Encoding

If there is a message having 100 chara

$a = 5$, $b = 9$, $c = 12$

$d = 13$, $e = 16$, $f = 45$

Total characters $\Rightarrow$ 100

ASCII Encoding

Each character takes 8 bits

So Total bits for this message $= 100 \times 8$

$= 800$ bits.

Another method of encoding:-

As there are 6 unique charact

So 3 bits are needed to assign unique code to each character.

codes

a → 000

b → 001

c → 010

d → 011

e → 100

f → 101

So Total bits for encoded message = 100 × 3

= 300 bits.

* Both above methods are used to assign "Fixed Length codes" to symbols.

In 2nd method, message has been compressed (size of bits has been reduced)

800 bits ⟶ 300 bits.

* Message can be decoded easily on the basis on scheme used for encoding. because codes are fixed Length.

Huffman encoding

It is a way to encode and compress message using "Variable Length codes" to represent characters depending on how frequently they appear.

The **idea** behind this is that

→ characters appeared more frequently should be assigned short codes,

→ while those appear more rarely should be assigned Longer codes

* Codes assigned to one character is not the prefix of code assigned to any other character.

So it makes sure that there is no ambiguity during decoding.

Algorithm

assuming index start from 1

```
HUFFMAN (N, Symbols[1..N], freq[1..N])
{
    for (i ← 1 To N)
    {  t ← TreeNode(symbol[i], freq[i])
       minHeap.insert(t, freq[i])
    }

    for (i ← 1 To N-1)
    {
        x ← minHeap.remove()
        y ← minHeap.remove()
        Z ← new TreeNode()
        Z.left ← x
        Z.right ← y
        Z.freq ← x.freq + y.freq
        minHeap.insert(Z, Z.freq)
    }
}   return minHeap.remove()
```

Time Complexity $= O(n \log n)$

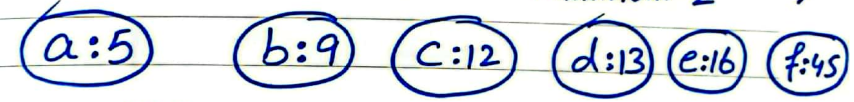
n are unique Characters.

Tree finally obtained is the desired Huffman tree.

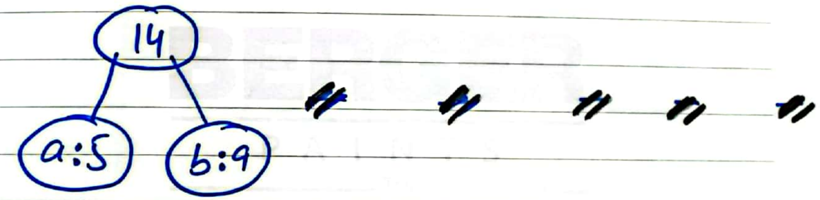
Dry Run

Create minHeap or array all nodes of characters in increasing order of their frequencies. (After each step arrange resulting node and Pick minimum 2)

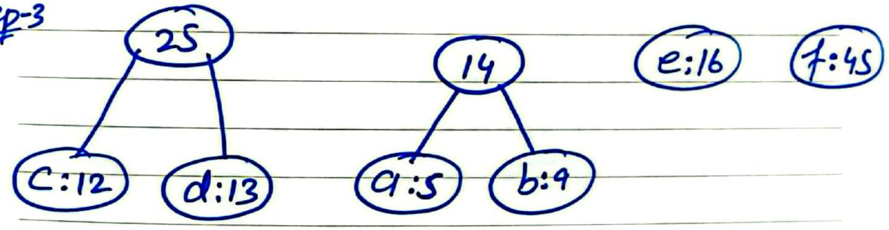
Step-1



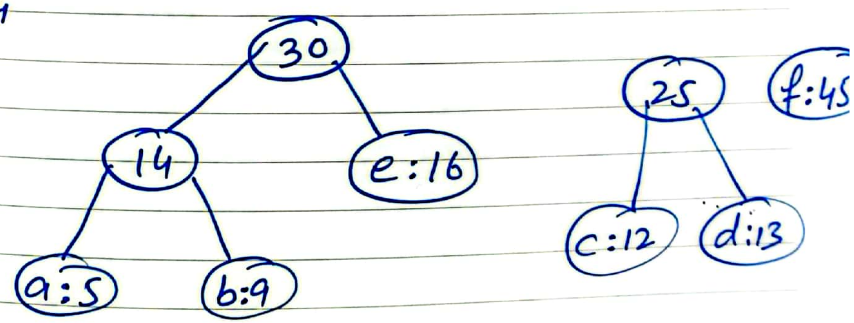
Step-2



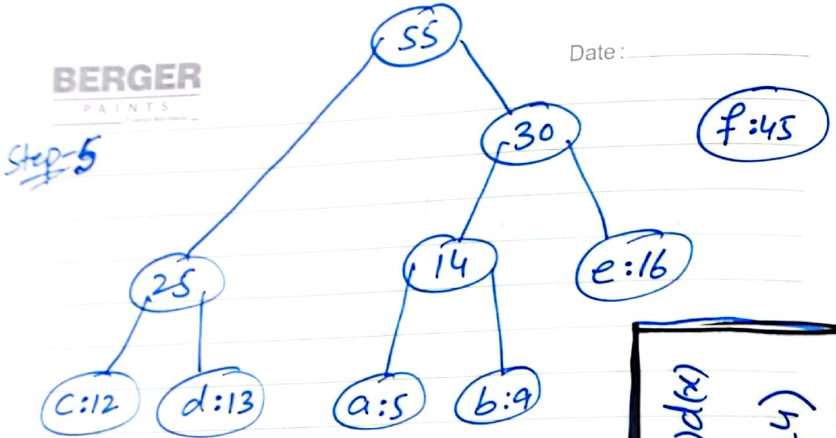
Step-3



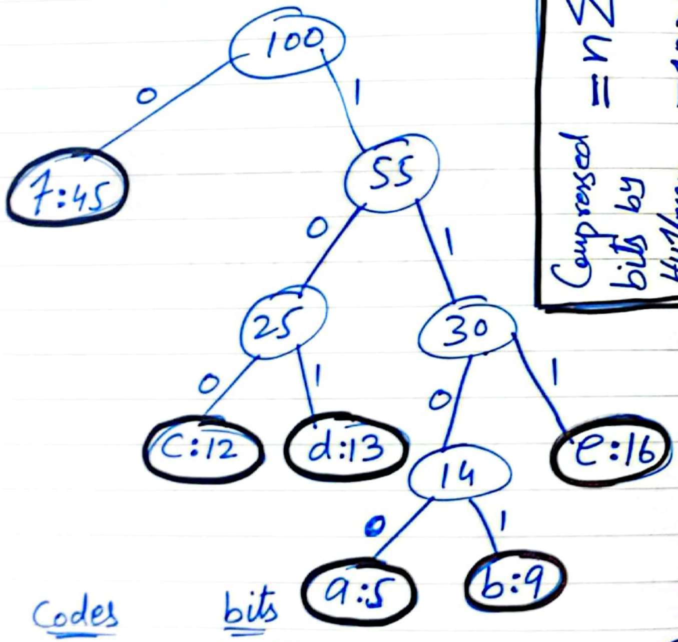
Step-4



Step-5



Step-6



Compressed bits by Huffman encoding $B(T) = n\sum P(x)d(x) = 100(2.24) = 224$ bits

	Codes	bits
$a \Rightarrow$	1100	$4\times5 = 20$
$b \Rightarrow$	1101	$4\times9 = 36$
$c \Rightarrow$	100	$3\times12 = 36$
$d \Rightarrow$	101	$3\times13 = 39$
$e \Rightarrow$	111	$3\times16 = 48$
$f \Rightarrow$	0	$1\times45 = 45$

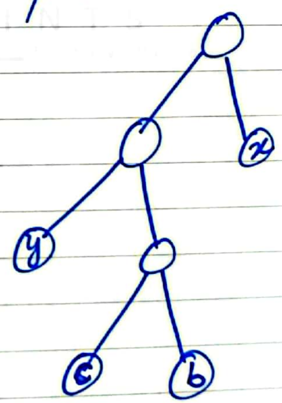
Total bits $= 224$

Proof of Correctness

Greedy choic Property:-

Let two characters 'x' and 'y' with smallest probabilities, Then there is optimal code tree in which 'x' and 'y' are siblings at maximum depth.

Proof Let 'T' be any optimal prefix code tree with two siblings 'b' and 'c' at max. depth.



As

$$P(b) \geq P(x)$$

$$P(b) - P(x) \geq 0 \quad \text{--- (1)}$$

&

$$d(b) \geq d(x)$$

$$d(b) - d(x) \geq 0 \quad \text{--- (2)}$$

Multiplying ① and ②

$$[P(b)-P(x)][d(b)-d(x)] \geq 0 \quad ③$$

Now try to change our own Tree "T" to prefix code tree of Huffman for that, swap 'x' with 'b' to be 'x' (with Lowest probability/freq.) at more depth.

Now tree becomes "T′"

$$B(T') = B(T) - P(x)d(x) + P(b)d(x) - P(b)d(b) + P(x)d(b)$$

$$= B(T) - [P(b)-P(x)][d(b)-d(x)]$$

$B(T') = B(T) - (+\text{ve factor})$

Implying that

T′ is optimal.

By swapping y and c in T, we'll get T″. So using previous argument we can say T″ is also optimal.

So This shows that greedy choice made by Huffman algo is proper one.

Optimal Substructure:-

(Proof by Induction)

Base Case:-

For base case,

$n=1$ or $n=2$

$n=1$	$n=2$
Tree consist of single leaf node (just 1 character)	Tree consist of two nodes
So that 1 character can be coded as '0' or '1'	code assigned to 1^{st} character = 0, 2^{nd} character = 1
So code length for that single character is 1 bit (which is optimal)	Again code length for them is 1 bit. (Obviously optimal for compression)

Inductive hypothesis:-

"Let Huffman tree for "n-1" charaters is optimal."

Inductive Case:-

we want to prove that Huffman tree is also optimal for exactly "n" charaters.

if there are "n" charaters in "T"

Now Remove two characters (say 'x' and 'y') with lowest propabilities at max depth; and replace with charater z such that

$$P(z) = P(x) + P(y)$$

Now there are "n-1" characters in tree (say "T′")

we can convert this tree T′ to the tree of 'n' characters T by replacing z with nodes x and y

Now cost of tree T is

$$B(T) = B(T') - P(z)d(z) + P(x)[d(z)+1] + P(y)[d(z)+1]$$

$$= B(T') - [P(x)+P(y)]d(z) + (d(z)+1)[P(x)+P(y)]$$

$$= B(T') + [P(x)+P(y)][\cancel{-d(z)}+\cancel{d(z)}+1]$$

$$B(T) = B(T') + [\underbrace{P(x)+P(y)}_{\text{(no depth involved)}}]$$

cost changes but change depends in no way on ~~struct~~ depth of tree

As our inductive hypothesis say that tree (T′) built optimal for n-1 characters [B(T′)]

So tree for n charaters will also be optimal (According to above equation)