Chapter 9 Multilevel Indexing before the B—tree

Why do we care?

- If we sort the index on 80,000,000 record file it still takes
- Log 80,000,000 = 27 probes on average to find what we are looking for
- Since the keys don't all fit this could mean more disk accesses than we are willing to tolerate
- This doesn't address the difficulties caused by moving the index elements around when we add, delete and update records

There are two problems that must be addressed

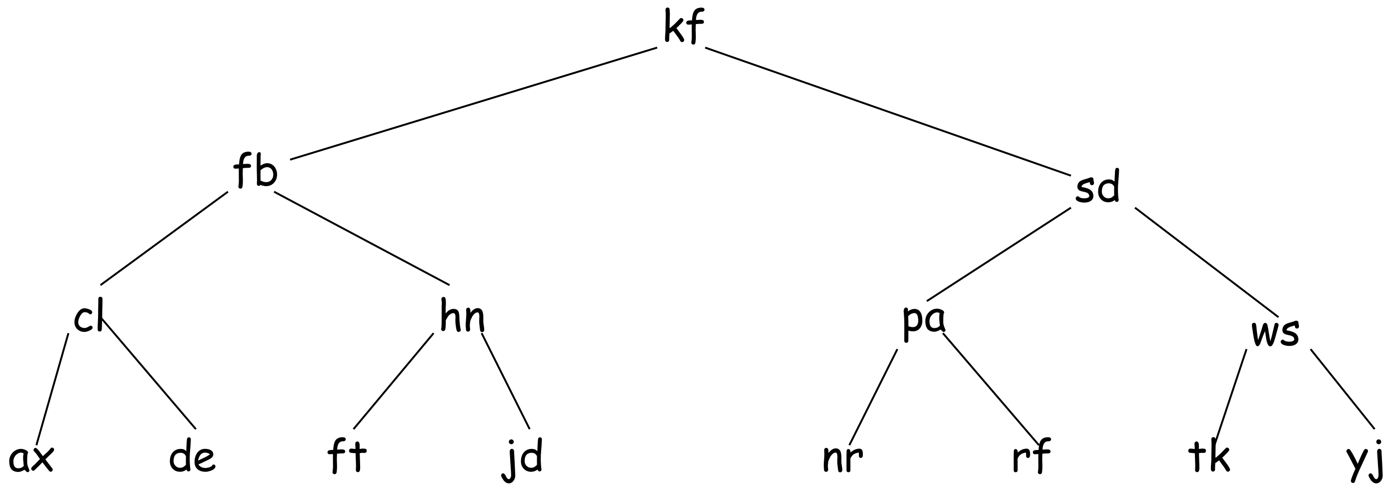
- Searching the index must be faster than binary searching
  - As many as 3 seeks is unsatisfactory
  - 9 or 10 is unbearable
- Insertion and deletion must be as fast as search
  - Inserting a key into the index requires moving other entries
  - This can mean rewriting the entire index

What are the options??

- Binary search tree
- AVL tree
- Paged Binary Tree
- Multilevel Indexes
- B-trees

Binary Search Trees : No duplicates

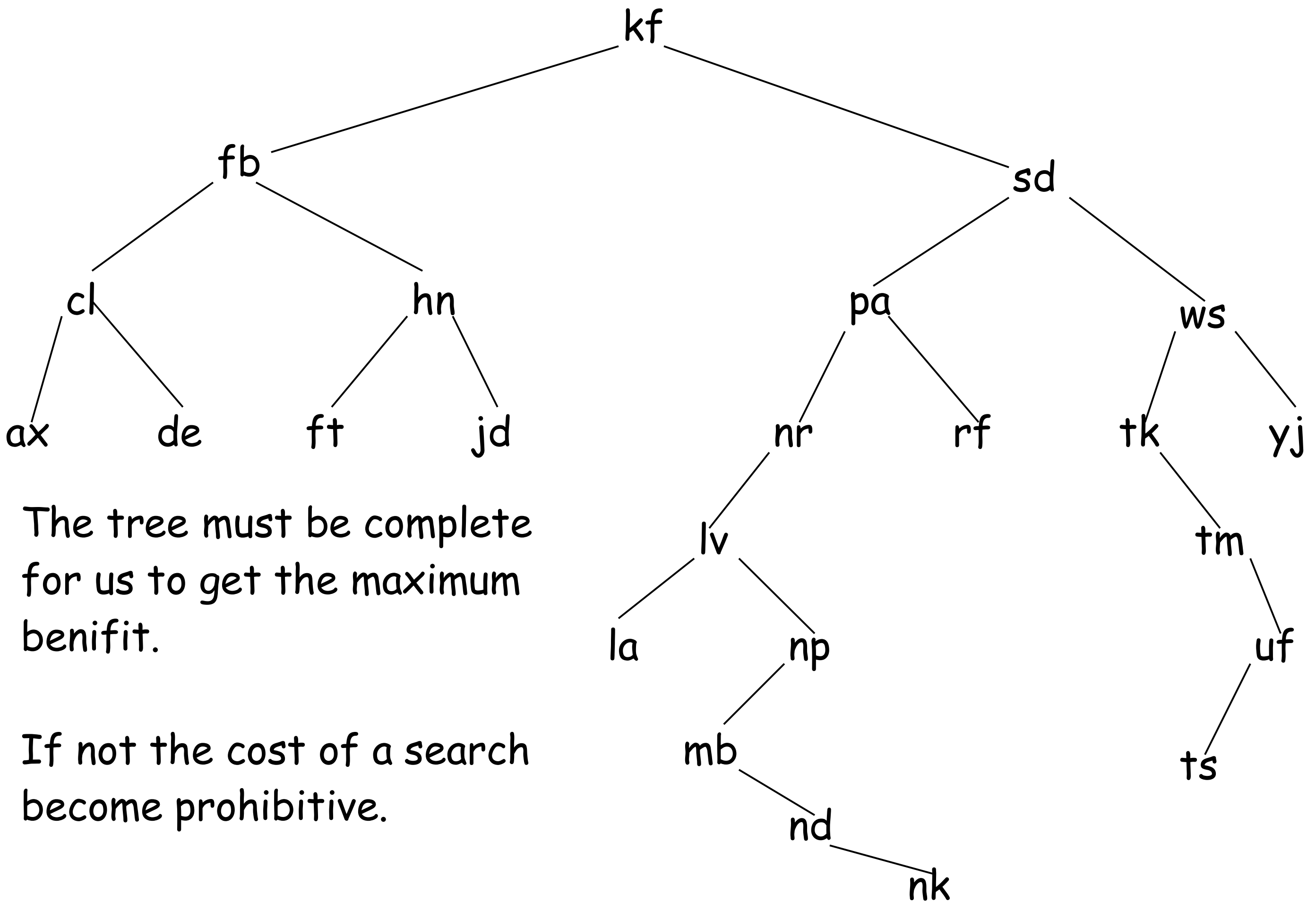
- Contents of Right Child is > Contents of Root
- Contents of Left Child is < Contents of Root



data: ax cl de fb ft hn jd kf nr pa rf sd tk ws yj

- As we said earlier this takes too long.
- It has other problems as well

- When the tree isn't balanced it can be as bad as sequential search
- Rebalancing can require seeks



The tree must be complete for us to get the maximum benifit.

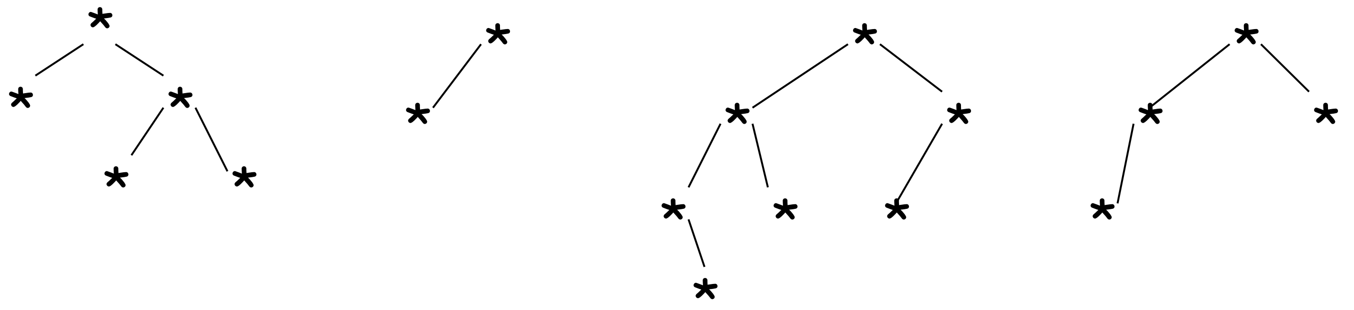
If not the cost of a search become prohibitive.

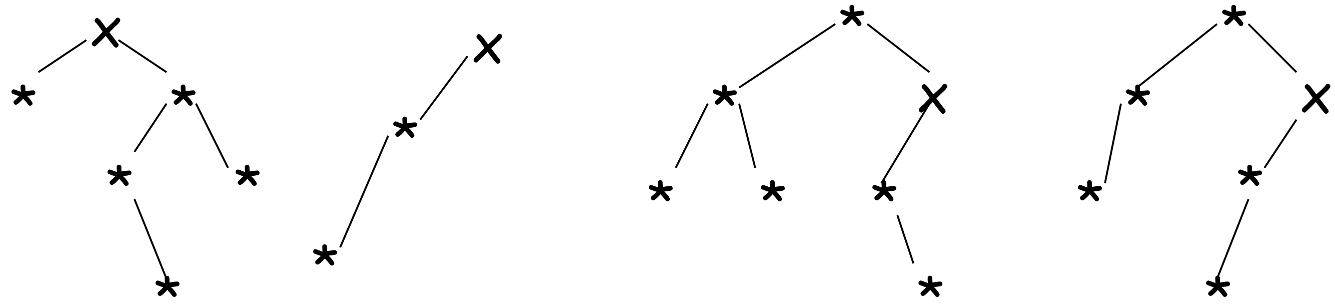
AVL-Trees: Russian Mathematicians Adel’son-Vel’skii and Landis

- Reorganize the nodes of the tree as we receive new keys
- This maintains the balance of the tree
- Trees don’t have to be complete just height balanced

- The maximum allowable difference between the heights of any two subtrees sharing a common root is one

AVL—trees





Not AVL—trees

Two important features of AVL—trees

- Guarantee a minimum level of performance in searching
    - $\log_2(N + 1)$ for the binary search tree
    - $1.44 \log_2(N + 2)$ for the AVL—tree
- Maintaining the balance is confined to a single, local area of the tree

But

- we still have to do too many seeks

Paged Binary Trees

- Divide the binary tree into pages
- $\log_{SizeOfPage}(N + 1)$ which can be much smaller than 2
- This has the same organizational problems for insertion and deletion and keeping the pages balanced.

So… We need a structure that makes maintaining the balance in the face of insertions and deletions fast.