CSC 212: Data Structures and Abstractions

Balanced trees

Prof. Marco Alvarez

Department of Computer Science and Statistics
University of Rhode Island

Spring 2025

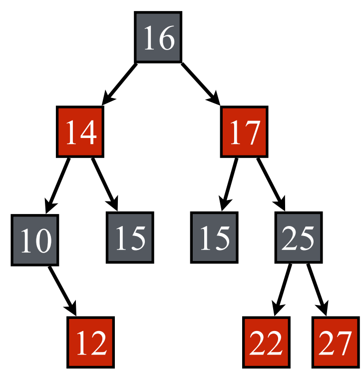
THINK BIG WE DO℠

Balanced search trees

- Balanced search trees are a type of BST that maintain a balanced structure to ensure that the height of the tree is logarithmic in the number of nodes
  - ✓ among the most useful data structures in computer science
  - ✓ many programming languages have built-in support: e.g. Java's `TreeSet` and `TreeMap`, C++'s `std::set` and `std::map`
- Examples of balanced trees:
  - ✓ AVL trees, **Red-Black trees**, B-trees, Splay trees, Treaps, etc.

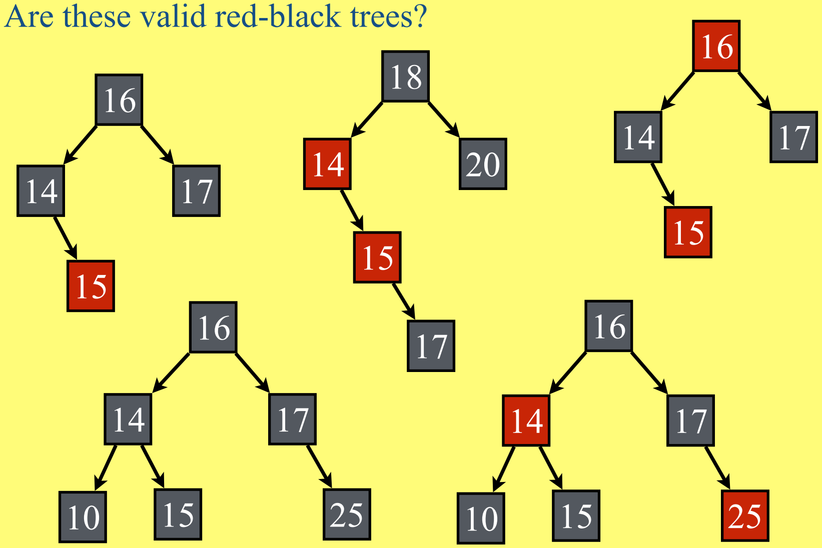
Red-black trees

- Red-black trees maintain a balanced structure by enforcing these properties on the nodes:
  - ✓ each node is colored either red or black
  - ✓ the root node is always black
  - ✓ *null* nodes are considered black (not shown in figures)
  - ✓ red nodes cannot have red children (no two red nodes can be adjacent)
  - ✓ every *root-to-null* path must have the same number of black nodes



Practice

- Are these valid red-black trees?



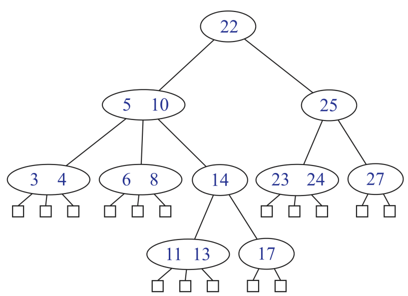
Analysis

- Theorem
  - a red-black tree on n nodes has $h = O(\log n)$
- Maintaining balance
  - after performing an insertion or deletion, the tree may become unbalanced
  - to restore balance, we can locally modify the tree in $O(\log n)$ time to satisfy the red-black properties
  - this is done by performing a sequence of **rotations** and **recoloring** nodes
- Equivalence to **B-trees**
  - red-black trees are equivalent to B-trees of order 4
  - it is easier to understand the complexity analysis and rebalancing operations of red-black trees by thinking of them as B-trees

B-Trees (interlude)

Multi-way search trees

- A multi-way search tree is a generalization of a BST that allows each node to have more than two children
  - the keys in each node are sorted in increasing order
  - the keys in the left subtree of a key k are less than k, and the keys in the right subtree are greater than k

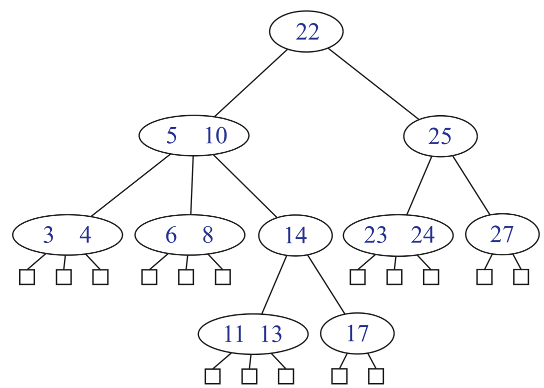


note that null pointers are illustrated as external nodes

Image credit: Data Structures and Algorithms in C++ 2e

Search on a multi-way search tree

- Perform search for 12, 17, 24, and 50 on the following tree
  - note that null pointers are illustrated as external nodes



Assume d denotes the maximum number of children of any node of T, and h denotes the height of T. What is the cost of search?

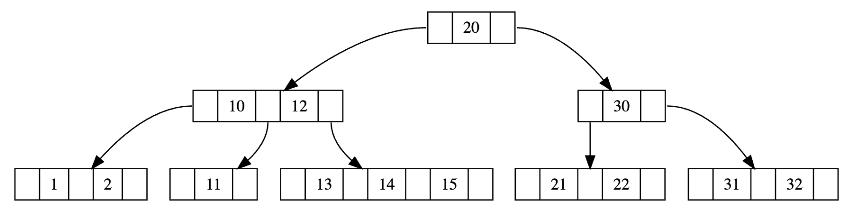
Image credit: Data Structures and Algorithms in C++ 2e

Balanced multi-way search trees

- A balanced multi-way search tree
  - ✓ cap the number of children to a fixed number and keep the leaf nodes at the same depth
  - ✓ add keys only to leaf nodes
    - split the nodes when they become too full sending the middle key up to the parent node (recursively)
  - ✓ the tree is **always balanced** => search, insertion, and deletion operations can be performed in $O(\log n)$ time
- **B-trees** are a specific type of balanced multi-way search trees
  - ✓ on a B-tree of order m, each node can have at most m children and $m - 1$ keys
    - there are differences in terminology including different "order" definitions
  - ✓ used in databases and file systems to store large amounts of data (common orders: 1024, 2048, 4096, …)

2-3-4 tree

- A 2-3-4 tree (a.k.a. 2-4 tree) is a B-tree of order 4
  - ✓ each node can have 2, 3, or 4 children
  - ✓ i.e. all nodes must have at least 1 key and at most 3 keys, except the root node that can have 0 keys when the tree is empty



Practice

- What is the max height h of a 2-3-4 tree with n nodes?
  - ✓ the greatest h such that the tree can still store only n keys
    - to maximize the height, want to minimize the number of keys per node
  - ✓ this is an instance of a worst-case
  - ✓ draw the tree and express h in terms of n

Practice

- What is the cost of search and insert on a 2-3-4 tree?
  - ✓ worst-case scenario

Red-black trees