HUFFMAN CODE ALGORITHM :-

Prefix Codes, means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character.

This is how huffman coding makes sure that there is no ambiguity when decoding the generated bitstream.

Example :- Let there be four characters a, b, c & d and their corresponding variable length codes 00, 01, 0 & 1. This coding leads to ambiguity because code assigned to c is the prefix of codes assigned to 'a' and 'b'.

If the compressed bit stream is 0001

The de-compressed o/p may be ⟶ "cccd" ← 0001
or
"ccb" ← 0001
or
"acd" ← 0001
or
"ab" ← 0001

See this for appl.n of Huffman Coding :-

There are mainly two major parts in Huffman Coding :

1) Build a Huffman Tree from i/p characters.
2) Traverse the Huffman Tree & assign codes to characters.

→ Data Compression method.

→ lossless (without loss of information) Compression.

Algorithm :-

Huffman (c) ← set of n character

1) n = |c|
2) Q = c // min. Priority Queue
3) for i=1 to n-1
4) do allocate a new node z
5) z.left ← x ← Extract.min(Q)
6) z.right ← y ← Extract.min(Q)
7) z.freq ← x.freq + y.freq
8) Insert (Q, z)
9) return Extract.min(Q) // return Root of the tree

Implementation ⇒

To understand the concept of fixed & variable code :-

fixed

	a	b	c	d	e	f
freq. in Thousands →	45	13	12	16	9	5

= (100)

Total freq. ⇒ 100 × 1000 = 1 lakh

If each character is represented by 3 bit ⇒

1 lakh × 3 = 3 lakh bit

Storage required

Variable.

But in Variable Code ⇒

1×45 + 3×13 + 3×12 + 3×16 + 4×9 + 4×5

= 45 + 39 + 36 + 48 + 36 + 20

= 224

224 × 1000 = 224000 bit

Storage required

See Soln in next Page →

Example:-

Character	frequency	fixed code	variable length
a	45	000	0
b	13	001	101
c	12	010	100
d	16	011	111
e	9	100	1101
f	5	101	1100

→ Solve using Huffman Code.

In fixed Code ⇒ we require more ~~we~~ storage to store.

like $45 \times 3 = 135$ (3 ↑ bit)

$13 \times 3 = 39$

$12 \times 3 = 36$

⋮

For reducing ~~less~~ storage we use variable length ⇒ by Huffman code.

1) $n = 6$

2) Q =

a	b	c	d	e	f
45	13	12	16	9	5

← min. Priority Queue.

3) for $i = 1$ — 5

4) do allocate a node (Z)

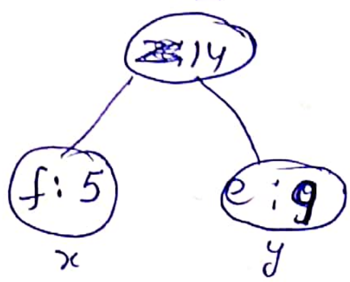
5) Z. left ← (f:5) x ← f

6) Z. right ← (e:9) y ← e

7) Z. freq ← x. freq + y. freq

Z. freq ← 5 + 9

Z. freq ← 14



Scanned with CamScanner

8) Now, Insert (Q, Z)

Again go to step 3)

3) i = 2 – 5

4) allocate Z

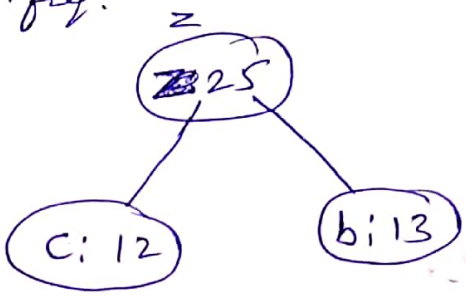
5) Z.left ← x ← c

6) Z.right ← y ← b

7) Z.freq. ← x.freq + y.freq.
← 12 + 13
← 25

a	45	
b	13	
c	12	
d	16	
e	9	
f	5	

14



8) Insert (Q, z)

Repeat
until
1 ~~root~~ node
is found.

SHORT CUT :- a b c d e f

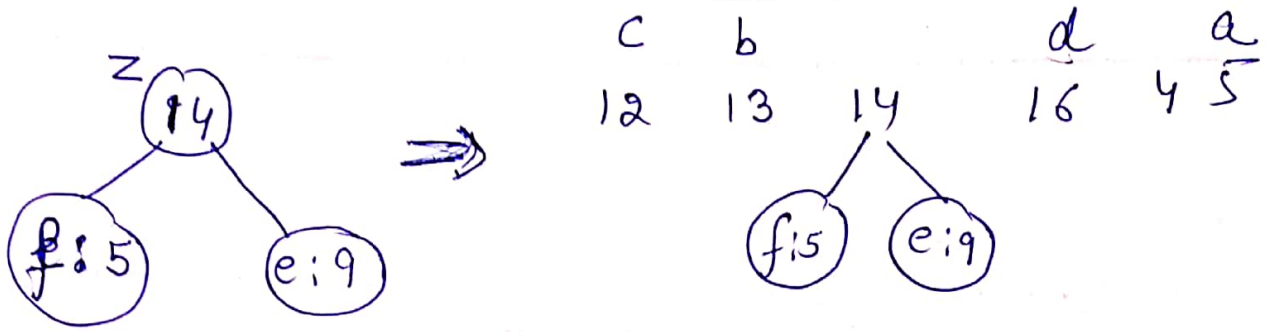
45 13 12 16 9 5

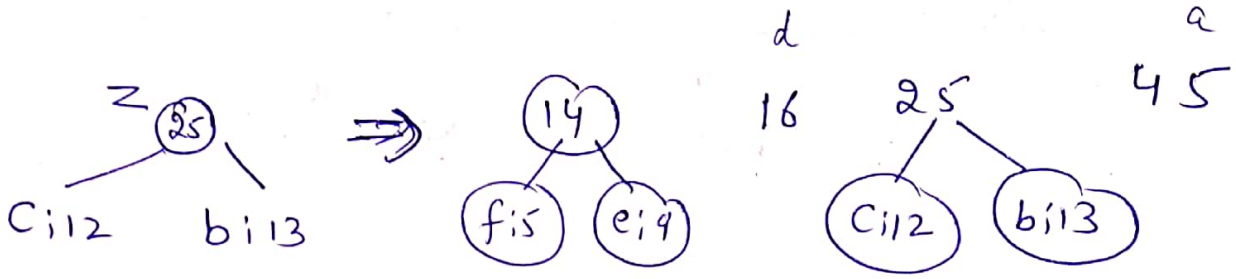
Arrange in ascending order :-

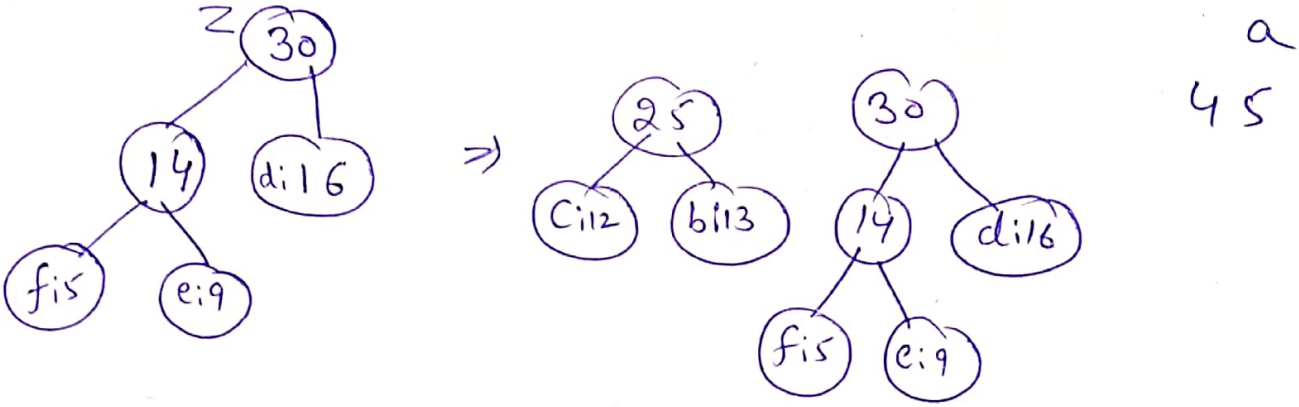
~~a b c d e f~~

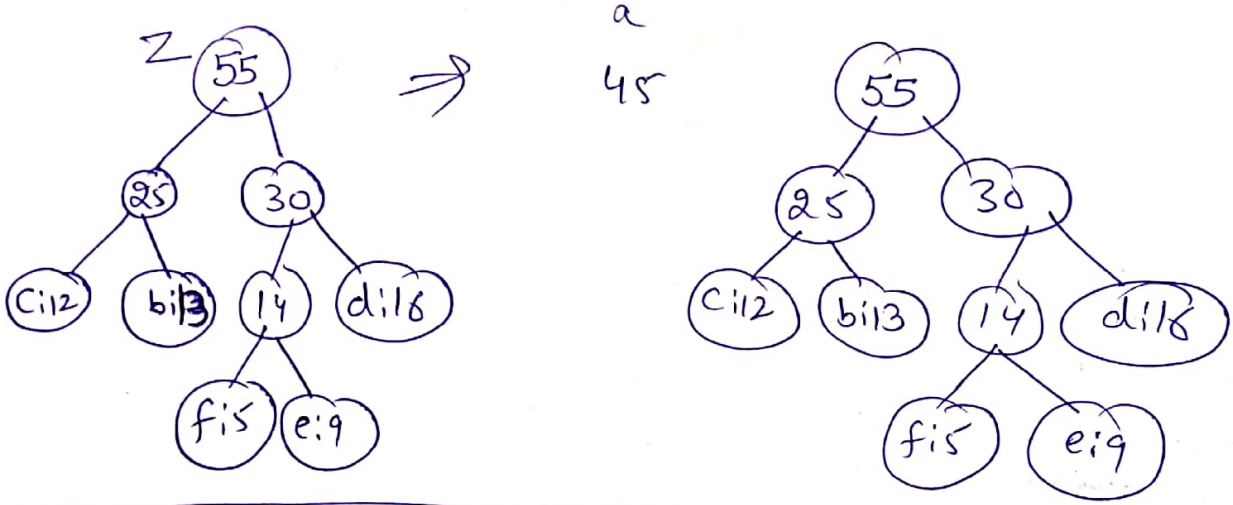
f e c b d a

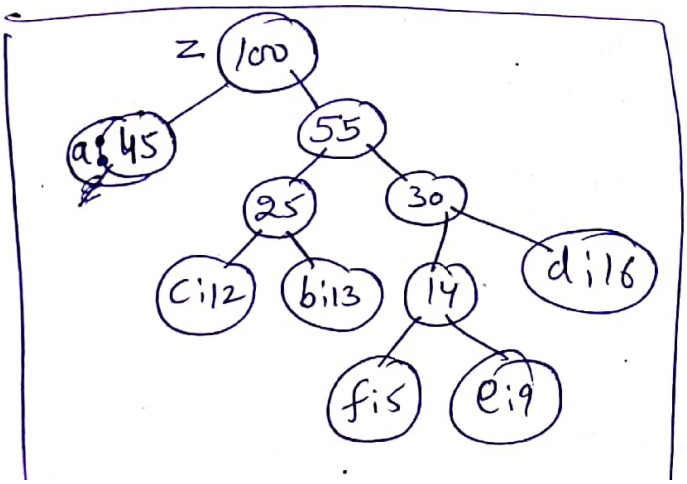
5 9 12 13 16 45



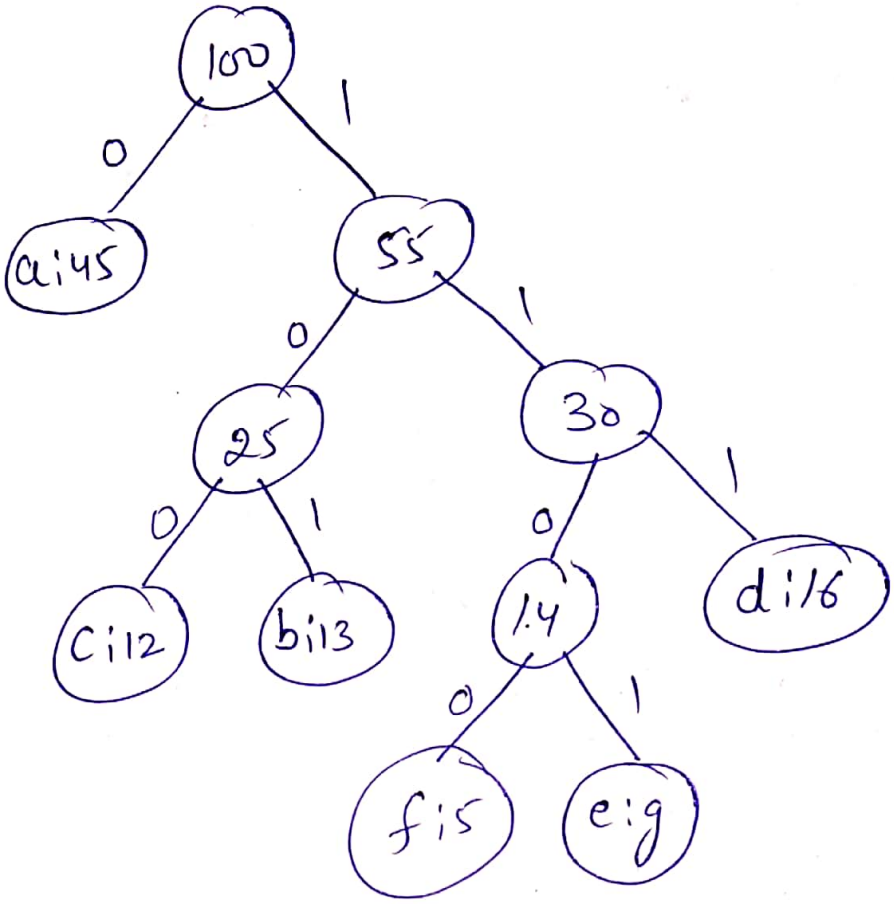








⟸ Now Assign (0) to left node &
(1) to right node



Now New Code :-

Variable Code

a	0
b	101
c	100
d	111
e	1101
f	1100

thus ⇒