B+ trees complexity

Complexity analysis of search

Assumption: N blocks in the index n order or the tree

Each search requires navigating along a path from the root to a leaf

Thus the complexity corresponds to the height h of the tree

Height h is maximum if the branching factor at the nodes is minimal

At root: 2

At non-leaf nodes: n/2

The root splits the tree into two trees of N/2 keys

We must find a number ‘a’ such that (n/2)a >= N/2

So a ~ log(n/2)(N/2) = O(logn(N))

If the branching factor is maximum at each nodes, the complexity analysis also gives O(logn(N))

Complexity analysis for insert and delete

In the worst case, for both insert and delete, processing is determined by a downward phase of h steps and an upward phase of also h steps. Consequently, the complexity of insert and delete is also O(logn(N))

To improve complexity, place the first several levels in main memory

For typical cases, search time is measured in terms of a few block I/O’s and this for very large data files

The leaf level provides a sorted list of the records in the data file.

Range searches can be accommodated: given range(k1, k2), locate the leaf holding k1 and then follow along the leaf level until reaching records with key value higher than k2