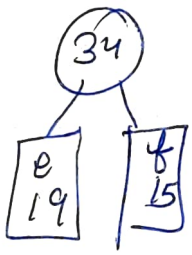
Tutorial sheet-7

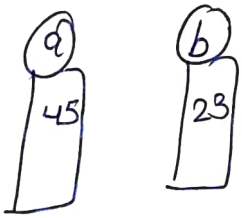
Sol 1: It is an algorithm paradigm that builds up a solution by adjoining smaller pieces together, always choosing the next piece that offers the most obvious & immediate benefit.

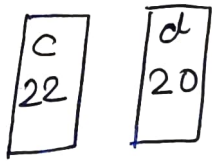
Sol 2:

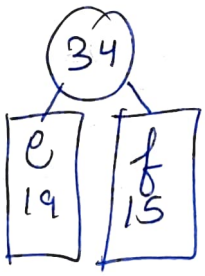
Name	TC	SC
Activity selection	$O(n \log n) - O(n)$	$O(n)$
Job sequencing	$O(n^2) - O(n \log n)$	$O(n)$
Fractional knapsack	$O(n \log n) - O(n)$	$O(n)$
Huffman Encoding	$O(n \log n) - O(\log n)$	$O(n)$

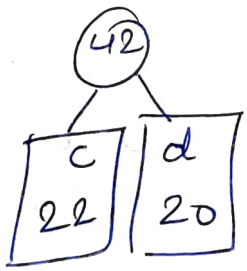
Sol 3: a = 45, d = 20, b = 23, e = 19, c = 22, f = 15

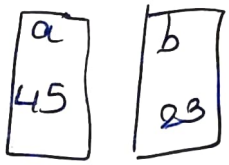


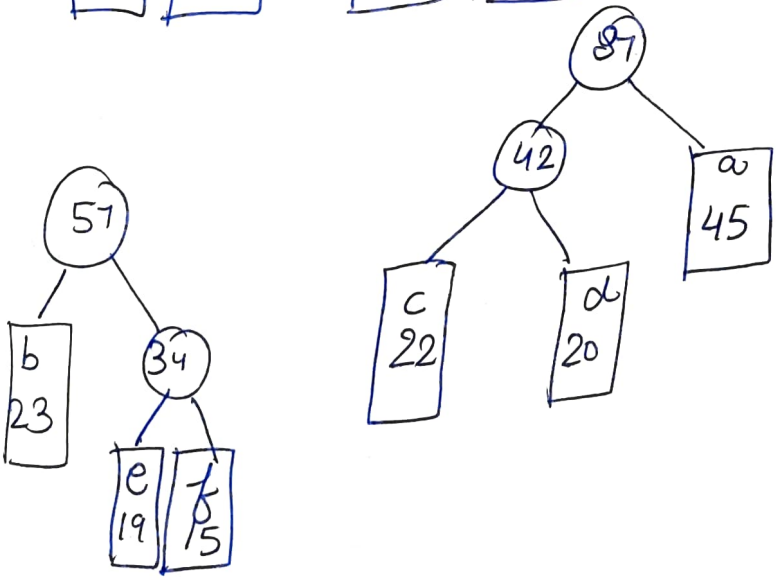




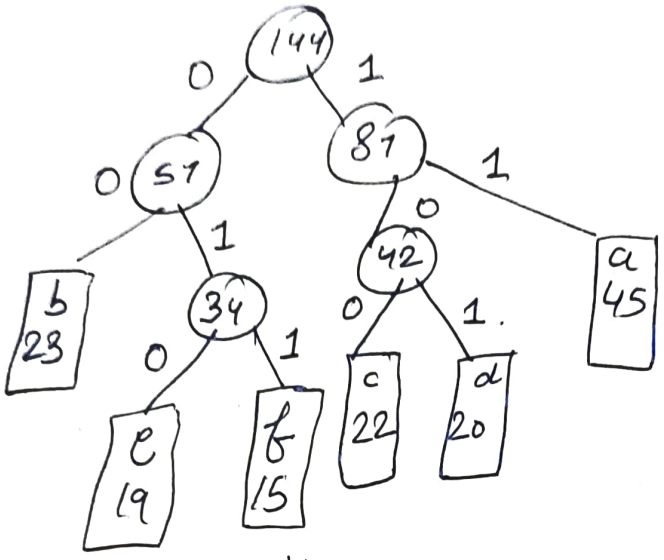








Total bits = (45+2) + (23+2) + (22+3) + (20+3) + (19+3) + (15+3) = 364 bits



a = 11
b = 00
c = 100
d = 101
e = 010
f = 011

Sol4⇒ A 2-tree is used to implement Huffman encoding algorithm. It is a binary tree where every node has either 2-child or no child.

- Applications of Huffman Encoding :-
  - Data compression in long files without any loss.
  - To implement traffic routes with traffic magnitude.

Sol5⇒

V	10	5	15	7	6	18	3
ω	2	3	5	7	1	4	1
V/ω	5	1.67	3	1	6	4.5	3

$k = 15 - 1 - 2 - 4 - 5 - 1 - 2 = 0$

Profit $= 30 + 10 + 18 + 15 + 3 + 3.34$

$= 79.34$

V	6	10	18	15	3	5
w	1	2	4	5	1	3
V/w	6	5	4.5	3	3	1.67

Sol 6 :- fractional knapsack : It is using greedy approach as we have divided our profits to the smallest unit possible & then builds upon it.

huffman Encoding : It is using the greedy approach as we have divided our profits to the smallest unit possible & then builds upon it.

huffman Encoding - It is using the greedy approach as it always places the node with the lower frequency further from the parent node.

Sol 7 =>

Start	1	2	0	6	9	10
End	3	5	7	8	11	12
Index	0	1	2	3	4	5

Jobs to do = [0], [3], [4] or [5]

i.e. => Max = 4.

Sol 8 ⇒

	Profit	Deadline
a	20	2
b	15	2
c	10	1
d	5	3
e	1	3

0	1	2
b	a	d

0 1 2 3

Profit = 20+15+5
= 40

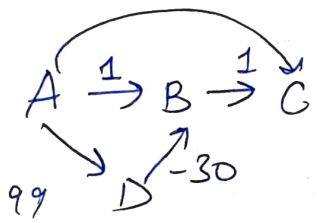
Sol 9 ⇒ Times when not to use greedy algorithm :-

(1) When approach involves a lot of assumptions, such as "pick always then".

(2) When we need compiler implementation.

(3) When we are making performance-critical application.

eg : Dijkstra's algo is very unoptimised for graphs with negative edges. Here, we cannot find distance of pair [A, C] → it gives 0, though it is -200.



Sol 10 ⇒ T.C. of job sequencing = O(n2), but we can improve it using priority queue by ~~algorithm~~ using (Max Heap).

Algorithm :-

(1) sort based on deadlines

(2) Iterate the end & calculate the available slots b/w two consecutive deadlines put everything in max heap.

(3) If slots available & there are jobs in max. Heap, include JOB ID with max profit & deadlines in the result.

(4) Sort the array based on deadline.

$$TC = O(n \log n); \quad SC = O(n).$$