* **Huffman Codes** (Variable size coding)

Ex: Message :- BCCABBDDAECCBBAEDDCC (len = 20)
i/p

General way :- ASCII char of each symbol (8 bit)

20 x 8 = 160 bits (to send)

	Char	freq count	Own code (3 bit)
5 char	A →	3	0 0 0
	B →	6	0 0 1
	C →	6	0 1 0
	D	:	0 1 1
	E	:	1 0 0

fixed sized code

$2^1 = 2$

$2^2 = 4$

$2^{3} = 8$ //

- total size of msg = 20 x 3 = 60 bits //

- total bits for table => $5 \times \overset{\text{ASCII}}{8 \text{ bits}} = 40$

  $+$

  $5 \times \underset{\text{own code}}{\underline{3 \text{ bits}}} = 15$

  $\overline{55 \text{ bits}}$

* Final size of msg = $\underset{\text{msg}}{60} + \underset{\text{table}}{55}$ = (115 bits)

* Variable Size Coding (Huffman coding)

↳ optimal merge pattern tree

Msg → B C C A B B D D A E C C B B A E D D C C

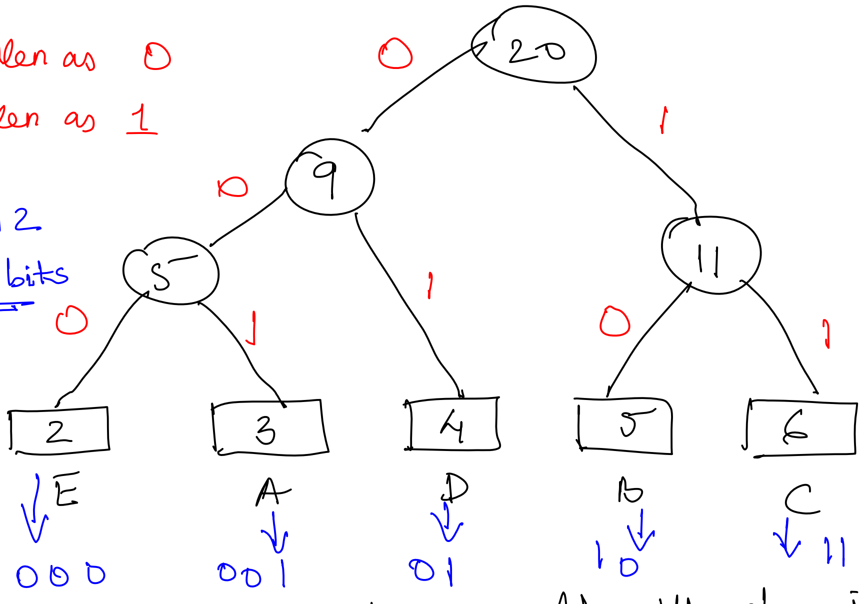
10 11 11 001 10 10 ... 11 11

Char	freq count	code (Variable size code)	
A	3 ✓	001	3 × 3 = 9
B	5 ✓	10	5 × 2 = 10
C	6 ✓	11	6 × 2 = 12
D	4 ✓	01	4 × 2 = 8
E	2 ✓	000	2 × 3 = 6
5 × 8 = 40 bits	20 bits	12 bits	Msg size ⇒ 45 bits

(ASCII)

* left child taken as 0
* rt child taken as 1

* Total table size ⇒ 40 + 12 = 52 bits



optimal merge pattern tree

Huffman code →

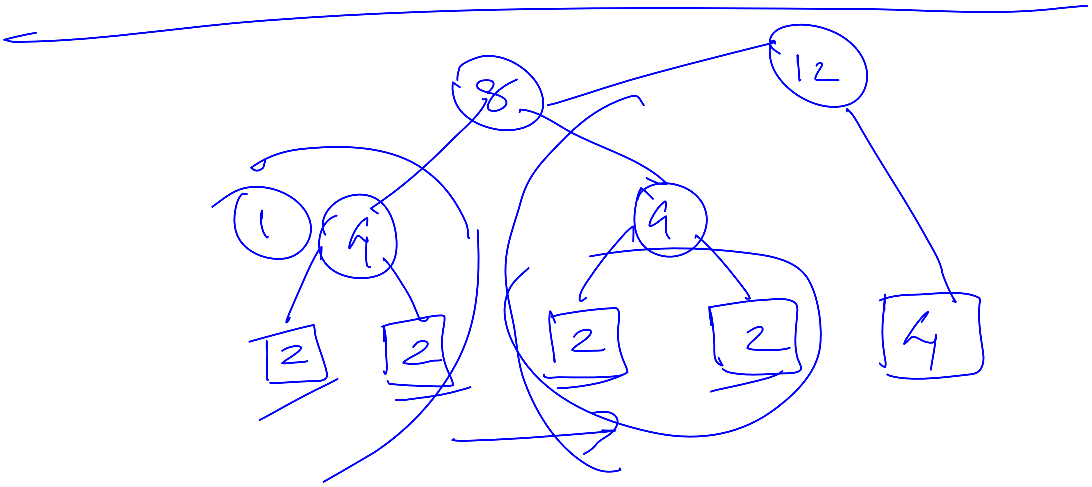
arrange the i/p char in asc order seq

(Huffman coding)

45 bits	52 bits
Msg size	Table

= 97 bits (Final size)

Relation ① More the no of times a char occurrs in an i/p text → lesser is the size of Huffman code


12
8
1
4
4
2
2
2
2
4