Huffman Codes

Introduction and Motivation

Algorithms: Design and Analysis, Part II

Binary Codes

Binary code: Maps each character of an alphabet Σ to a binary string.

Example: $\Sigma =$ a-z and various punctuation (size 32 overall, say)

Obvious encoding: Use the 32 5-bit binary strings to encode this Σ (a fixed-length code)

Can we do better? Yes, if some characters of Σ are much more frequent than others, using a variable-length code.

Ambiguity

Example: Suppose Σ ={A,B,C,D}. Fixed-length encoding would be {00,01,10,11}.

Suppose instead we use the encoding {0,01,10,1}. What is 001 an encoding of?

A) AB $\rightarrow$ Leads to 001

B) CD

C) AAD $\rightarrow$ Also leads to 001

D) Not enough info to answer question

Prefix-Free Codes

Problem: With variable-length codes, not clear where one character ends + the next one begins.

Solution: Prefix-free codes - make sure that for every pair $i, j \in \Sigma$, neither of the encodings $f(i), f(j)$ is a prefix of the other.

Example: $\{0,10,110,111\}$

Why useful? Can give shorter encodings with non-uniform character frequencies.

Example

Example:

A	60%	00	0
B	25%	01	10
C	10%	10	110
D	5%	11	111
Σ	frequencies	fixed-length	variable-length (prefix free)

Fixed-length encoding: 2 bits/character

Variable-length encoding: How many bits needed on average?

A) 1.5 B) 1.55 C) 2 D) 2.5

$0.6 \cdot 1 + 0.25 \cdot 2 + (0.1 + 0.05) \cdot 3 = 1.55$