CSCI 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

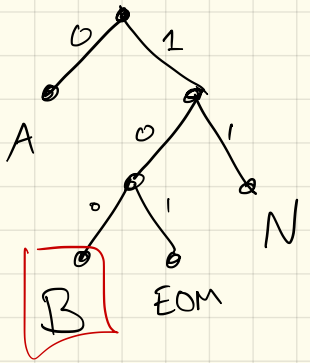
↳ $2^8 = 256$ letters

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.

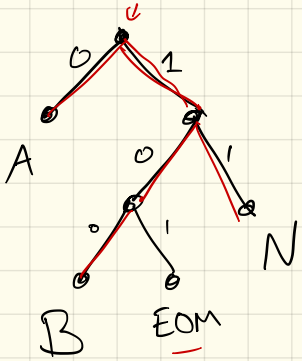
Encode: BAN

→ 100011

Decode:

1 0 0 0 1 1 0 1 1 0 1 0 1

EOM



B A N

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

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

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 & spaces (just to keep it simple), we get:

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 letters should be deeper (or shallower)?

(ie: How to be greedy?)

Put least frequent at bottom.

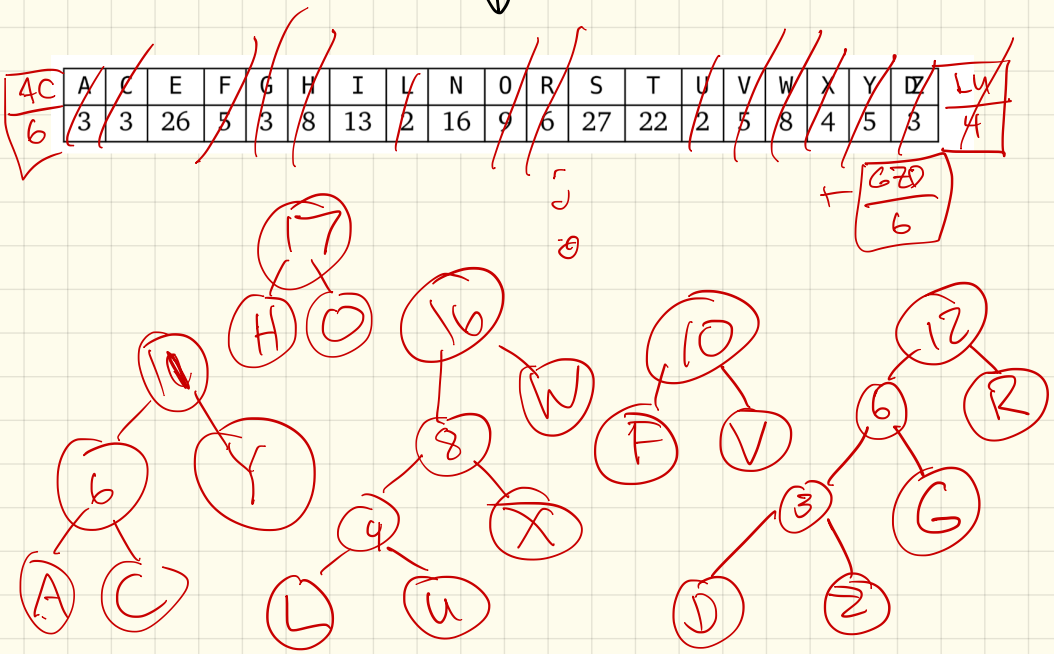
Huffman's alg:

Take the two least frequent characters.

Merge them in to 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

↓

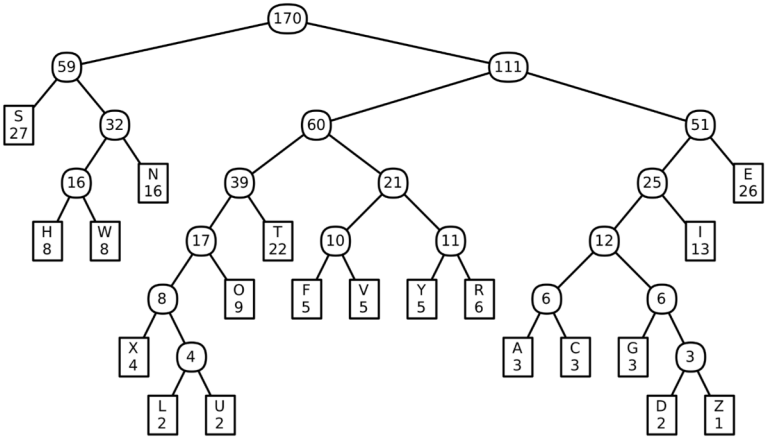


Example (cont):

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

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	…
T	H	I	S	S	E	N	T	E	N	C	E	C	O	N	T	A	

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

Total is $\sum f[i] \cdot \text{depth}(i)$

$= 646$ bits here

How would ASCII do on these 170 letters

170×8

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.

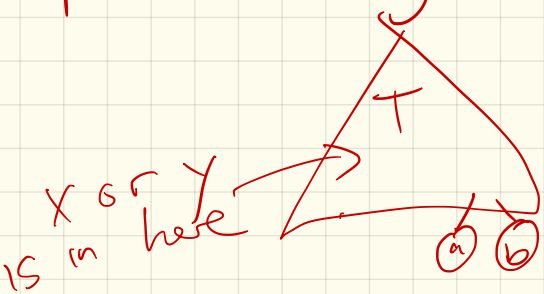
Lemma: Let x & y be 2 least common characters.

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

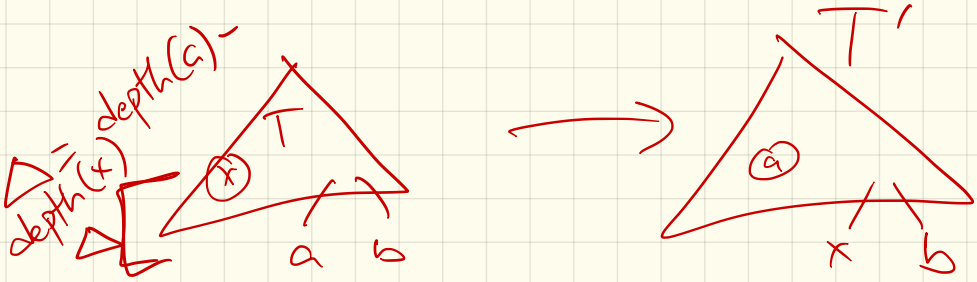
pf: Spps not:

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

Let a & b be T's deepest siblings.



Create T' by swapping x with a



$$cost(T') = cost(T) \underbrace{- f[a] \cdot \Delta + f[x] \cdot \Delta}$$

know $f[x] \leq f[a]$

since x was least frequent

T was optimal, so cost(T') must not be better

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

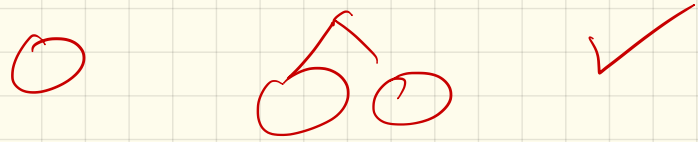
$\Rightarrow f[x] \geq f[a]$

So can assume least freq. was a leaf

Pf: (of thm that Huffman codes are optimal)

Induction on the # of characters

Base case: $n = 1$ or 2

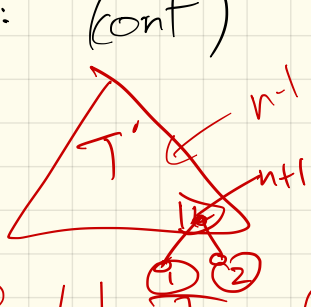


IS: Take $f[1..n]$. WLOG assume $f[1]$ & $f[2]$ are least frequent.

↳ Create $f[3..n+1]$ where $f[n+1] = f[1] + f[2]$.

By IH, Huffman tree T' of $f[3..n+1]$ must be a best possible tree.

Pf: (cont)



know $f[n+1]$ is a leaf of T'

Build T from T' by putting $1 \to 2$ back under it

Claim: T is optimal:

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

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

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

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

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

Implementation: use priority queue

```
BUILDHUFFMAN(f[1..n]):
  for i ← 1 to n
      L[i] ← 0; R[i] ← 0
      INSERT(i, f[i])

  for i ← n to 2n − 1
      x ← EXTRACTMIN( )
      y ← EXTRACTMIN( )
      f[i] ← f[x] + f[y]
      L[i] ← x; R[i] ← y
      P[x] ← i; P[y] ← i
      INSERT(i, f[i])

  P[2n − 1] ← 0
```

into my heap

put new sum of freq in heap

get 2 least freq. i's

+ 3 arrays L, R, P:

L[i] is left "pointer" of node i

right

parent