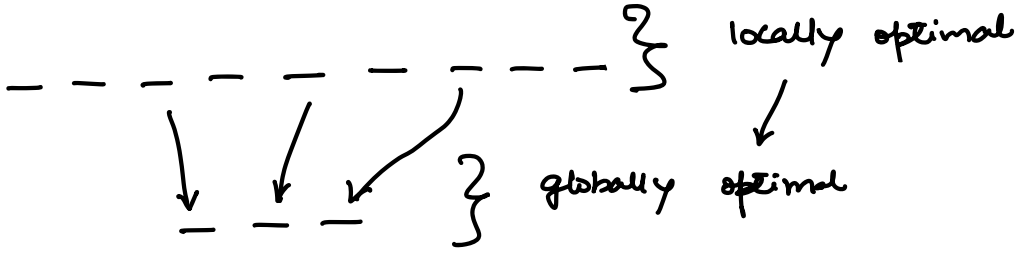
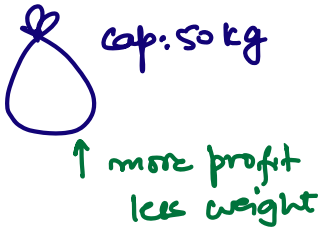
GREEDY ALGORITHM

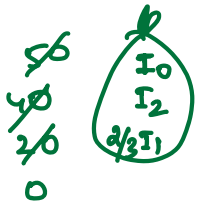


KNAPSACK

- 0/1 Knapsack (DP)
- Fractional Knapsack (Greedy)

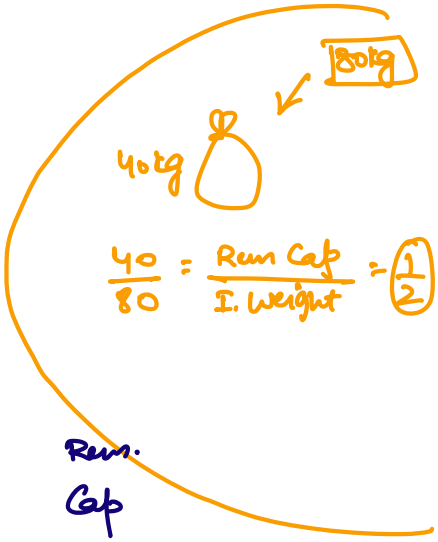
	I_0	I_1	I_2
Price :	60	120	100
Weight :	10	30	20
$\frac{\text{Price}}{\text{weight}}$:	6	4	5





Profit : 60 + 100 + $\frac{2}{3}$ · 120

$\frac{20}{30} = \frac{2}{3}$



Items	Price	Total Profit	weight	Rem. Cap
		0		50
I_0	60	60	10	40
I_2	100	160	20	20
$\frac{2}{3} \cdot I_1$	$\frac{2}{3} \cdot 120 = 80$	240	$\frac{2}{3} \cdot 30 = 20$	0

↳ Time Complexity:

Sort + Loop
$n\log n + n$
$= O(n\log n)$

a b c d e f

true: no swapping
false: swapping

a b

5 4

ra > rb

a b

4 5

ra > rb
4 > 5 : false

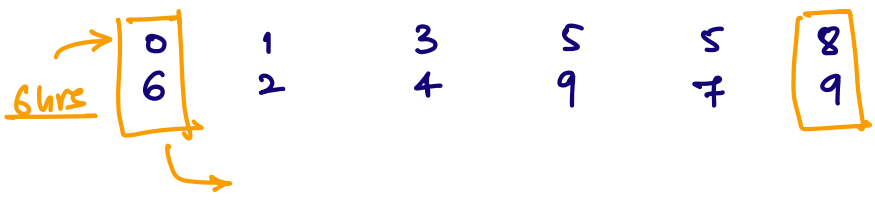
5 4
b a

ACTIVITY SELECTION PROBLEM

	A_0	A_1	A_2	A_3	A_4	A_5
Start:	5	1	3	0	5	8
finish:	9	2	4	6	7	9

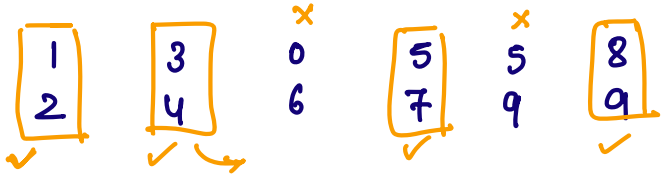
maximum no. of activities that you can perform?

Method 1: Sort the activities (classes) on basis of start time.



2 classes created

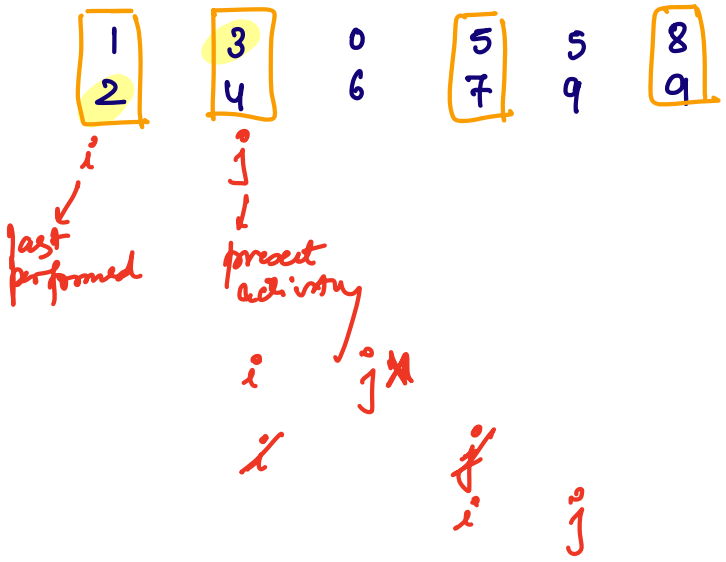
Method 2: Sort finish time basis



: 4 classes

	a	b	
finish:	4	6	: no swapping

4 < 6 : true



TC: sort + loop

nlogn + n

= O(nlogn)

JOB SEQUENCING PROBLEM

Q4. (a) Find the greedy solution for the following job sequencing problem with deadlines with a total number of jobs n = 7.

Job S. No	1	2	3	4	5	6	7
Profit	3	5	20	18	1	6	30
Deadline	1	3	4	3	2	1	2

74

Ques 4:

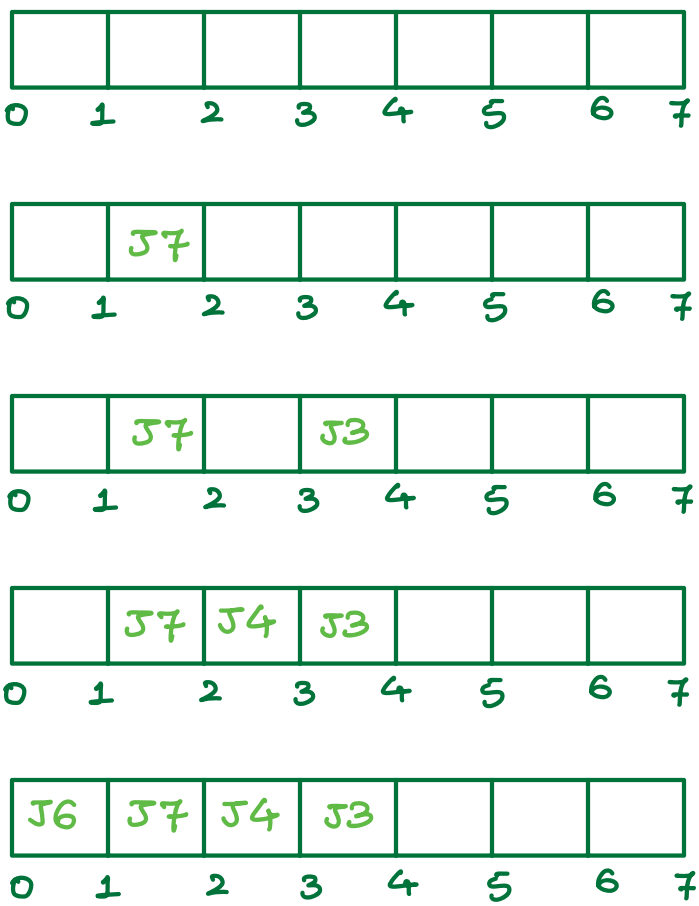
a).

Jobs	J1	J2	J3	J4	J5	J6	J7
Profits	3	5	20	18	1	6	30
Deadlines	1	3	4	3	2	1	2

- Sort the jobs in decreasing order of profit

Jobs	J7	J3	J4	J6	J2	J1	J5
Profits	30	20	18	6	5	3	1
Deadlines	2	4	3	1	3	1	2

- Iterate over the jobs and assign the last slot available



J2, J1, J5 cannot be completed because deadlines are 3, 1, 2 respectively and all slots are occupied till 3.

Profit = 74

HUFFMAN CODING → GREEDY ALGO.

- Lossless data compression algorithm.

- Idea is to assign variable length codes to input characters.
  Length of codes is based on frequencies of corresponding characters.

"abca"
↓
110 011 111 110
4×3 = 12

a → 110
b → 011
c → 111

"abca"
"0 110 100"
≠

a → 0
b → 110
c → 10

- Variable length codes assigned to input characters are 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 other character.

a → 0
b → 01
c → 1
✗

01
↙ ↘
b ac ?

↓
This is how huffman coding make sure there is no ambiguity while decoding.

You should know about frequently occuring characters.

characters	frequency
a	4
b	2
c	1
d	5
e	6

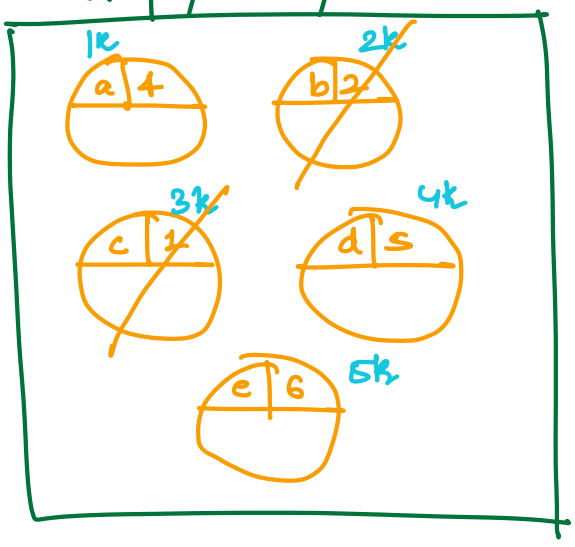
English:
vowels are more frequently occuring as comp. to consonants.

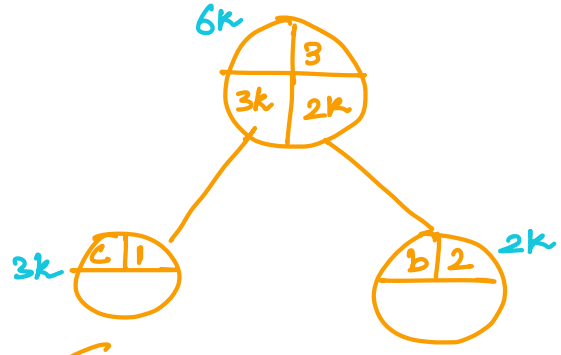
assign codes ?

Implementation:

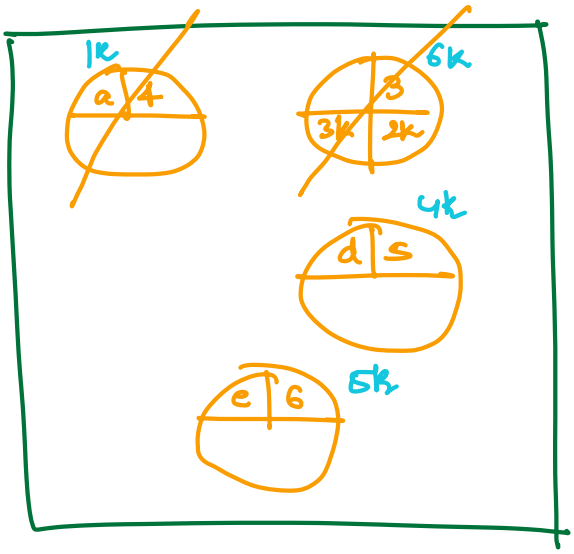
Node:

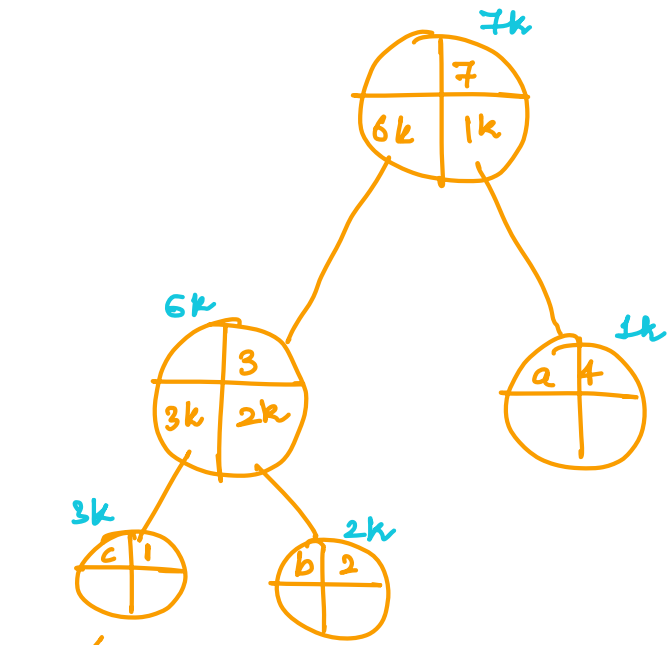
(Min) Heap / Priority Queue



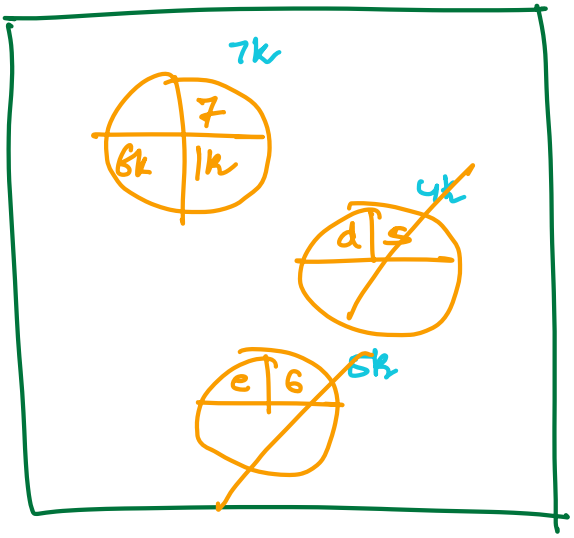


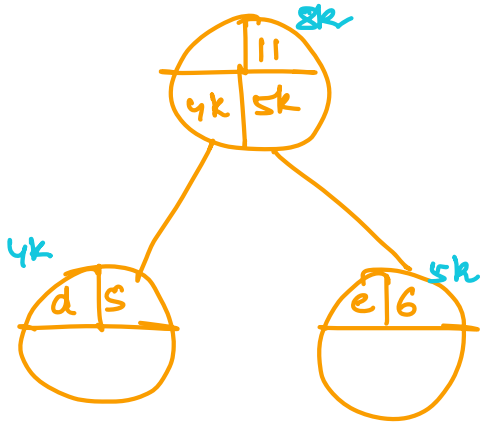
put it back

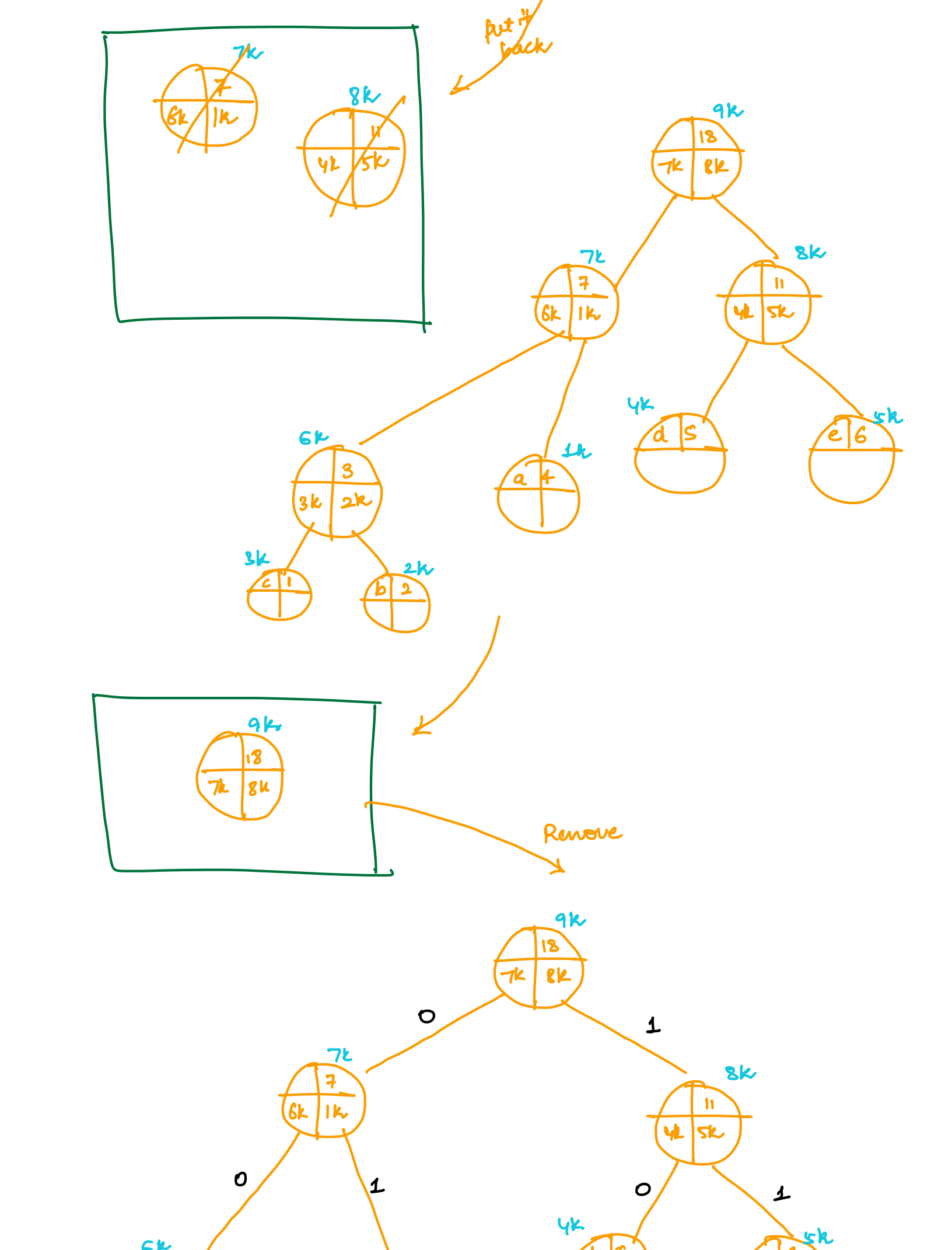


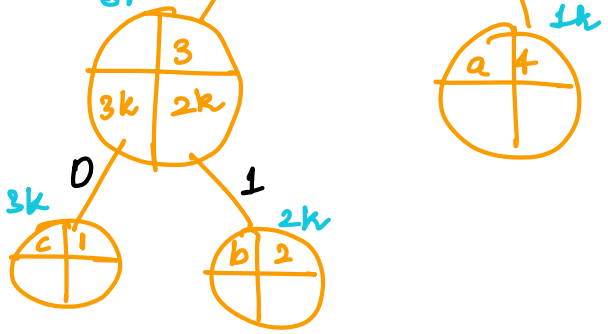


put it back






put it back
7k
7
6k
1k
8k
11
4k
5k
9k
18
7k
8k
7k
7
6k
1k
8k
11
4k
5k
4k
d
5
5k
e
6
6k
3
3k
2k
1k
a
4
3k
c
1
2k
b
2
9k
18
7k
8k
Remove
9k
18
7k
8k
0
1
7k
7
6k
1k
8k
11
4k
5k
0
1
0
1
4k
5k



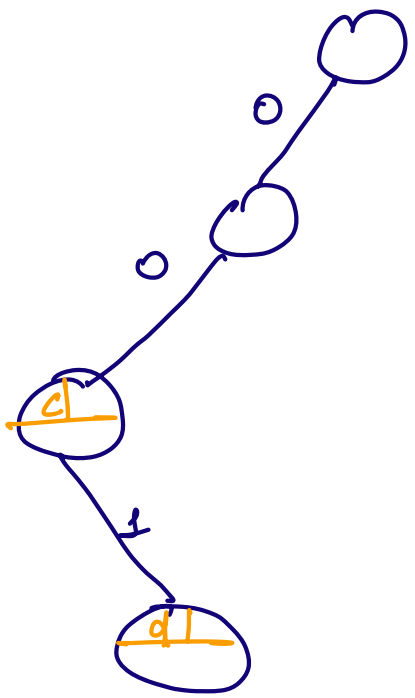
c → 000 } larger length code
b → 001
a → 01 } smaller length code
d → 10
e → 11

Encoding:

adc

01 10 000 = 011000

c → 00
d → 001] X



c → 000
b → 001
a → 01
d → 10
e → 11

Decoding:

011000

a d c

Application:

- fax or text transmitting
- Compression format: bzip, Gzip
- Multimedia codecs: jpeg, png, mp3 prefix code