Chapter 8, Strings:

- General introduction; string sorting (last week Friday)
- Tries (**Today**)
- Other string topics (Wednesday)
  - Regular expressions
  - ~~Huffman encoding~~
  - ~~Edit distance~~
  - ~~Grammars and parsing~~
- Review for Tests 3 and 4 (next week Friday)

Today:

- Problem statement
- Main idea behind tries
- Code details:
  - Node class
  - Find
  - Insertion
  - Deletion

Coming up (the last):

Catch up on old projects . . .
Do **Perfect Hashing** *project (due today, Monday, Dec 9)*
Do **Trie** *project (due Friday, Dec 13)*

Due **Today, Mon, Dec 9**
Read Section 8.2

End-of-semester important dates

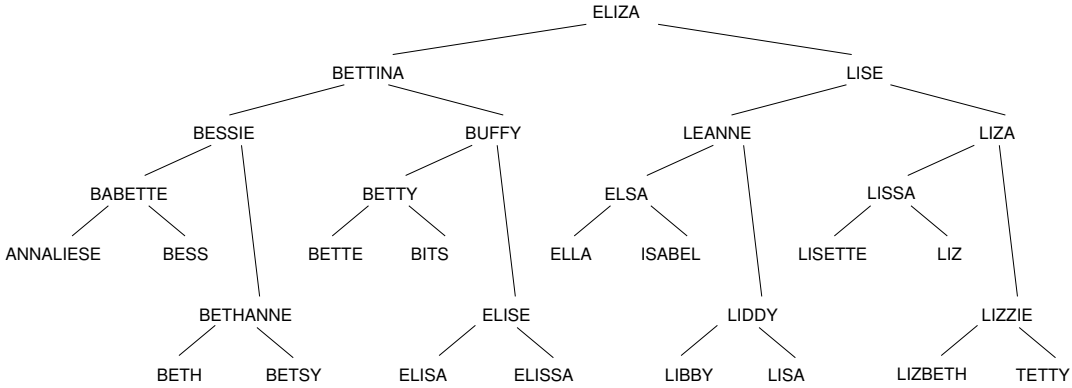
- Mon, Dec 9: Last project assigned
- Tues, Dec 10: Last "normal" running of project grading script
- Wed, Dec 11: Test 3 & 4 Review sheet distributed, Test 4 practice problems made available.
- Thurs, Dec 12: Review lab (pick practice problems for Test 4)
- Fri, Dec 13, AM: "Two-minute warning" running of project grading script (Canvas gradebook will not be updated—see project report in your turn-in file)
  Note that Fri, Dec 13 is the *Last Day of Classes*.
- Fri, Dec 13, midnight: Official project deadline
- Sat, Dec 14, when I wake up: Permissions to turn-in folders turned off
- Mon, Dec 16: Project grading script run for final/semester grades
- Wed, Dec 18, 10:30am-12:30pm: Tests 3 and 4 (in lab)
  - Test 3: On paper (like Test 1) covering BSTs (ch 5), DP (Ch 6), hashtables (Ch 7) and strings (ch 8).
  - Test 4: At a computer (like Test 2) covering DP (Ch 6), hashtables (Ch 7) and strings (ch 8).

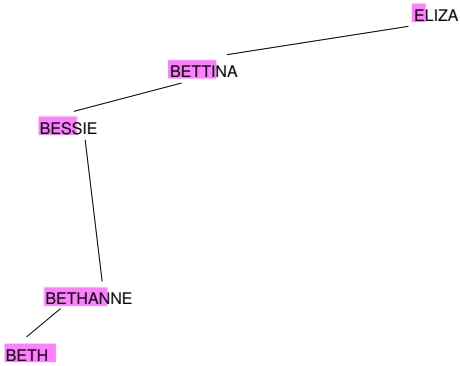
In class and in the text, we see an iterative implementation of tries.

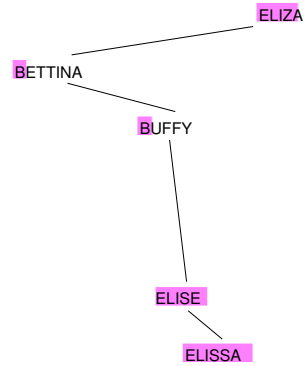
In the accompanying project, you'll implement the trie operations recursively in the node.

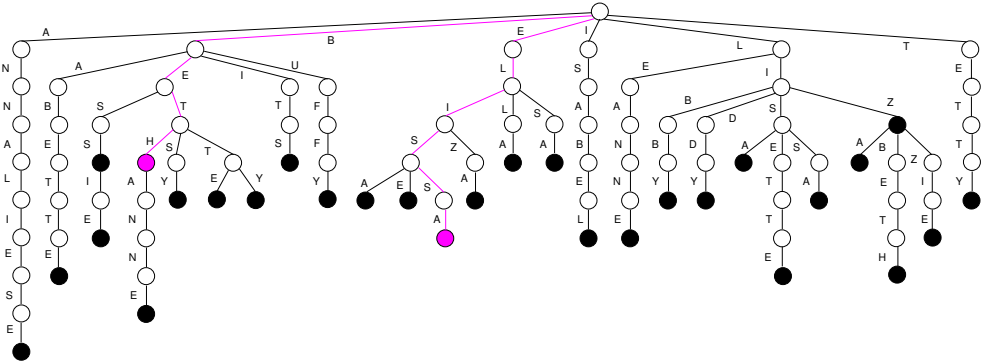
```
public class LinkedList {
    class Node {
        int datum;
        Node next;
    }

    Node root;

    public boolean contains(int item) {
        boolean found = false;
        for (Node current = root;
             ! found and current != null;
             current = current.next)
            found = current.datum == item;
        return found;
    }
}
```

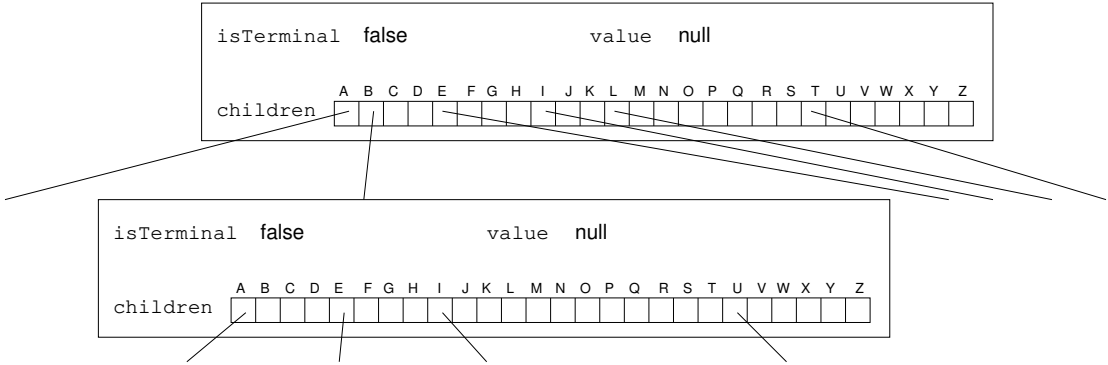
```
public class LinkedList {
    class Node {
        int datum;
        int next;
        boolean contains(int item) {
            if (item == datum) return true;
            else if (next == null) return false;
            else return next.contains(item);
        }

        Node root;

        public boolean contains(int item) {
            if (root == null) return false
            else return root.contains(item);
        }
}
```


ELIZA
BETTINA
LISE
BESSIE
BUFFY
LEANNE
LIZA
BABETTE
BETTY
ELSA
LISSA
ANNALIESE
BESS
BETTE
BITS
ELLA
ISABEL
LISETTE
LIZ
BETHANNE
ELISE
LIDDY
LIZZIE
BETH
BETSY
ELISA
ELISSA
LIBBY
LISA
LIZBETH
TETTY


ELIZA
BETTINA
BESSIE
BETHANNE
BETH


ELIZA
BETTINA
BUFFY
ELISE
ELISSA


A
B
E
I
L
T
N
A
E
I
U
L
S
E
I
E
N
B
S
T
T
F
I
L
S
A
A
B
D
S
Z
T
A
E
S
H
S
T
S
F
S
Z
A
A
B
N
B
D
A
E
S
A
B
Z
T
L
T
I
A
Y
E
Y
Y
A
E
S
A
E
N
Y
Y
T
A
E
I
Y
I
T
E
N
A
L
E
T
T
E
E
E
E
E
H
S
E
E


isTerminal false
value null
children
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
isTerminal false
value null
children
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Invariant 39. [Class invariant of `TrieMap`]

(a) For all nodes, the path to that node is a prefix to at least one key in the map.

(b) For all nodes, the node is terminal iff the path to that node is a key in the map.

In class and in the text, we see an iterative implementation of tries.

In the accompanying project, you'll implement the trie operations recursively in the node.

```
public class LinkedList {
    class Node {
        int datum;
        Node next;
    }

    Node root;

    public boolean contains(int item) {
        boolean found = false;
        for (Node current = root;
             ! found and current != null;
             current = current.next)
            found = current.datum == item;
        return found;
    }
}
```

```
public class LinkedList {
    class Node {
        int datum;
        int next;
        boolean contains(int item) {
            if (item == datum) return true;
            else if (next == null) return false;
            else return next.contains(item);
        }

        Node root;

        public boolean contains(int item) {
            if (root == null) return false
            else return root.contains(item);
        }
}
```