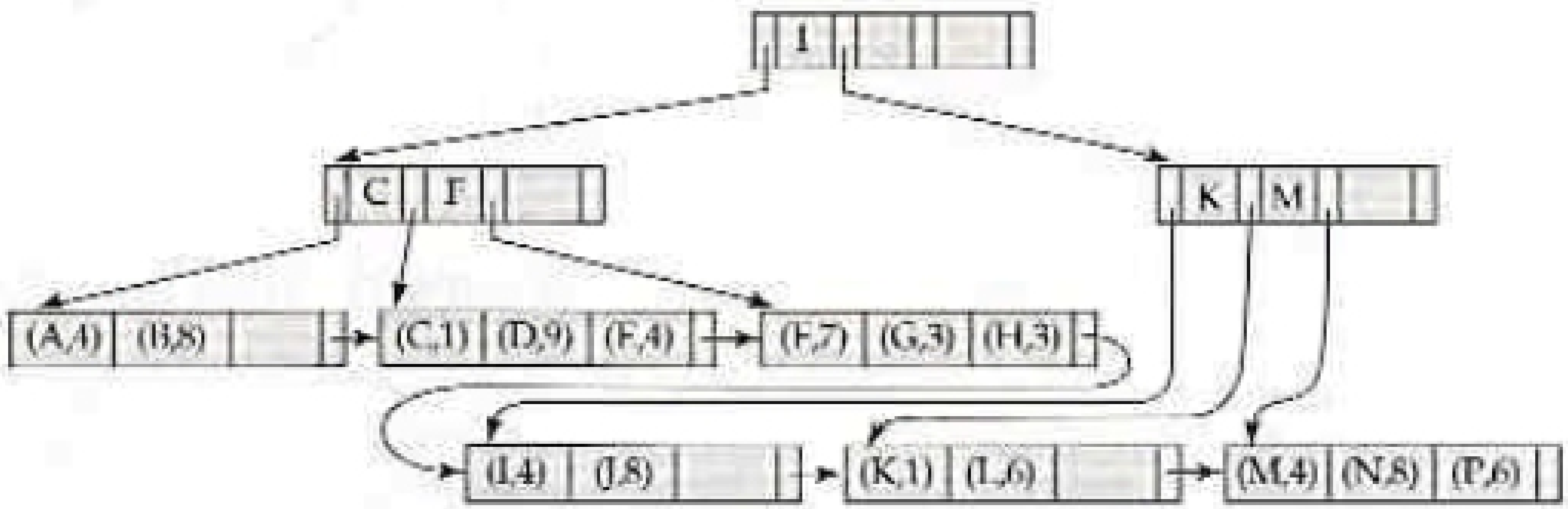
Quick Review of material covered Apr 8

- B+-Tree Overview and some definitions
  - balanced tree
  - multi-level
  - reorganizes itself on insertion and deletion
  - built so each node fits on a single disk page
- Examined mechanics of B+-tree Insertion and Deletion
  - looked at several examples
- We'll finish up B+-trees with two more concepts:
  - B+-tree File Organization
  - B-tree index files

B+-tree File Organization

- B+-Tree Indices solve the problem of index file degradation. The original data file will still degrade upon a stream of insert/delete operations.
- Solve data-file degradation by using a *B+-tree file organization*
- Leaf nodes in a *B+-tree file organization* store records, not pointers into a separate original datafile
  - since records are larger than pointers, the maximum number of recrods that can be stored in a leaf node is less than the number of pointers in a non-leaf node
  - leaf nodes must still be maintained at least half full
  - insert and delete are handled in the same was as insert and delete for entries in a B+-tree index

B+-tree File Organization Example



- Records are much bigger than pointers, so good space usage is important
- To improve space usage, involve more sibling nodes in redistribution during splits and merges (to avoid split/merge when possible)
  - involving one sibling guarantees 50% space use
  - involving two guarantees at least 2/3 space use, etc.