Binary Coding of files

- Suppose a file F has m letters & there are n alphabets in the language.

- Qn: How large is the binary coding?

- Ans: At least $m \cdot \lg n$.

- Additional assumption: Suppose we know the frequency distribution of the alphabets in F.

  Can we use the distribution to have a smaller coding of the file?

- Eg. in English texts 'e' appears 13%, 't' appears 9%, but 'j' 'q' 'x' 'z' appear $< \frac{1}{2}\%$!

- Idea: More frequent alphabets should map to shorter strings.

- Eg. Alphabet A: a b c d e

Frequency f:	0.45	0.18	0.15	0.12	0.10
Coding γ:	0	10	110	101	111

- This gives an average bit length ABL of:

$0.45 \times 1 + 0.18 \times 2 + (0.15 + 0.12 + 0.10) \times 3$

$= 1.92 = \sum_{a \in A} f(a) \cdot |\gamma(a)|$

$\Rightarrow$ A file of size m has an encoding of size $\approx 1.92\,m$, which is smaller than $3m$.

- But, this coding has ambiguity.

01010111 is abbe

adae

▷ 'b' is a prefix of 'd'.

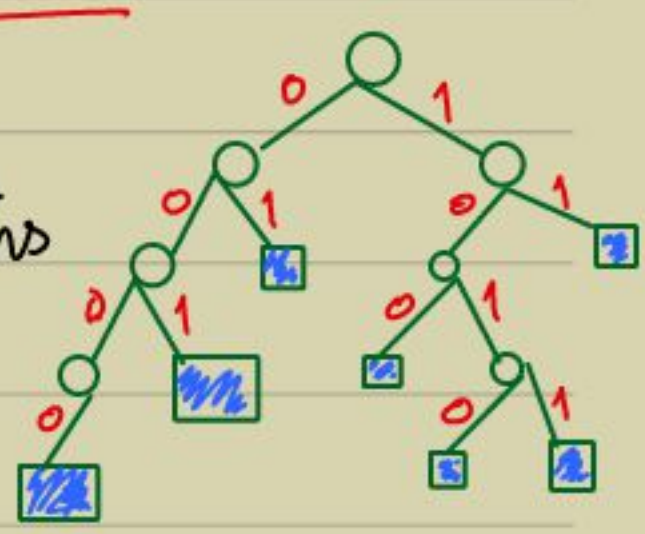
- Prefix coding: If $\nexists\, x \neq y \in A$ s.t. $\gamma(x)$ is a prefix of $\gamma(y)$.

- Algorithmic problem:
  Given A of n alphabets with their frequencies, compute a prefix encoding γ s.t. ABL(γ) is minimum.

- Brute-force: A naive algorithm would go over all the permutations of n.
  $\Rightarrow \Omega(n!)$ time taken.

- Instead, we can model a prefix code as a labeled binary tree.

- Eg. the blue leaves' paths form a prefix code!



(Exercise)

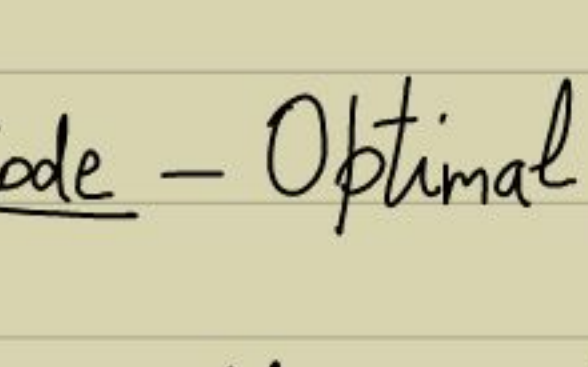
Huffman Code – Optimal prefix code

- We make the following observations about the labelled binary tree T of the optimal prefix code γ.

Lemma 1: T must be a full binary tree.

Proof:

- If there is a node with out-degree ≤ 1 then we can shrink it.

eg. 

.

$\Rightarrow$ Every node has out-deg $= 2$.

$\Rightarrow$ T is full. □

Lemma 2: More frequent alphabets are close to the root.

Proof:

- If $f(a_1) < f(a_2)$ & a_1 is deeper in T than a_2, then swapping them reduces $\sum_{a \in A} f(a) \cdot |\gamma(a)|$. □

Lemma 3: Let $A=\{a_1,\dots,a_n\}$ & $f(a_1)\le\dots\le f(a_n)$. There is an optimal T where a_1 & a_2 are siblings.

Proof:

- a_1, a_2 are deepest in the full tree T
  $\Rightarrow$ Wlog we can make them siblings in the deepest level. □

– By Lemma 3 we can modify the instance $A=\{a_1,a_2,\dots a_n\}$ to $A':=\{a_3,a_4,\dots a_n\}\cup\{a'\}$ by merging the two alphabets a_1 & a_2 to a'.

▷ If we set $f(a'):=f(a_1)+f(a_2)$ then $OPT(A')$ will also give us $OPT(A)$.

Pf:

$$OPT(A')=\sum_{b\in A'} f(b)\cdot|\gamma'(b)| = OPT(A)-f(a')$$

minimizing $ABL(\gamma)$. □

- Developing the algorithm to find T:
  - First, compute T' for alphabet A'.
  - Where is a'? A leaf in T'.
  - Replace it by 

.
  - Return this tree T.

- <u>Time</u>: It takes $O(n^2)$ time as we reduce the alphabet size by one.

<u>Theorem</u> (Huffman '52): Optimal prefix code can be found in $O(n^2)$ time.