

B-Trees

Disk access time is relatively slow,
but disk transfer rate is relatively fast.

Efficiency can be gained by moving many records
per disk transfer.

A node in a B-tree is intended to correspond to
one disk transfer.

B-trees are search trees with the properties:

1. $t =$ minimum degree, each internal node (except for root) has $\geq t$ children.
2. The root node contains 1 to $2t - 1$ keys.
3. Other nodes contain $t - 1$ to $2t - 1$ keys.
4. Keys in a node are in sorted order.
5. An internal node with k keys has $k + 1$ children.
6. $key[x, i-1] \leq$ keys in i th subtree $\leq key[x, i]$
7. Every leaf node has the same depth.

The Height of B-Trees

n keys and minimum degree t imply

$$h \leq \log_t \frac{n+1}{2}$$

Let $T(d)$ = minimum number of nodes at level d assuming $d \leq h$.

Basis: $T(1) = 2 = 2t^0$

Assume: $T(d-1) = 2t^{d-2}$ for $1 < d \leq h$

Show: $T(d) = 2t^{d-1}$ if $1 < d \leq h$

Induction: Each node at depth $d-1$ has $\geq t$ children, so $T(d) = t(2t^{d-2}) = 2t^{d-1}$

Root node has ≥ 1 key.

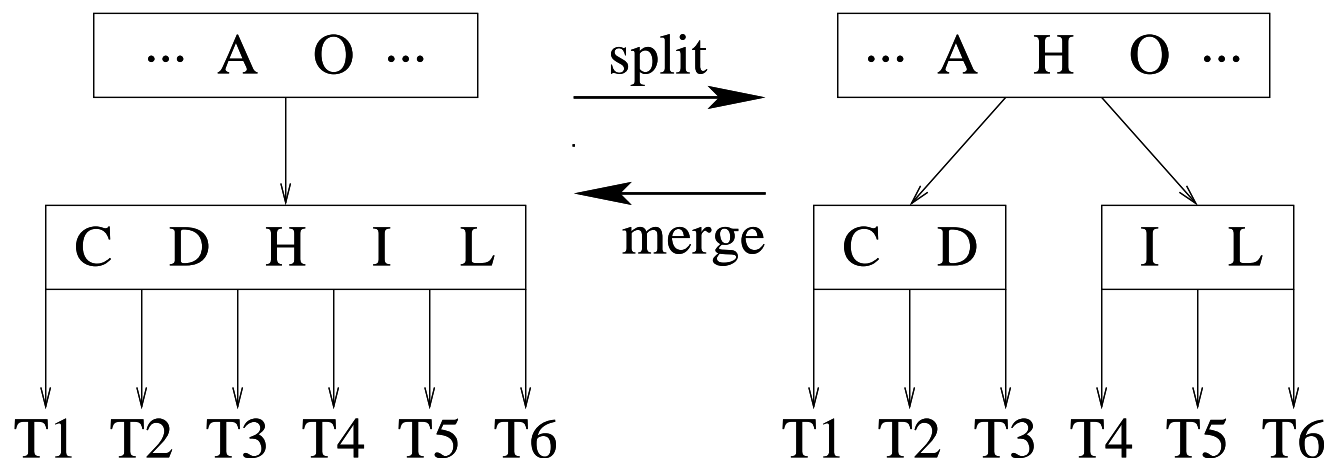
Other nodes have $\geq t-1$ keys.

$$n \geq 1 + (t-1) \sum_{d=1}^h 2t^{d-1} = 2t^h - 1$$

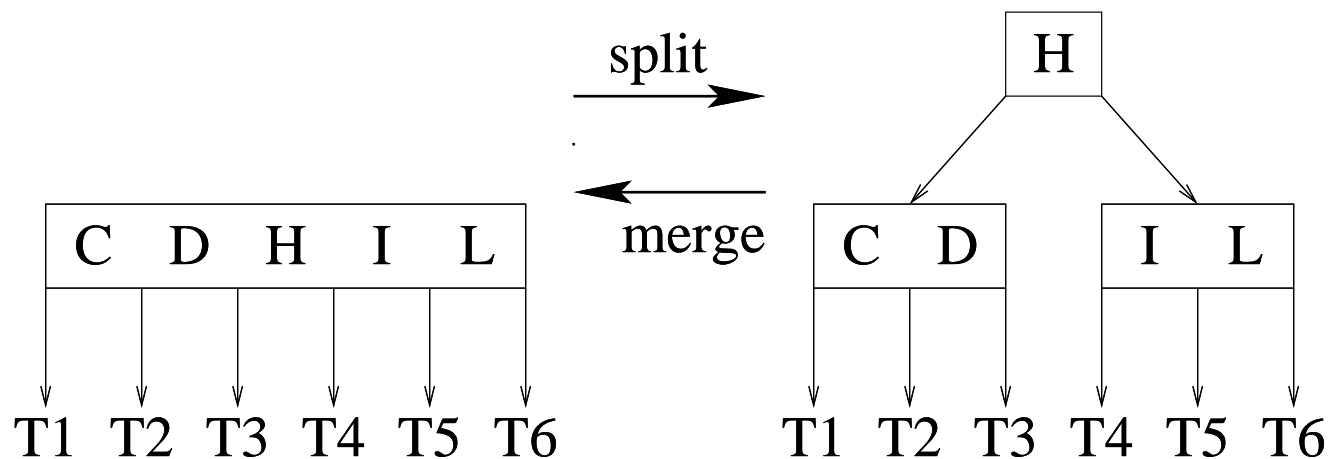
This is equivalent to:

$$\frac{n+1}{2} \geq t^h$$

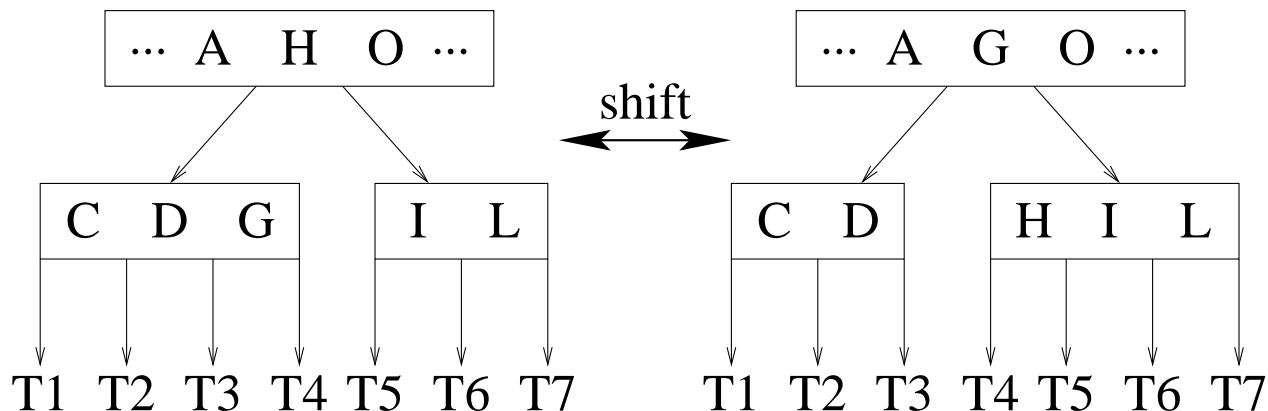
Splitting and Merging Children



Splitting and Merging Root



Shifting from Siblings



B-TREE-INSERT(T, k)

$x \leftarrow \text{root}[T]$

loop

if x is full

then split x

$x \leftarrow$ left or right node of split

if x is a leaf node **then** exit loop

$x \leftarrow$ next node down

end loop

insert k into x

B-TREE-DELETE(T, k)

$x \leftarrow \text{root}[T]$

loop

if x has $t - 1$ keys and $x \neq \text{root}[T]$

then shift from or merge with sibling of x

if x contains k **then** exit loop

$x \leftarrow$ next node down

end loop

if x is a leaf node

then delete k from x

else $k' \leftarrow$ successor of k

 delete k' starting search from x

 replace k in x with k'