Huffman Codes

Represent a text $T[1, 2, \ldots, n]$ with $S[1, 2, \ldots, k]$ different characters using a binary code - How efficiently can it be done?

* Represent each character using $\log k$ bits & the text using $n \log k$ bits

  Can we do better?

  Get a representation that uses $< n \log k$ bits

- Not all characters occur with the same frequency
  - encode the more frequent letters with shorter representations
- Issues

  a : 101 b : 10 c = 1

  10110

  101 → a

  → bc

Average bit length: f_i = fraction of occurrences of i in T

$$abl(T) = \sum_{i \in [k]} f_i \cdot \ell(i)$$

↳ length of the representation of i in the code

* if all lengths are to be the same $abl(T) = \log k$

Prefix codes: $C : \{1, 2, .., k\} \to \{0,1\}^*$

s.t $\forall\, j, l$ $C[j]$ is not a prefix of $C[l]$

Example a, b, c, d

$C(a) = 00$ $C(b) = 01$ $C(c) = 10$ $C(d) = 11$

$f_a = 0.1$ $f_b = 0.5$ $f_c = 0.3$ $f_d = 0.1$

$abl(C) = 2$

$C(a) = 001$ $C(b) = 1$ $C(c) = 01$ $C(d) = 000$

$abl(C) = 3 \times 0.1 + 1 \times 0.5 + 2 \times 0.3 + 3 \times 0.1$

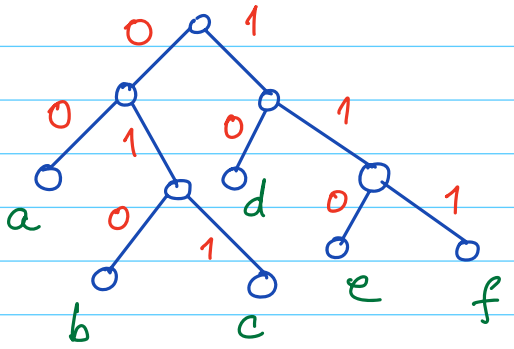
$= 1.7$

$C(abccd) \mapsto 00110101000$

Shannon-Fano: Which is the binary prefix code with the smallest abl?

Prefix codes & Binary trees

* Every binary tree corresponds to a prefix code



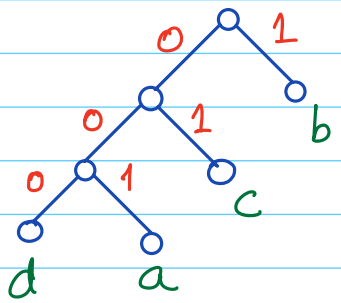
$C(a) = 00$ $C(d) = 10$

$C(b) = 010$ $C(e) = 110$

$C(c) = 011$ $C(f) = 111$

* Every prefix code corresponds to a binary tree

$C(a) = 001$ $C(b) = 1$ $C(c) = 01$ $C(d) = 000$

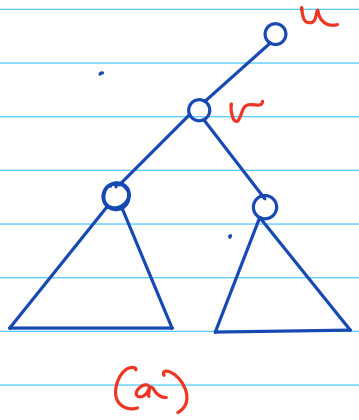


- All codes that start with 0 on the left subtree & all those with 1 on the

* Depth of the leaf ≡ length of the code

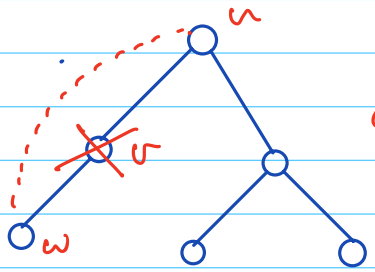
$abl(C) = avg\text{-}depth(T)$

* Optimal prefix code corresponds to full binary trees



(a)

delete u and make v the root



(b)

Make w, a child of u & delete v

Both (a) & (b) can only reduce the average depth

* Intuitively, the lowest frequency characters should have the longest codes