Announcements

- Midterm 2 in 1 week!
  - Practice midterm in center table
  - Review Friday (no lab this week)

- All HW due Thursday

- Next HW will be posted, but not due until a week after the exam (in pairs)
We want to transmit information using as few bits as possible.

Standard ASCII: 8 bits

(Extended - 64 bits)

Encode: 40,000 characters \times 8 bits
So— how can we do better?

What if we don't use every character?

Hello: 5x8 bits

More common characters should use fewer bits.
Problem:
Confusion while parsing:

Codes:

- E: 11
- A: 00
- S: 01
- T: 10
- R: 110
- M: 001
- B: 010
- N: 100

Decode this:

- 11001
  - EM
  - or
  - RS
Prefix-free codes

An unambiguous way to send information when we have characters that are not of a fixed length.

No letter's code is the prefix of another letter.

Encode: BAN

1000111
Decode:

1000 11011 0101
↓↓↓↓↓↓↓↓↓↓↓
B ANANA (EOM)

13 bits
versus
7x9 = 56 bits
So how do we do this? With exact frequency counts!

This sentence contains three a’s, three c’s, two d’s, twenty-six e’s, five f’s, three g’s, eight h’s, thirteen i’s, two l’s, sixteen n’s, nine o’s, six r’s, twenty-seven s’s, twenty-two t’s, two u’s, five v’s, eight w’s, four x’s, five y’s, and only one z.

A C D E F ...
3 3 2 26 5

Goal: Make a tree:

E S
Using frequency counts, build one of these trees. Which ones should get few bits?
Huffman's algorithm

Take the two least frequent characters.

Merge them into 1 letter, which becomes a new "leaf".
Example:

| A | C | D | E | F | G | H | I | L | N | O | R | S | T | U | V | W | X | Y | Z |
| 3 | 3 | 2 | 26 | 5 | 3 | 8 | 13 | 2 | 16 | 9 | 6 | 27 | 22 | 2 | 5 | 8 | 4 | 5 | 1 |

Merge D + 2:

| A | C | E | F | G | H | I | L | N | O | R | S | T | U | V | W | X | Y | Z |
| 3 | 3 | 26 | 5 | 3 | 8 | 13 | 2 | 16 | 9 | 6 | 27 | 22 | 2 | 5 | 8 | 4 | 5 | 3 |
Next?
In end, build a tree:
Using the tree:

```
1001 0100 1101 00 00 111 011 1001 111 011 110001 111 110001 10001 011 1001 110000 1101
```

THIS SENTENCE CONTAINS
How many bits?

| freq. | 3 | 3 | 2 | 26| 5 | 3 | 8 | 13| 2 | 16| 9 | 6 | 27| 22| 2 | 5 | 8 | 4 | 5 | 1 |
| depth | 6 | 6 | 7 | 3 | 5 | 6 | 4 | 4 | 7 | 3 | 4 | 4 | 2 | 4 | 7 | 5 | 4 | 6 | 5 | 7 |
| total | 18| 18| 14| 78| 25| 18| 32| 52| 14| 48| 36| 24| 54| 88| 14| 25| 32| 24| 25| 7 |

total = 646 bits

How many bits would ASCII use to send these 170 letters?

170 × 8 = 1360
Exercise: 01001111000010100001010001

Message? HELLO
How many bits? 26 versus 5x8 = 40
Thm: Huffman codes are optimal in the sense that they use the fewest # of bits possible.

(See page 374 to see the proof, or read supplemental notes on the schedule page.)

This is a greedy algorithm.
Next program: Decode

Given an input which describes a tree and a set of bits which are a message:

1) Create the tree (our BinaryTree.h)
2) Use it to decode the message