Announcements

- Exam back Monday

- HW due Sat.

- Next HW—checkpoint will be due right after break

- Final: Dec. 17 (Monday) at noon
Idea

We want to transmit information using as few bits as possible.

\[ 2^8 = 256 \text{ characters total} \]

Standard ASCII: 8 bits per character

Hello: \( 5 \times 8 \text{ bits total} \)

Extended ASCII: 64 bits?
So—how can we do better?

What if we don't use every character?

Use fewer bits for more common characters

Penalty: less common characters will need more bits.

Problem: variable length codes
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

1000011
Decode:

1000 1101 0101

T T I Y I
end of BANANA message

A

B EoM

Even though each letter is different length I scan once use the tree to detect letters.

How do we get a good tree?
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.

\[
\begin{array}{cccc}
A & C & D & E \\
3 & 3 & 2 & 26 \\
\end{array}
\]

pull exact letter counts
Using frequency counts, build one of those trees.

| 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 |

Which ones should get few bits? Many bits

```
\text{build up} \quad \text{bottom leaf}
```

\( \text{s} \quad \text{e} \quad \text{t} \)
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 + Z:

Merge L + U

D + Z, together have Req. 1 + 2
Next?\[ A \times C \]
\[ G \times D \]
In end, build a tree:
Using the tree:
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 |

\[
\text{total} = 646 \text{ bits}
\]

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

\[170 \times 8 \text{ (bigger)}\]
Exercise: 01001111 00000100 00001010 0011

Message? HELLO

How many bits? 26 bits (versus 40 w/ASCII)
Thm: Huffman codes are optimal in the sense that they use the fewest # of bits possible.

(60 take 314 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
2) Use it to decode the message