Huffman Coding

↳ lossless algorithm
data compression

string → "aaabcdaag" → 7 x 2 bytes or
7 x 2 x 8 = 112 bits
size of file

---

2 Hashmaps :

encoder

char	string
a	
b	
c	
d	

decoder

string	char
	a
	b
	c
	d

steps :

① pass the main string that we wanna compress
(data, aka feeder)

② make a frequency map → (a is used sixty times)

a	60
b	30
c	8
d	2

(3) For every key in a frequency map, create a node and insert that node in a min heap / priority queue.

Node data : char data; int cost → frequency

a 60

It will also have node left and right

a	60
X	X

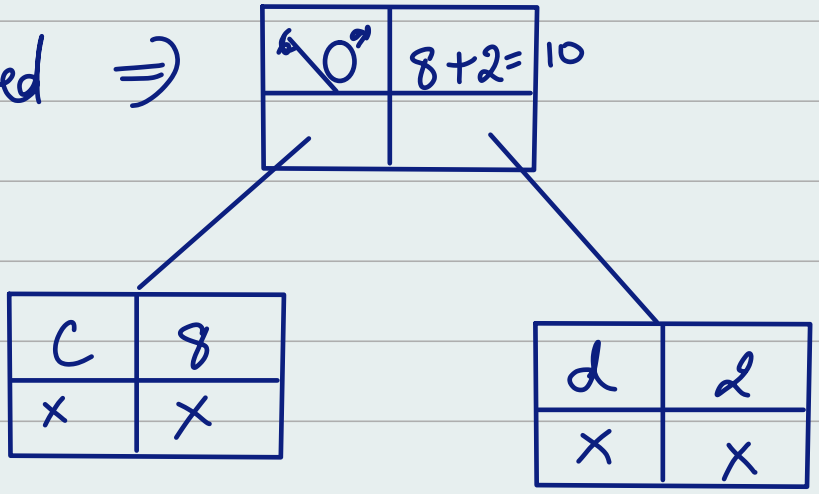
b	30
X	X

c	8
X	X

d	2
X	X

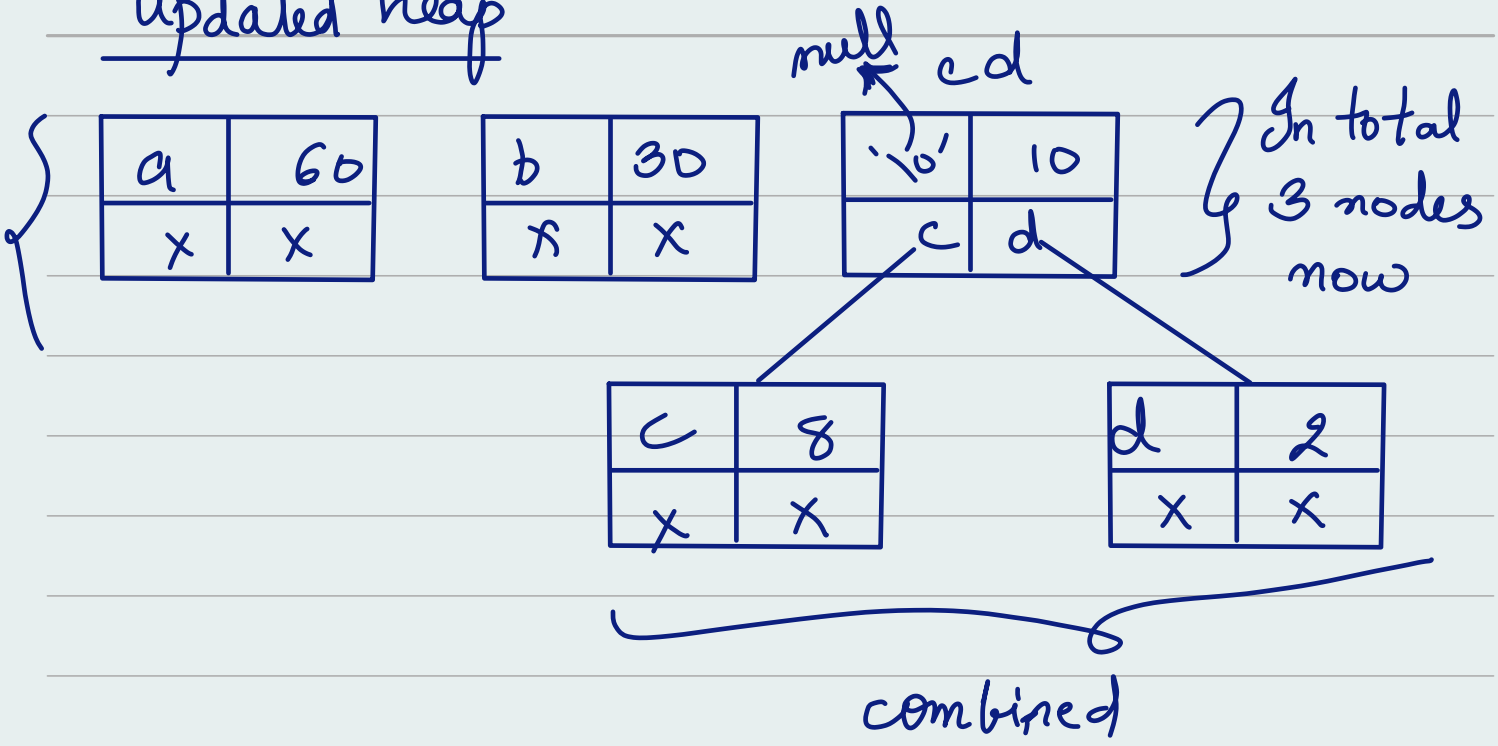
(4) Remove 2 elements from heap & combine (c & D will be removed because it's min heap).

c, d removed ⇒

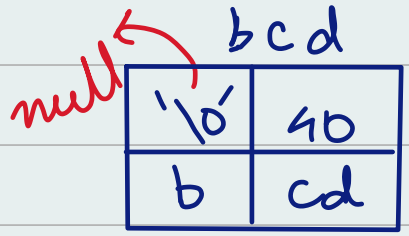


This can also be d & c (Interchanged) is fine

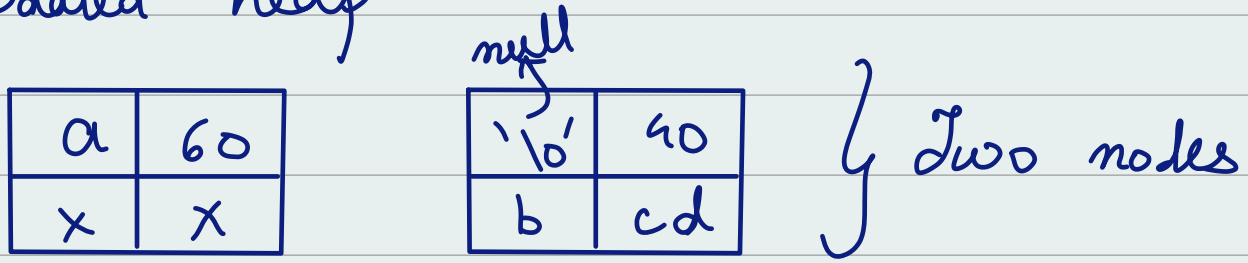
updated heap:



c d & b removed:

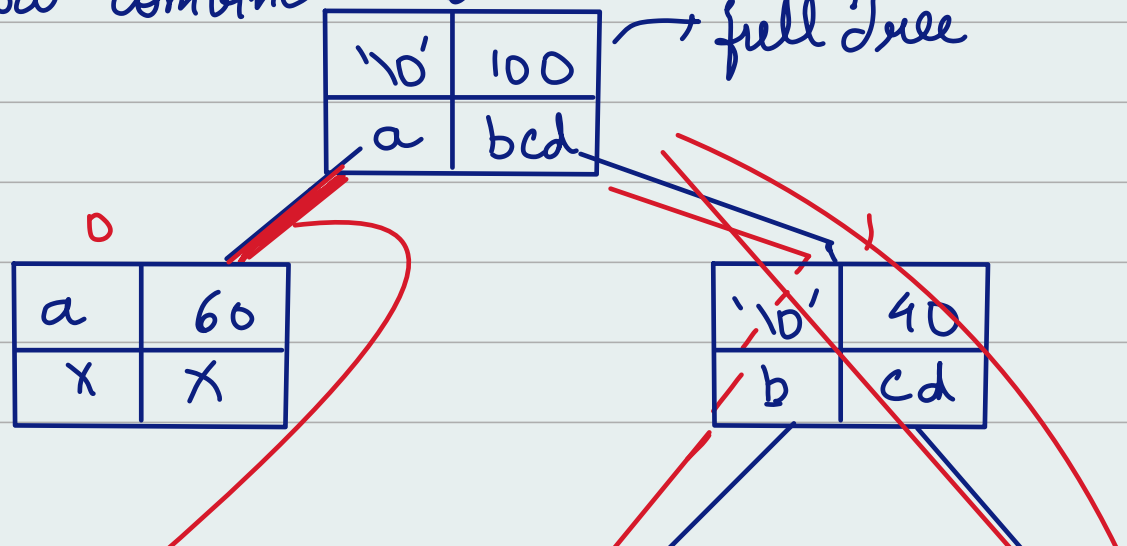


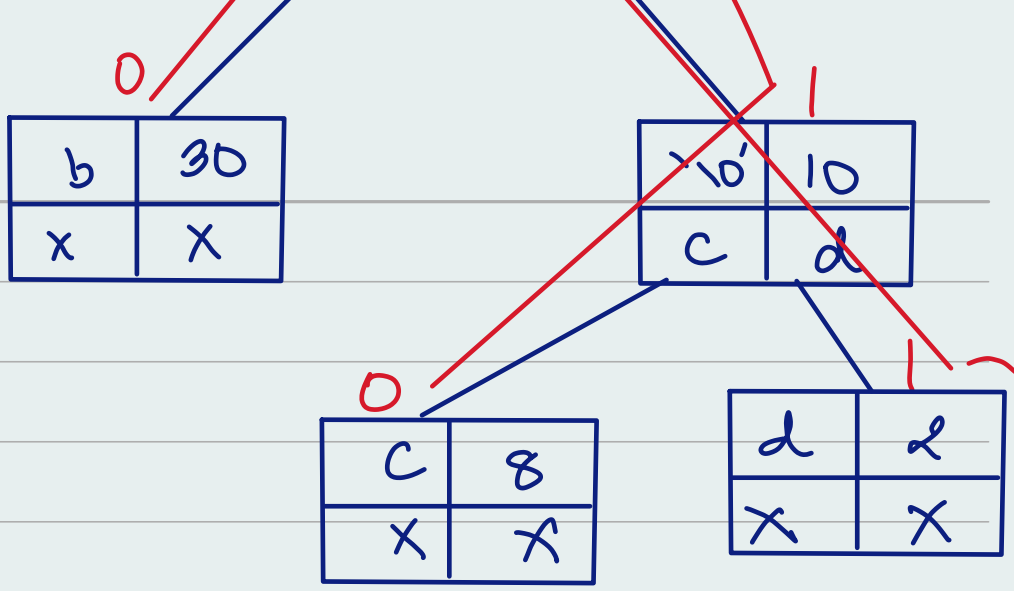
updated heap



Now combined: abcd

(5)





all the left will be zero and the right will be 1

encoder

a	0
b	10
c	110
d	111

here no value is prefix of another value
a → 0 & if a started from 0 no other element started from zero,
similarly b started with 10 no other element started with 10 everyone is unique because of the tree structure

decoder

0	a
10	b
110	c
111	d

(6)

(7) How to encode & decode?

string = a b b c c d a
= 112 bits (7 x 2 x 8)

a b b c c d a
↓
0 10 10 110 110 111 0 → 15 bits
instead of 112 bits

- Now converting back to original message

→ Does 0 exist ; yes → a (In the encoder)

→ Does 1 exist ; no Does 10 exist yes → b

→ again 10 → b similarly 11 → Not existing

110 → yes → c 11 → yes → c

011 → no 111 → yes → d

0 → yes → a

a b b c a d a

Space Complexity → O (N)

Time Complexity = 2 * (n-1) times

(Here every element is inserted or removed twice but not the very first element that is why (n-1))

≈ O[n log (N)] → for in heap for insertion or deletion

→ Code :-