

2-14 *B*-Trees

Hengfeng Wei

hfwei@nju.edu.cn

June 02, 2020



Organization and Maintenance of Large Ordered Indexes

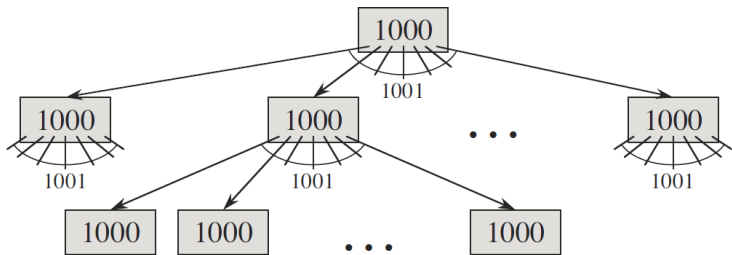
R. BAYER and E. MCCREIGHT

Received September 29, 1971

Summary. Organization and maintenance of an index for a dynamic random access file is considered. It is assumed that the index must be kept on some pseudo random access backup store like a disc or a drum. The index organization described allows retrieval, insertion, and deletion of keys in time proportional to $\log_k I$ where I is the size of the index and k is a device dependent natural number such that the performance of the scheme becomes near optimal. Storage utilization is at least 50% but generally much higher. The pages of the index are organized in a special data-structure, so-called *B-trees*. The scheme is analyzed, performance bounds are obtained, and a near optimal k is computed. Experiments have been performed with indexes up to 100000 keys. An index of size 15000 (100000) can be maintained with an average of 9 (at least 4) transactions per second on an IBM 360/44 with a 2311 disc.

ID	Name	Gender	Age	...
----	------	--------	-----	-----

*“Bayer and McCreight introduced B-trees in 1972;
they **did not** explain their choice of name.”*



2-way *vs.* multi-way

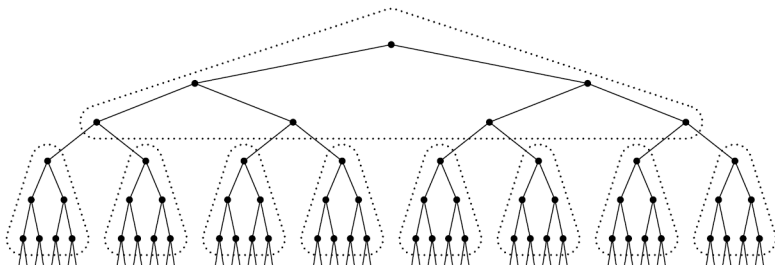
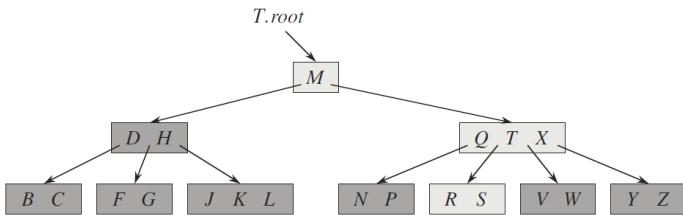


Fig. 29. A large binary search tree can be divided into “pages.”

indexes (keys) *vs.* pages

Minimum (TC 18.2-3)

Explain how to find the **minimum** key stored in a *B*-tree.



the leftmost key in the leftmost node

Predecessor (TC 18.2-3)

Explain how to find the predecessor of a given key (x, i) stored in a B -tree.



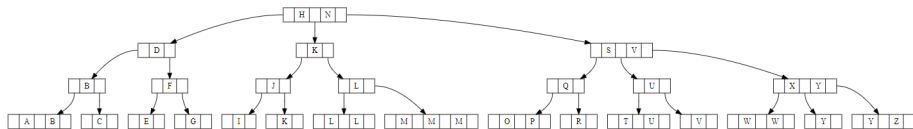
B -tree with 37 characters.

$x.leaf = 0$

$H \quad N \quad S \quad V$

find the rightmost key in $x.c_i$

(x, i)



$x.leaf = 1$

$P \quad U \quad T \quad O \quad A$

$i \geq 2 \implies (x, i - 1)$

$i = 1 \implies$ find (y, j) such that x is the **leftmost key** in $y.c_{j+1}$

A is the **only** key which has no predecessor.

```

1: procedure B-TREE-PREDECESSOR( $T, x, i$ )    ▷  $x.key_i$  in  $B$ -tree  $T$ 
2:   if  $x.leaf = 0$  then
3:     return the rightmost key in  $x.c_i$ 
4:   else if  $i \geq 2$  then                                ▷  $x.leaf = 1$ 
5:     return  $(x, i - 1)$ 
6:   else                                                ▷  $x.leaf = 1 \wedge i = 1$ 
7:      $y \leftarrow x.p$ 
8:     while  $y \neq T.root \wedge x = y.c_1$  do ▷ exit:  $y = T.root \vee x \neq y.c_1$ 
9:        $x \leftarrow y$ 
10:       $y \leftarrow y.p$ 
11:     if  $x = y.c_1$  then                                ▷  $y = T.root \wedge x = y.c_1$ 
12:       return “no predecessor”
13:     else                                              ▷  $x \neq y.c_1$ 
14:        $j \leftarrow 2$ 
15:       while  $y.c_j \neq x$  do
16:          $j \leftarrow j + 1$ 
17:       return  $(y, j - 1)$                                 ▷  $x = y.c_j$ 

```


Insertion (TC 18.2-4 ★)

Suppose that we insert the keys $\{1, 2, \dots, n\}$ in increasing order into an empty B -tree with minimum degree 2.

How many nodes, denoted X_n , does the final B -tree have?

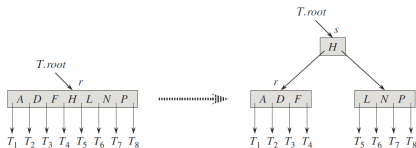


$$X_0 = 1$$

By Yangjing Dong (June 2018)

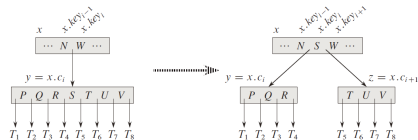
<https://maxmute.com/TC18.2-4.html>

Only **SPLIT** can create new nodes.



root SPLIT

+2



non-root SPLIT

+1

(I) Which nodes will SPLIT? S

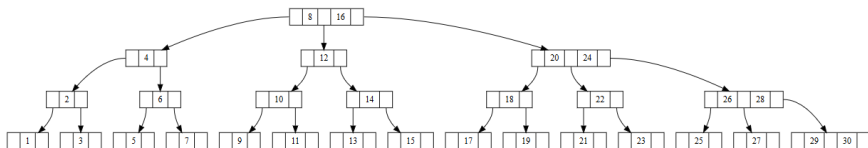
(II) When does each node $s \in S$ SPLIT? $T_s = \langle s_1, s_2, \dots \rangle$

(III) How does it SPLIT, as a root or a non-root? $T_s = s_R \uplus s_{NR}$

$$X_n = 1 + \sum_{s \in S} (2 |s_R| + |s_{NR}|)$$

$$X_n = 1 + \sum_{s \in S} (2 |s_R| + |s_{NR}|)$$

(I) Which nodes will SPLIT?



1, 2, ..., 30

$S = \left\{ \begin{array}{l} \text{the nodes in the } \textcolor{blue}{\text{rightmost}} \text{ chain} \end{array} \right\}$

(II) When does each node $s \in S$ SPLIT?

$$T_s = \langle s_1, s_2, \dots \rangle$$

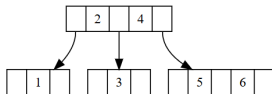
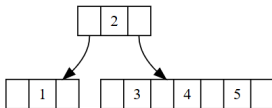
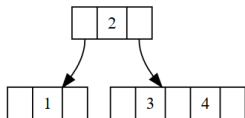
Let's focus the **rightmost** node first, denoted A .

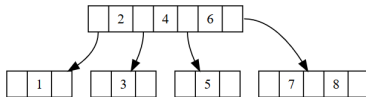
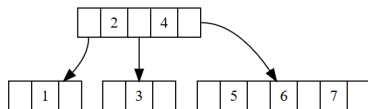
	1	
--	---	--

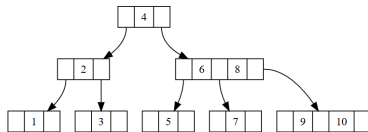
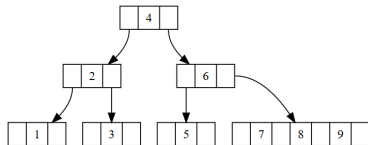
--

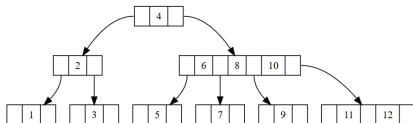
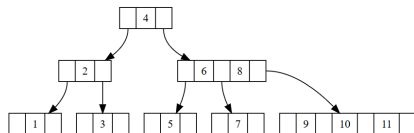
	1		2	
--	---	--	---	--

	1		2		3	
--	---	--	---	--	---	--







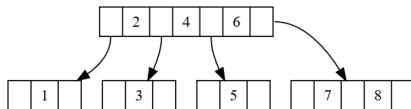
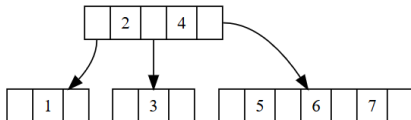
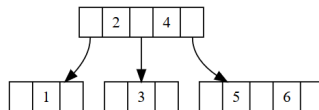
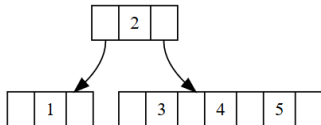
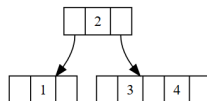
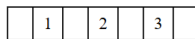


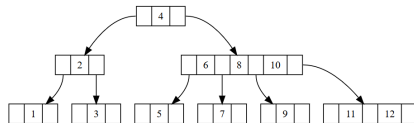
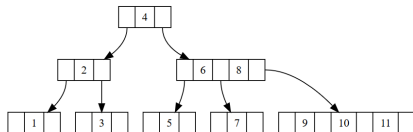
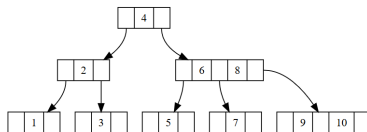
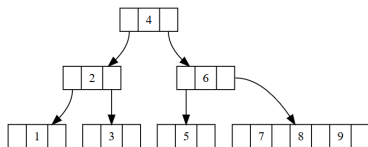
A SPLIT : 4, 6, 8, 10, 12, ...

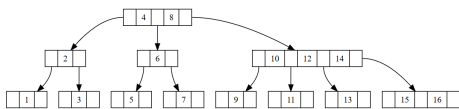
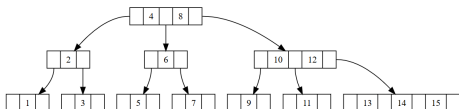
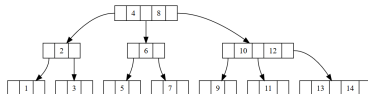
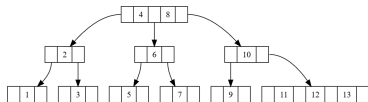
(II) When does each node SPLIT?

Let's consider the parent of A , denoted $B \triangleq p(A)$.

Every time A splits, B obtains a new key.







A SPLIT : 4, 6, 8, 10, 12, ...

B SPLIT : 9, 13, 17, 21, 25, ...

(II) When does each node SPLIT?

Let's consider the parent of B , denoted $C = p(B)$.

A SPLIT : 4, 6, 8, 10, 12, ...
 B SPLIT : 9, 13, 17, 21, 25, ...
 C SPLIT : 18, 26, 34, 42, 50, ...

$A : 1 \quad B : 2 \quad C : 3$

T_i : the first time point the i -th node splits

$$T_1 = 4$$

$$T_i = \underbrace{T_{i-1}}_{\text{its right child first split}} + \underbrace{2 \times 2^{i-1}}_{\text{its right child split twice more}} + \underbrace{1}_{\text{insert one more}}$$

$$T_i = 2^{i+1} + i - 1$$

$$X_n = 1 + \sum_{s \in S} \left(2 \, |s_R| + |s_{NR}| \right)$$

$$(T_s = s_R \uplus s_{NR})$$

(III) How does it SPLIT, as a root or a non-root?

$$s_R = \{s_1\} \quad s_{NR} = \{s_2, s_3, \dots\}$$

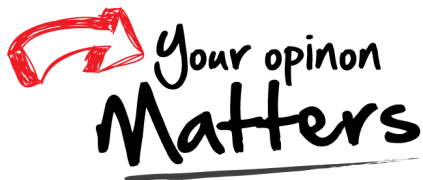
$$|s_R| = 1 \quad |s_{NR}| = |T_s| - 1$$

$$X_n = 1 + \sum_{s \in S} \left(2 + |T_s| - 1 \right) = 1 + \sum_{s \in S} \left(|T_s| + 1 \right)$$

$$X_n = 1 + \sum_{s \in S} (|T_s| + 1) \quad T_i = 2^{i+1} + i - 1$$

$$\begin{aligned}
 X_n &= 1 + \sum_{i=1}^{\infty} [T_i \leq n] \left(\left(\left\lfloor \frac{n - T_i}{2^i} \right\rfloor + 1 \right) + 1 \right) \\
 &= 1 + \sum_{i=1}^{\infty} [T_i \leq n] \left(\left\lfloor \frac{n - T_i}{2^i} \right\rfloor + 2 \right) \\
 &= 1 + \sum_{i=1}^{\infty} [T_i \leq n] \left(\left\lfloor \frac{n - 2^{i+1} - i + 1}{2^i} \right\rfloor \right) \\
 &= 1 + \sum_{\substack{i \geq 1 \\ 2^{i+1} + i - 1 \leq n}} \left\lfloor \frac{n - i + 1}{2^i} \right\rfloor \\
 &= 1 + \sum_{\substack{i \geq 0 \\ 2^{i+2} + i \leq n}} \left\lfloor \frac{n - i}{2^{i+1}} \right\rfloor
 \end{aligned}$$

Thank
You!



Office 302

Mailbox: H016

hfwei@nju.edu.cn