B Trees

Jonathan Windle

University of East Anglia

J.Windle@uea.ac.uk

June 3, 2017

Overview I

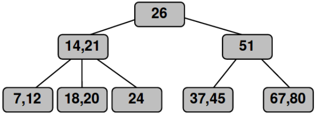
1. Intro

2. Inserting
   - Inserting - Case 1
   - Inserting - Case 2
   - Inserting - Case 3
   - Inserting - Case 4

3. B-Trees
   - Comparison

Intro

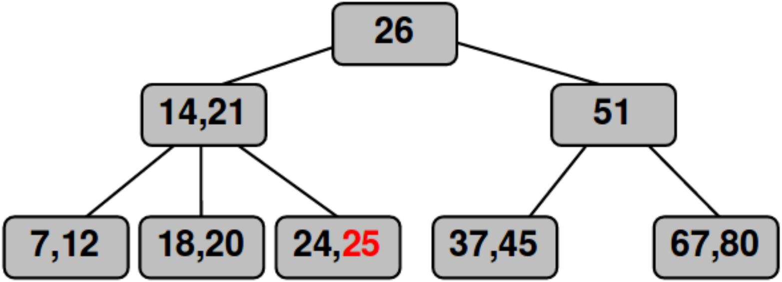
- Let K be a totally ordered set.
- A 2-3 tree, t is a tree on $K \cup (K \times K)$ possessing the following properties:
  - Each non-leaf node has 2 or 3 children
  - If p is a node with 2 children then it contains one key sych that:
    - All keys in the left subtree of p $<$ $key(p)$ $<$ all keys in the right subtree of p.
  - If p is a node with 3 children, then it contains two keys, $ltkey$ and $rtkey$, such that:
    - All keys in the left subtree of p:
    - $<$ $ltkey$
    - $<$ all keys in the middle subtree of p
    - $<$ $rtkey$
    - $<$ all keys in right subtree of p.
  - All leaf nodes are at the same level.



Inserting - Case 1

Insertion in a leaf node containing a single key:

E.g. Insert 25:

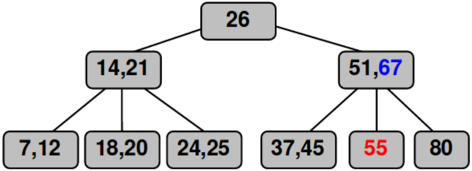


Inserting - Case 2

Node whose children should contain new key has only two children, both with two keys:

E.g. Insert 55:

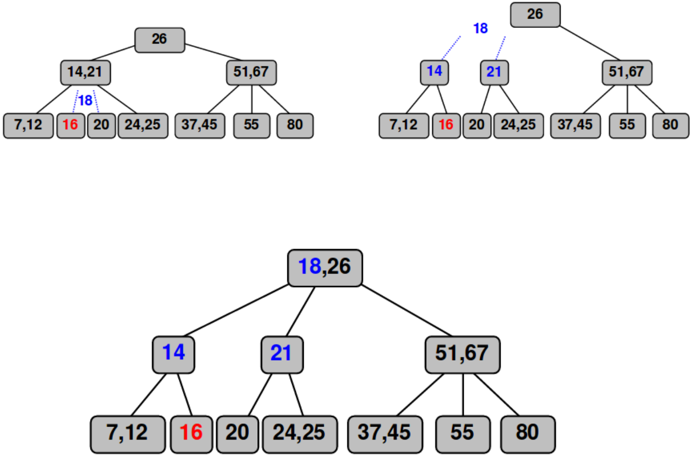
- Go to right child of node containing 51; inserting 55 in this node would give 55,67,80;
- Make the middle key of these 3 keys the right key of the parent;
- Split the other two values into two nodes.



Inserting - Case 3

Node whose children should contain new key already has 3 children, and new key cannot be inserted in a leaf containing just 1 key.

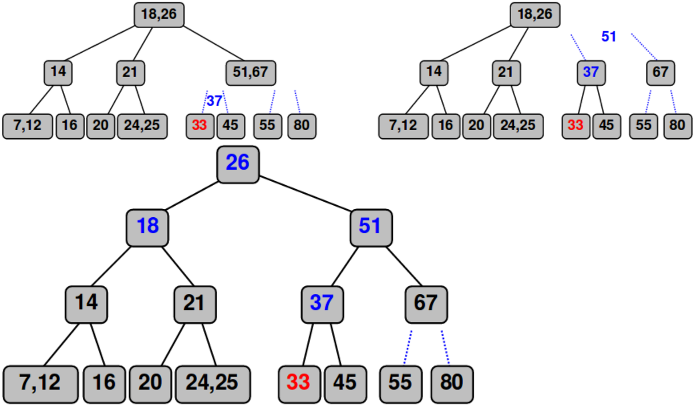
E.g. insert 16:



Inserting - Case 4

Splitting the root

E.g. Insert 16 in



Intro

- A 2-3 tree is a B-Tree of order 3.
- More generally,aB-tree of order m is an m-way search tree such that:
  - If the root is not a leaf nodethen it has between 2 and m children.
  - All non-leaf nodes except the root have between $[m/2]$ and m children.
  - All leaf nodes are at the same level.
- The path length when searching in a B-tree is bounded above by: $log_{[m/2]}(\frac{n+1}{2})$

Comparison of Height-Balanced Binary Search Trees and B-Trees

- The expected path length in a height-blances BST containing n keys is:
  $\boxed{[log_2(n+1)] - 1}$
- Height of a B-tree of order m containing n keys is:
  $\boxed{\leq log_{[m/2]}(\frac{n+1}{2})}$.
- It is easier to code BST algorithms
- B-Trees waste space if used for RAM-based dictionaries.
- B-Trees on disk units have $m = 256$ or more:
  shallow tree $\Rightarrow$ *fewer disk accesses*

The End