CS180 - Huffman trees

Note Title 11/16/2011

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.

$2^8 = 256$ characters total

Standard ASCII : 8 bits per character

Hello : 5×8 bits total

Extended ASCII : 64 bits ?

So – how can we do better?

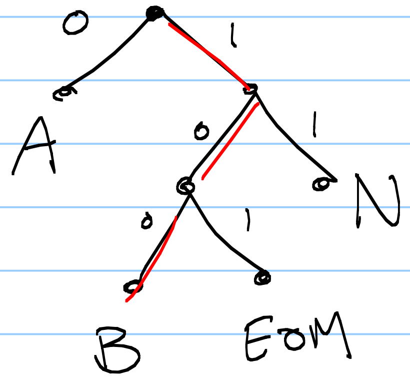
What if we don't use every character?

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.

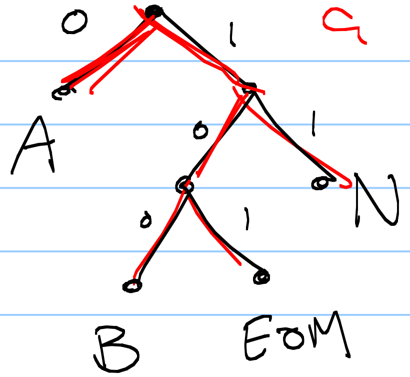
Encode: BAN

100011

Decode:

1000 11011 0101

B A N A N A end of message



how do we get a good tree?

Even though each letter is different length, scan & use the tree to detect letters.

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	

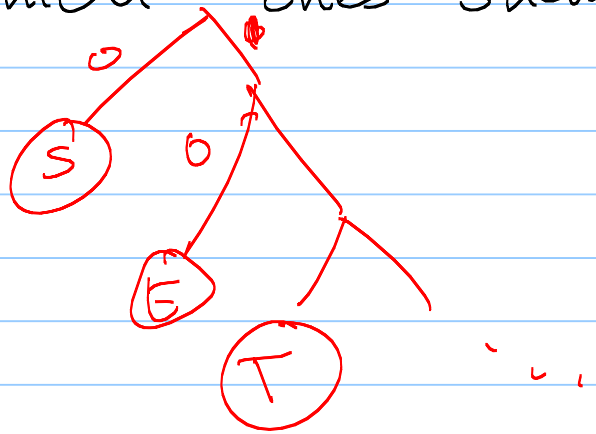
pull exact letter counts

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

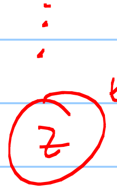
Which ones should get few bits?

many bits



build up

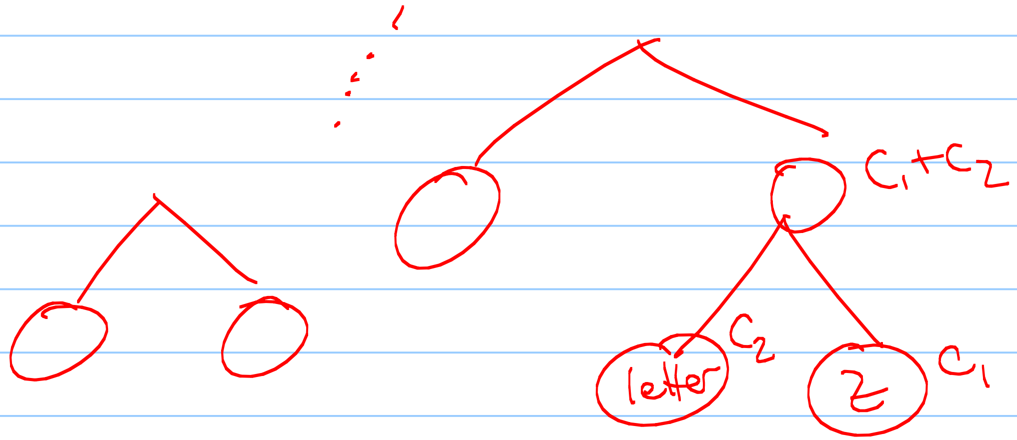
bottom leaf



Huffman's algorithm

Take the two least frequent characters.

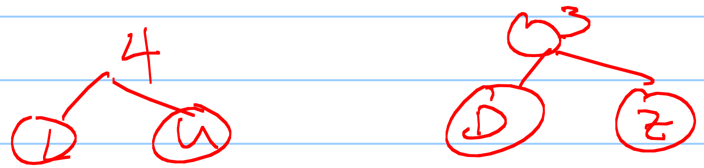
Merge them into 1 letter, which becomes a new "leaf".



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:



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

Merge L & U

D+Z, together have freq. 1+2

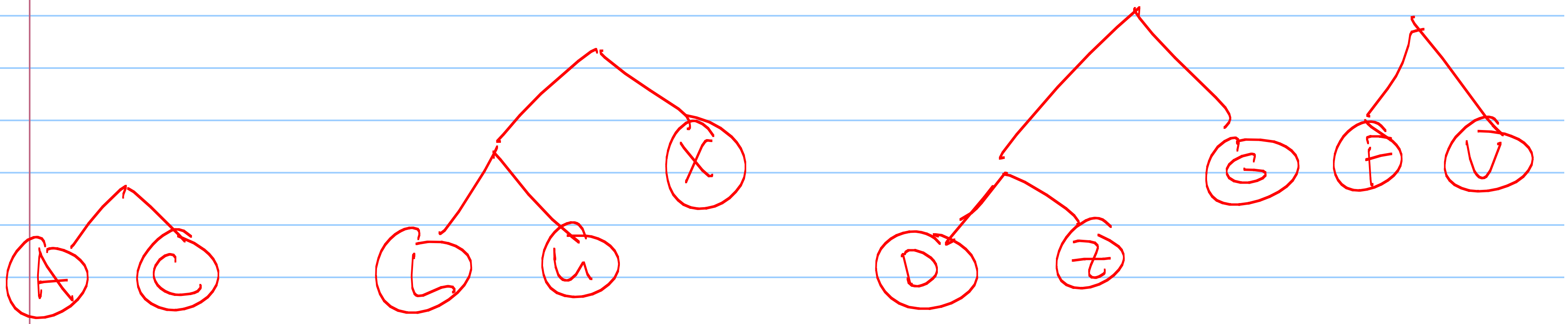
~~A~~	~~C~~	E	~~F~~	~~G~~	H	I	~~L~~	N	O	R	S	T	~~U~~	~~V~~	W	~~X~~	Y	~~DZ~~	~~LU~~	AC	GDZ
~~3~~	~~3~~	26	~~5~~	~~3~~	8	13	~~2~~	16	9	6	27	22	~~2~~	~~5~~	8	~~4~~	5	~~3~~	~~4~~	6	6

XLU	FV
8	10

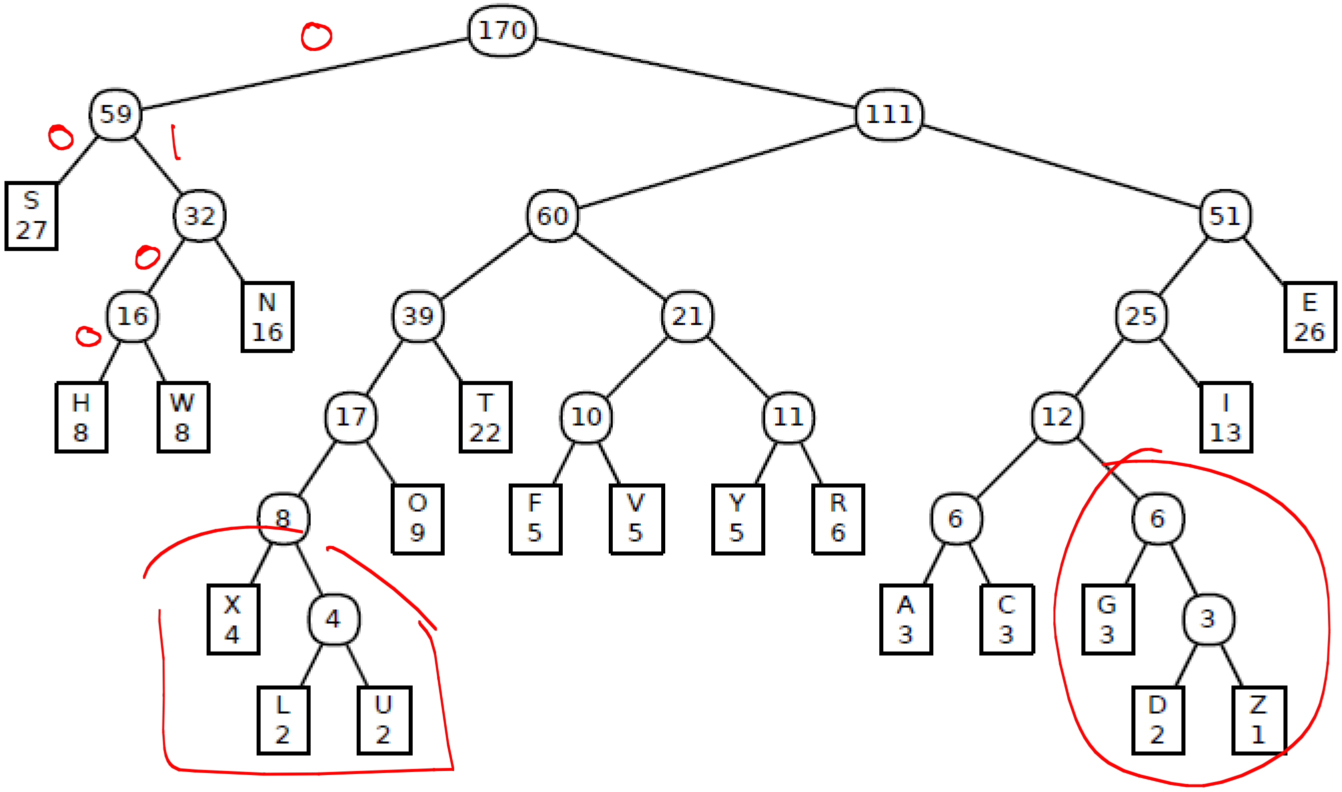
Next?

A & C

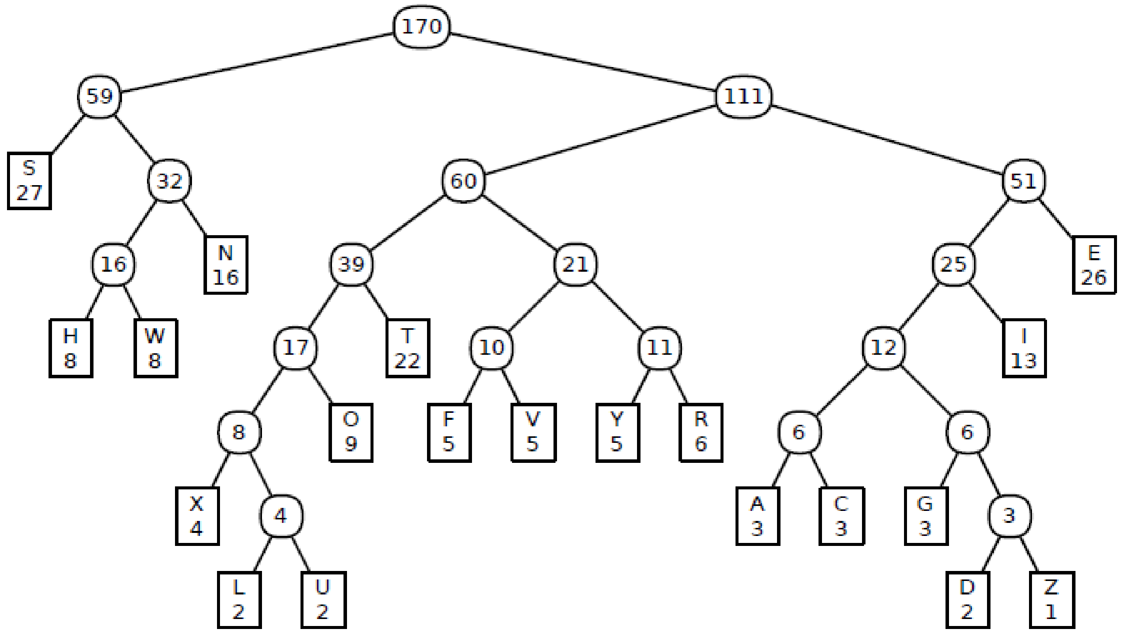
G & DZ



In end, build a tree:

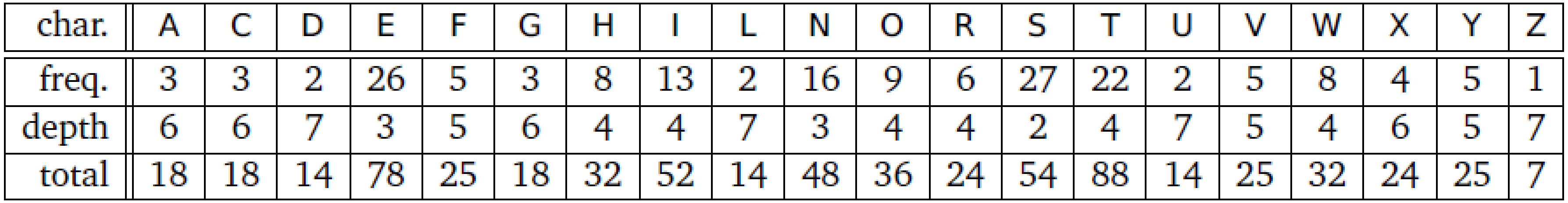


Using the tree:



1001	0100	1101	00	00	111	011	1001	111	011	110001	111	110001	10001	011	1001	110000	1101	...
T	H	I	S	S	E	N	T	E	N	C	E	C	O	N	T	A	I	

How many bits?



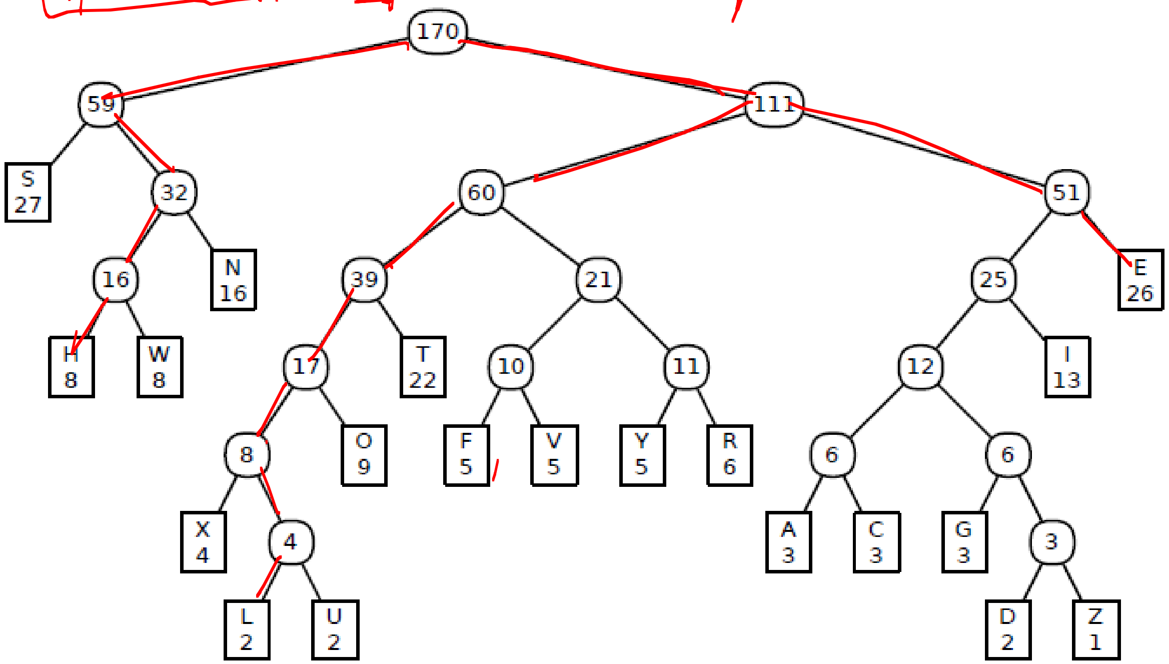
char.	A	C	D	E	F	G	H	I	L	N	O	R	S	T	U	V	W	X	Y	Z
freq.	3	3	2	26	5	3	8	13	2	16	9	6	27	22	2	5	8	4	5	1
depth	6	6	7	3	5	6	4	4	7	3	4	4	2	4	7	5	4	6	5	7
total	18	18	14	78	25	18	32	52	14	48	36	24	54	88	14	25	32	24	25	7

total = 646 bits

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

170 x 8 (bigger)

Exercise: 01001111000010100001010001



Message? H E L L O

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.

(Go 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