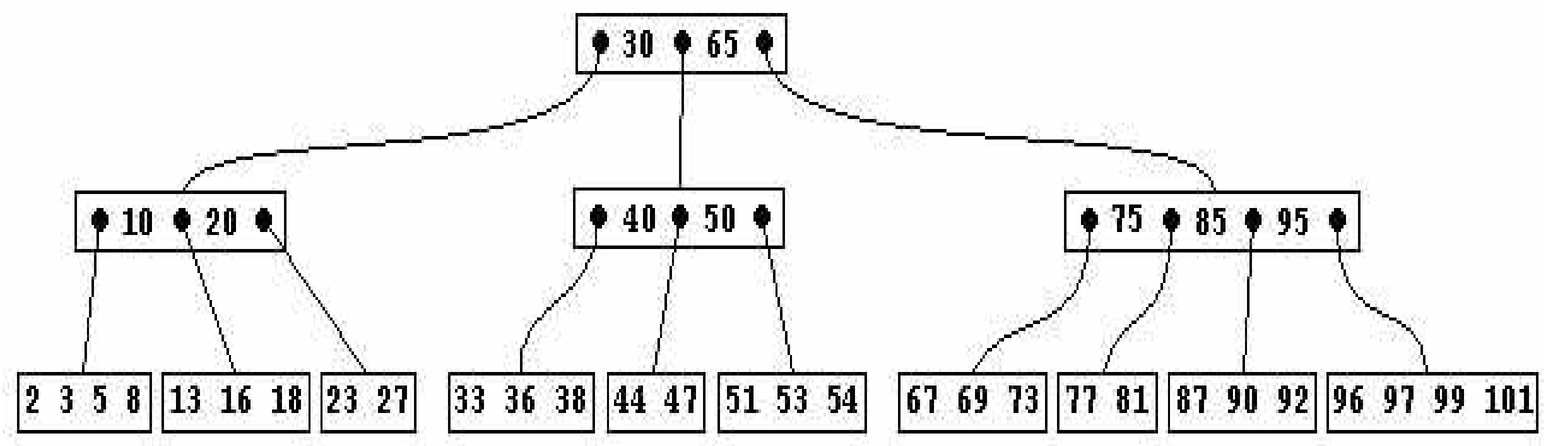
- Disadvantage: key weight is not taken into account.
  Cannot guarantee any lower bound greater than 0 with the weight taken into account

Weight

- $\mu(k)$ – key weight
- $\mu(S)$ – node weight
- $M(T)$ – total weight of all keys in tree T
- $\mu_{max}(K) = max\{\mu(k) \mid k \text{ in } K\}$
- p – node capacity: $\mu(S) \leq p$
- *Utilization* $$\Delta(T) = \frac{M(T)}{np}$$

8
Sweep

- Neighboring nodes, delimiting keys.
- A sequence $\sigma = S_0, k_1, S_1, \ldots, k_m, S_m$ of vertices and keys of a tree T is called a ***sweep*** iff each pair S_{i-1}, S_i is a pair of neighbors and k_i is their delimiting key.



9
S(b)-tree properties

- b – locality parameter
- q – tree order: $|k(S)| \geq q$
- p – tree rank: $\mu(S) \leq p$
- $\mu_{max}(K)$ – *maximal key weight*
- Sweep σ composed of $m+1$ nodes is **dense** if $\mu(\sigma) \geq mp$
- Sweep σ composed of $m+1$ nodes is **incompressible w.r.t. *p* and *q*** if nodes of σ cannot be "compressed" into m nodes with the same rank p and order q.
- T is b-locally dense if all its sweeps of length b are dense.
- T is b-locally incompressible if all its sweeps of length b are incompressible.

10
S(b)-tree definitions

Let *K* be a weighted linearly ordered set of keys.

1. A structured *b*-locally dense tree *T* of order *q* and rank *p* is called a DS(b)-tree of order order *q* and rank *p,* if its parameters *b*, *q*, and *p* are natural numbers satisfying

$$q > 0, \quad q \geq b, \quad p \geq 2q\, \mu_{max}(K)$$

2. A structured *b*-locally incompressible tree *T* of order *q* and rank *p* is called a S(b)-tree of order order *q* and rank *p,* if its parameters *b*, *q*, and *p* are natural numbers satisfying

$$q > 0, \quad q \geq b, \quad p \geq 2q\, \mu_{max}(K)$$

Respective tree classes are denoted by *DS(b,q,p)* and *S(b,q,p).*

11
Hierarchy of balanced trees

- Class of all structured trees is $\bigcup_{q>0} \bigcup_{p>2q} S(0,q,p)$
- If $\mu \equiv 1$ on K then class of B-trees of order q is $S(0,q,2q)$.
- Class of 2-3-trees is $S(0,1,2)$.
- $S(0,q,p) = DS(0,q,p)$
  $S(1,q,p) = DS(1,q,p)$
  $S(1,q,p) \subset DS(1,q,p)$ for all $b > 1$
- If $b' < b < q \leq p/ 2q\, \mu_{max}(K)$ then
  $S(b,q,p) \subset S(b',q,p)$
- The same is not true for DS(b)-trees
- If $b < q < q' \leq p/ 2q\, \mu_{max}(K)$ then
  $S(b,q,p) \subset S(b,q',p)$
  $DS(b,q,p) \subset DS(b,q',p)$

Lower bounds

Theorem 1. Let $T \in DS(b,q,p)$ and $n > q+1$ number of tree nodes.

Then
$$\Delta(T) = \frac{b}{b+1}\frac{q}{q+1} - \frac{b+1}{n}$$

Theorem 2. If locality parameter $b > 0$ is fixed, then for any $\varepsilon > 0$ two parameters $q \geq b$ and $p \geq 2q\,\mu_{max}(K)$ can be chosen such that for any tree $T \in DS(b,q,p)$ having $n \geq (b+1)(q+1)$ nodes its utilization is

$$\Delta(T) = \frac{b}{b+1} - \varepsilon$$

Theorem 3. For any $\varepsilon > 0$ three parameters $b > 0$, $q \geq b$ and $p \geq 2q\,\mu_{max}(K)$ can be chosen such that for any tree $T \in DS(b,q,p)$ having $n \geq (b+1)(b+1)$ nodes its utilization is

$$\Delta(T) = 1 - \varepsilon$$

13 Algorithms

Theorem 4.
Search, insertion, and deletion of a key in a S(b)-tree containing *n* nodes can be performed in time *O(log n)*.