DS286 | 2016-10-21

L19: B Trees

Yogesh Simmhan
simmhan@cds.iisc.ac.in

Slides courtesy:
www.cs.nott.ac.uk/~psznza/G52ADS/btrees2.pdf
http://www.cs.carleton.edu/faculty/jgoldfea/cs201/spring11/inclass/Tree/BTreefinalNew.pdf

Searching External Storage

- Main memory (RAM) is fast, but has limited capacity
- Different considerations for in-memory vs. on-disk data structures for search
- Problem: Database too big to fit memory
  - Disk reads are slow
- Example: 1,000,000 records on disk
- Binary search might take 20 disk reads
  - log2(1M) ~= 20

Searching External Storage

- But disks are accessed “block at a time” by OS
- Blocks are typically 1KiB–4KiB in size
  - Can span multiple “sectors” on HDD
  - Access time per block
  - ~12ms for HDD
  - <1ms for SSD
- Say 1KiB block, 100B per record
  - 10,000 blocks for 1M records

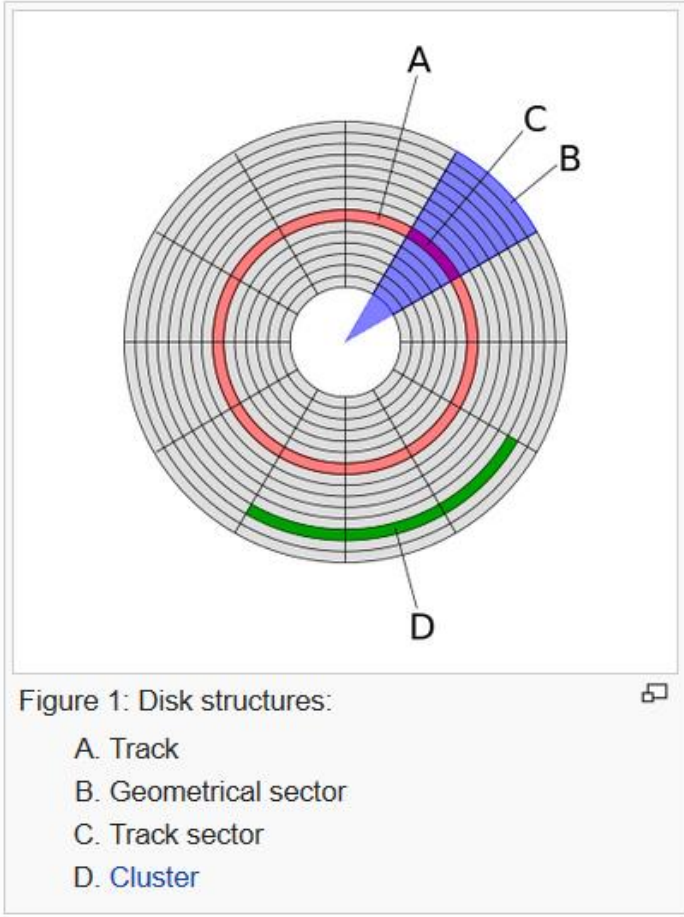
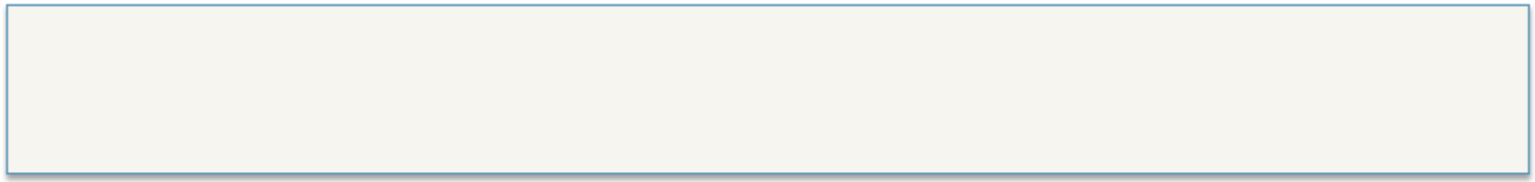


Figure 1: Disk structures:

A. Track
B. Geometrical sector
C. Track sector
D. Cluster

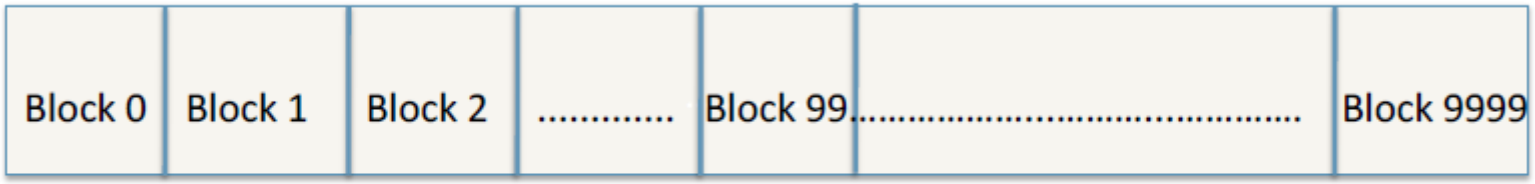
https://en.wikipedia.org/wiki/Disk_sector

Storing & Searching on External Storage



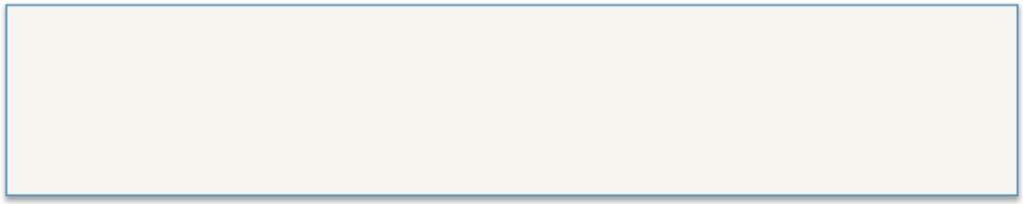
1,000,000 Records

Storing & Searching on External Storage

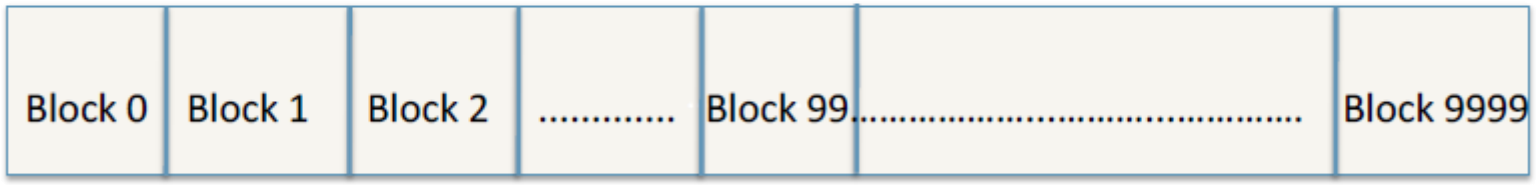


1,000,000 Records

Storing & Searching on External Storage

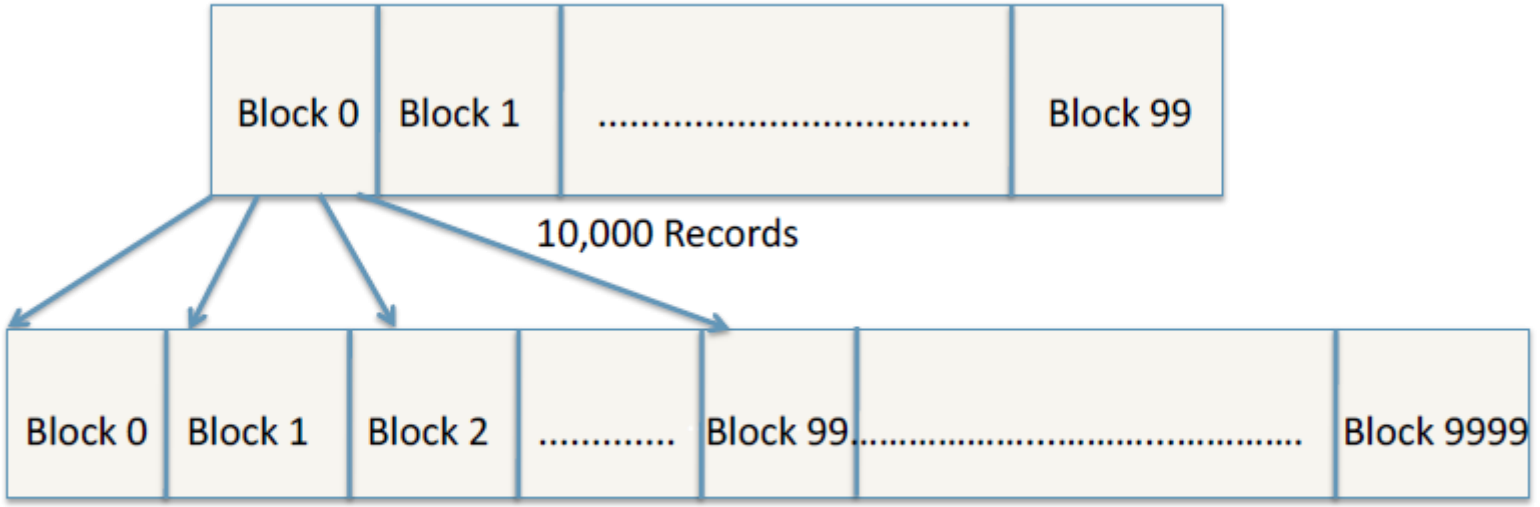


10,000 Records



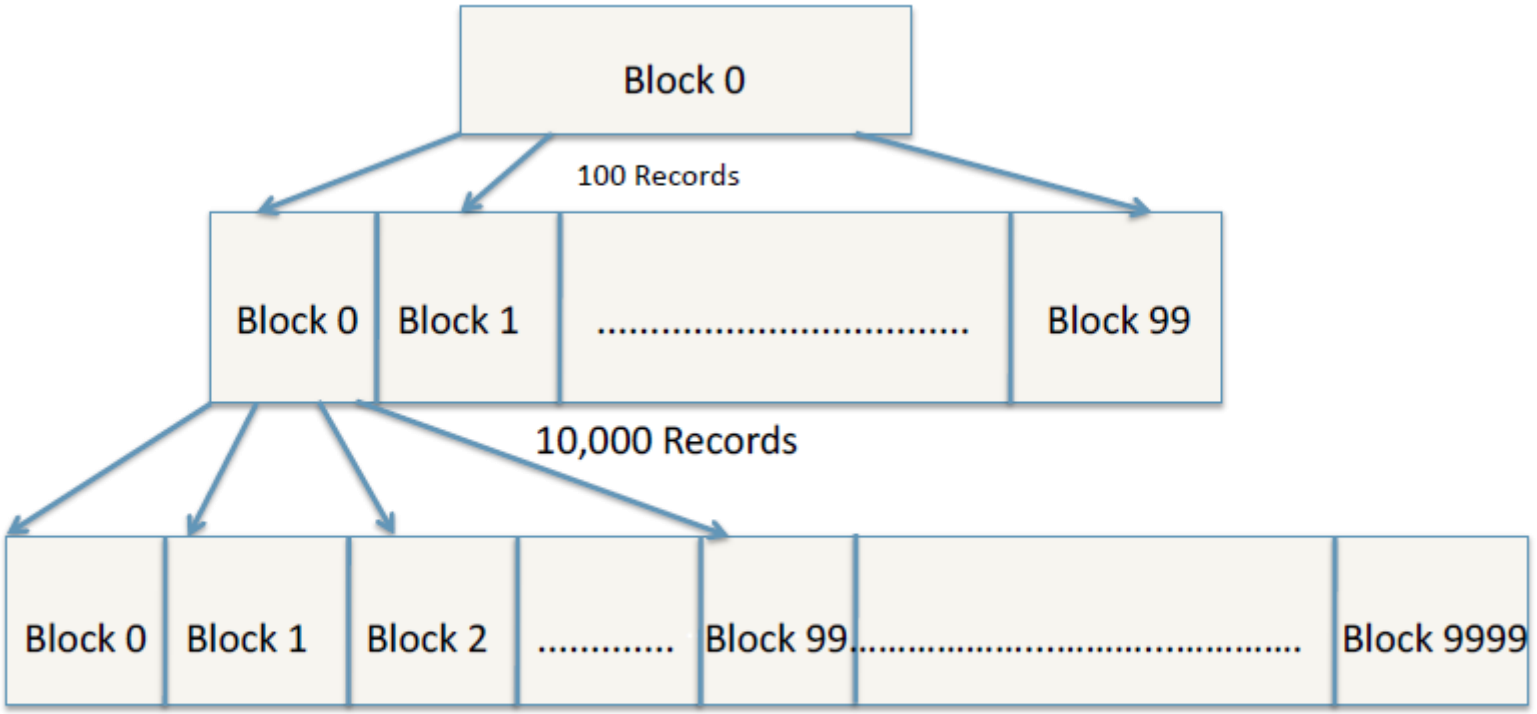
1,000,000 Records

Storing & Searching on External Storage



1,000,000 Records

Storing & Searching on External Storage



B-Trees

- Data structures for external memory, not main memory
  - Goal is to reduce number of block accesses, not number of comparisons
- B-Trees
  - Proposed by R. Bayer and E. M. McCreigh in 1972.
  - “Bayer”, “Balanced”, Bushy”, “Boeing” trees?
  - Different from **binary** trees
- NOTE:
  - In-memory data structure will be better than on-disk
  - milliseconds vs. nano seconds
  - So *in-memory binary tree* will be better than *on-disk B Tree*
  - But on-disk B Tree better than on-disk binary tree

B-Trees

- Definition: A B-Tree of order m is an m-way tree with
  - All leaf nodes are at the same level.
  - All non-leaf nodes (except the root) have at most m and at least m/2 children.
  - The number of keys is one less than the number of children for non-leaf nodes and at most m-1 and at least m/2 for leaf nodes.
  - The root may have as few as 2 children unless the tree is the root alone.

B Tree of order 5

- A B- Tree of order 5 is an 5-way tree such that
  - All leaf nodes are at the same level.
  - All non--leaf nodes (except the root) have at most 5 and at least 2 children
  - The number of keys is one less than the number of children for non-leaf nodes and at most 4 and at least 2 for leaf nodes
  - The root may have as few as 2 children unless the tree is the root alone.

Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P

Creating a B-tree of order 5

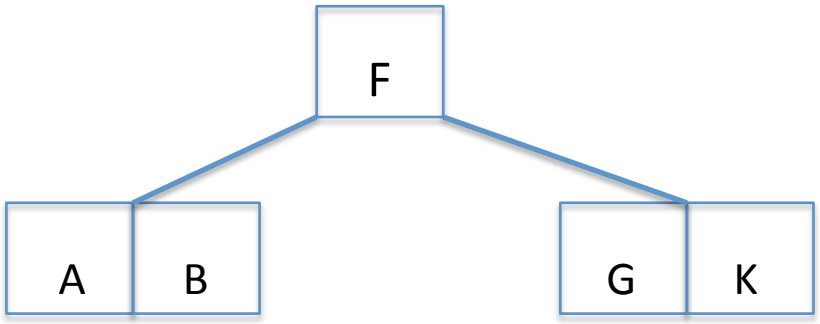
A G F B K D H M J E S I R X C L N T U P

Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P

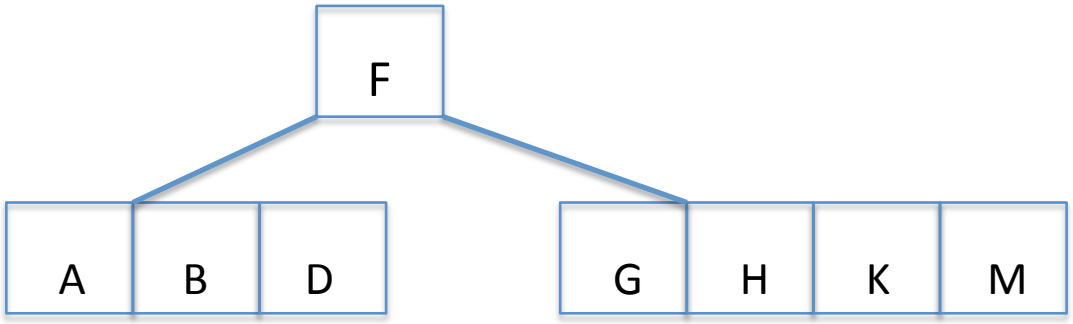
Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P



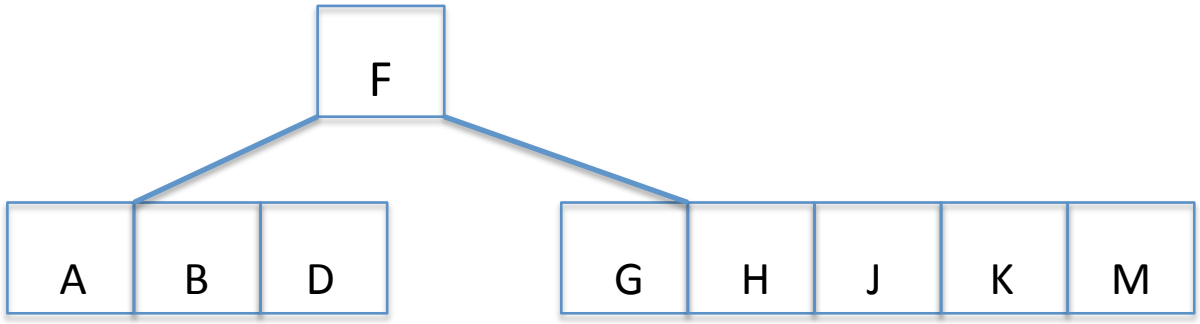
Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P



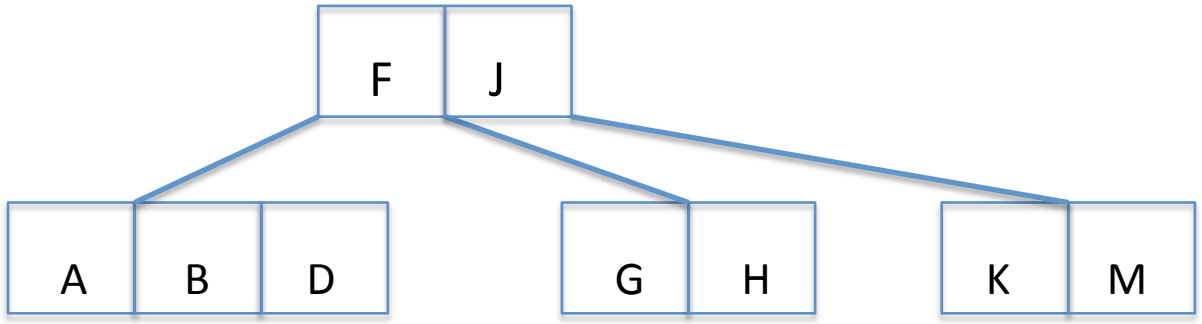
Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P



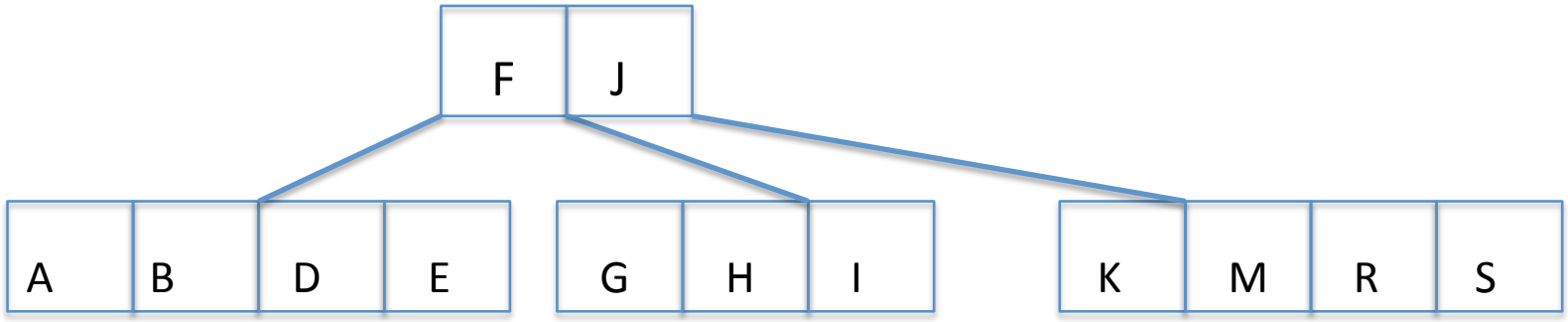
Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P



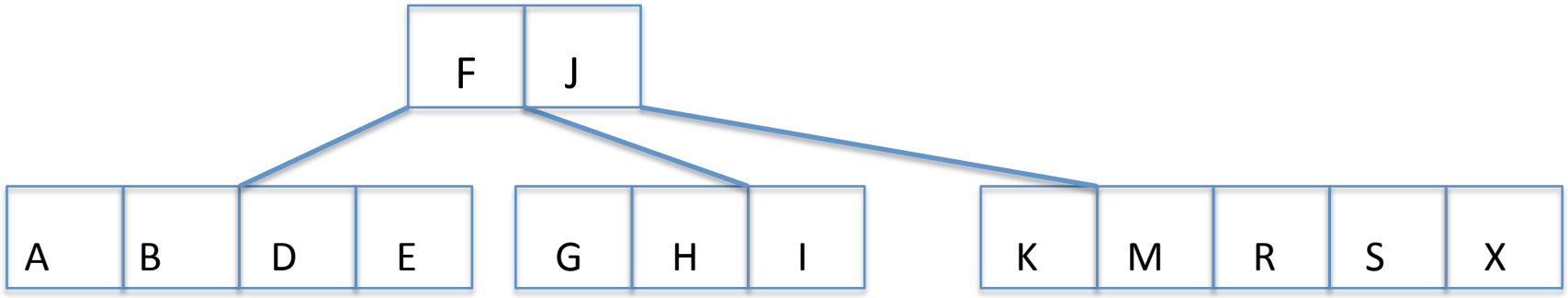
Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P



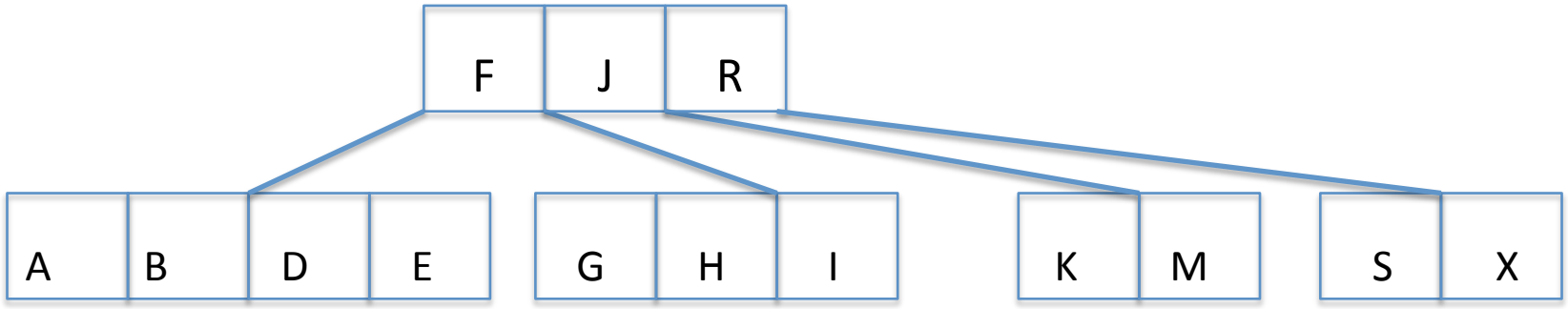
Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P



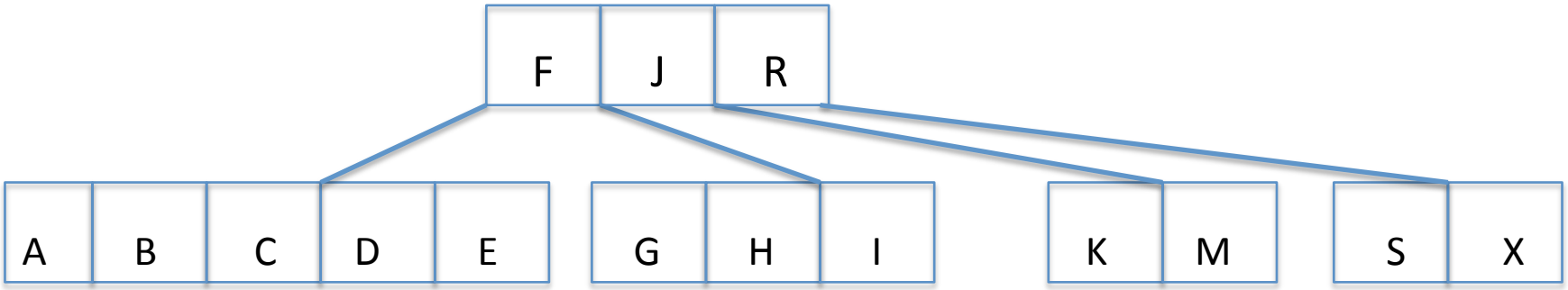
Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P



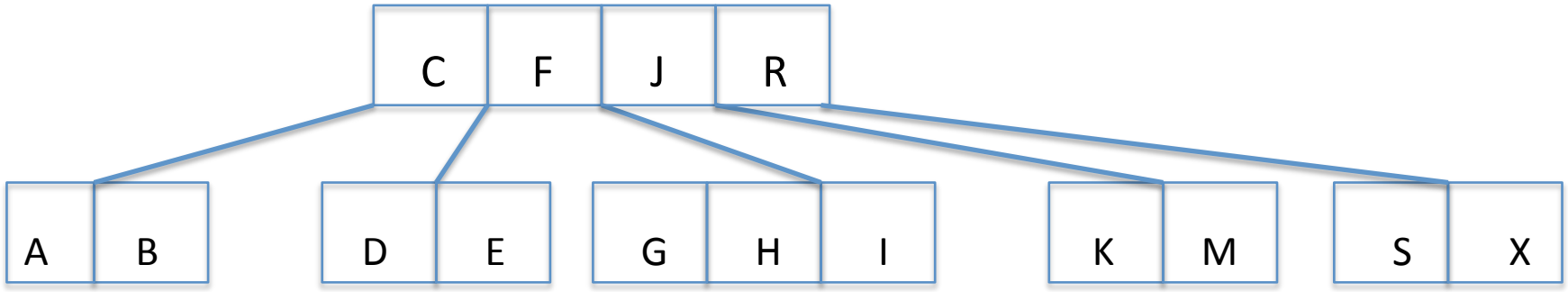
Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P



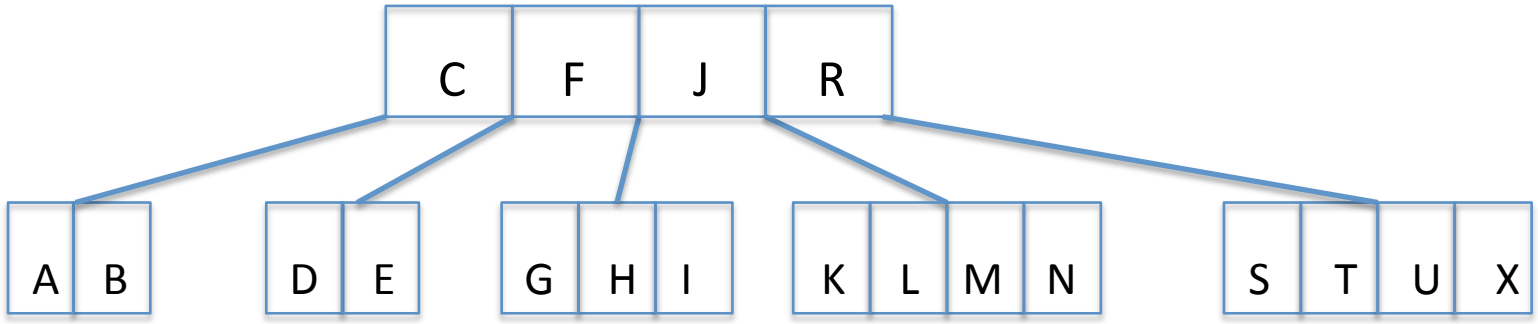
Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P



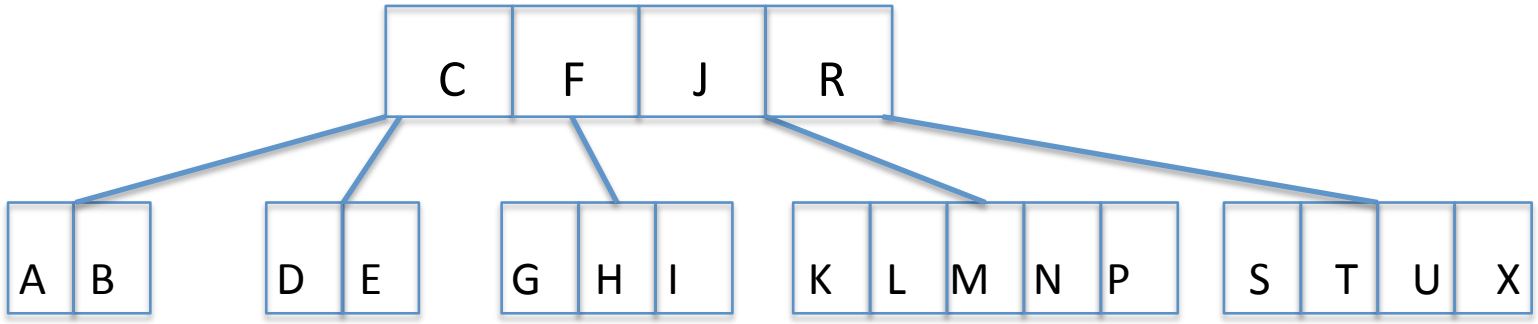
Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P



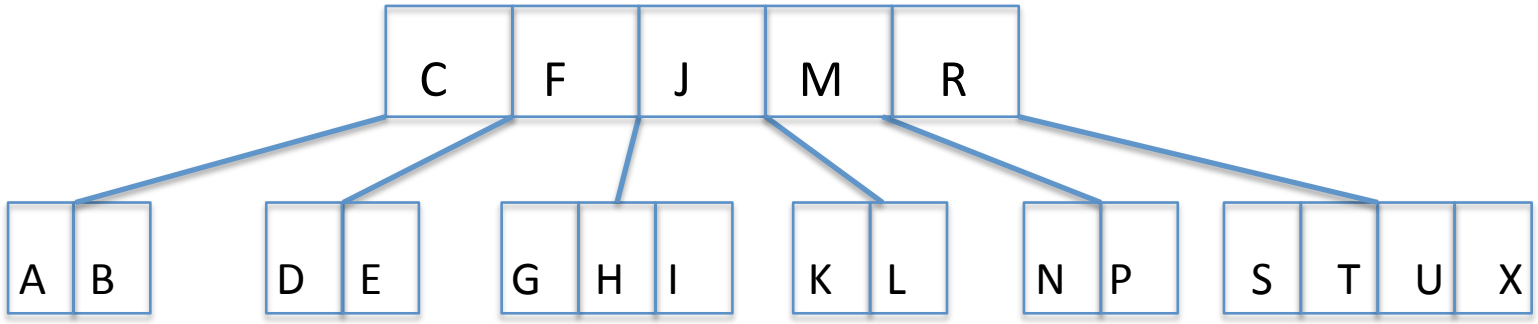
Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P



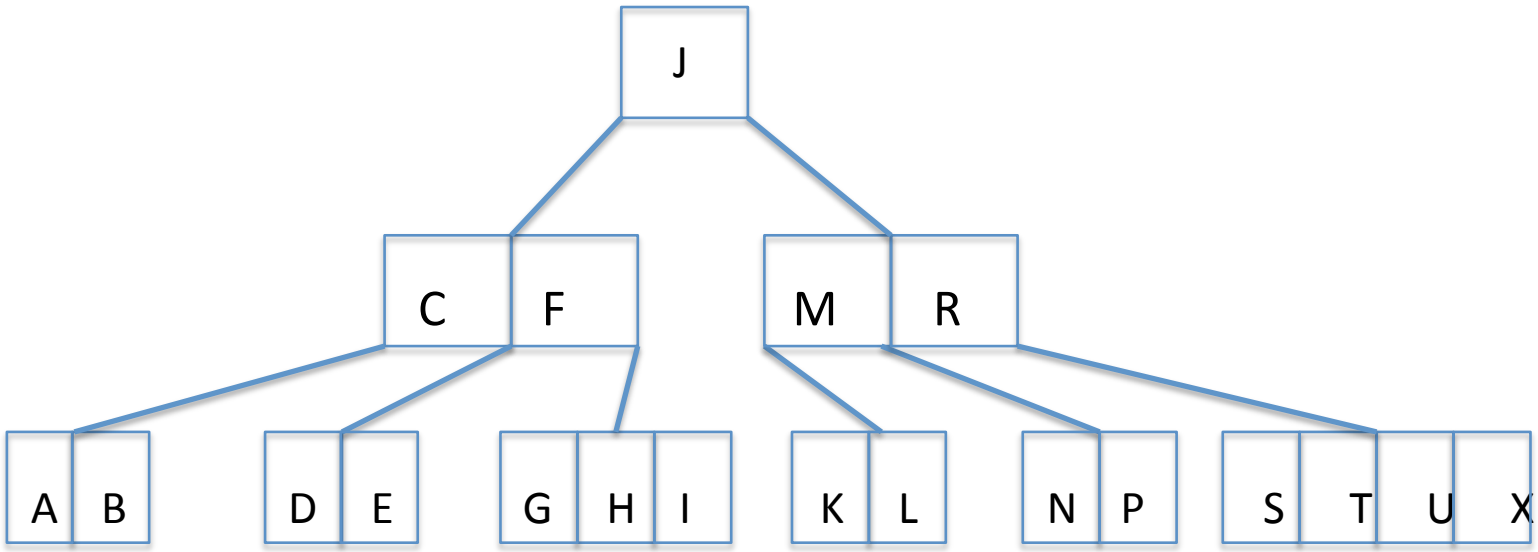
Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P



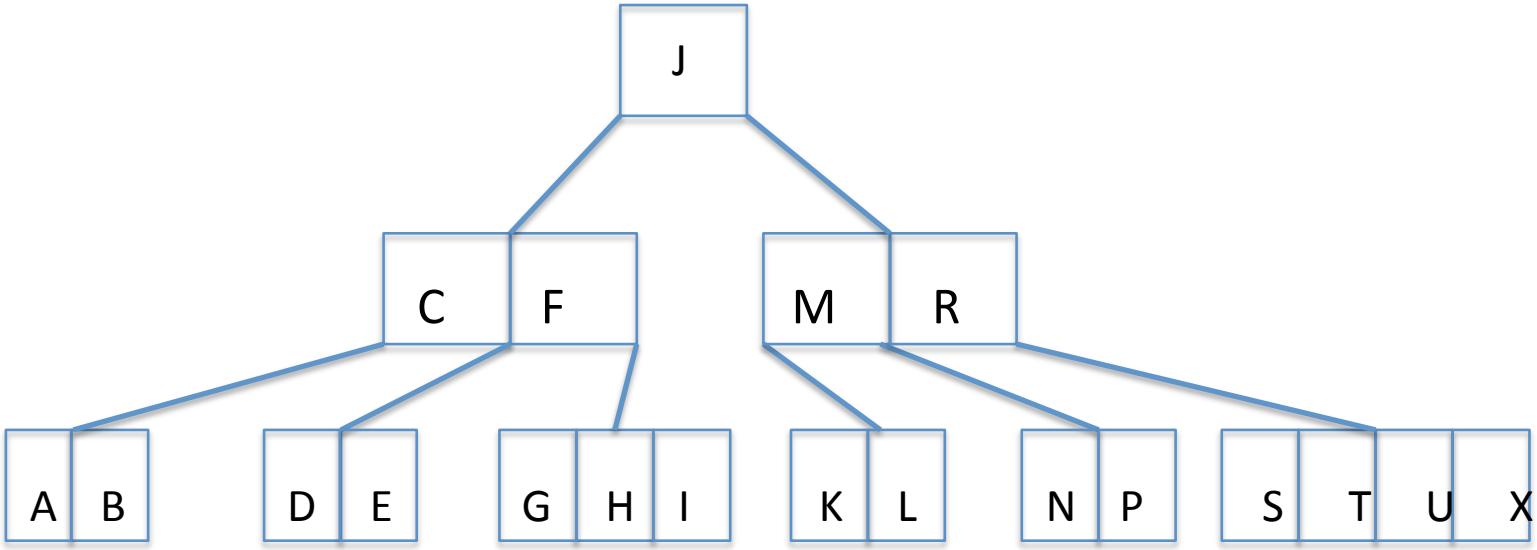
Creating a B-tree of order 5

A G F B K D H M J E S I R X C L N T U P



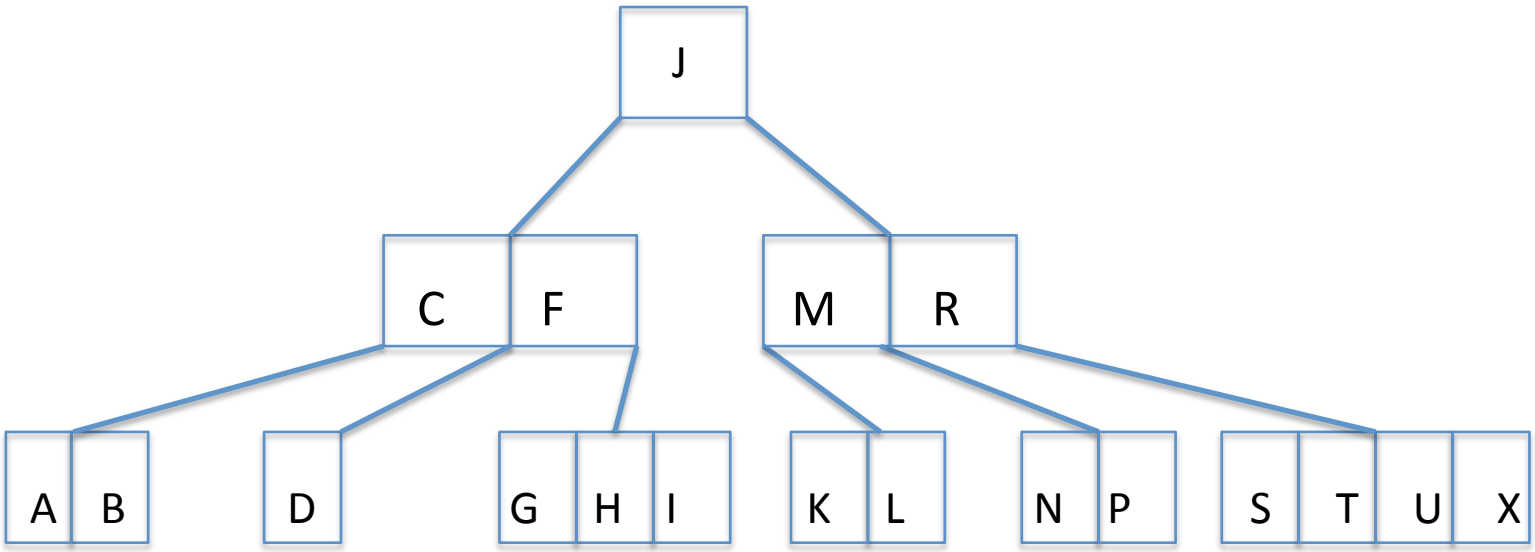
Deleting Nodes

- Delete E from leaf node



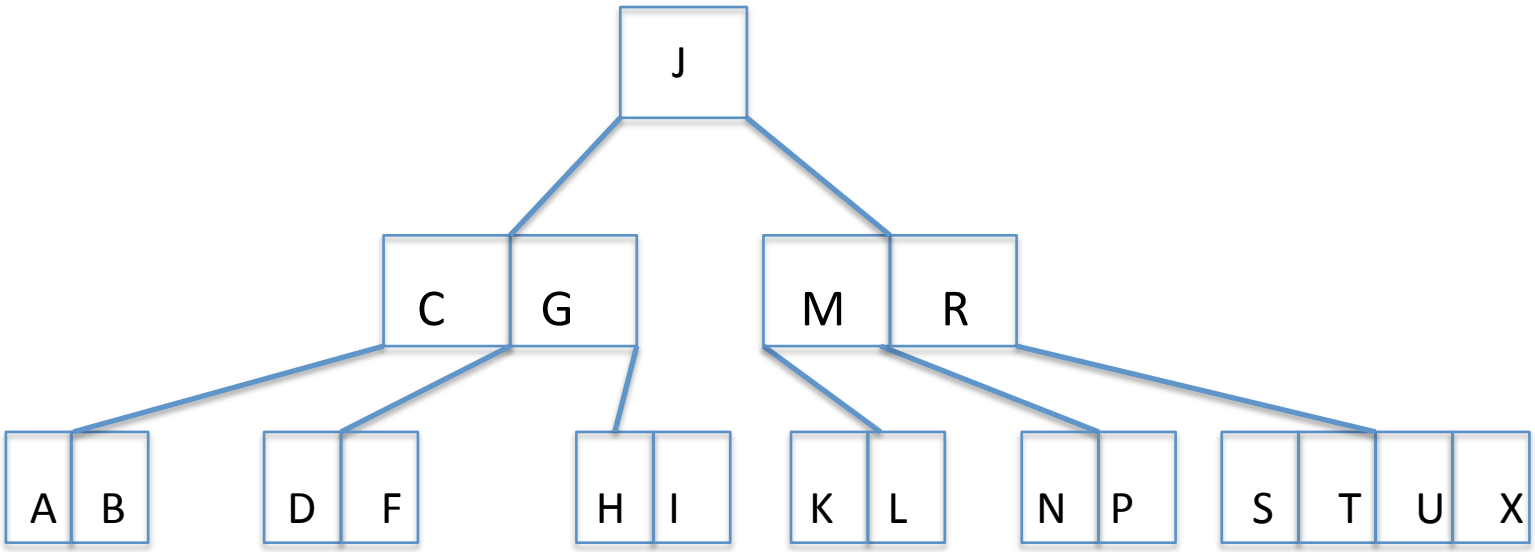
Deleting Nodes

- Delete E



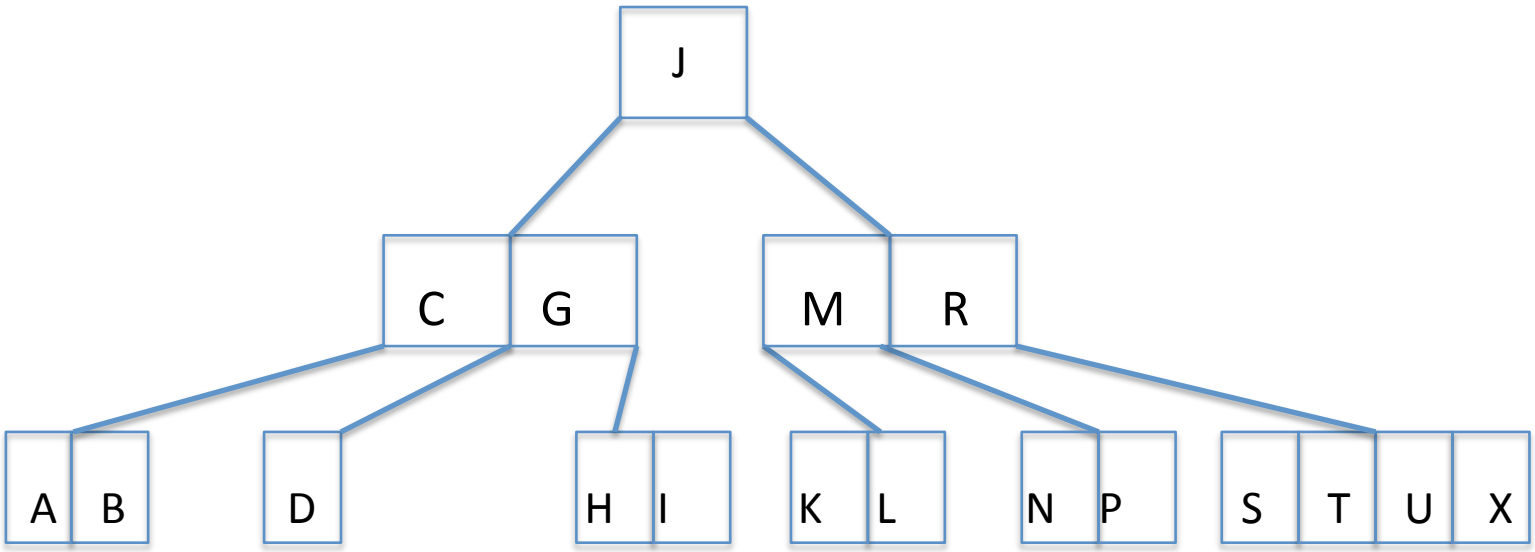
Deleting Nodes

- Borrow from a neighbor



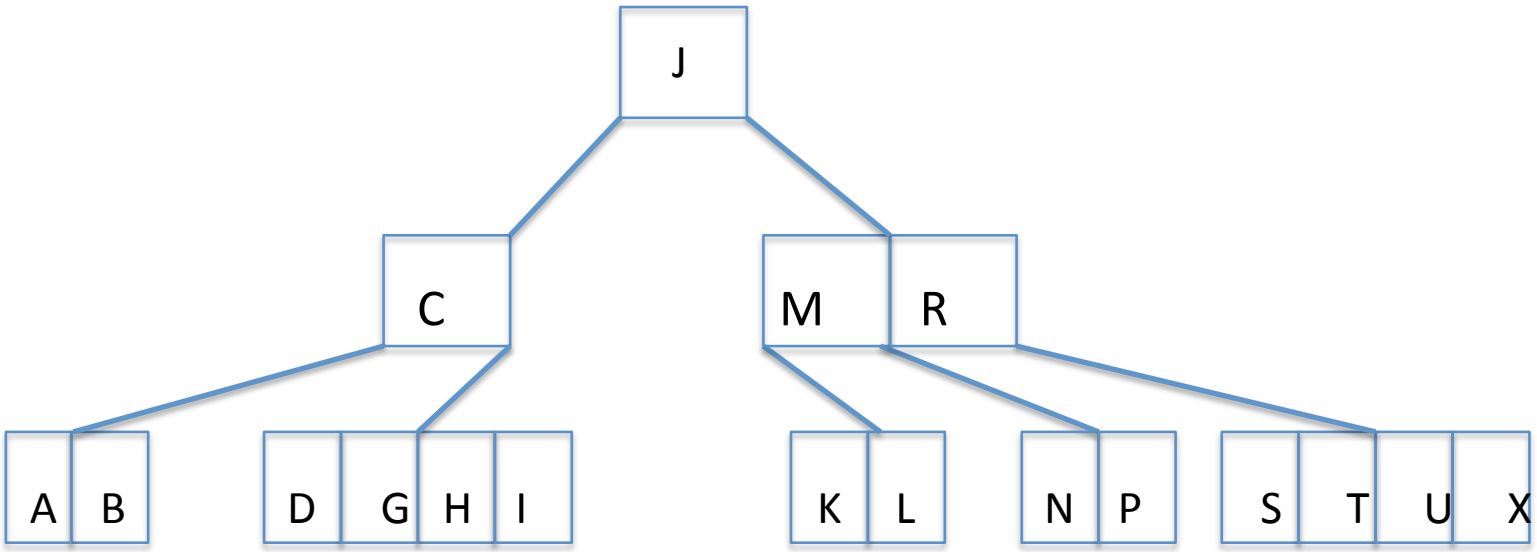
Deleting Nodes

- Delete F --- but can't borrow from a neighbor



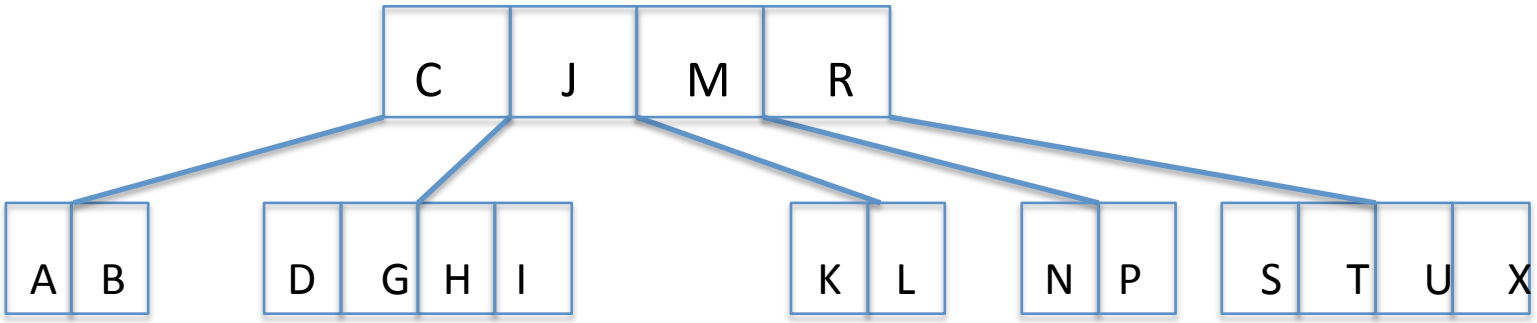
Deleting Nodes

Combine and push the problem up one level



Deleting Nodes

Can’t borrow so combine

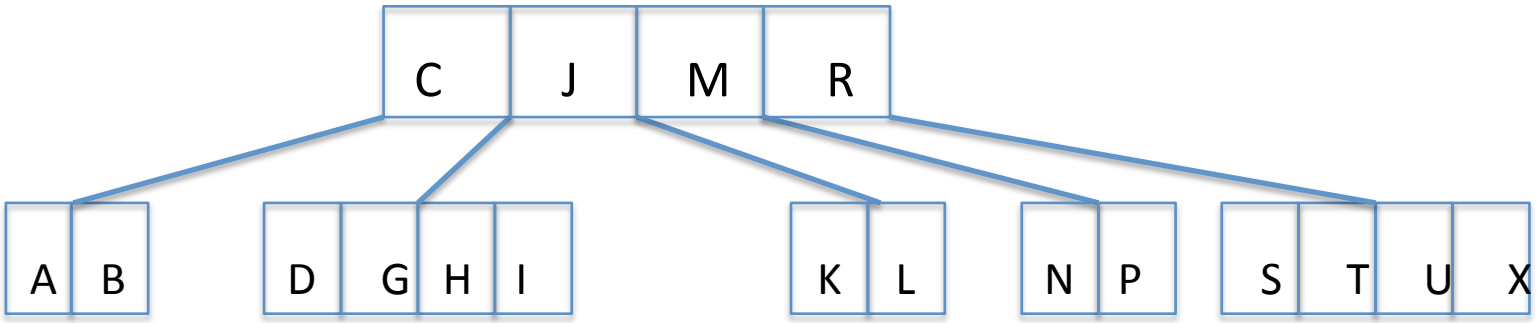


Deleting Nodes

Delete M from non-leaf node

Note: immediate predecessor in non-leaf

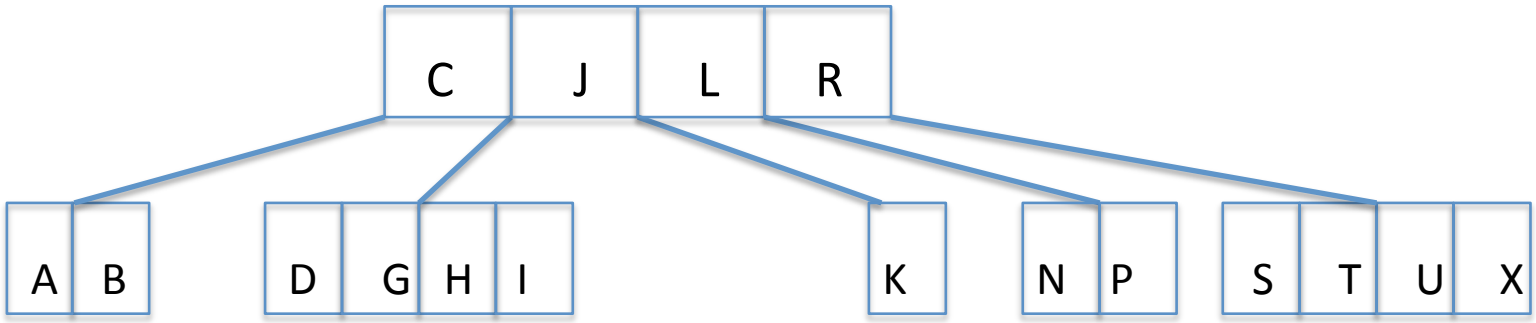
Is always in a leaf.



Deleting Nodes

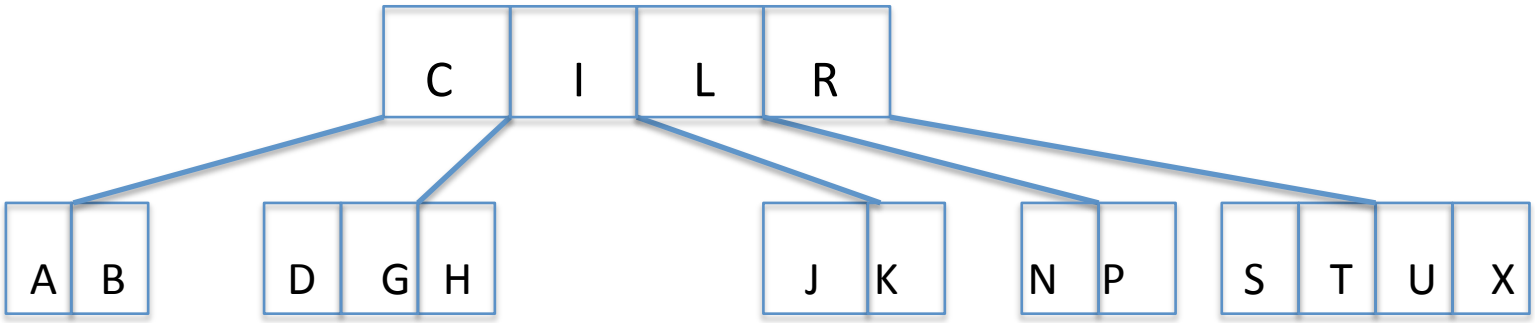
Delete M from non-leaf node

Overwrite M with immediate predecessor



Deleting Nodes

Borrow from a neighbor



Practical considerations

- Since each node correspond to a block/page, their size is usually a power of 2.
- The number of records in a node is therefore usually even.
- The number of children (the order of the tree) is therefore usually odd.

Efficiency of B-trees

- If a B-tree has order d, then each node (apart from the root) has at least d/2 children. So the depth of the tree is at most log d/2 (size)+1. In the worst case have to make d-1 comparisons in each node (linear search, but d-1 is a constant factor).
- Fewer disk accesses than for a binary tree.
- Each node could correspond to one block of data (plus addresses of successor nodes).

Variant of B-trees

- Only leaves contain data
- Non-leaves contain only keys and block numbers
- Access speed is higher because each block can hold more block numbers.
- Programming is more complicated because there are two kinds of nodes: leaves and non-leaves.

Indexing

- Index: list of key/block pairs.
- If small enough, could be kept in the main memory.
- If the index is too large to be loaded in the main memory, it is often kept in a B-tree.
- Multiple index files: if different kinds of search have to be performed, can keep one index for every set of keys.

Figure 1.2 B-tree Structure

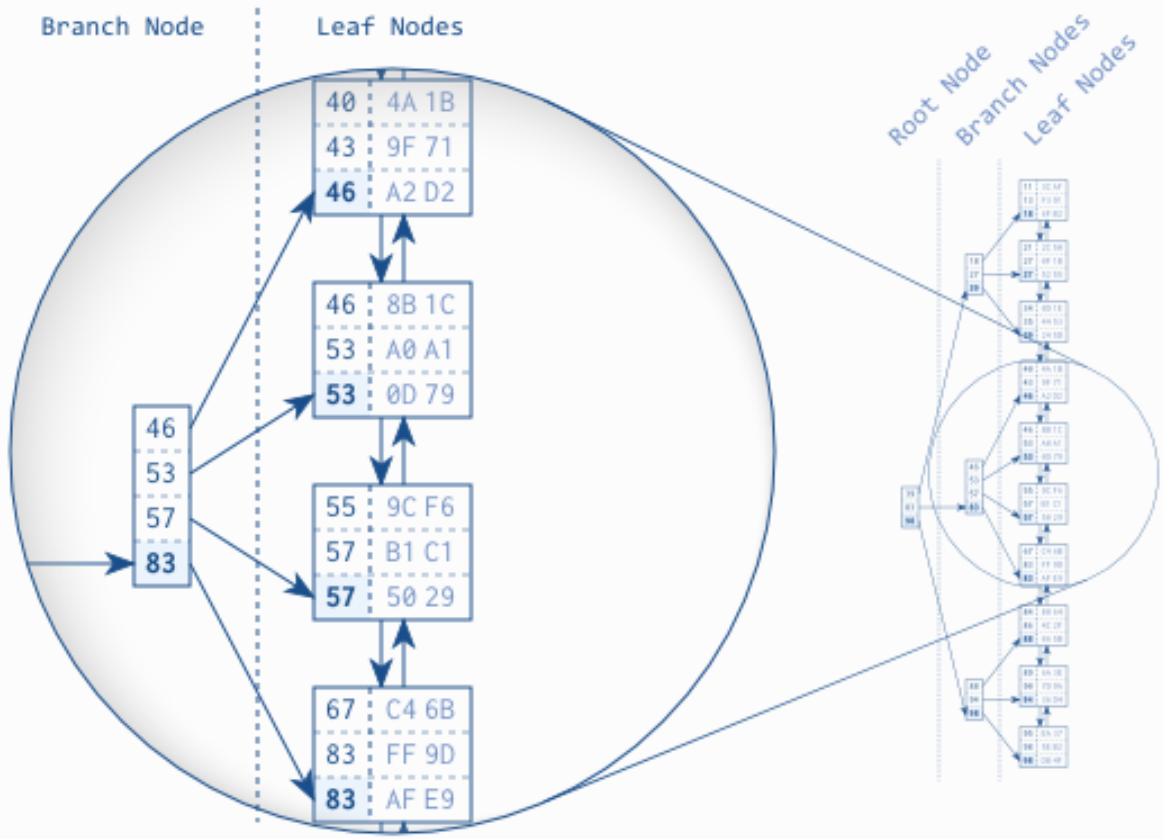


Figure 1.2 shows an example index with 30 entries. The doubly linked list establishes the logical order between the leaf nodes. The root and branch nodes support quick searching among the leaf nodes.

http://use-the-index-luke.com/sql/anatomy/the-tree

Tasks

- Self study
  - **Read**: B Trees (online sources)
- Finish Assignment 4 by Wed Oct 26 *(75 points)*
- Make progress on CodeChef (100 points)

Questions?