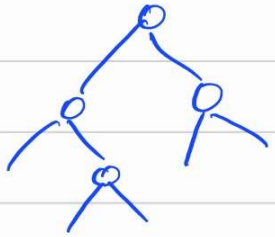
Goal now is to come up with a binary tree with as small avg. depth as possible. wrt frequencies.

—x —x —x —x

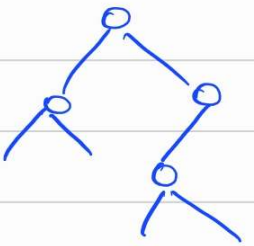
21/02/2022

Prefix-free codes ⟷ Binary trees.

Defn :- A binary tree is called full if every non-leaf node has exactly two children.



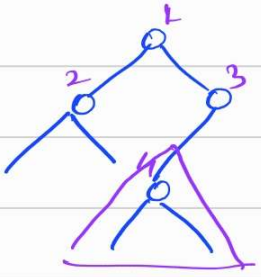
full binary tree

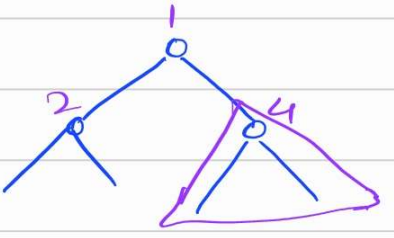


NOT a full binary tree

Lemma :- The binary tree corresponding to an optimal prefix-free code is full.

Proof: Suppose not.





this tree has strictly lower avg. depth

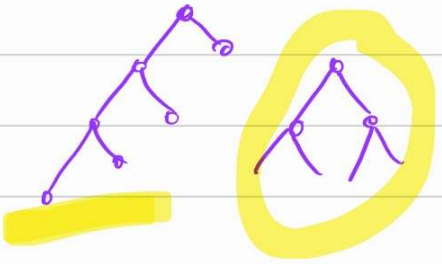
∎

Shannon – Fano Codes:

alphabet = $\{a_1, \ldots, a_k\} := A$

frequencies = $\{f(a_1), \ldots, f(a_k)\}$

$$\sum_{i=1}^{k} f(a_i) = 1.$$



A partition

$\{a_{i_1}, a_{i_2}, \ldots\} := A_1$ $\quad$ $A_2 := \{a_{j_1}, a_{j_2} \ldots\}$

s.t. $\sum$ frequencies in A_1 = $\sum$ frequencies in A_2

if equal partition is not possible then partition them as "nearly" as possible.

↳ sort frequencies in decreasing order

$\left| \sum_{i=1}^{s} f(a_i) - \sum_{i=s+1}^{k} f(a_i) \right|$ is minimized

$A = \{a, b, c, d, e\}$

frequency = $f(a) = 0.32$, $f(b) = 0.25$

$f(c) = 0.20$, $f(d) = 0.18$

$f(e) = 0.05$.

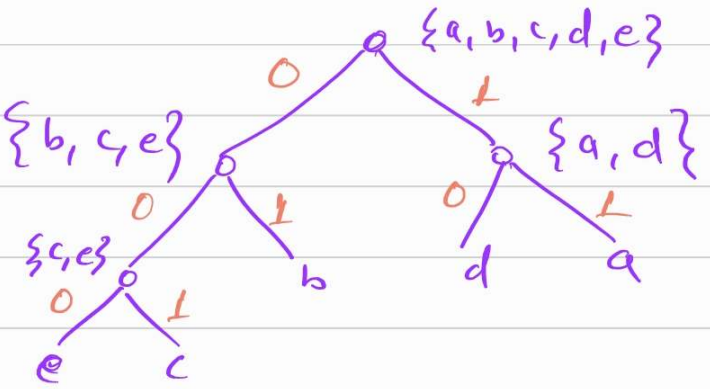
$\Gamma_2 : A \to \{0,1\}^3$

$\Gamma_2(a) = 11$, $\Gamma_2(b) = 01$, $\Gamma_2(c) = 001$

$\Gamma_2(d) = 10$, $\Gamma_2(e) = 000$.

Applying Shannon-Fano codes to above example.

$A_1 = \{a, d\}$ $\quad$ $A_2 = \{b, c, e\}$



$\Gamma_3 : A \rightarrow \{0,1\}^3$

$\Gamma_3(a) = \underline{11}$, $\Gamma_3(b) = \underline{10}$ $\Gamma_3(c) = \underline{01}$

$\Gamma_3(d) = \underline{001}$, $\Gamma_3(e) = \underline{000}$

avg. encoding length per letter

$= 2 \cdot 0.32 + 2 \cdot 0.25 + 2 \cdot 0.20 + 3 \cdot 0.18 + 3 \cdot 0.05$

$= 0.64 + 0.50 + 0.40 + 0.54 + 0.15 = 2.23$

0 1 11 10 $\quad$ 01 $\quad$ 011

c

Huffman Code:

Suppose the optimal tree structure is given to you.

Now you need to label the leaves of this tree.

Lemma:- Suppose T^* is an optimal tree for a prefix-free code.
Let u, v be two leaves in T^* s.t. depth(u) < depth(v). Then

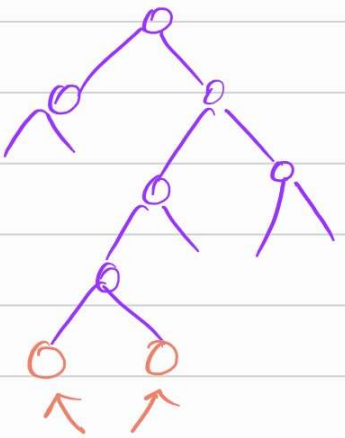
$$f(a_u) \geq f(a_v).$$

Proof:- Suppose not. Then exchange the labels of u and v. and this new tree has smaller avg. depth.

labeling leaves of an optimal tree:

→ leaves at depth 1 (if any) are labeled with highest-frequency letter

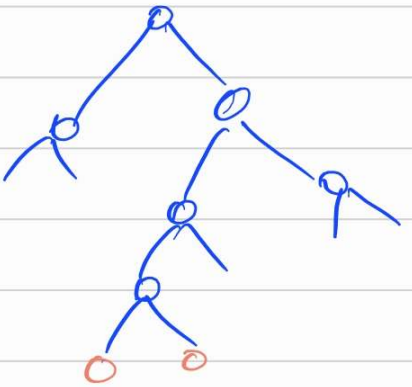
→ leaves at depth 2 and so on.



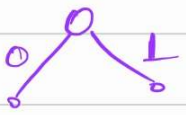
two least-frequent letters

Lemma :- There is an optimal tree T^* s.t. the two least frequent letters label two leaves that are siblings and are at the maximum depth.

Proof :- Start with any optimal tree. Suppose this structure was not present then you exchange labels.



Huffman's Algorithm

→ if $|A| = 2$ then 

→ find two least frequent letters, say, a and b

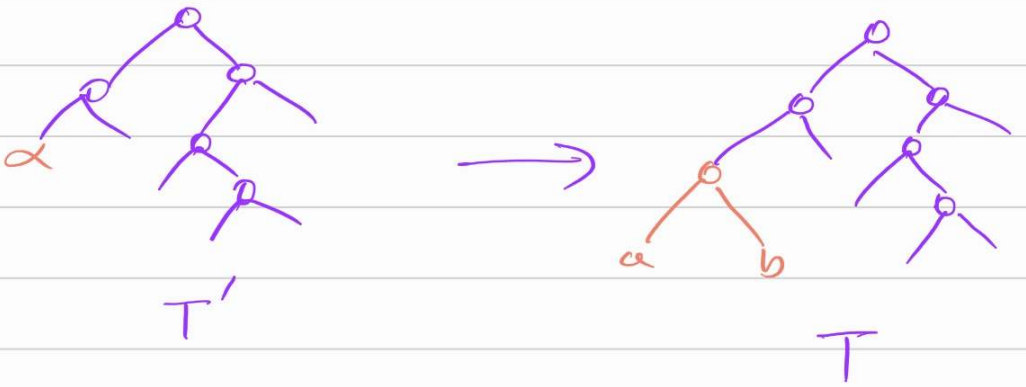
define a new letter α with frequency $f(\alpha) = f(a) + f(b)$

$A' := A \setminus \{a, b\} \cup \{\alpha\}$

$|A'| = |A| - 1$

→ recurse on A'.

→ Let T' be an optimal tree for A'.



T' $\qquad\qquad$ T

extend the leaf labelled d as shown above.

$f(a) = 0.32$
$f(b) = 0.25$
$f(c) = 0.20$
$f(d) = 0.18$
$f(e) = 0.05$

$\longrightarrow$

$f(a) = 0.32$
$f(b) = 0.25$
$f((de)) = 0.23$
$f(c) = 0.20$

$\downarrow$

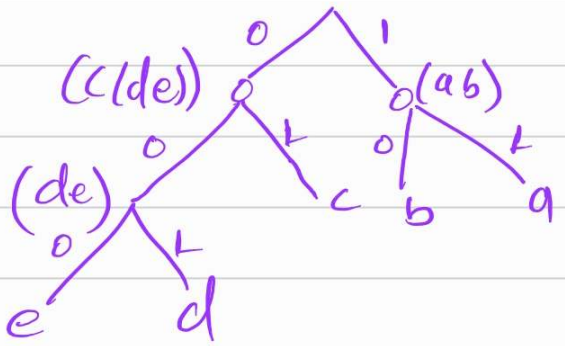
$f(a) = 0.32$
$f(b) = 0.25$
$f((c(de))) = 0.43$

$\longleftarrow$

$f((ab)) = 0.57$

$f((c(de))) = 0.43$

$f((ab)) = 0.57$



$\equiv T_3$

Lemma:- Huffman's algorithm produces a prefix-free code that has minimal average length encoding per letter.

Proof:-

Base Case: alphabet of size 2.

Induction hypothesis: produces Optimal tree for alphabet of size k-1.

Induction Step: Consider an alphabet A of size k.

A is of size k $\xrightarrow{\text{algorith}}$ A' of size k-1

T for A. $\longleftarrow$ T' optimal for A'

Claim:- average depth of T = avg. depth of T' + the sum of two least frequencies.

Implementation :

→ $O(K)$ time to find the two least frequencies and add the new letter.

→ $K-1$ times.

→ $O(K^2)$ — worst case time.

better implementation : priority queue

$O(K \log K)$.