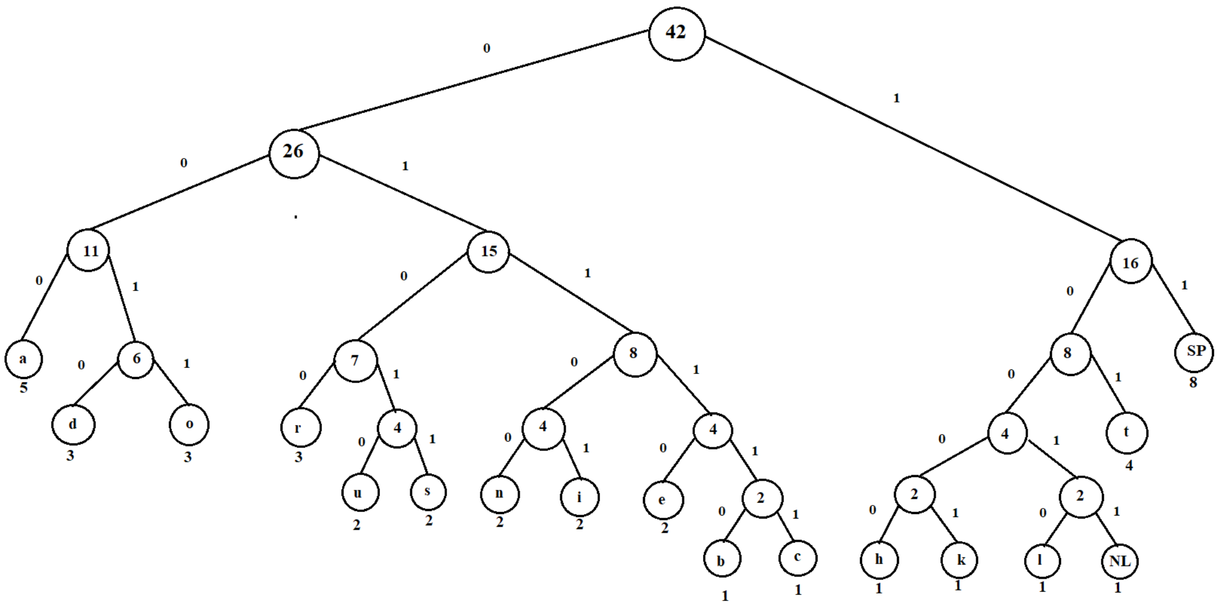
Lab10: Learn to build frequency table and Huffman encoding tree

Consider the message “**the clouds are dark and its about to rain**” and construct frequency table and Huffman encoding tree.

Frequency Table:

Character	Frequency	Character	Frequency
a	5	n	2
b	1	o	3
c	1	r	3
d	3	s	2
e	2	t	4
h	1	u	2
i	2	SP	8
k	1	NL	1
l	1		

Huffman Encoding Tree:



Huffman Character Codes:

Character	Huffman Code	Character	Huffman Code
a	000	n	01100
b	011110	o	0011
c	011111	r	0100
d	0010	s	01011
e	01110	t	101
h	10000	u	01010
i	01101	SP	11
k	10001	NL	10011
l	10010		

There are **42 characters** in the message. If we represent them using **ASCII encoding**, where each character is represented by **8 bits**, the total number of bits required would be:
42 × 8 = 336 bits.

However, using **Huffman encoding**, each character is represented by a **variable number of bits**, as shown in the table above. To calculate the total number of bits used in the Huffman-encoded message, we multiply the **frequency** of each character by the **length of its Huffman code**, and then sum the results.

Character	Frequency	Huffman Code	Code Length	Total Bits
a	5	000	3	15
b	1	011110	6	6
c	1	011111	6	6
d	3	0010	4	12
e	2	01110	5	10
h	1	10000	5	5
i	2	01101	5	10
k	1	10001	5	5
l	1	10010	5	5
n	2	01100	5	10
NL	1	10011	5	5
o	3	0011	4	12
r	3	0100	4	12
s	2	01011	5	10
t	4	101	3	12
u	2	01010	5	10
SP	8	11	2	16
			Total:	**181**

Using **Huffman encoding**, the message requires **only 181 bits**, compared to **336 bits** using standard ASCII.
This results in a **46.13% reduction in size**, demonstrating the efficiency of Huffman coding for data compression.

By: Javaid Iqbal
| GitHub: JavaidIqbal786
| LinkedIn: JavaidIqbalAwan