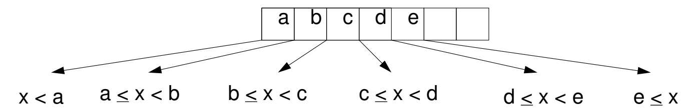
CSE 326
Lecture 12

Spring Quarter 1998

B - Trees

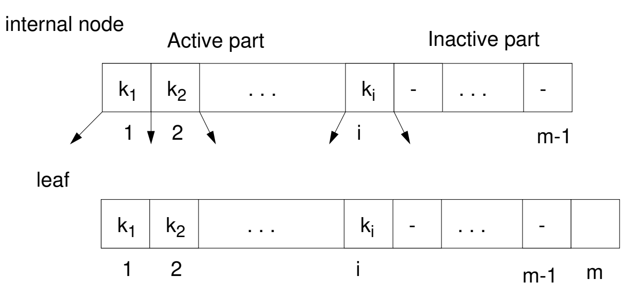
- B-trees are multiway trees which are not binary



- B-tree nodes contain search keys only, actual data is stored at the leaves.
- B-tree developed for database applications where minimizing the number of disk accesses is important. B-tree nodes are large enough to balance the disk access time with the internal search time within a B-tree node.

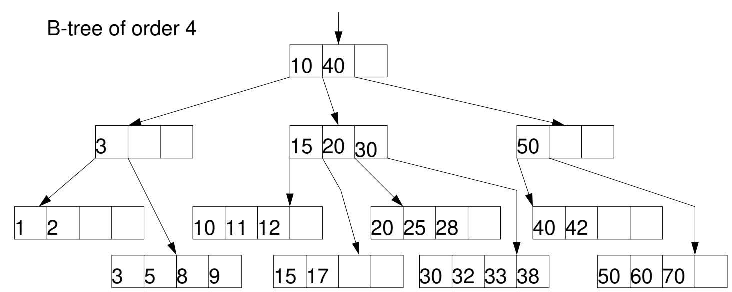
Properties of B-Trees of Order m

- The root is either a leaf or has between 2 and m children
- All non-leaf node (except the root) have between $\lceil m/2 \rceil$ and m children.
- All leaves are at the same depth and contain between $\lceil m/2 \rceil$ and m keys.



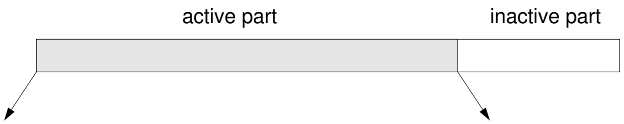
Example

B-tree of order 4

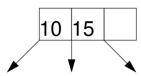


Find in a B-Tree

- If m is large then binary search within a B-tree node.



- If m is small then linear search is fine.

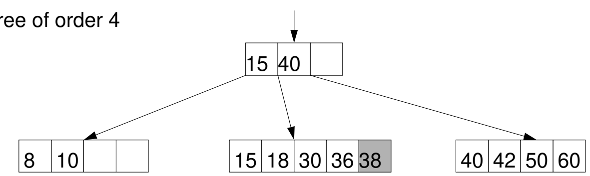


Insertion in a B-tree

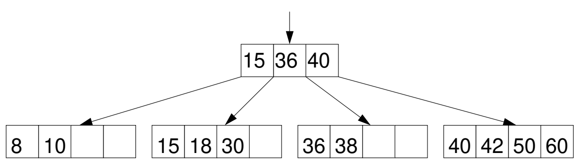
- Begin by inserting in the proper leaf.
- If the leaf ends up with m items, it has overflowed.
- An overflowed leaf node is split into two nodes with $\lceil (m+1)/2 \rceil$ in one and $\lfloor (m+1)/2 \rfloor$ in the other. A new child is added to the parent.
- An overflowed internal node with m+1 children is split into two nodes, one with $\lceil (m+1)/2 \rceil$ children and the other with $\lfloor (m+1)/2 \rfloor$ children. A new child is added to the parent.
- If overflow propagates to the root, then the node is split into two and a new root with two children is formed.

Insertion with Overflow - Leaf Case

B-tree of order 4

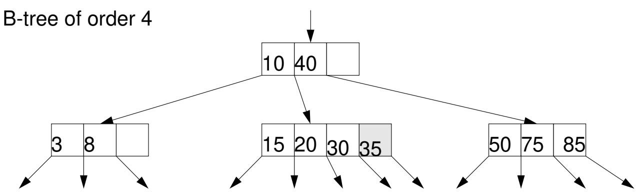


Splitting a node

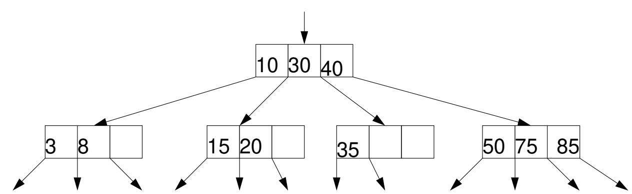


Insertion with Overflow - Internal Node Case

B-tree of order 4



Splitting a node



Example of Insertion

- Insert 1,2,3,4 5,6 7,8, ... into a B-tree of order 3

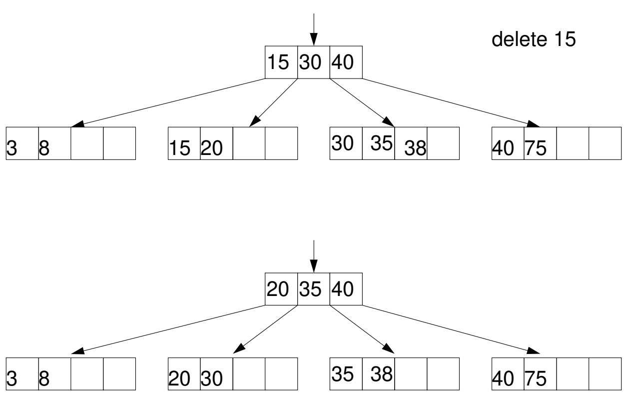
B-Tree Deletion

- In deletion a leaf may underflow, that is, have < $\lceil m/2 \rceil$ items.

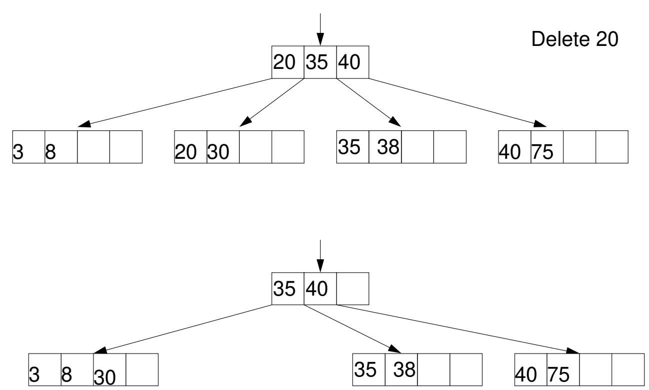
  1. Borrow from a sibling.

  2. If borrowing is impossible then give all to siblings evenly and physically delete the node. Deletion may propagate to the parent.

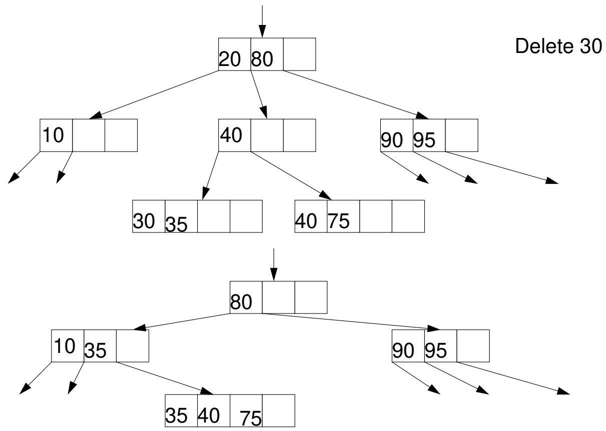
Borrowing



Elimination without Propagation



Deletion with Propagation



Summary of B-Trees

- B-trees of order 3 are often called 2-3 trees and B-trees of order 4 are often called 2-3-4 trees.
- B-trees are used when accessing a new node can be costly, like in large data bases. The root and next level are often kept in main memory.
- B-tree insertion causes splitting which may propagate.
- B-tree deletion causes borrowing (good) or physical deletion which may propagate. Propagation is quite rare if the order of the tree is large.
- In standard B-trees there can be thrashing with repeated insertions followed by deletions. This can be overcome with B* trees.
- If order of the tree is large like 64 or 128, then only a few levels are need to store very large data sets. For example, if m = 128 then a B-tree of height 4 will store at least 30 million items.

Summary of Search Trees

- Plain vanilla binary search trees

  Good on average - simple to understand and use.
- AVL trees - height balanced trees

  Good in worst case - more complicated.
- Splay trees

  Good amortized cost and good locality properties - slightly more complicated than AVL.
- B-trees

  Good in worst case when node access is expensive - most complicated of the algorithms. Heavily used in database applications.

Summary of Dictionary Data Structures

	Find	Insert	Delete	Find-min	List-all	n Finds
Sorted Array	log n	n	n	1	n	n log n
Ordered List	n	n	n	1	n	n^2
Binary Search Tree	n	n	n	n	n	n^2
AVL Tree	log n	log n	log n	log n	n	n log n
Splay Tree[1]	n	n	n	n	n	n log n
B-tree[2]	log n	log n	log n	log n	n	n log n

1. Splay Tree efficiently supports the additional operations of split and merge.

2. B-tree is suitable for out of memory applications while other data structures are for in memory applications.