CSCE 3100

Greedy Alg
Announcements

- HW2 is back
- Grades on Blackboard, please double check
- HW3 is up due in 1 week
- Midterm: Fri, Oct 13 (not 20th)
Huffman codes - the idea:
We would like to transmit info using as few bits as possible.

What does ASCII do?
- 8 bits per character
- $2^8 = 256$ letters
- Fixed length encoding

How can we do better?
- Common characters should use fewer bits.
Prefix-free codes

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

Key: No letter's code will be the prefix of another.

Encode: B A N

→ 100011
Decode:

1000110110, 0

BAN
Goal: Minimize cost

Here, minimize total length of encoded message:

Input: Frequency counts $f[1..n]$

Char $i$ has freq. $f[i]$

Compute: Tree $T$, with $i$'s placed at leaves

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

depth of $i$ in $T$
To do this, we'll need to use the array \( f \):

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.

If we ignore punctuation and spaces (just to keep it simple), we get:

\[
\begin{array}{cccccccccccccccccccccccccccc}
A & C & D & E & F & G & H & I & L & N & O & R & S & T & U & V & W & X & Y & Z \\
3 & 3 & 2 & 26 & 5 & 3 & 8 & 13 & 2 & 16 & 9 & 6 & 27 & 22 & 2 & 5 & 8 & 4 & 5 & 1 \\
\end{array}
\]

Which letters should be deeper (or shallower)? (i.e.: How to be greedy?) Put least frequent at bottom.
Huffman's alg:
Take the two least frequent characters.
Merge them into one letter, which becomes a new "leaf".

| 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 |

\[ \frac{40}{6} \]
Example (cont):

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

The tree:
In the end, get a tree with letters at the leaves:

A Huffman code for Lee Sallows’ self-descriptive sentence; the numbers are frequencies for merged characters.

| 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 |

If we use this code, the encoded message starts like this:

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

THISSSENTENCECONTENTA
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 |

\[ \text{Total is } \sum f[i] \cdot \text{depth}(i) \]
\[ = 646 \text{ bits here} \]

How would ASCII do on these 170 letters?

\[ 170 \times 8 \]
Thm: Huffman codes are optimal: they use the fewest # of bits possible.

pf: Greedy—so how to start?

Contradiction—compare ours to some other optimal.
Lemmas: Let $x$ and $y$ be two least common characters.

There is an optimal tree in which $x$ and $y$ are siblings and have largest depth.

pf: Spps not:

Take opt tree $T$, where $x$ and $y$ are not siblings at deepest level.

Let $a$ and $b$ be $T$'s deepest siblings.

Create $T'$ by swapping $x$ with $a$.
\[
\text{cost}(T') = \text{cost}(T) - f[a] \cdot \Delta + f[x] \cdot \Delta
\]

Know \( f[x] \leq f[a] \)

Since \( x \) was least frequent

\( T \) was optimal, so \( \text{cost}(T') \) must not be better

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

\[ f[x] \geq f[a] \]

So can assume least freq. was a leaf.
Pf: (of them that Huffman codes are optimal)

Induction on the # of characters

Base case: \( n = 1 \) or \( 2 \)

IS: Take \( f[1..n] \). WLOG assume \( f[\hat{1}] + f[\hat{2}] \) are least frequent.

Create \( f[3..n+1] \) where \( f[n+1] = f[\hat{1}] \times f[\hat{2}] \).

By IH, Huffman tree of \( f[3..n+1] \) must be a best possible tree.
Claim: $T$ is optimal:

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

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

Since $f[n+1] = f[3] + f[2]$ and $\text{depth}(1) = \text{depth}(n+1) + 1$,

$$\Rightarrow \text{cost}(T) = \text{cost}(T') + f[3] + f[2]$$
Implementation: use priority queue

**BUILDHUFFMAN(f[1..n]):**

- for \( i \leftarrow 1 \) to \( n \)
  - \( L[i] \leftarrow 0 \); \( R[i] \leftarrow 0 \)
  - INSERT(\( i, f[i] \))

- for \( i \leftarrow n \) to \( 2n - 1 \)
  - \( x \leftarrow \text{EXTRACTMIN()} \)
  - \( y \leftarrow \text{EXTRACTMIN()} \)
  - \( f[i] \leftarrow f[x] + f[y] \)
  - \( L[i] \leftarrow x \); \( R[i] \leftarrow y \)
  - \( P[x] \leftarrow i \); \( P[y] \leftarrow i \)
  - INSERT(\( i, f[i] \))

- \( P[2n - 1] \leftarrow 0 \)

---

4 arrays \( L, R, P \):

- \( L[i] \) is left "pointer" of node \( i \)
- \( R[i] \) is right "pointer" of node \( i \)
- \( P[i] \) is parent of node \( i \)

Scan heap of new heap put in heap into my heap