CS 314 - Huffman Codes

Note Title 11/16/2011

Announcements

- HW due Monday

Idea

We want to transmit information using as few bits as possible.

Standard ASCII: 8 bits

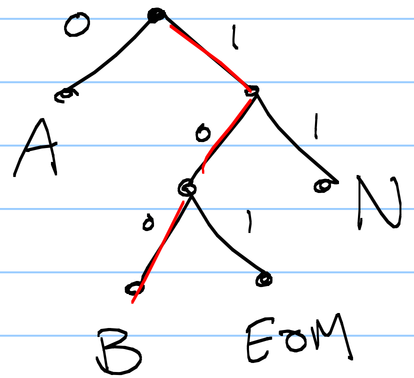
(Extended: 16 bit)

So – how can we do better?

What if we don't use every character?

↳ assign more frequent characters a string of fewer bits

Prefix-free codes



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

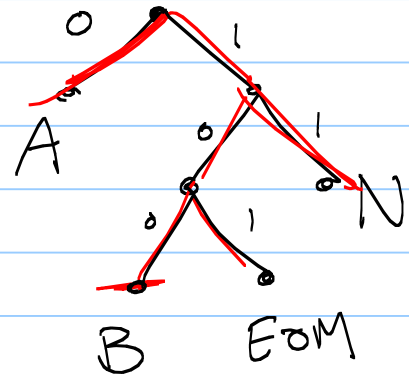
No letter's code is the prefix of another letter.

Encode: BAN
100011

Decode:

1000011011 0101

BANANA



Goal: minimize cost

Here, minimize total length of encoded message:

$$\text{cost}(T) = \sum_{i=1}^{n} \underbrace{f[i]}_{\substack{\text{frequency}\\ \text{count}}} \cdot \underbrace{\text{depth}(i)}_{\text{in a tree}}$$

Input: $f[1..n]$

So how do we do this? With exact frequency counts!

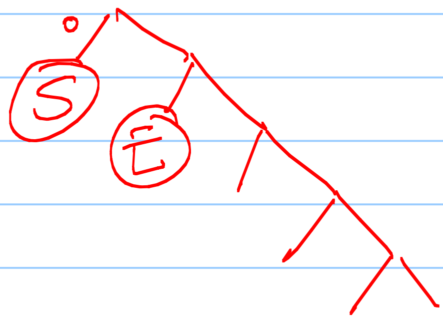
This sentence contains three a's, three c's, two d's, twenty-six e's, five f's, three g's, eight h's, thirteen i's, two l's, sixteen n's, nine o's, six r's, twenty-seven s's, twenty-two t's, two u's, five v's, eight w's, four x's, five y's, and only one z.

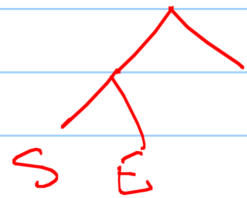
A	C	D	E	...
3	3	2	26	

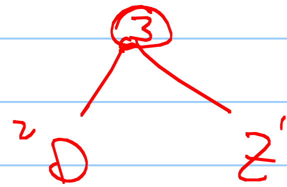
Using frequency counts, build one of those trees.

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

```
BUILDHUFFMAN(f[1..n]):
    for i ← 1 to n
        L[i] ← 0; R[i] ← 0
        INSERT(i, f[i])          ← log n
    for i ← n to 2n − 1
        x ← EXTRACTMIN()
        y ← EXTRACTMIN()         ] heap O(log n)
        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
```

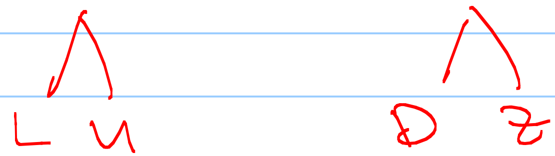
Q: Which data structure do we need?

Example:

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

Merge D & Z:

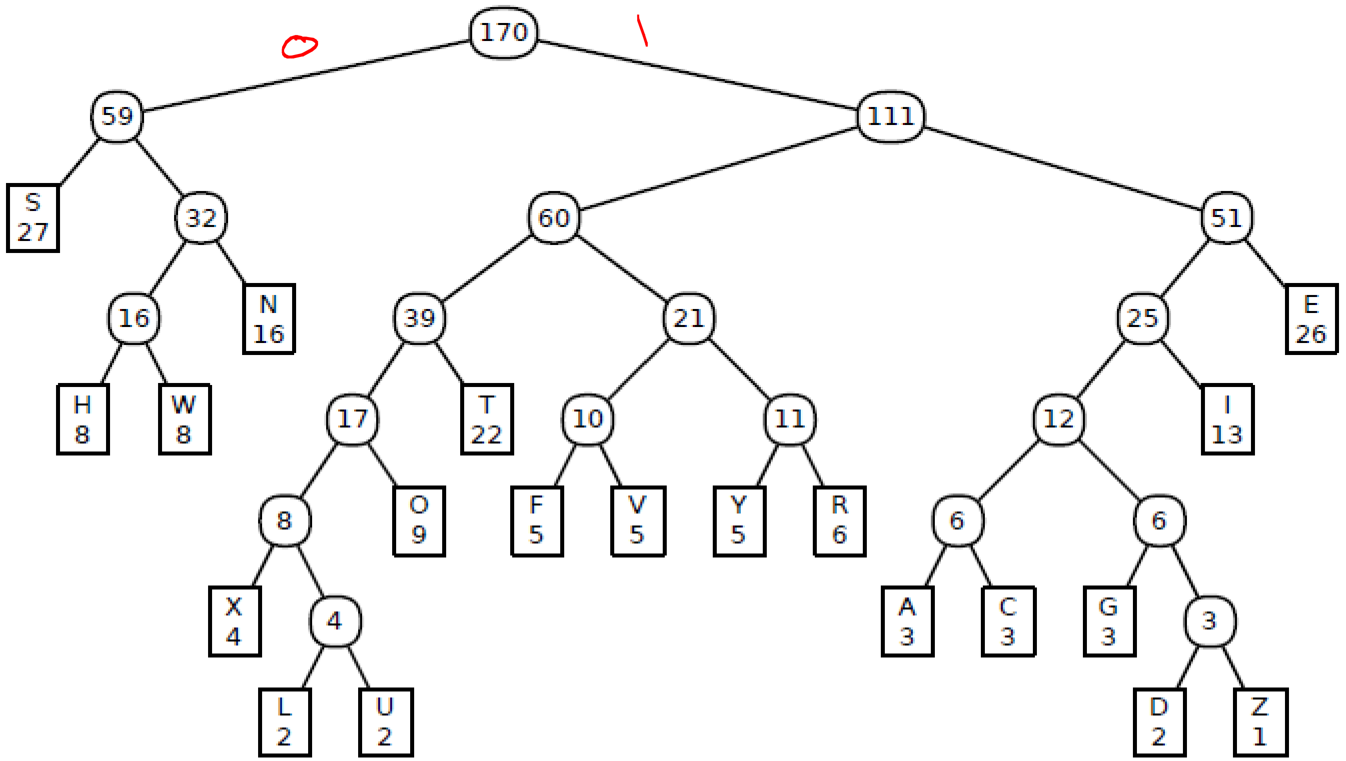
A	C	E	F	G	H	I	~~L~~	N	O	R	S	T	~~U~~	V	W	X	Y	DZ	LU
3	3	26	5	3	8	13	~~2~~	16	9	6	27	22	~~2~~	5	8	4	5	3	4



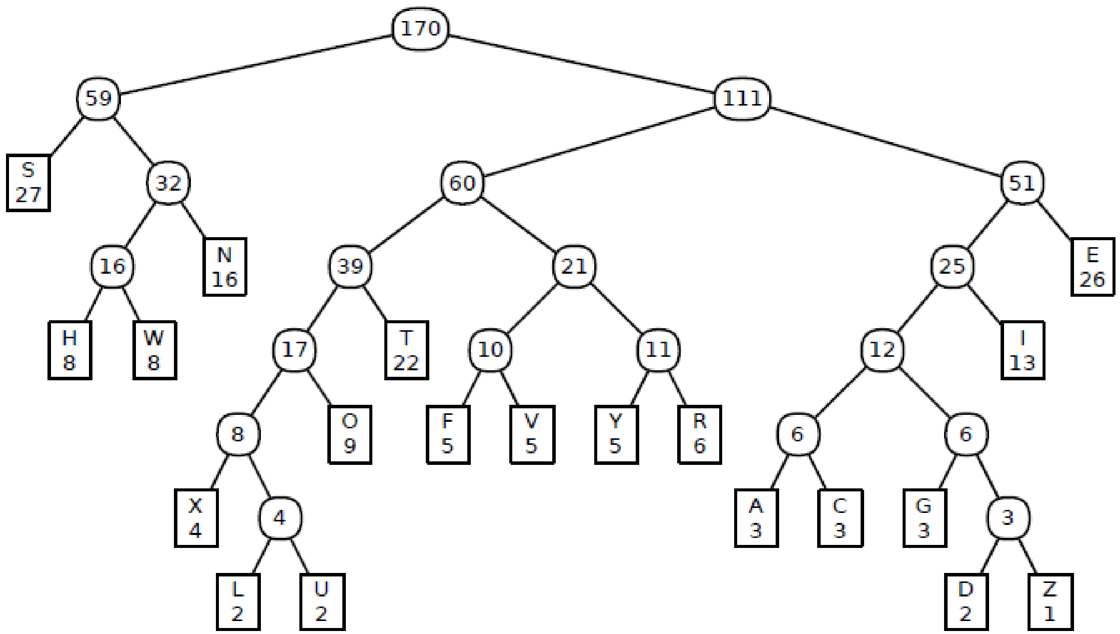
A	C	E	F	G	H	I	L	N	O	R	S	T	U	V	W	X	Y	DZ
3	3	26	5	3	8	13	2	16	9	6	27	22	2	5	8	4	5	3

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	...
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

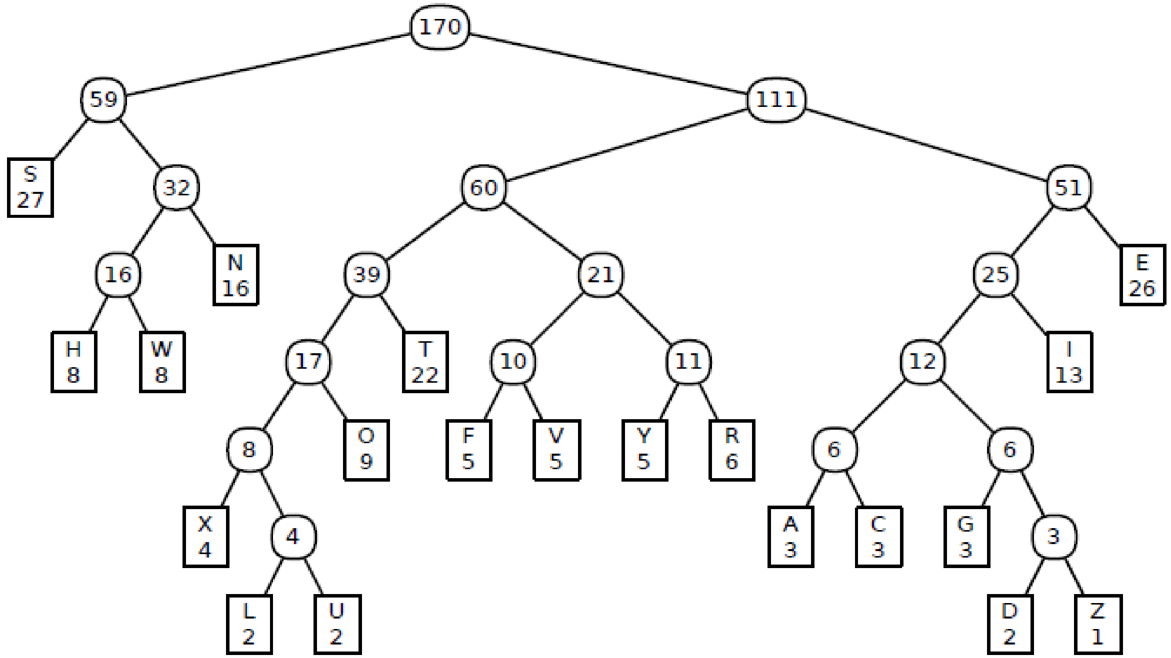
$$\text{Cost} = \sum_{i=1}^{\#\text{letters}} f[i] \cdot \text{depth}[i]$$

total = 646 bits

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

170×8

Exercise: 01001111000010100001010001



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?

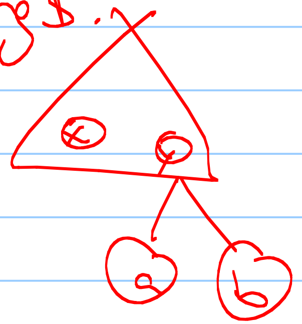
Compare to "OPT", & show ours is just as good.

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.

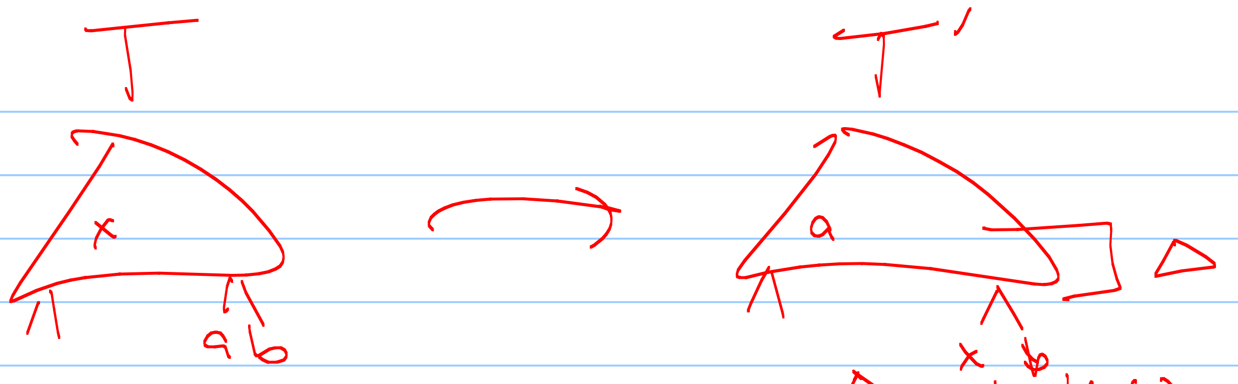
Take optimal tree T, where x & y are not siblings at largest depth.

Let a & b be deepest siblings.



Create T' by swapping x and a

cont:



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

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

know $\underline{f[x] \le f[a]}$ since a is more frequent.

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

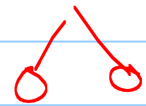
$\Rightarrow \Delta(f[x] - f[a]) \ge 0 \Rightarrow f[x] \ge f[a]$

$\Longrightarrow f[x] = f[a]$.

Pf that Huffman codes are optimal:

induction on # of characters:

Base case: $n=1$ or 2 obvious



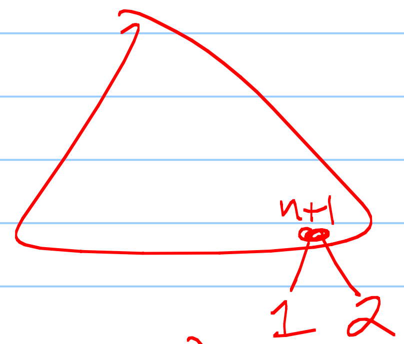
IS: Take $f[1..n]$. WLOG assume $f[1]$ & $f[2]$ are least frequent.

Let $f[n+1] = f[1] + f[2]$.

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

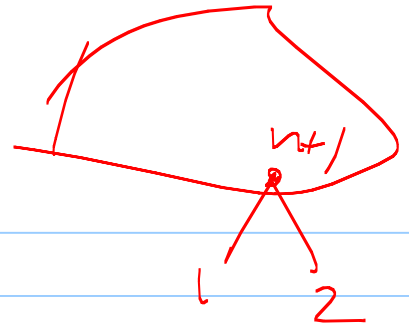
create T →



Claim: T is optimal.

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

$$= \underbrace{\sum_{i=3}^{n+1} f[i] \cdot \text{depth}(i)}_{\text{cost}(T')} + f[1] \cdot \text{depth}(1) + f[2] \cdot \text{depth}(2) - f[n+1] \cdot \text{depth}(n+1)$$



$$\mathrm{cost}(T) = \quad \cdots$$

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

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

$$\because f[n+1] = f[1]+f[2]$$

$$\Longrightarrow \quad = \mathrm{cost}(T') + f[1] + f[2]$$