

Announcements

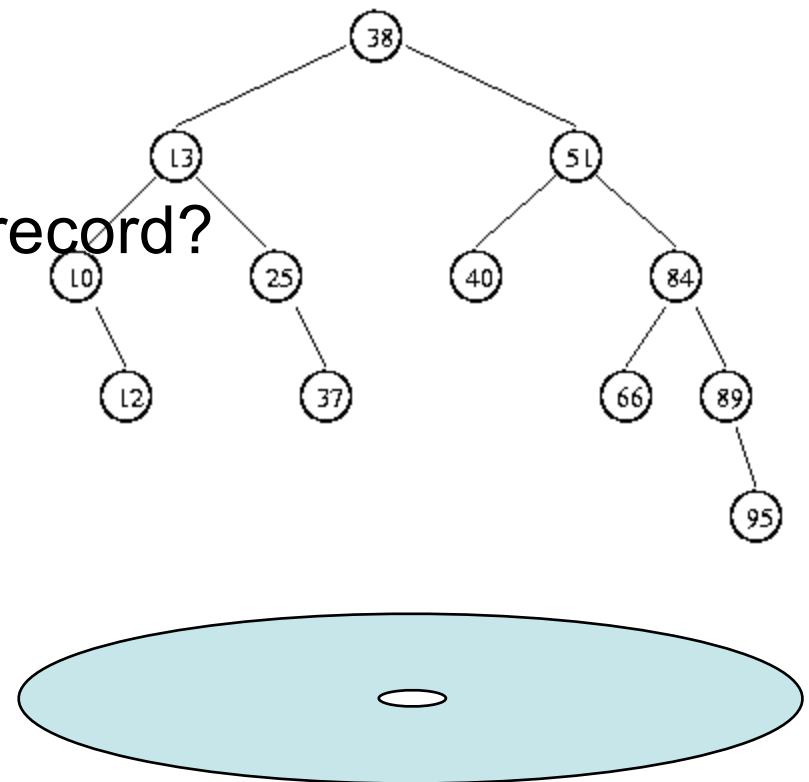
MP5 av, due 11/1, 11:59p.

Imagine an AVL tree storing _____.

How many records?

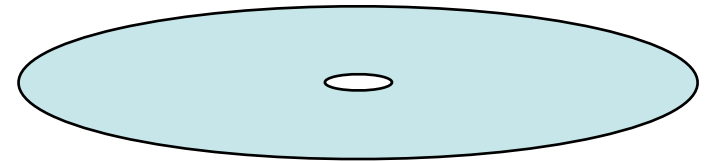
How deep is the AVL tree?

How many disk seeks to find a record?



B Tree of order m

12	18	27	52	58	63	77	89
----	----	----	----	----	----	----	----



Goal: Minimize the number of reads from disk

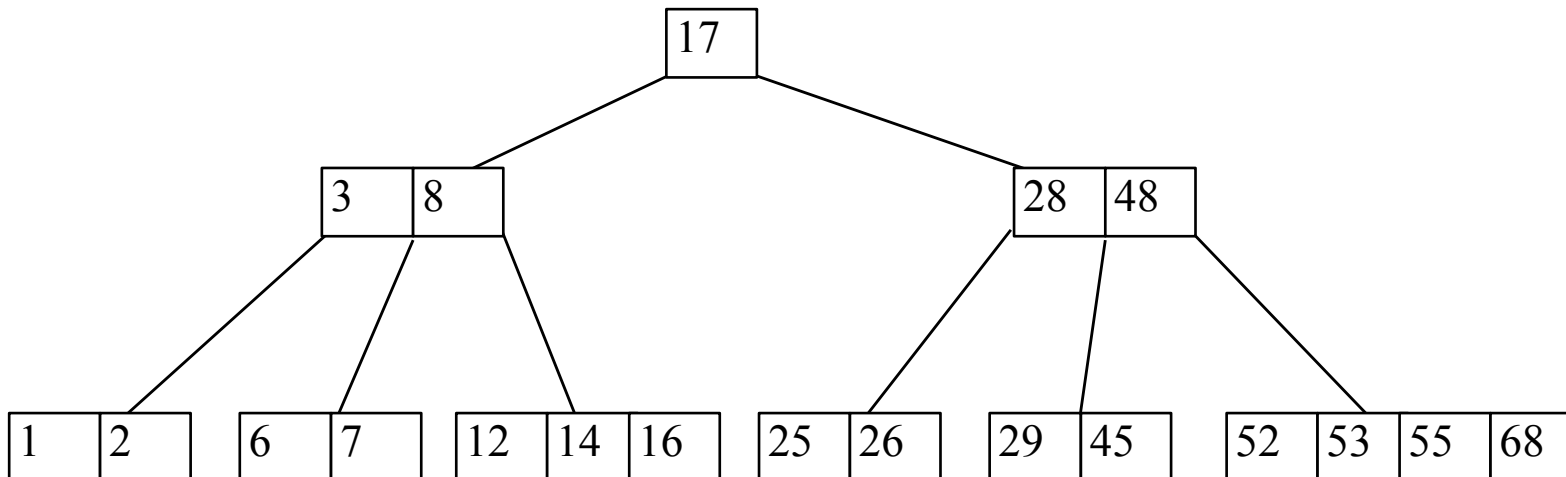
- Build a tree that uses 1 disk block per node
 - Disk block is the fundamental unit of transfer
- Nodes will have more than 1 key
- Tree should be balanced and shallow
 - In practice branching factors over 1000 often used

<http://people.ksp.sk/~kuko/bak/big/>

Definition of a B-tree

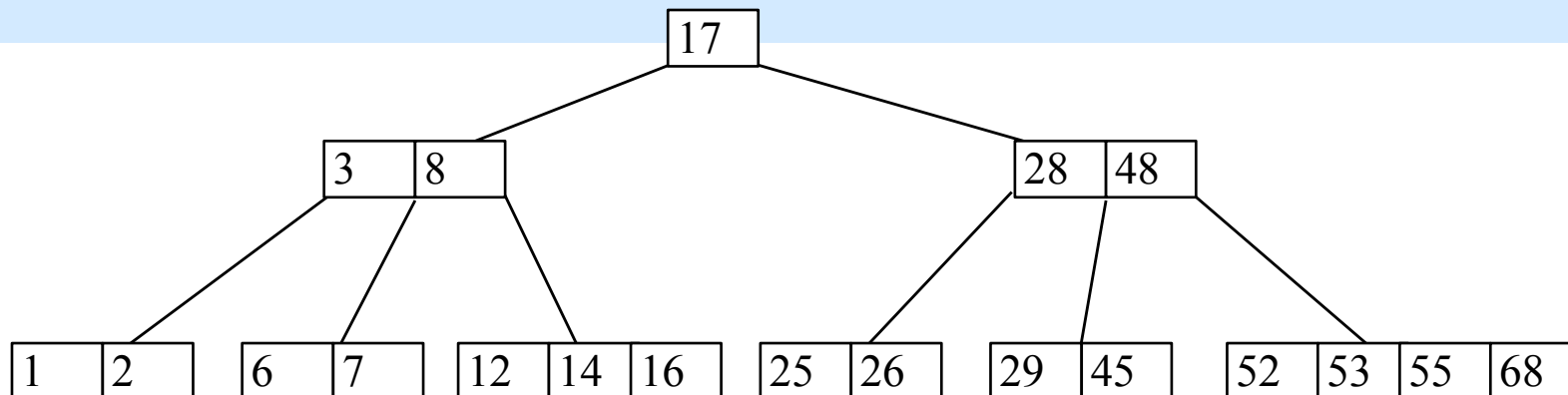
B-tree of order m is an m -way tree

- For an internal node, # keys = #children - 1
- All leaves are on the same level
- All leaves hold no more than $m-1$ keys
- All non-root internal nodes have between $\lceil m/2 \rceil$ and m children
- Root can be a leaf or have between 2 and m children.
- Keys in a node are ordered.



Searching a B-tree

```
bool B-TREE-SEARCH(BtreeNode & x, T key){
    int i = 0;
    while ((i < x.numkeys) && (key > x.key[i]))
        i++;
    if ((i < x.numkeys) && (key == x.key[i]))
        return true;
    if (x.leaf == true)
        return false;
    else{
        BtreeNode b=DISK-READ(x.child[i]);
        return B-TREE-SEARCH(b, key);
    }
}
```



Analysis of B-Trees (order m)

The height of the B-tree determines the number of disk seeks possible in a search for data.

We want to be able to say that the height of the structure and thus the number of disk seeks is no more than _____.

As we saw in the case of AVL trees, finding an upper bound on the height (given n) is the same as finding a lower bound on the number of keys (given h).

We seek a relationship between the height of the structure (h) and the amount of data it contains (n).

Analysis of B-Trees (order m)

We seek a relationship between the height of the structure (h) and the amount of data it contains (n).

- The minimum number of *nodes* in each level of a B-tree of order m :
(For your convenience, let $t = \underline{\hspace{2cm}}$.)

root

level 1

level 2

. . .

level h

- The total number of nodes is the sum of these:

- So, the least **total** number of *keys* is:

Analysis of B-Trees (order m)

We seek a relationship between the height of the structure (h) and the amount of data it contains (n). (continued...)

- So, the least **total** number of *keys* is:
- rewrite as an inequality about n , the total number of keys:
- rewrite **that** as an inequality about h , the height of the tree (note that this bounds the number of disk seeks):

Summary

B-Tree search:

$O(m)$ time per node

$O(\log_m n)$ height implies $O(m \log_m n)$ total time

BUT:

Insert and Delete have similar stories.

What you should know:

Motivation

Definition

Search algorithm and analysis

What you should not know:

Insert and Delete