Trees in the Real World

Only 326 just stores numbers in a tree

Use key to index data
Trees in the Real World

```c
struct Node {
    int key;
    Data *data;
    Node *left, *right;
};
```

Store *pointer to data* in node with *key*

---

Achieving Perfect Balance

2-3 Trees
- 2-node: 2 children, 1 key
- 3-node: 3 children, 2 keys
- Any key $k$ is *between* all keys in the subtrees adjacent to $k$
- All leaves are at the *same depth*
- What is depth with respect to size?
- How long to search?

---

Insertion

Search for leaf location
Insertion

Squeeze new key into leaf

Split to make space

Many need to split further up the tree
Insertion

If root is full, increase height

Total time to insert?

Deletion

(a, b)-Trees

Why Stop at 2-3?

- Each node has between $a$ and $b$ children
  - $a \geq 2, b \geq 2a - 1$, root may be smaller
  - Why is $b$ at least twice $a$?
- All leaves at same depth
- Advantages?
- Disadvantages?
Disks

- Entire blocks transferred at a time
- Transfer Time \(<<\) Block Search Time
- Need to minimize disk accesses

B-Trees

- \# disk access to search = \# levels in tree
- Each node may as well fill up disk block
  \(i.e. b = \text{block size}\)
- Only store index in node
- Store data in leaves

\[ a = 100, b = 199: \]
- How many levels to store \(10^6\) items?

- Many variations, very practical