16/02/2022

# Huffman Code

→ An encoding algorithm that takes a text over, say english alphabet, and converts it to a string of 0s and 1s.

→ Also a decoding algorithm to go back from binary strings to the text.

→ 26 letters in alphabet, space to separate words, period, question mark, comma.

$26 + 4 = 30$ characters.

Use 5 bits to encode every character.

$2^5 = 32$

Can we do it with 4 bits? NO.

$2^4 = 16 < 30$

"uniform" encoding := every character has a code word of same length.

Letter frequency in english alphabet:

e t a i o — appears much more frequently than → x y z j

→ text with K characters and each character has some frequency, where frequency is the fraction of times a character appears in the text.

length of the text = $n$

Alphabet size = $k := \{a_1, \ldots, a_k\}$

$$\text{frequency of } a_i = \frac{\#\text{ times } a_i \text{ appears in the text}}{n}$$

you want to bring down the no. of bits required to encode each character on an average.

$$\text{length of encoding for the text} = \sum_{\text{character}} |\text{length of character}| \cdot \#\text{ of times it appears}$$

$$\text{average length per character} = \frac{\text{total length of encoding}}{n}$$

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

$\Gamma_1$

a = 10
b = 11
c = 010
d = 001
e = 000

uniform encoding. $\Gamma_0$

Use 3 bits

000 ← a
001 ← b
010 ← c
011 ← d
100 ← e

a a b c a c

↳ 000000001010000010

↳ a a b c a c

aabcade

↳ 10 10 11 010 10 001 000

↳ a a b c a d e

total length of encoding = 17

total length of text = 7

average encoding length = $\frac{17}{7} = 2\frac{3}{7}$

$$\text{average encoding length} = \frac{\sum_{\text{Character}} |\text{length of character}| \cdot \#\text{ times it appears}}{n}$$

$$= \sum_{\text{Character}} |\text{length of character}| \cdot \frac{\#\text{ times it appears}}{n}$$

↑ frequency of the character

**Problem:-** Given an alphabet $A = \{a_1, \ldots, a_k\}$ and frequencies of each letter in the alphabet $\{f(a_1), f(a_2), \ldots, f(a_k)\}$.

Come up with an encoding scheme with minimal average length per letter.

$$= \sum_{i=1}^{k} \text{length of encoding}(a_i) \cdot f(a_i)$$

→ **Morse codes:**

e → 0 — (dot)

t → 1 — (dash)

a → 01 — (dot-dash)

OLOL → a a
→ e t e t

} include a "pause" between two letters.

⇒ prefix-free codes.

no codeword is a prefix of another codeword.

$a_1 \to \Gamma(a_1)$
$a_2 \to \Gamma(a_2)$
$\vdots$
$a_k \to \Gamma(a_k)$

then $\forall\ i \neq j,\ 1 \leq i < j \leq k$

$\Gamma(a_i)$ is not a prefix of $\Gamma(a_j)$

and vice-versa.

Uniform encoding ⇔ all codewords have the same length is a prefix-free code.

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

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

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

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

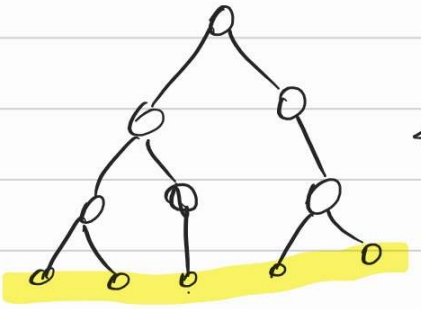
$\Gamma_2(d) = 10$, $\Gamma_2(e) = 000$.

Average length of $\Gamma_2 = 2 \cdot 0.32 + 2 \cdot 0.25 + 3 \cdot 0.20 + 2 \cdot 0.18 + 3 \cdot 0.05$

$= 2.25 < 3$

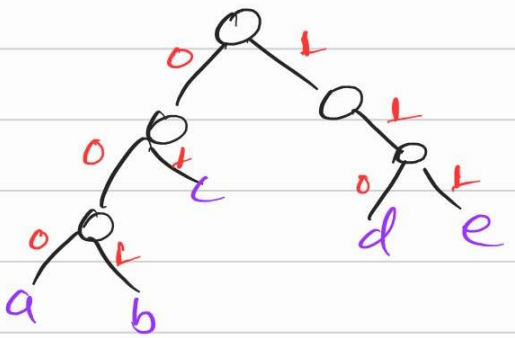
## Prefix-free Codes:

**Defn:** Binary trees: A tree where every node has at most 2 children.




← binary tree

→ Suppose I give a binary tree with 5 leaves where each leaf is labelled by one of the letters in $\{a, b, c, d, e\}$




– fix a label for edges in the tree

left edge = 0

right edge = 1

$a \rightarrow 000$

$b \rightarrow 001$

$c \rightarrow 01$

$d \rightarrow 110$

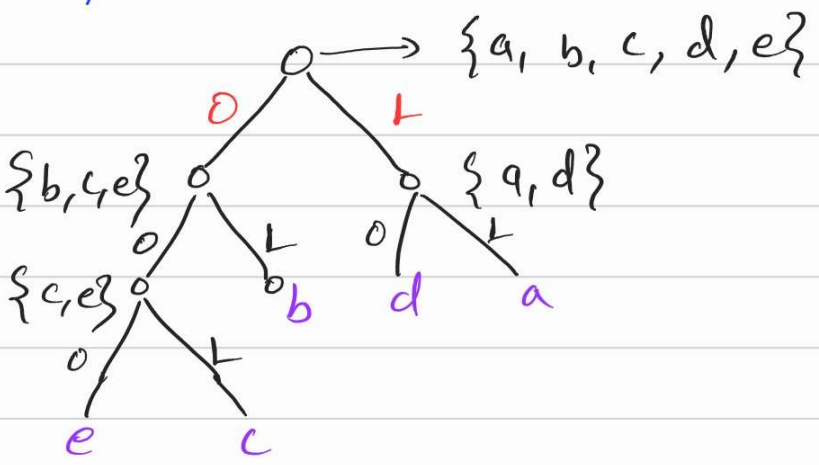
$e \rightarrow 111$

**Claim:** It is a prefix-free code.

you can go from Binary tree → Prefix free codes.

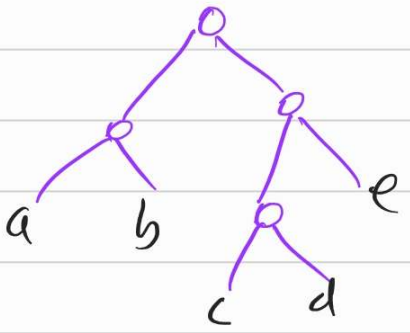
(⇐) Prefix-free code ⟶ Binary tree

$\Gamma_2(a) = 11$
$\Gamma_2(b) = 01$
$\Gamma_2(c) = 001$
$\Gamma_2(d) = 10$
$\Gamma_2(e) = 000$




All of this implies that to come up with a prefix-free code all you need to do is to come up with a binary tree with leaves labelled by the letters in your alphabet.

What is average encoding length in terms of binary trees?




$\sum_{\text{letters}} \text{depth}_T(a) \cdot f(a)$

average depth of the tree w.r.t the frequencies.

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

wrt frequencies.