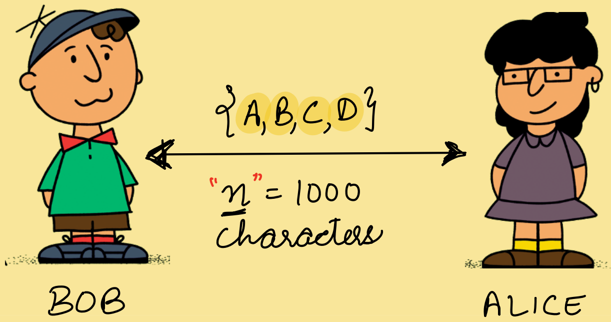
COL 351: Analysis and Design of Algorithms

Lecture 6

MESSAGE ENCODING



FREQUENCY TABLE

A	40%
B	10%
C	20%
D	30%

msg C A B A D . . . .

enco 10 00 01 00 11 . . .

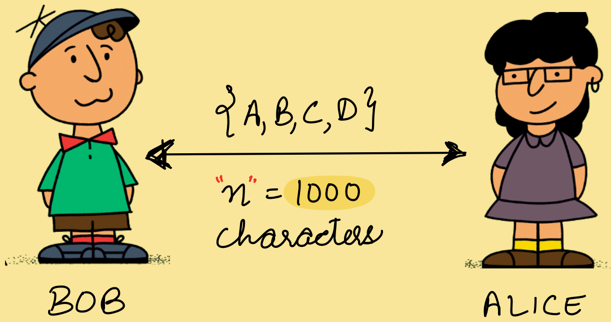
Natural approach:

A	00
B	01
C	10
D	11

msg length = 2000

Question: Can we compress beyond 2000 characters?

MESSAGE ENCODING



FREQUENCY TABLE

A	40%
B	10%
C	20%
D	30%

Way 2:

A	0
B	100
C	101
D	11

length = 400(1) + 100(3) + 200(3) + 300(2) = 1900

Ques: Can there be ambiguity in decoding?

eg. C A B A D

encode ⇒ 1 0 1 0 1 0 0 0 1 1

C A B A D

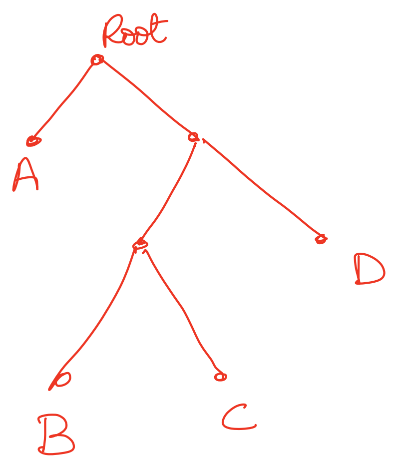
PREFIX FREE ENCODING

Def^n: If $(x_1, \ldots, x_k)$ is code-word then NO prefix of it can be code-word.

Example:

A	0
B	100
C	101
D	11

Binary Tree Representation



Remark: Symbols can only be leaf nodes.

PROBLEM

Given: symbols $(a_1 \cdots a_n)$ with frequency $F = (f_1 \cdots f_n)$

Find: Prefix free encoding for which "encoded-msg" has MINIMUM length.

Equivalently, find a Binary tree T _ _ _ that minimizes:

$$\sum_{i=1}^{n} f_i * depth(a_i, T).$$

Property

Lemma: If symbols $a_1 \dots a_n$ satisfy $f_1 \geq f_2 \geq \dots \geq f_n$. Then

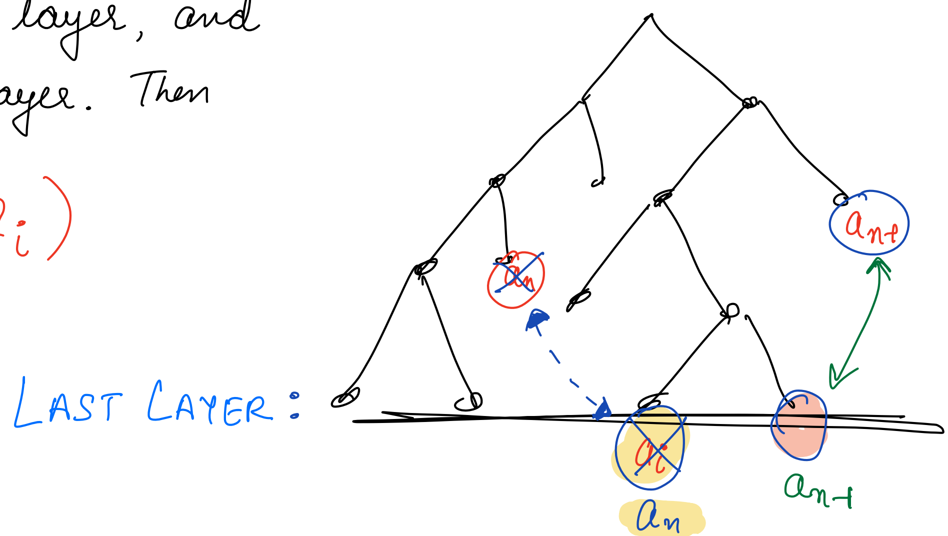
(i) $\exists$ opt tree where a_n has maximum depth.

(ii) $\exists$ opt tree where $\boxed{a_n, a_{n-1} \text{ are siblings}}$

Proof Idea:

→ If a_n not in last layer, and a_i is in last layer. Then swap them.

(Recall $f_n \leq f_i$)



Example

F

A	40%
B	10%
C	20%
D	30%

size n

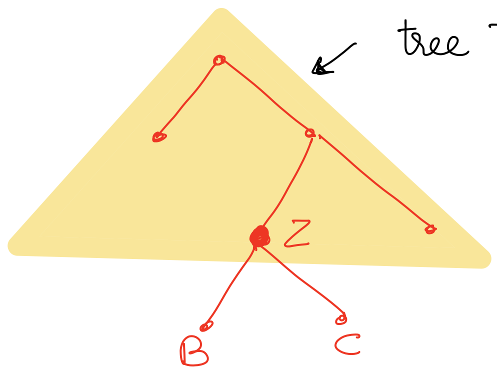
$\rightarrow$

New Freq Table (F^*)

A	40%
Z	30%
D	30%

size $(n-1)$

Ques: How to get this remaining tree T^*?



$$(a_1 \quad a_2 \quad \ldots \quad a_{n-2} \quad \underbrace{a_{n-1} \quad a_n}_{\Downarrow\; Z = \{a_{n-1},\, a_n\}})$$

$$\text{old-opt} = \sum_{i=1}^{n-2} f_i \,\text{depth}(a_i, T) + f_{n-1} \cdot \overbrace{\text{depth}(a_{n-1}, T)}^{\text{" } L \text{ (say)}} + f_n \cdot \overbrace{\text{depth}(a_n, T)}^{\text{" } L \text{ (say)}}$$

$$= \text{———"———} \quad + (f_{n-1} + f_n) \cdot L$$

$$= \underbrace{\text{—"—} \quad + (f_{n-1} + f_n) \cdot \text{depth}(Z, T^*)}_{\text{Should be new-opt}} + f_n + f_{n-1}$$

Theorem: Consider a problem instance $F = (f_1 \ldots f_n)$.

Let $i, j \leq n$ satisfy that $\exists$ opt tree in which a_i & a_j are siblings.

Then for NEW problem $F^* = (F \setminus \{f_i, f_j\}) + f^*$ where $f^* = f_i + f_j$,

$$\boxed{opt(F) = opt(F^*) + f_i + f_j}$$

Theorem: Consider a problem instance $F = (f_1 \dots f_n)$.

Let $i, j \le n$ satisfy that $\exists$ opt tree in which a_i & a_j are siblings.

Then for NEW problem $F^* = (F \setminus \{f_i, f_j\}) + \tilde{f}$ where $f^* = f_i + f_j$,

$$\boxed{opt(F) = opt(F^*) + f_i + f_j}$$

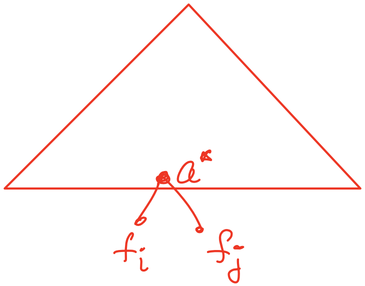
Proof

Part (i) $opt(F) \le opt(F^*) + f_i + f_j$

Let $T^* = \text{opt sol}^n(F^*)$, and a^* be node of freq $f^* = f_i + f_j$.

Add children a_i + a_j to a^* to get tree T.

$$\sum_{k=1}^{n} f_k \cdot depth(a_k, T) = opt(F^*) + f_i + f_j$$



So, $opt(F) \le opt(F^*) + f_i + f_j$

Theorem: Consider a problem instance $F = (f_1 \ldots f_n)$.

Let $i, j \leq n$ satisfy that $\exists$ opt tree in which a_i & a_j are siblings.

Then for NEW problem $F^* = (F \setminus \{f_i, f_j\}) + \tilde{f}$ where $\tilde{f} = f_i + f_j$,

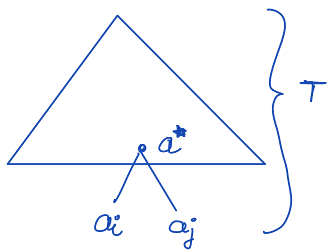
$$\boxed{opt(F) = opt(F^*) + f_i + f_j}$$

Proof

Part (ii) $\quad opt(F^*) \leq opt(F) - (f_i + f_j)$

Take an opt tree T for F in which a_i and a_j are siblings.

Mark $parent(a_i, a_j, T)$ as a^*, and let $T^* = T \setminus \{a_i, a_j\}$.



Set $freq(a^*) = f_i + f_j$.

Then T^* is valid solⁿ for F^*.

Now observe $opt(F^*) \leq \sum_{\substack{\text{leaves } v \\ \text{in } T}} depth(v, T^*) \cdot Freq(v) = opt(F) - (f_i + f_j)$ (Why?)

This proves the claim.

Huffman Encoding

① Replace symbols a_n, a_{n-1} (the letters with least frequency) by new symbol "a^*"

② $f^* := f_n + f_{n-1}$ be freq of a^*.

③ Recursively solve $\{a_1, a_2, \ldots, a_{n-2}, a^*\}$, and find opt tree T^*

④ Add a_{n-1} & a_n as children of a^* in the tree T^*, and return.

H.W. – Can you get an $O(n \log n)$ time implementation?

CHALLENGE PROBLEMS

→ Can you find an $O(n \log n)$ time implementation for Huffmann encoding?

→ What if we have additional constraint that ALL code words have length $\leq$ "L". $=50$

Can you solve this in $O(n^{c})$ time?