Huffman's Encoding Problem

That Is Math

Dheer Avashia

University of the South

2nd March 2023

The Problem

Encode the message **‘That is Math’** using an optimal binary code without losing any information during decoding.

What we know

1. **Binary Encoding Requirement**
2. **No information loss**
3. **Characters and Frequency**: $\Gamma = \{A, T, H, I, S, M, \phi\}$

Character	**Frequency**
A	2
T	3
H	2
I	1
S	1
M	1
ϕ	2

4. **Optimal Code**:

$$B(C) = \sum (f(\Gamma_i) \cdot L(C(\Gamma_i))$$

Approach 1

Randomly assign unique binary numbers to each character.

$C_1 = \{\mathrm{A} = 0,\ \mathrm{T} = 01,\ \mathrm{H} = 10,\ \mathrm{I} = 001,\ \mathrm{S} = 101,\ \mathrm{M} = 100,\ \phi = 010\}$

$$011000101000110101010000011 0$$

Is this uniquely decipherable?

Approach 1

$C_1 = \{\text{A} = 0, \text{T} = 01, \text{H} = 10, \text{I} = 001, \text{S} = 101, \text{M} = 100, \phi = 010\}$

$$\textcolor{red}{\mathbf{01100}}010100011010101000001 10$$

Can be read as can be read as either ‘THA’ or ‘TM’. There is loss of information.

Approach 1

Cost of C_1

Character	**Frequency**	**Length**
A	2	1
T	3	2
H	2	2
I	1	3
S	1	3
M	1	3
ϕ	2	3

$$\begin{aligned} B(C_1) &= (2 \cdot 1) + (3 \cdot 2) + (2 \cdot 2) + (1 \cdot 3) + (1 \cdot 3) + (1 \cdot 3) + (2 \cdot 3) \\ &= 2 + 6 + 4 + 3 + 3 + 3 + 6 \\ &= \textbf{27 Bits} \end{aligned}$$

Approach 2

Fixed-Length Code Approach

$C_2 = \{$A $=$ 0100 0001, T $=$ 0101 0100, H $=$ 0100 1000, I $=$ 0100 1001, S $=$ 0101 0011, M $=$ 0100 1101, $\phi =$ 0000 1000$\}$

0101 0100 0100 1000 0100 0001 0101 0100 0000 1000 0100 1001 0101 0011 0000 1000 0100 1101 0100 0001 0101 0100 0100 1000

This is uniquely decipherable.

Approach 2

Cost of C_2

Character	Frequency	Length
A	2	8
T	3	8
H	2	8
I	1	8
S	1	8
M	1	8
ϕ	2	8

$$
\begin{aligned}
B(C_2) &= (2 \cdot 8) + (3 \cdot 8) + (2 \cdot 8) + (1 \cdot 8) + (1 \cdot 8) + (1 \cdot 8) + (2 \cdot 8) \\
&= 16 + 24 + 16 + 8 + 8 + 8 + 16 \\
&= \mathbf{96\ Bits}
\end{aligned}
$$

Problem Is Still A Problem

C_1 uses lower bits, looses information.

C_2 uniquely decipherable, possibly uses too many bits.

Huffman was given the same problem.

Huffman's Greedy Algorithm

Step 1: Characters with the lowest frequency. Create sub-tree with these two characters as leaves. Label the root of the tree some arbitrary z.

Step 2: Set frequency of z as addition of frequencies. Create new set of alphabet with z replacing letters.

Step 3: Repeat merging of variables using the new alphabet until one character left.

Huffman's Greedy Algorithm

Character	Frequency
A	2
T	3
H	2
I	1
S	1
M	1
ϕ	2

M and S, create a sub-tree with root MS. New alphabet $\Gamma_1 = \{A, T, H, I, MS, \phi\}$

Character	**Frequency**
A	2
T	3
H	2
I	1
MS	2
ϕ	2

Huffman's Greedy Algorithm

I and A, root IA. New alphabet $\Gamma_2 = \{IA, T, H, MS, \phi\}$

Character	**Frequency**
IA	3
T	3
H	2
MS	2
ϕ	2

Huffman's Greedy Algorithm

ϕ and H, root ϕH. $\Gamma_3 = \{IA, T, MS, \phi H\}$

Character	**Frequency**
IA	3
T	3
MS	2
ϕH	4

MS and T, root MST. $\Gamma_4 = \{IA, MST, \phi H\}$

Character	**Frequency**
IA	3
MST	5
ϕH	4

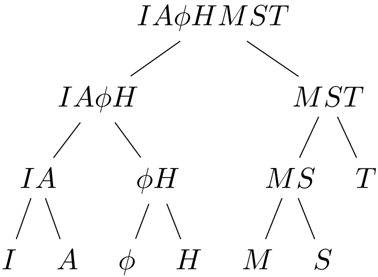
IA and ϕH, root $IA\phi H$ and $\Gamma_5 = \{IA\phi H, MST\}$

Character	**Frequency**
$IA\phi H$	7
MST	5

Last Merge: $IA\phi H$ and MST, root $IA\phi HMST$. $\Gamma_6 = \{IA\phi HMST\}$

Character	**Frequency**
$IA\phi HMST$	12

Huffman's Greedy Algorithm



Corresponding Huffman encoding:
$C_3 = \{$A = 001, T = 11, H = 011, I = 000, S = 101, M = 100, ϕ = 010$\}$

110110011101000010101010000111011

Huffman's Greedy Algorithm

Cost of C_3

Character	**Binary Code** (C_3)	**Frequency**
A	001	2
T	11	3
H	011	2
I	000	1
M	100	1
S	101	1
ϕ	010	2

$$\begin{aligned} B(C_3) &= (2\cdot 3) + (3\cdot 2) + (2\cdot 3) + (1\cdot 3) + (1\cdot 3) + (1\cdot 3) + (2\cdot 3) \\ &= 6+6+6+3+3+3+6 \\ &= \textbf{33 Bits} \end{aligned}$$

Proof Of Optimally

Theorem

Huffman's algorithm produces an optimal prefix code tree

Lemma (1)

Every tree will yield a prefix-free code and conversely

Lemma (2)

The tree for any optimal prefix code must be full, every internal node has exactly two children.

Lemma (3)

Consider two letters, x and y with the smallest frequencies. Then there is a optimal code tree in which these two letters are siblings leaves in the tree in the lowest level.