Trees 4

The B-Tree

Section 4.7
Problem with Big 'O' notation
Problem with Big `O` notation

- Big `O` assumes that all operations take equal time
- Suppose all data does not fit in memory
- Then some part of data may be stored on hard disk.

- CPU speed is in millions of instructions per second
  - 3GHz machines common now
    - Equals roughly 3000 million instructions per seconds
- Typical disk speeds about 7,200 RPM
  - Roughly 120 disk accesses per second

- So accessing disk is incredibly expensive.
- So we may be willing to do more computation to organize our data better and make fewer disk accesses.
Problem with binary trees

There is no guarantee that binary trees will be balanced
Problem with binary trees

- There is no guarantee that binary trees will be balanced

If stored on disk, we have potentially $O(N)$ disk operations
**M-ary Trees**

- Allows up to $M$ children for each node
  - Instead of max. 2 for binary trees
- A complete $M$-ary tree of $N$ nodes has a depth of $\log_M N$
- Example of complete 5-ary tree of 31 nodes
M-ary search tree

- Similar to binary search tree, except that
  - Each node has \((M-1)\) keys to decide which of the \(M\) branches to follow.

- Larger \(M\) \(\Rightarrow\) smaller tree depth

- But how to make \(M\)-ary tree balanced?
B-Tree (or Balanced Trees)

- B-Tree is an M-ary tree with some restrictions:
  - Data items are stored at the leaves
  - Non-leaf (internal) nodes store up to \( M-1 \) keys
    - Key \( i \) represents the smallest key in subtree \((i+1)\)
  - Leaves also store keys, where key \( i \) represent smallest key in the \( i+1 \) data block
**B-tree (continued)**

- Root is a leaf or has between 2 to M children.
- Internal nodes have between \([M/2]\) and M children.
- All leaves are at the same depth and have between \([L/2]\) and L children, for some L.
**B-Tree: Inserting a value**

- After insertion of 57:
  - Simply rearrange data in the correct leaf
- What about 55?
B-Tree: Inserting a value (contd)

- Inserting 55:
  - Splits a full leaf into two leaves
  - What about 40?

[Diagram of a B-tree with nodes 8, 18, 26, 35, 48, 51, 54, 57, 41, 66, 87, 72, 78, 83, 92, 97, 87, 89, 93, 95, 98, 99, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 31, 32, 35, 36, 37, 38, 39, 41, 42, 44, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 66, 68, 69, 70, 72, 73, 74, 76, 78, 79, 81, 83, 84, 85, 87, 89, 92, 93, 95, 97, 98, 99]
B-Tree: Inserting a value (contd)

- Insertion of 40:
  - Splits a leaf into two leaves
  - Also splits the parent node
- What about deleting 99?
$B$-tree: Deletion of a value

- Deletion of 99:
  - Causes combination of two leaves into one
  - Can recursively combine non-leaves
Questions

- How does the height of the tree grow?
- How does the height of the tree reduce?
- How else can we handle insertion, without splitting a node?