CS2100:

Huffman codes

Recap

Lab due Sunday ←

Reading for Zybook –
check later today (Graphs)

Next HW is posted

Final exam:

Wed at 2pm

↳ Last day of class:
review session

Huffman Codes - the idea:

We would like to transmit info using as few bits as possible.

What does ASCII do?

8-bit rep. of letters
↳ 256 characters

x letters ⇒ 8x bits

How can we do better?
↳ Well, what if we don't use all the characters?

fewer than 8 bits
↳ shorter

use fewer bits for more common letters

Problem:

If not fixed length, hard to tell when a character is finished.

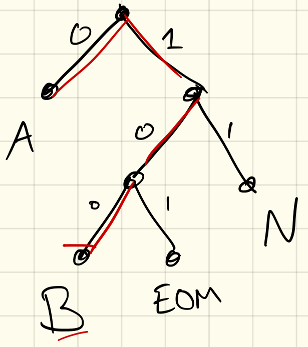
Ex:

E	11	2 bits
A	00	
S	01	
T	10	
R	110	3 bits
M	001	
B	010	
N	100	
⋮	etc.	

Decode: 11001

$$\left.\begin{array}{l}\underbrace{11}_{E}\,\underbrace{001}_{M} \\ \underbrace{110}_{R}\,\underbrace{01}_{S}\end{array}\right] \times$$

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.

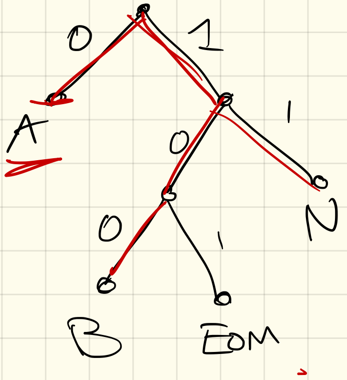
Encode: BAN

100011

Decode:

1000110110101

B A N A N A EOM

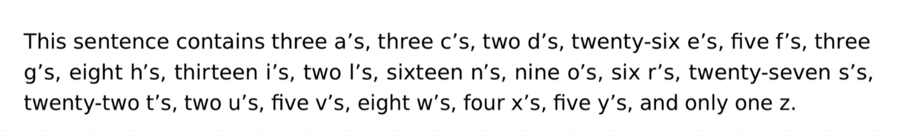


Bits : 13

ASCII: 6x8

How should we do this?

Use frequency counts to make a good prefix-free code (or tree):



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.

lower frequency
= more bits
↳ lower in tree

higher frequency
= fewer bits
↳ higher in tree

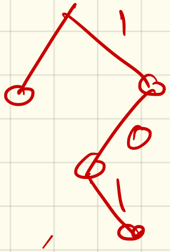
Goal: Minimize cost

↳ here, minimize total length of encoded message:

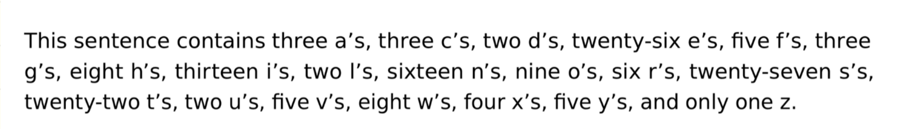
Input: frequency counts $f[1..n]$

Compute: tree with minimum "cost"

$$\text{cost}(T) = \sum_{i=1}^{n} f[i] \cdot \underbrace{\text{depth}(i)}_{\text{\# of bits to transmit}}$$



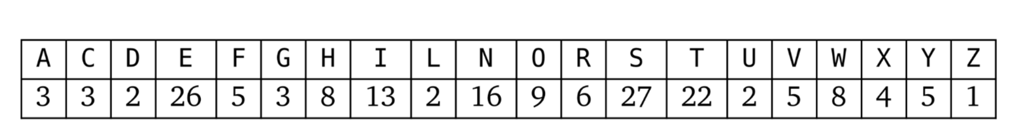
To do this, we'll need to use the array f:



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

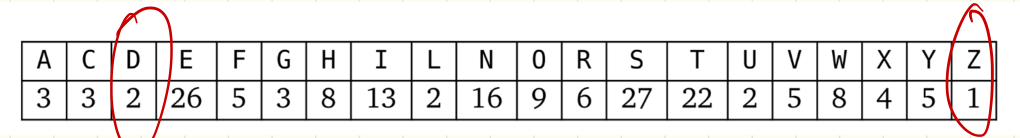
Which letters should be deeper (or shallower)?

(ie: How to be greedy?)

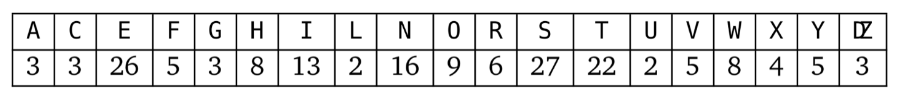
Huffman's alg:

Take the two least frequent characters.

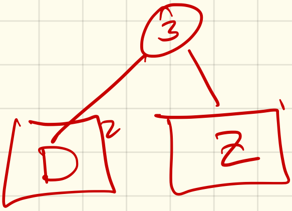
Merge them in to one letter, which becomes a new "leaf":



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



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

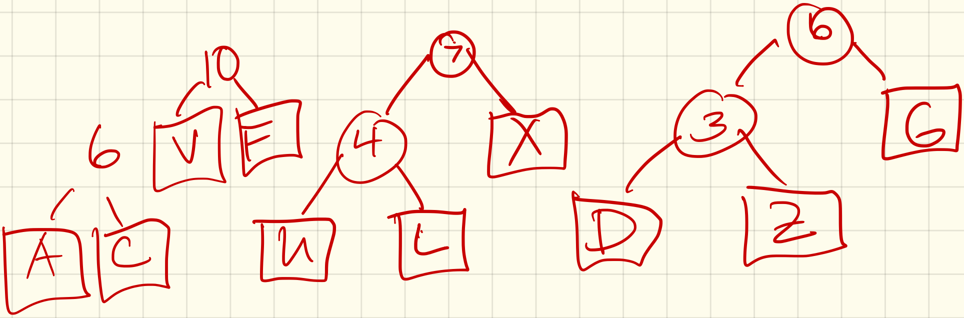


Example (cont):

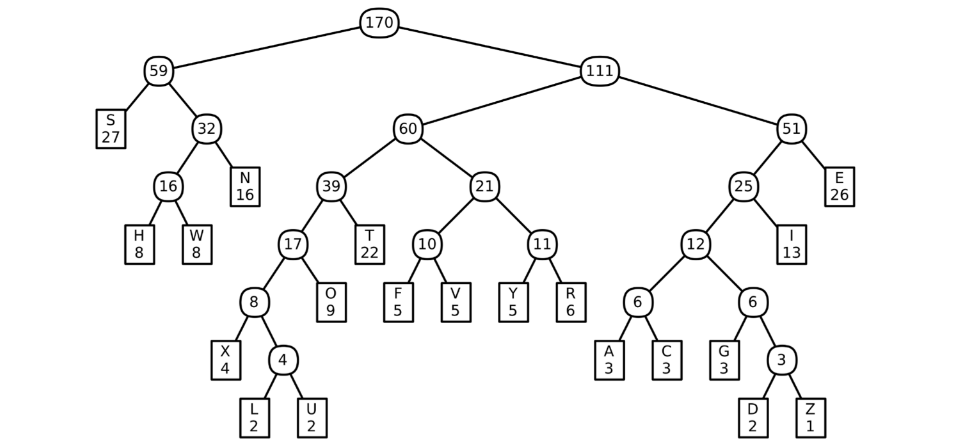
AC
6

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

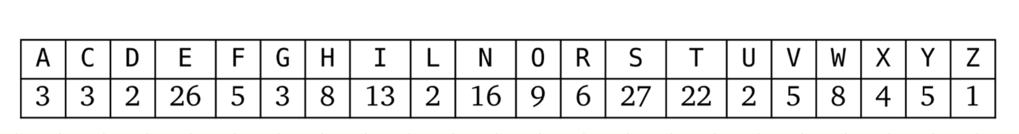
Lu
4



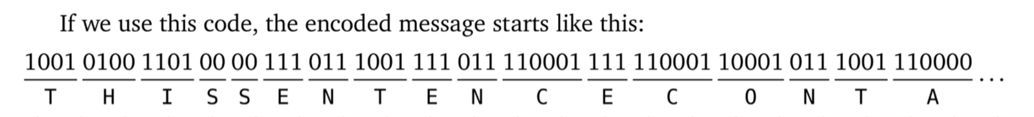
In the end, get a tree with letters at the leaves:



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



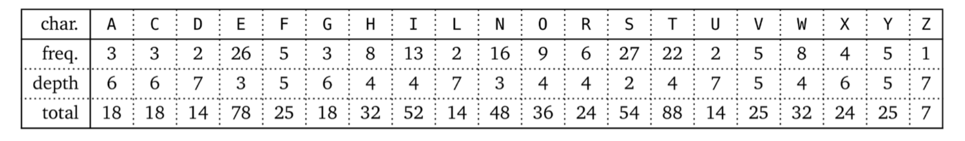
A	C	D	E	F	G	H	I	L	N	O	R	S	T	U	V	W	X	Y	Z
3	3	2	26	5	3	8	13	2	16	9	6	27	22	2	5	8	4	5	1



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 $\sum f[i] \cdot depth(i)$

$= 646$ bits here

How would ASCII do on these 170 letters

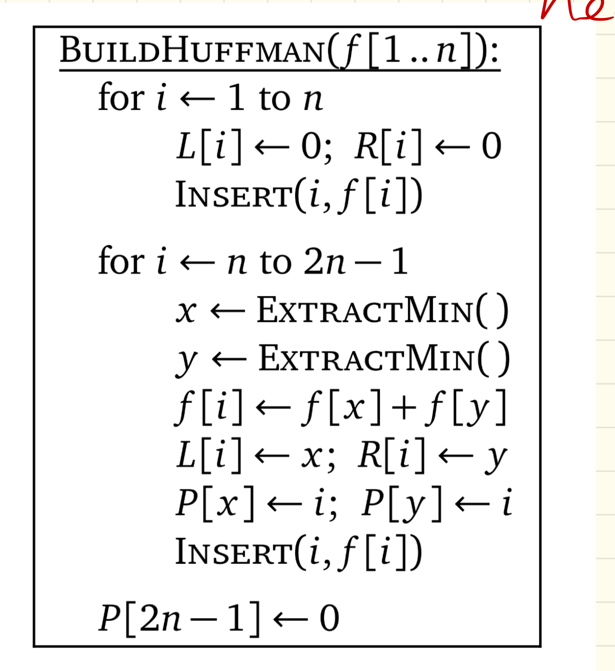
↳ $\underline{170 \times 8}$

Thm: Huffman codes are optimal: they use the fewest # of bits possible.

Pf: (go take 3100)

Side note: This is known as a greedy algorithm.

Implementation: use ~~priority queue~~ heap!



```
BuildHuffman($f[1..n]$):
    for $i \gets 1$ to $n$
        $L[i] \gets 0$;  $R[i] \gets 0$
        Insert($i, f[i]$)

    for $i \gets n$ to $2n-1$
        $x \gets$ ExtractMin()
        $y \gets$ ExtractMin()
        $f[i] \gets f[x] + f[y]$
        $L[i] \gets x$;  $R[i] \gets y$
        $P[x] \gets i$;  $P[y] \gets i$
        Insert($i, f[i]$)

    $P[2n-1] \gets 0$
```

+ 3 arrays L, R, P:

(go take algorithms)

Next HW:

decode:

Given an input which describes a tree & a message:

1) Create the tree

2) Use it to decode the message.

One thing I skipped: do need to store the tree.

Overview of assignment...