WS 2007/2008

# Fundamental Algorithms


Dmytro Chibisov, Jens Ernst

Fakultät für Informatik
TU München


http://www14.in.tum.de/lehre/2007WS/fa-cse/

Fall Semester 2007

# 1. (a,b)-Trees

As we saw in the previous section, the efficiency of standard operations on binary search trees depends on the maximum tree height. Using height balancing, we ensure that trees cannot degenerate linearly but instead have logarithmic height. Let us extend this approach to more general trees.

**Motivation**: assume tree nodes are stored in secondary storage (hard disk). Comparisons of keys of binary trees would be too time expensive due to mechanical positioning of the read-write head of the hard drive. Reading blocks of data (sectors, pages, etc.) is relatively fast provided the read-write head is positioned.

**Idea**: store blocks of data in nodes of trees !

**Advantages**: faster access to the data and decreasing height of trees !

(a,b)-Trees have been invented by Rudolf Bayer and Edward M. McCreight (1976).

## 1. (a,b)-Trees

As we saw in the previous section, the efficiency of standard operations on binary search trees depends on the maximum tree height. Using height balancing, we ensure that trees cannot degenerate linearly but instead have logarithmic height. Let us extend this approach to more general trees.

**Motivation**: assume tree nodes are stored in secondary storage (hard disk). Comparisons of keys of binary trees would be too time expensive due to mechanical positioning of the read-write head of the hard drive. Reading blocks of data (sectors, pages, etc.) is relatively fast provided the read-write head is positioned.
**Idea**: store blocks of data in nodes of trees !
**Advantages**: faster access to the data and decreasing height of trees !
(a,b)-Trees have been invented by Rudolf Bayer and Edward M. McCreight (1976).

## 1. (a,b)-Trees

As we saw in the previous section, the efficiency of standard operations on binary search trees depends on the maximum tree height. Using height balancing, we ensure that trees cannot degenerate linearly but instead have logarithmic height. Let us extend this approach to more general trees.

**Motivation**: assume tree nodes are stored in secondary storage (hard disk). Comparisons of keys of binary trees would be too time expensive due to mechanical positioning of the read-write head of the hard drive. Reading blocks of data (sectors, pages, etc.) is relatively fast provided the read-write head is positioned.
**Idea**: store blocks of data in nodes of trees !
**Advantages**: faster access to the data and decreasing height of trees !
(a,b)-Trees have been invented by Rudolf Bayer and Edward M. McCreight (1976).

## 1. (a,b)-Trees

As we saw in the previous section, the efficiency of standard operations on binary search trees depends on the maximum tree height. Using height balancing, we ensure that trees cannot degenerate linearly but instead have logarithmic height. Let us extend this approach to more general trees.

**Motivation**: assume tree nodes are stored in secondary storage (hard disk). Comparisons of keys of binary trees would be too time expensive due to mechanical positioning of the read-write head of the hard drive. Reading blocks of data (sectors, pages, etc.) is relatively fast provided the read-write head is positioned.
**Idea**: store blocks of data in nodes of trees !
**Advantages**: faster access to the data and decreasing height of trees !
(a,b)-Trees have been invented by Rudolf Bayer and Edward M. McCreight (1976).

## Definition 1

Consider a node $v$ of a search tree and let $deg(v)$ be the number of sons of $v$. (a,b)-Tree is a tree with following properties:

- All keys are located on the same level
- For every vertex $v$ internal $b \geq deg(v) \geq a$
- $a \geq 2$ and $b \geq 2a - 1$
- For the root $b \geq deg(v) \geq 2$
- For every vertex $v$ all keys stored in the $i$th subtree are less than keys stored in the $(i+1)$th subtree
- For every internal node $v$, let $m_v = deg(v)$. Then
  - $v$ has $(m_v - 1)$ key values
  - $k_1 < k_2 < \cdots < k_{m_v - 1}$
  - For $1 \leq i \leq m_v$ the following is satisfied: $k_{i-1} <$ keys in the $i$th subtree $\leq k_i$

## Definition 1

Consider a node $v$ of a search tree and let $deg(v)$ be the number of sons of $v$. (a,b)-Tree is a tree with following properties:

- All keys are located on the same level
- For every vertex $v$ internal $b \geq deg(v) \geq a$
- $a \geq 2$ and $b \geq 2a - 1$
- For the root $b \geq deg(v) \geq 2$
- For every vertex $v$ all keys stored in the $i$th subtree are less than keys stored in the $(i+1)$th subtree
- For every internal node $v$, let $m_v = deg(v)$. Then
  - $v$ has $(m_v - 1)$ key values
  - $k_1 < k_2 < \cdots < k_{m_v - 1}$
  - For $1 \leq i \leq m_v$ the following is satisfied: $k_{i-1} <$ keys in the $i$th subtree $\leq k_i$

## Definition 1

Consider a node $v$ of a search tree and let $deg(v)$ be the number of sons of $v$. (a,b)-Tree is a tree with following properties:

- All keys are located on the same level
- For every vertex $v$ internal $b \geq deg(v) \geq a$
- $a \geq 2$ and $b \geq 2a - 1$
- For the root $b \geq deg(v) \geq 2$
- For every vertex $v$ all keys stored in the $i$th subtree are less than keys stored in the $(i+1)$th subtree
- For every internal node $v$, let $m_v = deg(v)$. Then
  - $v$ has $(m_v - 1)$ key values
  - $k_1 < k_2 < \cdots < k_{m_v - 1}$
  - For $1 \leq i \leq m_v$ the following is satisfied: $k_{i-1} <$ keys in the $i$th subtree $\leq k_i$

**Definition 1**

Consider a node $v$ of a search tree and let $deg(v)$ be the number of sons of $v$. (a,b)-Tree is a tree with following properties:

- All keys are located on the same level
- For every vertex $v$ internal $b \geq deg(v) \geq a$
- $a \geq 2$ and $b \geq 2a - 1$
- For the root $b \geq deg(v) \geq 2$
- For every vertex $v$ all keys stored in the $i$th subtree are less than keys stored in the $(i+1)$th subtree
- For every internal node $v$, let $m_v = deg(v)$. Then
  - $v$ has $(m_v - 1)$ key values
  - $k_1 < k_2 < \cdots < k_{m_v - 1}$
  - For $1 \leq i \leq m_v$ the following is satisfied: $k_{i-1} <$ keys in the $i$th subtree $\leq k_i$

## Definition 1

Consider a node $v$ of a search tree and let $deg(v)$ be the number of sons of $v$. (a,b)-Tree is a tree with following properties:

- All keys are located on the same level
- For every vertex $v$ internal $b \geq deg(v) \geq a$
- $a \geq 2$ and $b \geq 2a - 1$
- For the root $b \geq deg(v) \geq 2$
- For every vertex $v$ all keys stored in the $i$th subtree are less than keys stored in the $(i+1)$th subtree
- For every internal node $v$, let $m_v = deg(v)$. Then
  - $v$ has $(m_v - 1)$ key values
  - $k_1 < k_2 < \cdots < k_{m_v - 1}$
  - For $1 \leq i \leq m_v$ the following is satisfied: $k_{i-1} <$ keys in the $i$th subtree $\leq k_i$

## Definition 1

Consider a node $v$ of a search tree and let $deg(v)$ be the number of sons of $v$. (a,b)-Tree is a tree with following properties:

- All keys are located on the same level
- For every vertex $v$ internal $b \geq deg(v) \geq a$
- $a \geq 2$ and $b \geq 2a - 1$
- For the root $b \geq deg(v) \geq 2$
- For every vertex $v$ all keys stored in the $i$th subtree are less than keys stored in the $(i+1)$th subtree
- For every internal node $v$, let $m_v = deg(v)$. Then
  - $v$ has $(m_v - 1)$ key values
  - $k_1 < k_2 < \cdots < k_{m_v - 1}$
  - For $1 \leq i \leq m_v$ the following is satisfied: $k_{i-1} <$ keys in the $i$th subtree $\leq k_i$

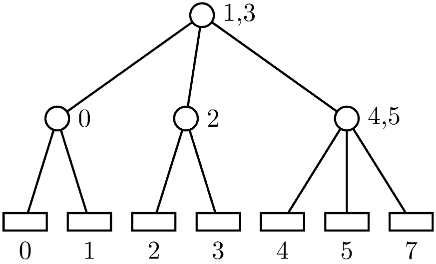
## Definition 1

Consider a node $v$ of a search tree and let $deg(v)$ be the number of sons of $v$. (a,b)-Tree is a tree with following properties:

- All keys are located on the same level
- For every vertex $v$ internal $b \geq deg(v) \geq a$
- $a \geq 2$ and $b \geq 2a - 1$
- For the root $b \geq deg(v) \geq 2$
- For every vertex $v$ all keys stored in the $i$th subtree are less than keys stored in the $(i+1)$th subtree
- For every internal node $v$, let $m_v = deg(v)$. Then
  - $v$ has $(m_v - 1)$ key values
  - $k_1 < k_2 < \cdots < k_{m_v - 1}$
  - For $1 \leq i \leq m_v$ the following is satisfied: $k_{i-1} <$ keys in the $i$th subtree $\leq k_i$

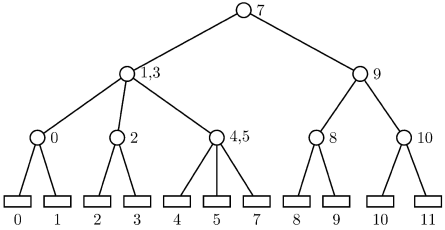
## Example 2

The (2,3)-tree:



## Example 3

The (2,3)-tree:



# 2. Operations for (a,b)-Trees

## 2.1 is_element

This operation has to be implemented like for general binary search trees. The only difference is that the higher branching factor has to be treated appropriately.

Algorithm:

data is_element(key $k$){
  $v$:=root of the tree
  while ($v$ is not a leaf) do
    $i := min\{j|1 \leq j \leq deg(v) \wedge k \leq k_j\}$
    $v := i$th child of $v$
  od
  $location := v$
  if ($v.key = k$) then $location := v$; return $v.data$
    else return NULL;
  fi
}

## 2.2 insert

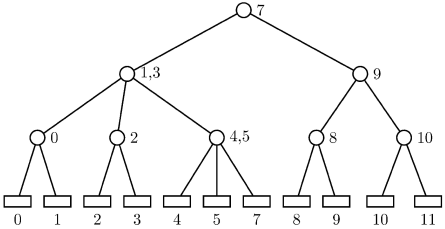
- is_element finds the position for the element to be inserted (stored in $location$)
- attach new leaf to the leaf in $location$
- If the branching factor of the leaf in $location \geq b + 1$ – do rebalancing

## 2.3 insert: Rebalancing

- Split the node $w$, $deg(w) \geq b + 1$ into $v_1$ and $v_2$.
- Assign first $a$ sons of $w$ to $v_1$, and the remaining $b + 1 - a$ sons to $v_2$.
  - Since $b \geq 2a - 1$ (see Definition 1), we obtain that $deg(v_1) \geq a$, $deg(v_2) \geq a$.
- This may increase the degree of the ancestor of $w$ – repeat splitting for the ancestor of $w$.
- In necessary – proceed up to the root.
- The root may also be divided into two nodes, then create a new root – the height of the tree increases.
  - Since according to Definition 1 $b \geq deg(root) \geq 2$, splitting of the root into two nodes is valid.

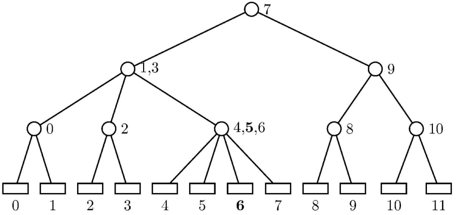
## Example 4

Insert $6$:



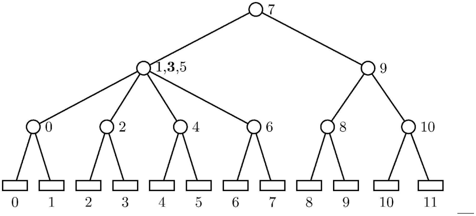
## Example 5

Rebalancing:



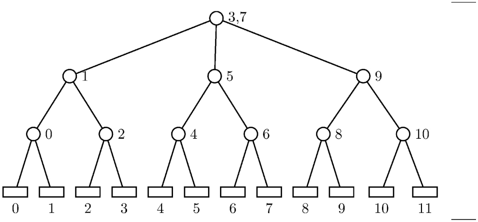
## Example 6

Rebalancing:



## Example 7

Rebalancing:



## 2.4 delete

- is_element finds the position for the element to be removed (stored in $location$)
- remove the element stored in $location$
- If the branching factor of the ancestor of the node in $location$ $< a$ – do rebalancing

## 2.5 delete: Rebalancing

- Let $deg(v_1) < a$ – merge $v_1$ and its brother $v_2$ into the new node $w$.
- If $deg(w) > b$, split $w$ into two new nodes $v_1$, $v_2$ and assign fist $a$ sons to the first node.
  - Since $b \geq 2a - 1$ (see Definition 1), we obtain that $deg(v_1) \geq a$, $deg(v_2) \geq a$.
  - The number of sons of the ancestor of $w$ is not changed in this case.
- If $deg(w) \leq b$ the merging may decrease the degree of the ancestor of $w$ – repeat merging for the ancestor of $w$.
- In necessary – proceed up to the root.

## Theorem 8

*For the (a,b)-Tree with $n$ nodes and height $h$ the following is satisfied:*

- $2a^{h-1} \leq n \leq b^h$
- $\log_b(n) \leq h \leq \log_a(n/2) + 1$

Proof.

Easy. Homework. □

### Theorem 8

*For the (a,b)-Tree with $n$ nodes and height $h$ the following is satisfied:*

- $2a^{h-1} \leq n \leq b^h$
- $\log_b(n) \leq h \leq \log_a(n/2) + 1$

### Proof.

Easy. Homework. □