Huffman Codes

AND

Data Compression

- Computers understand only binary language.
- An encoding scheme maps symbols to binary strings.
- A simple encoding scheme assigns fixed number of bits to each symbol.
- With b bits you can encode 2^b symbols.
- ASCII is a fixed length encoding scheme.

Any encoding scheme should satisfy following two objectives:

(1) store the data Compactly.

(2) Reconstruct the data Unambiguously.

Objective - I (Data Compression)

- Encode frequently occuring letters (eg. **e, t, a, o, i, n**) with short strings.

- Encode infreequently occuring letters (eg **x, z, q**) with long strings.

Objective - 2 (Unambiguous reconstruction)

- Use prefix codes.
- A prefix code is one in which the code of one symbol is not a prefix of the code of other symbol.

* S is the set of Symbols.
* n is the # of Symbols in the text.
* $\forall x \in S$: f_x is the fraction of letters eqal to x
* $f_x \cdot n$ is the # of letters equal to x
* $\sum_{x \in S} f_x = 1$
* Let function r be the P. Code
* Encoding Length $= \sum_{x \in S} n \cdot f_x |r(x)|$
* $ABL(r) = \sum_{x \in S} f_x \cdot |r(x)|$

EXAMPLE - 1

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

$f_a = 0.32$, $r(a) = 11$

$f_b = 0.25$, $r(b) = 01$

$f_c = 0.20$, $r(c) = 001$

$f_d = 0.18$, $r(d) = 10$

$f_e = 0.05$, $r(e) = 000$

Calculate the average bits per letter (ABL)

$$ABL(r) = 2.25$$

EXAMPLE - 1

IMPROVEMENT IN CODE

$r(a) = 11$

$r(b) = 10$

$r(c) = 01$

$r(d) = 001$

$r(e) = 000$

The frequency of each letter remains same.

Now Calculate ABL(r) again.

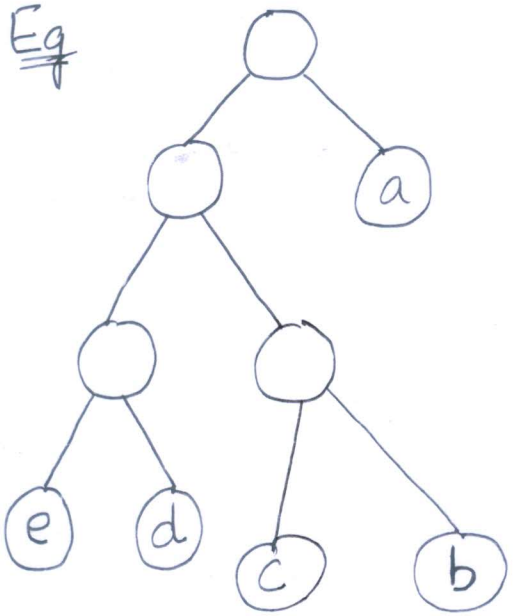
PROBLEM

- We are given an alphabet S and a frequency f_x for each letter $x \in S$.
- Produce a prefix code γ for which **ABL(γ)** is minimum.
- In other words, produce an optimal code γ.

PREFIX CODES

BINARY TREES

Eg



a = 1
b = 011
c = 010
d = 001
e = 000

# of leaves = |S|

PROBLEM RESTATED

We are given an alphabet S and a frequency for each $x \in S$. Produce a binary tree T s.t $\sum_{x \in S} f_x \cdot \text{depth}(x)$ is MIN.

OR.

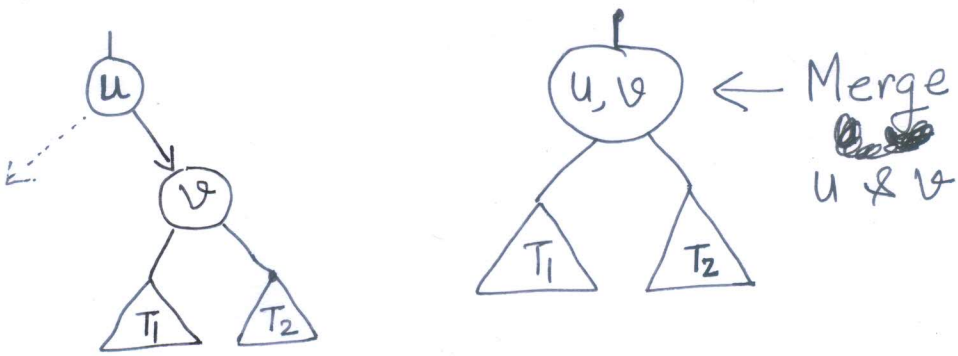
PRODUCE an optimal Tree.

$$ABL(T) = \sum_{x \in S} f_x \cdot \text{depth}(x)$$

LEMMA - 1

THE OPTIMAL BINARY TREE T^* IS A PROPER BINARY TREE.

Proof: If not, you can Convert it into one whose ABL value will be less than T^*, a contradiction.!



LEMMA - 2

SUPPOSE y and z are leaves of an optimal binary tree.

If $\text{depth}(y) < \text{depth}(z)$,

then $f_y \geq f_z$

PROOF : By Contradiction

THEOREM

IN OPTIMAL BINARY TREE T^*, THE TWO LOWEST FREQUENCY LETTERS ARE ASSIGNED TO LEAVES THAT ARE SIBLINGS.

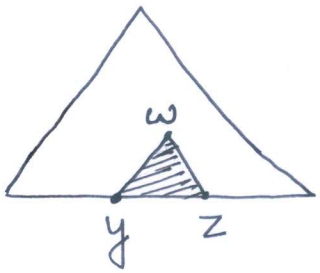
USING THIS THEOREM THE TREE CAN BE CONSTRUCTED AS FOLLOWS:

- Initially the Algo is called on alphabet size K
- In the second step, the Algo invokes itself on alphabet size K-1
- In the final step, the Algo invokes itself on alphabet size 2.

THE FIRST STEP

(1) In the Ist step, the two lowest frequency letters y & z are merged into a single letter w.

(2) The size of the alphabet reduces from k to $k-1$.



(3) T and T' are Trees built from alphabet size k and $k-1$, resp.

- $f_w = f_y + f_z$
- $\text{depth}_T(x) = \text{depth}_{T'}(x) \quad \begin{bmatrix} x \neq y \wedge \\ x \neq z \end{bmatrix}$
- $\text{depth}_{T'}(w) = \text{depth}_T(y) - 1$
- $\text{depth}_{T'}(w) = \text{depth}_T(z) - 1$

Theorem $ABL(T') = ABL(T) - f_\omega$

Proof:

$$ABL(T) = \sum_{x \in S} f_x \cdot depth_T(x)$$

$$= f_y \cdot depth_T(y) + f_z \cdot depth_T(z) + \sum_{x \neq y,z} f_x \cdot depth_T(x)$$

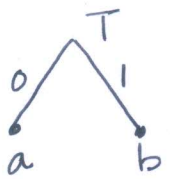
$$= (f_y + f_z)(1 + depth_{T'}(\omega)) + \sum_{x \neq y,z} f_x \cdot depth_{T'}(x)$$

$$= f_\omega + f_\omega \cdot depth_{T'}(\omega) + \sum_{x \neq y,z} f_x \cdot depth_{T'}(x)$$

$$= f_\omega + ABL(T')$$

PROOF OF CORRECTNESS

- We prove by MI on size of the alphabet
- **Base Case** Size = 2 $S = \{a, b\}$ say



T is optimal

- **Induction Step** Suppose the tree built when size = K is sub optimal.
- It means, $\exists$ tree Z s.t ABL(Z) < ABL(T)
- Convert Z into Z' by removing two least frequency symbols y and z, and labeling their parent as w
- Note both y and z were leaves with common parent.
- Now by previous theorem

$$ABL(Z) = ABL(Z') + f_w$$
$$ABL(T) = ABL(T') + f_w$$

- Since $ABL(Z) < ABL(T)$,
  we have $ABL(Z') < ABL(T')$.

- T' is suboptimal for alphabet of size $k-1$.

- What we have proved is:
  If T is suboptimal for alphabet of size k, then T' is suboptimal for alphabet of size $k-1$.

- The contrapositive of above is:
  If T' is optimal for alphabet of size $k-1$, then T is optimal for alphabet of size k.

- The Induction Step is over.

COMPLEXITY OF THE HUFFMAN ALGO

(1) Construct a heap of the k letters based on theirs frequencies. : $O(k)$

(2) Fetch two min freq letters say x and y from the heap: $O(\log k)$.

(3) Add a new letter z into the heap s.t $f_z = f_x + f_y$: $O(\log k)$

(4) Repeat 2, 3 k - times. :

$$O(k \cdot \log k)$$