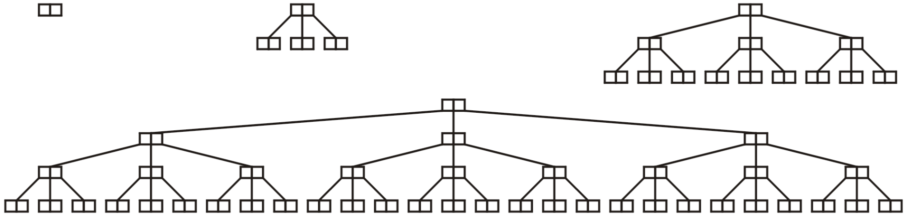
M-way Trees and B-Trees



Carlos Moreno
cmoreno@uwaterloo.ca
EIT-4103

https://ece.uwaterloo.ca/~cmoreno/ece250

These slides, the course material, and course web site are based on work by Douglas W. Harder

M-way Trees and B-Trees

Standard reminder to set phones to silent/vibrate mode, please!

M-way Trees and B-Trees

- Once upon a time... in a course that we all like to call ECE-250...
  - We talked about trees (hierarchical data structures)
  - In particular, binary search trees (non-hierarchical; just take advantage of the tree structure)
    - Including balanced binary search trees (we talked about AVL trees)

M-way Trees and B-Trees

- Today, we'll discuss:
  - M-Way trees
    - In-order traversal of an M-way tree
  - B-Trees
    - Only basic concepts and rationale
      - The details will be optional material — meaning that it will be the topic for bonus marks questions on the assignments and on the final.

M-way Trees and B-Trees

- M-Way Trees
  - Not to be confused with N-ary trees, which are trees where the nodes have a fixed number of children (binary trees being a particular case, with N=2)
  - M-Way Trees are trees where the nodes store multiple values.
    - They are search trees (just not binary)
  - They also have multiple children (a fixed number of them, unlike with general trees)

M-way Trees and B-Trees

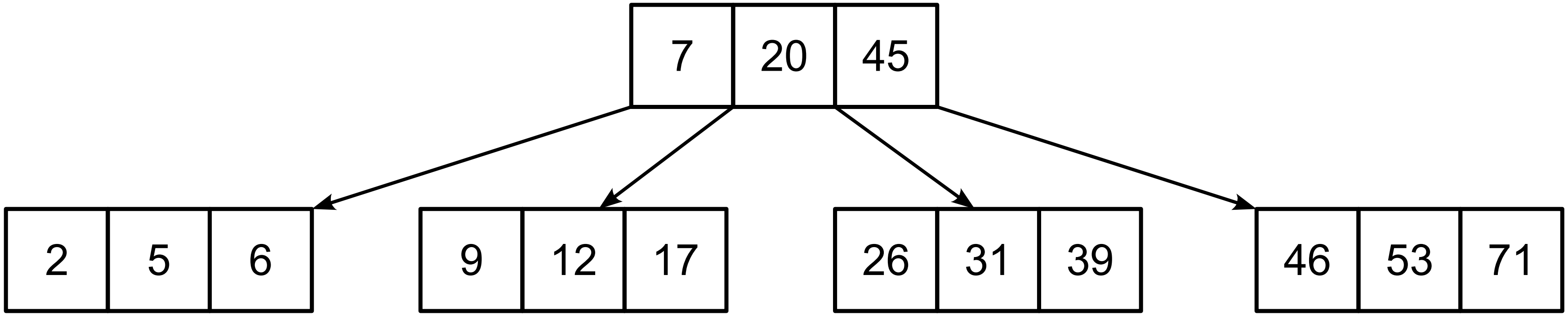
- M-Way Trees
  - In particular, a node in an M-Way tree has:
    - M-1 data values
    - M children
  - Notice: A Binary search tree is a particular case of an M-Way tree (M = ?)

M-way Trees and B-Trees

- M-Way Trees
  - The constraint that makes them search trees is that for each node in the tree, the values in the noide's sub-trees are related to the values in the node:
    - If the values are $\{v_1, v_2, \cdots, v_{M-2}\}$ and the children (sub-trees) are $\{T_1, T_2, \cdots, T_{M-1}\}$, then:
      - Every value in the tree T_k $(1 < k < M-1)$ is between v_{k-1} and v_k
      - For T_1, every value in the tree is less than v_1
      - For T_{M-1}, every value in the tree is greater than v_{M-2}

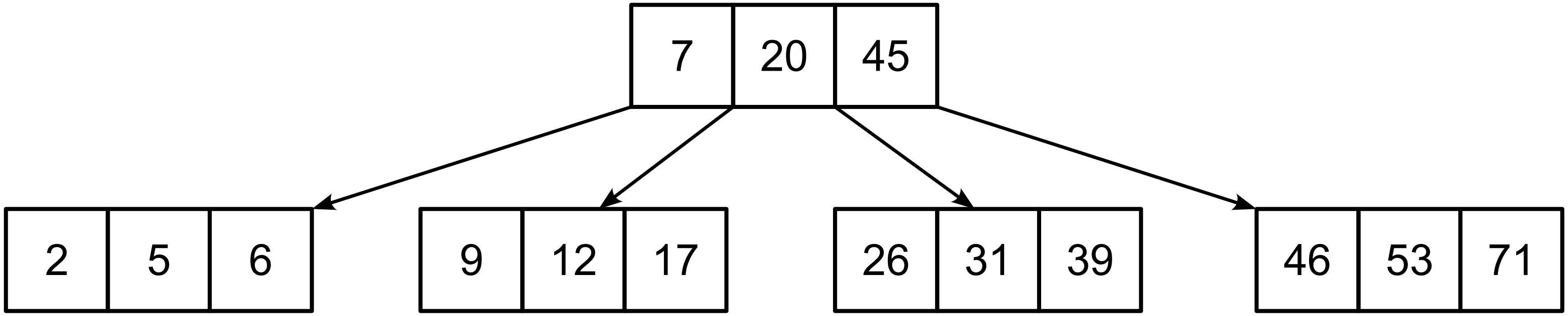
M-way Trees and B-Trees

- M-Way Trees
  - Example for a 4-Way tree:



M-way Trees and B-Trees

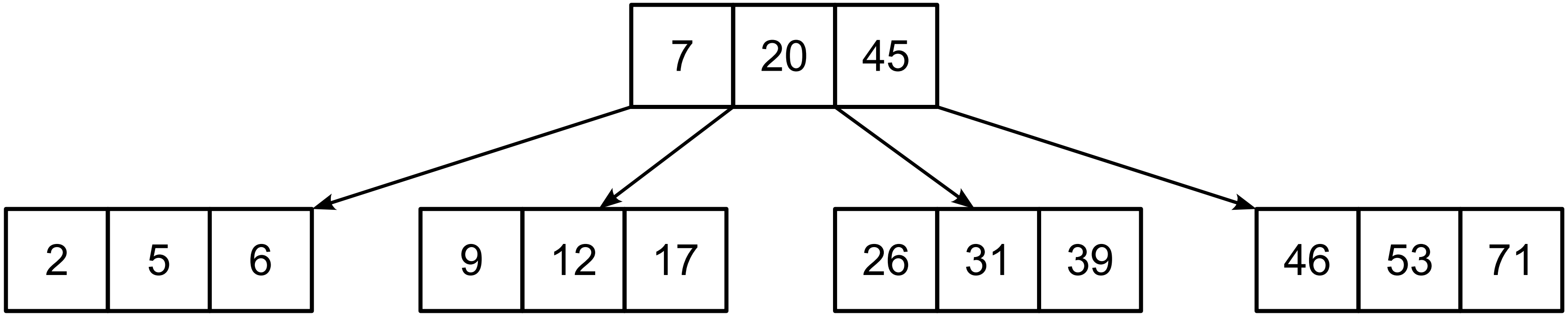
- M-Way Trees
  - Example for a 4-Way tree:



  - For in-order traversal, we extend the idea from binary trees: first, visit child T_k, then process value v_k, then visit child T_{k+1}.

M-way Trees and B-Trees

- M-Way Trees
  - Example for a 4-Way tree:



  - BTW... Do you notice anything interesting about this tree? (Hint: it contains 15 values)

M-way Trees and B-Trees

- M-Way Trees
    - The main point with M-Way trees is to reduce the height !
    - Of course, in terms of Landau symbols, there's no improvement — we're still $\Omega(\log n)$, with $\Theta(\log n)$ being reached if the tree is balanced.
    - But we have an improvement with respect to binary trees (a reduction of the height) by a constant factor of .... (you guys tell me?)

M-way Trees and B-Trees

- M-Way Trees
  - The main point with M-Way trees is to reduce the height !
  - Of course, in terms of Landau symbols, there's no improvement — we're still $\Omega(\log n)$, with $\Theta(\log n)$ being reached if the tree is balanced.
  - But we have an improvement with respect to binary trees (a reduction of the height) by a constant factor of .... Right, lg(M):

M-way Trees and B-Trees

- M-Way Trees
  - The number of nodes for an M-way tree of height h grows with M^h — thus, the height goes with $\log_M n$, and we have:

$$\begin{aligned} n = M^{\log_M n} \;\Rightarrow\; \log_2 n &= \log_2 M^{\log_M n} \\ &= \log_2(M) \cdot \log_M n \end{aligned}$$

M-way Trees and B-Trees

- M-Way Trees
  - Advantage: More efficient (though only by a constant speedup factor).
  - Disadvantage: More complex structure

M-way Trees and B-Trees

- M-Way Trees
  - Advantage: More efficient (though only by a constant speedup factor).
  - Disadvantage: More complex structure
  - When is this extra complexity justified?

M-way Trees and B-Trees

- M-Way Trees
  - Advantage: More efficient (though only by a constant speedup factor).
  - Disadvantage: More complex structure
  - When is this extra complexity justified?
    - Generally speaking, when the extra efficiency is necessary.

M-way Trees and B-Trees

- M-Way Trees
  - Advantage: More efficient (though only by a constant speedup factor).
  - Disadvantage: More complex structure
  - When is this extra complexity justified?
    - Generally speaking, when the extra efficiency is necessary.
    - One particular situation is when moving from one node to another is particularly expensive.

M-way Trees and B-Trees

- M-Way Trees
  - Advantage: More efficient (though only by a constant speedup factor).
  - Disadvantage: More complex structure
  - When is this extra complexity justified?
    - Generally speaking, when the extra efficiency is necessary.
    - One particular situation is when moving from one node to another is particularly expensive.
      - What would be an example where that would be the case?

M-way Trees and B-Trees

- M-Way Trees
  - Advantage: More efficient (though only by a constant speedup factor).
  - Disadvantage: More complex structure
  - When is this extra complexity justified?
    - Generally speaking, when the extra efficiency is necessary.
    - One particular situation is when moving from one node to another is particularly expensive.
      - What would be an example where that would be the case?
        Hint: What in a computer can be particularly slow?

M-way Trees and B-Trees

- B-Trees use this approach (among several other aspects) to store data on permanent storage (hard disk).
  - Typically used for database systems.

M-way Trees and B-Trees

- B-Trees use this approach (among several other aspects) to store data on permanent storage (hard disk).
  - Typically used for database systems.

M-way Trees and B-Trees

- B-Trees
  - Two key aspects that affect the design of this data structure:

M-way Trees and B-Trees

- B-Trees
  - Two key aspects that affect the design of this data structure:
    - Hard disks are slow

M-way Trees and B-Trees

- B-Trees
  - Two key aspects that affect the design of this data structure:
    - ~~Hard disks are slow~~ — scratch that...
    - Hard disks are slooooooooowwww....
      - They are *mechanical* beasts living in an electronic circuits world!

M-way Trees and B-Trees

- B-Trees
  - Two key aspects that affect the design of this data structure:
    - ~~Hard disks are slow~~ — scratch that...
    - Hard disks are slooooooooowwww....
      - They are *mechanical* beasts living in an electronic circuits world!
    - The other aspect being: access is by blocks (typical unit is 4kbytes — reading 1 byte or reading 4kbytes takes essentially the same amount of time)

M-way Trees and B-Trees

- About hard disks speed...
  - Two key parameters:
    - Access time
    - Transfer speed

  - Transfer speed is not that bad — once we start reading data, typically things move rather fast (hundreds of megabytes per second)
  - However, access time results from the head having to move (as in, *mechanical* movement) to the right place to read the data

M-way Trees and B-Trees

- About hard disks speed...
  - Two key parameters:
    - Access time
    - Transfer speed
  - Transfer speed is not that bad — once we start reading data, typically things move rather fast (hundreds of megabytes per second)
  - However, access time results from the head having to move (as in, *mechanical* movement) to the right place to read the data — typical figures in the order of 10 ms!! (that's an incredibly slow 100 Hz !!!!!)

M-way Trees and B-Trees

- B-Trees take these aspects into consideration by:
  - Storing internal nodes as 512-Way trees (why 512? A disk block is 4k, and typical key+pointer combinations are 8 bytes per item)
    - Nice side-effect: we load entire nodes into memory with a single disk access.

M-way Trees and B-Trees

- B-Trees take these aspects into consideration by:
  - Storing internal nodes as 512-Way trees (why 512? A disk block is 4k, and typical key+pointer combinations are 8 bytes per item)
    - Nice side-effect: we load entire nodes into memory with a single disk access.
  - But more importantly: we dramatically reduce the amount of disk accesses (notice that descending to each child node requires a new disk access — i.e., another 10 ms!!)
    - Why do we reduce the amount of accesses?

M-way Trees and B-Trees

- B-Trees take these aspects into consideration by:
  - Amount of disk accesses goes with the height of the tree (we're following the nodes, until reaching the appropriate leaf node, which is where the data is), and there are $h = \log_M(n)$ of them.

M-way Trees and B-Trees

- B-Trees take these aspects into consideration by:
  - Amount of disk accesses goes with the height of the tree (we're following the nodes, until reaching the appropriate leaf node, which is where the data is), and there are $h = \log_M(n)$ of them.
    - $M = 512 = 2^9$, meaning 9 times fewer disk accesses than with a binary search tree!

M-way Trees and B-Trees

- B-Trees are in a sense a “hybrid” structure: internal nodes are M-Way trees, and leaf nodes are just a block of records (essentially, a plain node containing an array structure)

M-way Trees and B-Trees

- B-Trees are in a sense a “hybrid” structure: internal nodes are M-Way trees, and leaf nodes are just a block of records (essentially, a plain node containing an array structure)
    - We make leaf nodes also the size of a block (4k), to make the most out of each disk access.

M-way Trees and B-Trees

- Additionally, the balancing is done in a way that we ensure that all leaf nodes are at the same depth.
  - Access time is uniform for all data in the database.

M-way Trees and B-Trees

- B-Trees take these aspects into consideration by:
  - Additionally, the balancing is done in a way that we ensure that all leaf nodes are at the same depth.
    - Access time is uniform for all data in the database.
  - We observe that we need to do array searches and various operations in memory — the point being, that time is *negligible* compared to disk access time; so really, the issues of asymptotic analysis when talking about data structure for disk storage become a secondary issue!

Summary

- During today's lesson:
  - Looked into the notion of M-Way trees
  - Discussed advantages, disadvantages, and when they are justified
  - Discussed the basic notions and rationale for B-Trees.
    - Search structure for disk storage of data (e.g., database systems)
    - Slow access time + block-based access:
      - Minimize number of accesses by widening the tree, thus reducing the height
      - Nodes are just wide enough that they fit within one 4k disk block