② Huffman coding & decoding :

Why Huffman coding and decoding is used ??
Let us take an example to understand.

Example :

Normal coding :

Char	Ascii value	frequency	code
A	65	3	000
B	66	5	001
C	67	6	010
D	68	4	011
E	69	2	100
	Total:	20	

To store 20 alphabets we need (20×3) = 60 bits

Huffman coding

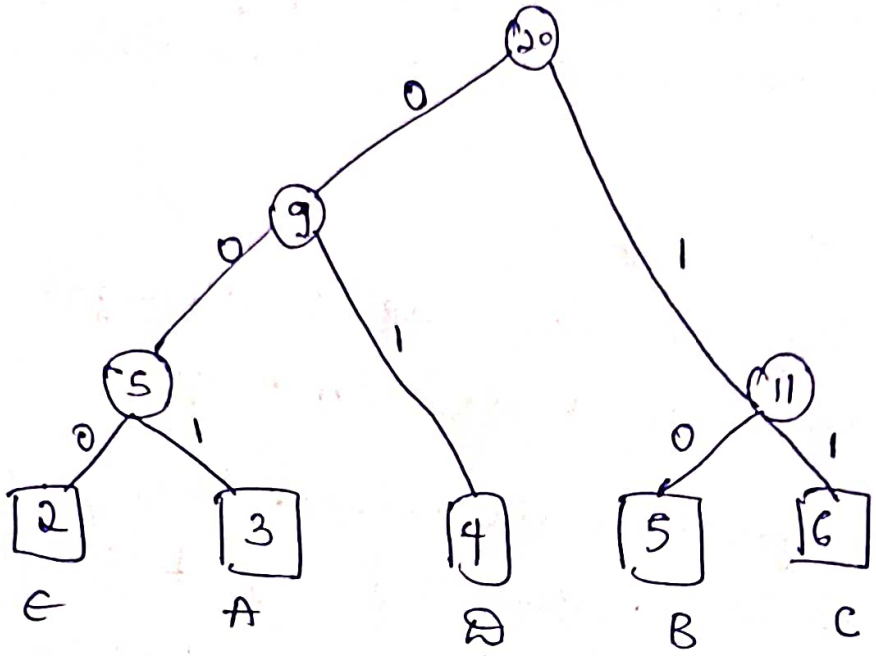
(i) sort according to frequency.

(ii) select the 2 smallest frequency and form a node.

(iii) Repeat the process untill all the nodes are covered.

(iv) Left path assign 0

Right path assign 1



	Huffman code
A	001
B	10
C	11
D	01
∈	000

To store 20 alphabets we need = $[3 \times 3]_{A} + (5 \times 2)_{B}$

(frequency | no of bits)

$+ (6 \times 2)_{C} + (4 \times 2)_{D} + (2 \times 3)_{\in}$

$= 35$ bits

{Difference b/w normal & Huffman} = 60 − 35 = 25 //

∴ we saved 25 bits using Huffman coding

Time complexity :

$O(n \log n)$

where n is the number of alphabets.

Since Huffman coding will use the min-heap. And min heap formation takes $O(n \log n)$ time for its creation. This algorithm also takes the same time.

Huffman decoding:

→ Given the character, its frequencies, we will be able to construct the min-heap.

→ Once you have the encoded string, travel left if '0' comes, travel right if '1' comes.

→ The decoded string will be found.

Time complexity :

Since this will also use min-heap. It is $O(n \log n)$

$O(n \log n)$

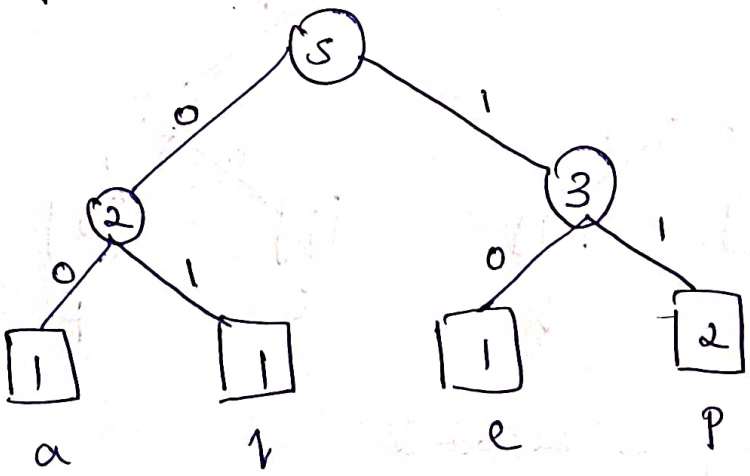
Huffman encoding

Input:

Enter a word: apple

	frequency
a	1
p	2
l	1
e	1

Output:



	Huffman code
a	00
e	10
l	01
p	11

Encoded string: a p p l e
00 11 11 01 10

Huffman decoding :-

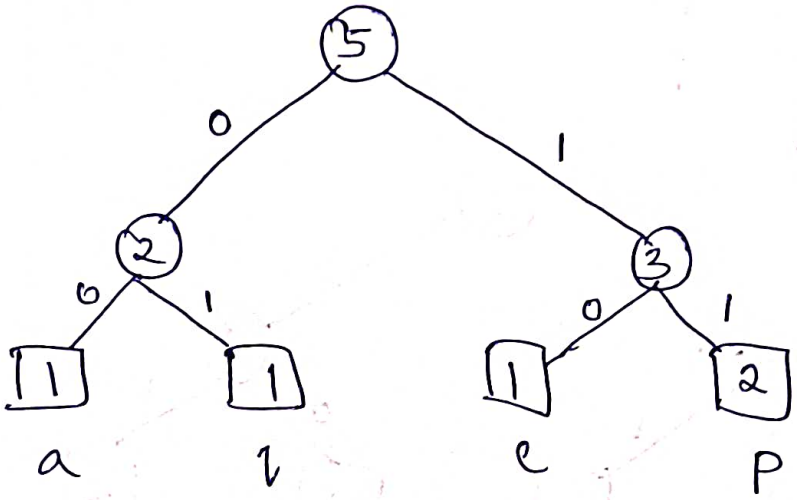
Input: Enter the frequency table.

	frequency
a	1
p	2
l	1
e	1

Enter the Encoded string : 00111 0110

Output :

create min-heap using frequency table.



Encoded string is decoded

00	11	11	01	10
a	p	p	l	e

∴ Decoded string : apple.

———— X ————