COMP2321 — DATA STRUCTURES

B-Tree

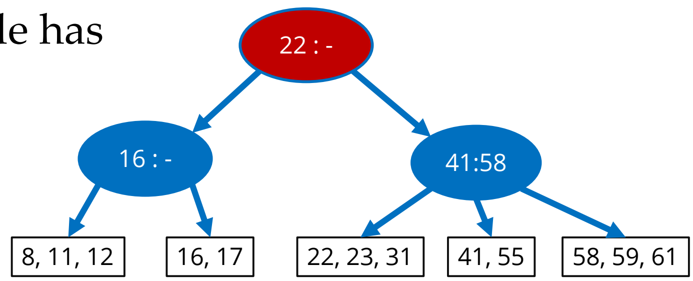
Dr. Radi Jarrar
Department of Computer Science
Birzeit University

B-Tree

- A B-Tree of order m is a tree with the following structural properties.
  - The root is either a leaf or has between 2 & m children.
  - All non-leaf nodes (except the root) have between ⌈m/2⌉ & m child nodes.
  - All leaves are at the same depth and have at most m data items.
  - All data is stored at the leaves.
- B-Tree is NOT a Binary Search Tree.

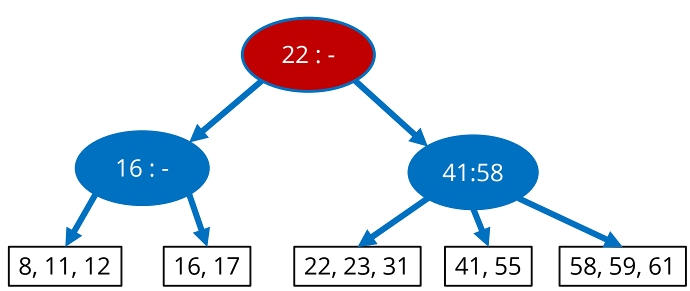
B-Tree (2)

- A B-tree order 4 is more popularly known as a 2-3-4 tree, and a B-tree of order 3 is known as a 2-3 tree.
- The leaves contain the actual data, which can be the keys themselves or pointers to records containing the keys.
- ‘Dash’ means this node has only 2-children.
- Keys in leaves are ordered.



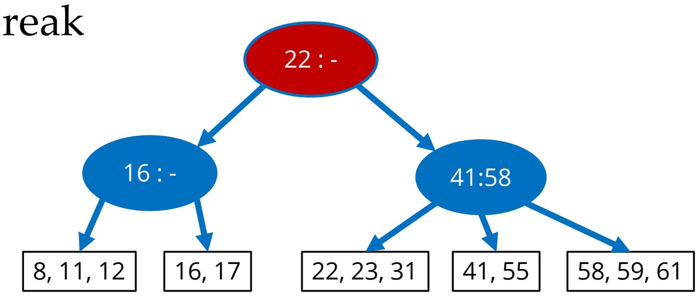
Find Operation

- Start at the root & branch in one of at most 3-directions depending on the relationship of the key we are looking for to the two (possible one) values stored at the node.



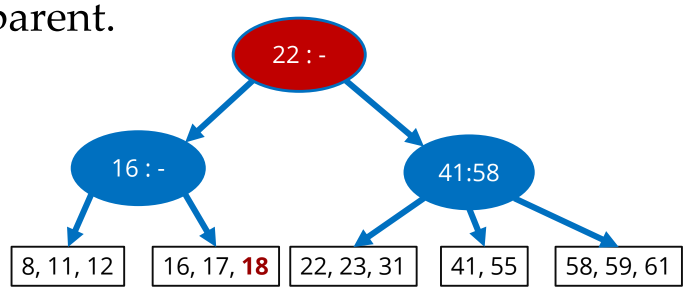
Insert Operation

- Follow the path as in Find. When reaching the leaf node, insert key X at the right position without violating the rules of the tree.
- E.g., Insert 18 won't break the rules of the 2-3 tree priority.



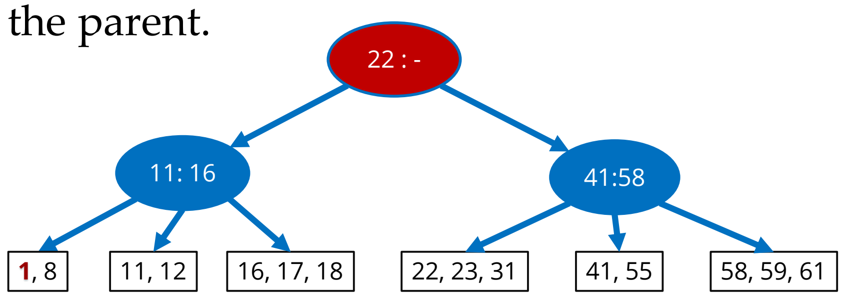
Insert Operation (2)

- E.g., Insert 1.
- We find where 1 belongs. It will violate the rule (a leaf node with 4 keys). Thus, create a 2-nodes of 2-keys & adjust the info at the parent.



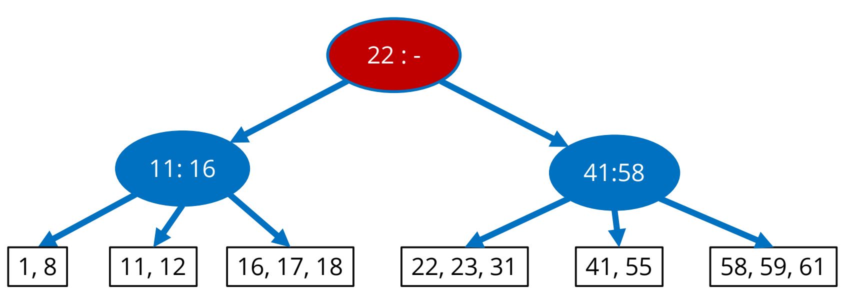
Insert Operation (2)

- E.g., Insert 1.
- We find where 1 belongs. It will violate the rule (a leaf node with 4 keys). Thus, create a 2-nodes of 2-keys & adjust the info at the parent.



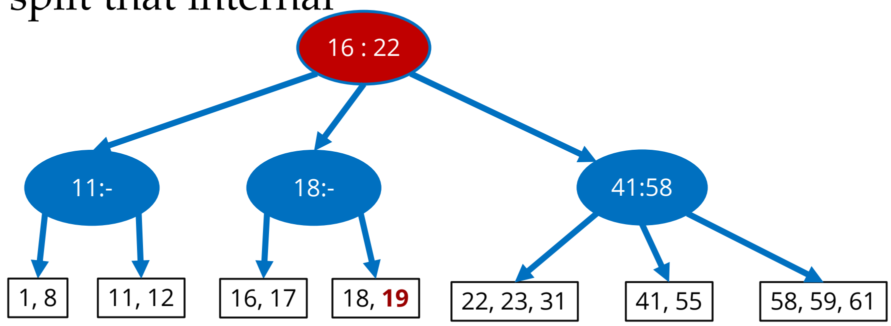
Insert Operation (3)

- E.g., Insert 19.
- Will break the rules as it causes 11:16 to have 4 leaf nodes which isn't allowed. Only 3 leaf nodes per internal nodes are allowed. So split that internal node into 2.



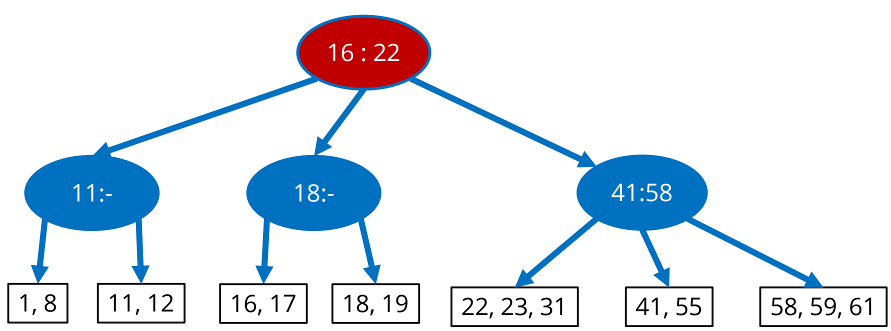
Insert Operation (3)

- E.g., Insert 19.
- Will break the rules as it causes 11:16 to have 4 leaf nodes which isn't allowed. Only 3 leaf nodes per internal nodes are allowed. So split that internal node into 2.



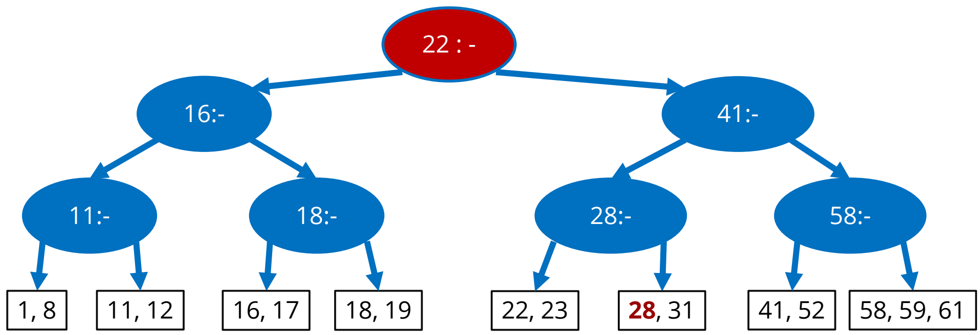
Insert Operation (4)

- E.g., Insert 28.
- Will break the rules as it causes the same to the right part of the tree.



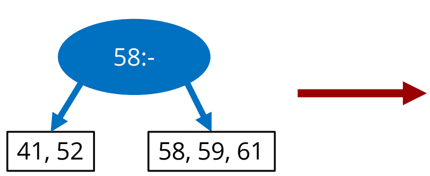
Insert Operation (4)

- E.g., Insert 28.



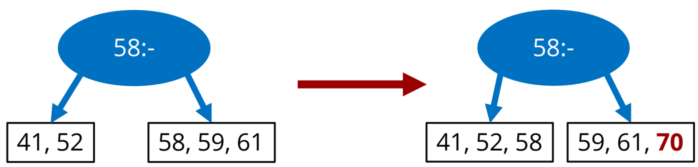
Insert Operation (5)

- Another way to handle the insertion at the leaf node without splitting the internal node is by searching for a sibling with 2 keys.
- E.g., Insert (70)
  - Move 58 to the leaf node on the left & then insert 70 there



Insert Operation (5)

- Another way to handle the insertion at the leaf node without splitting the internal node is by searching for a sibling with 2 keys.
- E.g., Insert (70)
  - Move 58 to the leaf node on the left & then insert 70 there



Delete

- Find & Delete.
- If it was one of the 2-keys, then combine the remaining key with a sibling node. If the sibling has 3-keys, steal one & have both keys to make a new node.

Applications

- Database systems.
  - The tree is kept on a physical disk instead of main memory.
- Accessing disk is typically several order of magnitude slower than any main memory operation. If we use a B-tree of order m, then the number of disk accesses is $O(\log_m n)$.

Applications

- Database systems.
  - The tree is kept on a physical disk instead of main memory.
- Accessing disk is typically several order of magnitude slower than any main memory operation. If we use a B-tree of order m, then the number of disk accesses is $O(\log_m n)$.

Algorithm	**Average**	**Worst case**
Space	$O(n)$	$O(n)$
Search	$O(\log n)$	$O(\log n)$
Insert	$O(\log n)$	$O(\log n)$
Delete	$O(\log n)$	$O(\log n)$