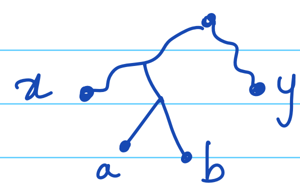
Theorem: Let x, y be two characters with the lowest frequencies. $\exists$ a prefix code s.t x, y are leaves of the max depth in the corr. binary tree

Proof: Consider an optimal prefix code and its tree T''. Let a, b be the leaves at the max depth s.t a, b are siblings



(i) Swap a & x to obtain T'

d – max depth of T

$d(x)$ – depth of x in T

$$abl(T') = abl(T'') - f_x \cdot d(x) + f_x \cdot d - f_a \cdot d + f_a \cdot d(x)$$

$$= abl(T'') + (d - d(x)) f_x - (d - d(x)) \cdot f_a$$

$$\left.\begin{array}{l} ① \ d - d(x) \geq 0 \\ ② \ f_x \leq f_a \end{array}\right\} \Rightarrow abl(T') \leq abl(T'')$$

(ii) swap b & y in T' to obtain T ∎

Recursively obtaining the code

$f[1] \leq f[2], \ \dots \leq f[k]$ – frequencies

* Obtain the Huffman code for

  $f[3], f[4], \dots, f[k], f[k+1]$ where

  $f[k+1] = f[1] + f[2]$

  ↳ Binary tree T'

* Make the leaf $(k+1)$ into an internal node with children 1 & 2 $\rightarrow T$

Theorem: T is an optimal prefix code

Proof: Suppose not. Let $\tilde{T}$ be the optimal prefix code. Wlog $\tilde{T}$ has 1 and 2 as siblings at the max depth

$$abl(T') = \sum_{i=3}^{n} f[i]\, d_{T'}(i) + f[n+1]\, d_{T'}(n+1)$$

$$= \sum_{i=3}^{n} f[i]\, d_{T'}(i) + (f[1] + f[2])(d_T(1) - 1)$$

$$= \sum_{i=3}^{n} f[i]\, d_T(i) + f[1]\, d_T(1) + f[2]\, d_T(2) - f[1] - f[2]$$

$$= abl(T) - f[1] - f[2]$$

Let $\hat{T}$ be the prefix code obtained from $\tilde{T}$ by combining 1 & 2

$$abl(\hat{T}) = abl(\tilde{T}) - f[1] - f[2]$$

$$abl(T') \leq abl(\hat{T}) \text{ (inductively)}$$

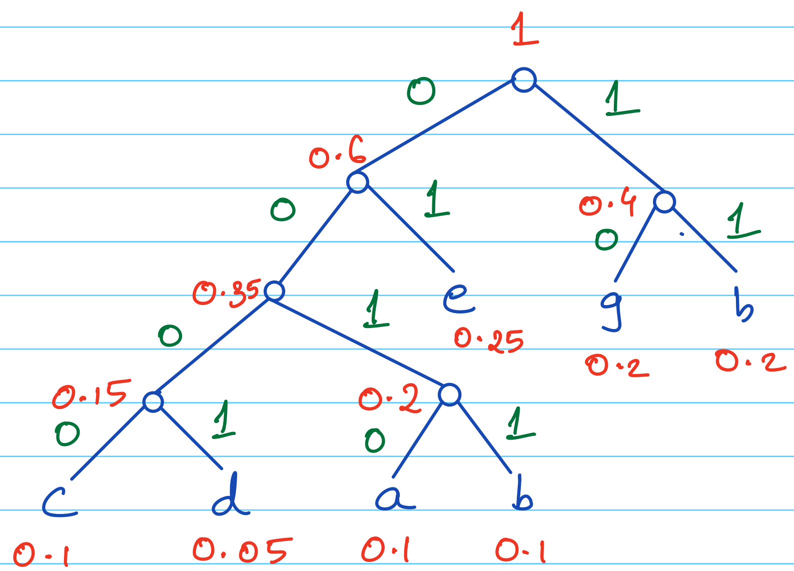
$$\Rightarrow abl(T) \leq abl(\tilde{T})$$

Example & Implementation details

$C = \{a, b, c, d, e, g, h\}$

$f[a] = 0.1 \quad f[b] = 0.1 \quad f[c] = 0.1 \quad f[d] = 0.05$

$f[e] = 0.25 \quad f[g] = 0.2 \quad f[h] = 0.2$



$E(a) = 0010 \quad E(b) = 0011 \quad E(c) = 0000$

$E(d) = 0001 \quad E(e) = 01 \quad E(g) = 10 \quad E(h) = 11$

- Store frequencies in a priority queue to extract the two smallest
- Insert the sum as a new node with the sum of the frequencies

* $O(n \log n)$ time to create the encoding
* $O(n)$ space to store the encoding
* $O(m)$ time to encode and decode the text