Announcements

- HW due Monday
We want to transmit information using as few bits as possible.

Standard ASCII: 8 bits

(Extended: 16 bit)
So—how can we do better?

What if we don’t use every character?

5. assign more frequent characters via strings of fewer bits
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
100011
Decode:

| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |

BANANA
Goal: minimize cost

Here, minimize total length of encoded messages.

\[ \text{Cost}(T) = \sum_{i=1}^{n} f[i] \cdot \text{depth}(i) \]

Input: \( f[1..n] \)
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...
3 3 2 26
Using frequency counts, build one of those trees.

| 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 |
| 3 | 3 | 2 | 26 | 5 | 3 | 8 | 13 | 2 | 16 | 9 | 6 | 27 | 22 | 2 | 15 | 8 | 4 | 5 | 1 | 3 | 3 |

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".
Pseudo code

\textbf{BUILDHUFFMAN}(f[1..n]):
\begin{itemize}
  \item for \(i \leftarrow 1\) to \(n\)
    \begin{itemize}
      \item \(L[i] \leftarrow 0\); \(R[i] \leftarrow 0\)
      \item \text{INSERT}(i, f[i])
    \end{itemize}
  \item for \(i \leftarrow n\) to \(2n-1\)
    \begin{itemize}
      \item \(x \leftarrow \text{EXTRACTMIN}()\)
      \item \(y \leftarrow \text{EXTRACTMIN}()\)
      \item \(f[i] \leftarrow f[x] + f[y]\)
      \item \(L[i] \leftarrow x\); \(R[i] \leftarrow y\)
      \item \(P[x] \leftarrow i\); \(P[y] \leftarrow i\)
      \item \text{INSERT}(i, f[i])
    \end{itemize}
  \item \(P[2n-1] \leftarrow 0\)
\end{itemize}

Q: Which data structure do we need? 
\[\log n\]
I: heap \[O(\log n)\]
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:

| 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 | 1 |
| 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
THISSSENTENCECONTAINS...
```
How many bits?

| char | A | C | D | E | F | G | H | I | L | N | O | R | S | T | U | V | W | X | Y | Z |
| 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 |

Cost = \sum_{i,j} F_{ij} \cdot \text{depth}[i]

Total = 646 bits

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

170 x 8
Exercise: 010011110000010100001010001

Message?

How many bits?
Thm: Huffman codes are optimal in the sense that they use the fewest # of bits possible.

proof: Greedy: so how to start?

Compare to "OPT", or show ours is just as good.
Lemma: Let $x \neq y$ be 2 least common characters. There is an optimal tree in which $x \neq y$ are siblings and have largest depth.

Proof: Suppose not.

Take optimal tree $T$, where $x \neq y$ are not siblings at largest depth. Let $a \neq b$ be deepest siblings.

Create $T'$ by swapping $x$ and $a$. Create $T''$ by swapping $y$ and $a$.

It is clear that $T''$ is also an optimal tree with $x \neq y$ as siblings at largest depth.
cont:

\[ \text{cost}(T') = \text{cost}(T) - f[s] \cdot \Delta + f[x] \cdot \Delta \]

\[ \Delta = \text{depth}(a) - \text{depth}(x) \]

We know \( f[x] \leq f[a] \) since \( a \) is more frequent.

Since \( T \) was optimal, \( -f[a] \cdot \Delta + f[x] \cdot \Delta \) can't be negative.

\[ \Rightarrow \Delta (f[x] - f[a]) \geq 0 \Rightarrow f[x] \geq f[a] \]

\[ \Rightarrow f[x] = f[a] . \]
prove that Huffman codes are optimal:

Induction on # of characters:

Base case: n=1 or 2 obvious

IS: Take f[i...n]. WLOG assume f[i] + f[2] are least frequent.
Let f[n+1] = f[i] + f[2].

Apply IH on f[3..n+1], says Huffman tree is optimal on those n-1 frequencies. Call this T'.
Take $T'$, know $n+1$ is a leaf

Claim: $T$ is optimal.

$$\text{cost}(T) = \sum_{i=1}^{n} f[i] \cdot \text{depth}(i)$$

$$= \sum_{i=3}^{n} f[i] \cdot \text{depth}(i) + f[1] \cdot \text{depth}(1) + f[2] \cdot \text{depth}(2)$$

$$= \text{cost}(T') - f[n+1] \cdot \text{depth}(n+1)$$
\[ \text{cost} (T) = \ldots \]

\[ = \text{cost}(T') + (f[1] + f[2]) \cdot \text{depth}(T') \]

\[ - f[n+1] \cdot \text{depth}[n+1] \]

\[ = \text{cost}(T') + (f[1] + f[2]) \cdot \text{depth}(T) \]

\[ - f[n+1] \cdot (\text{depth}(T) - 1) \]


\[ \text{cost} (T') + f[1] + f[2] \]