Can we always fit data into main memory?

Then, where do we keep the data?

Big-O analysis assumes uniform time for all operations.

But…

Considering disk access

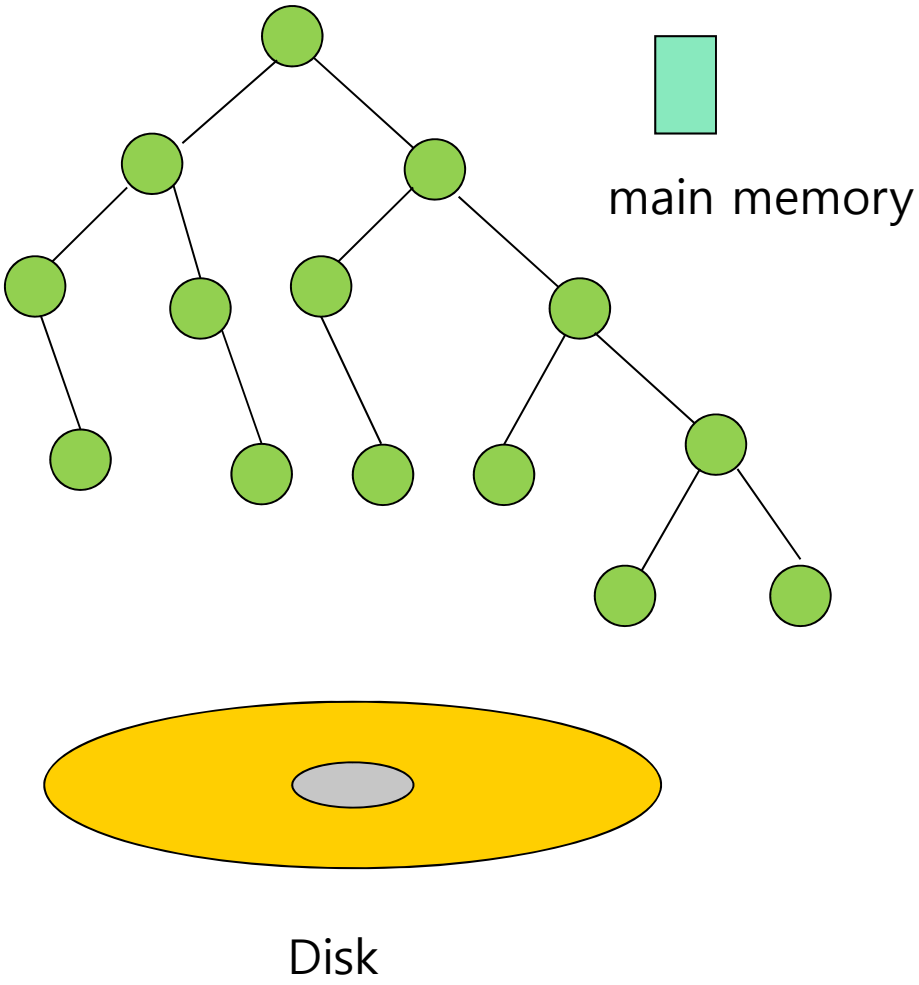
1GHz machine gives around 1m instructions per _____.

Seek time around _____ for a current hard disk.

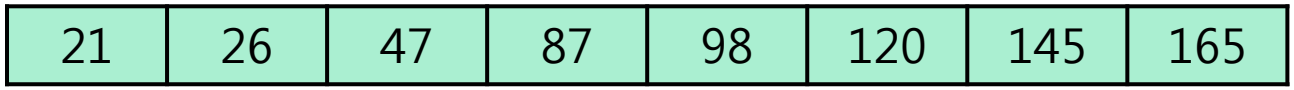
Consider an AVL tree storing 주민등록 records.

- ✓ How many records?
- ✓ How deep is the AVL tree?
- ✓ How many disk seeks to find a record?

B tree vs. AVL tree



B tree of order m

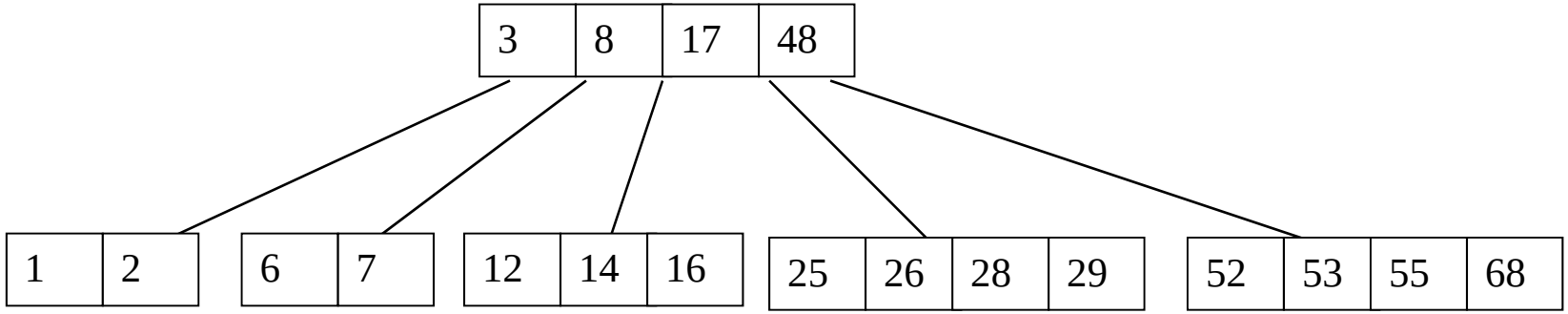


Goal: Minimize the number of reads from disk

- 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

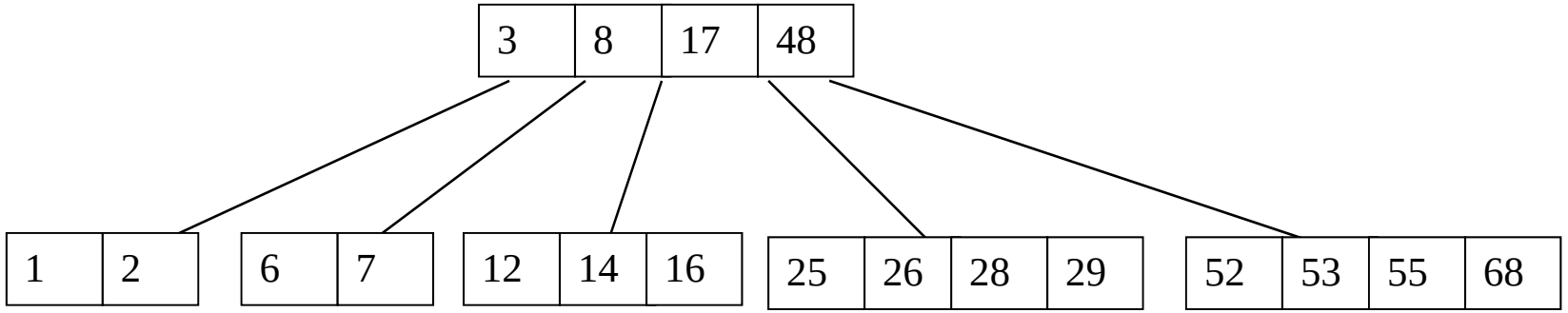
B-tree of order *m* is an *m*-way tree

- For an internal node, #keys = #children - 1
- All leaves are on the same level
- All nodes 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 = _____.)

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*)

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

rewrite as an inequality about n, the actual 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

Goal: Minimize # of disk accesses

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 are similar to that on AVL trees.

What you should know:

- Motivation
- Definition
- Search algorithm and analysis