Design & Analysis of Algorithms

Tutorial - 7

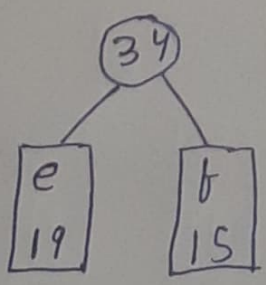
Name → MOHAK KALA
Section → D
Class R.No. → 29
Univ. R. NO. => 2014727

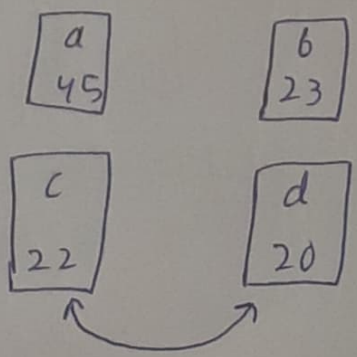
① It is an algorithmic paradigm that builds up a solution by adjoining smaller pieces together, always choosing the next piece that offers the most obvious and immediate benefit.
We should use greedy approach wherever a locally optimal solution is also globally optimal

②

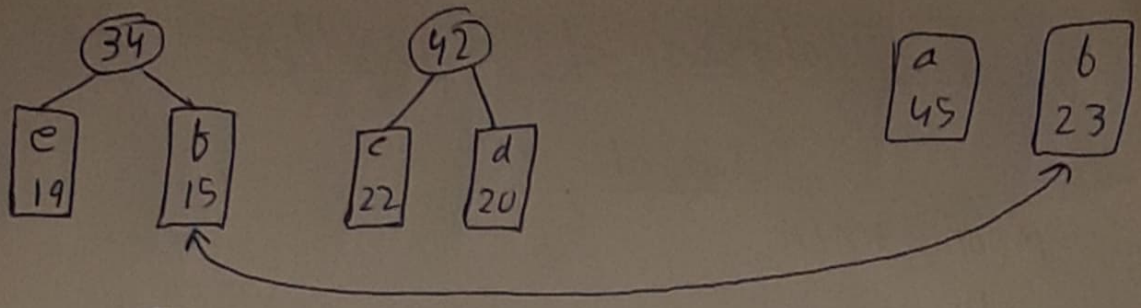
Name	TC	SC
Activity Selection	$O(n \log n) \longleftrightarrow O(n)$	$O(n)$
Job Sequencing	$O(n^2) \longleftrightarrow O(n \log n)$	$O(n)$
Fractional knapsack	$O(n \log n) \longleftrightarrow O(n)$	$O(n)$
Huffman Encoding	$O(n \log n) \longleftrightarrow O(\log n)$	$O(n)$

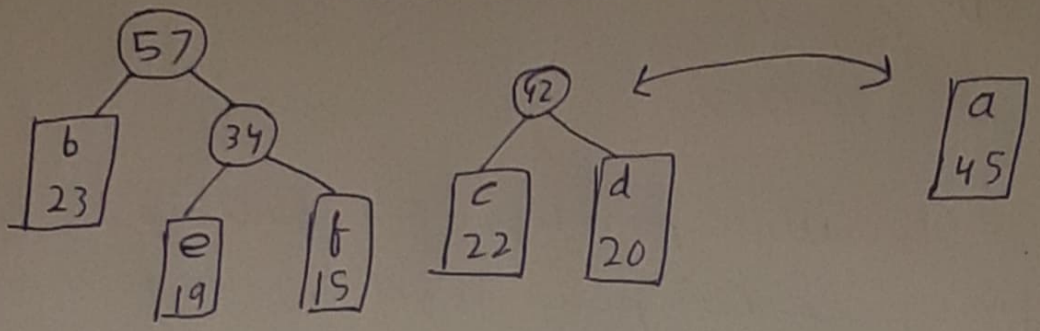
③ a = 45, d = 20, b = 23, e = 19, c = 22, f = 15

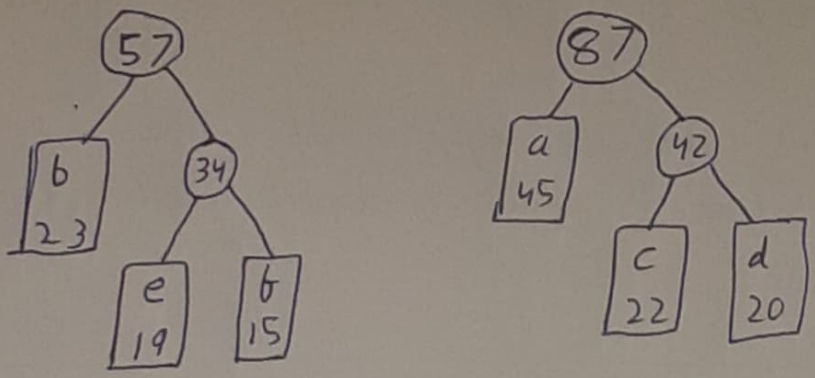




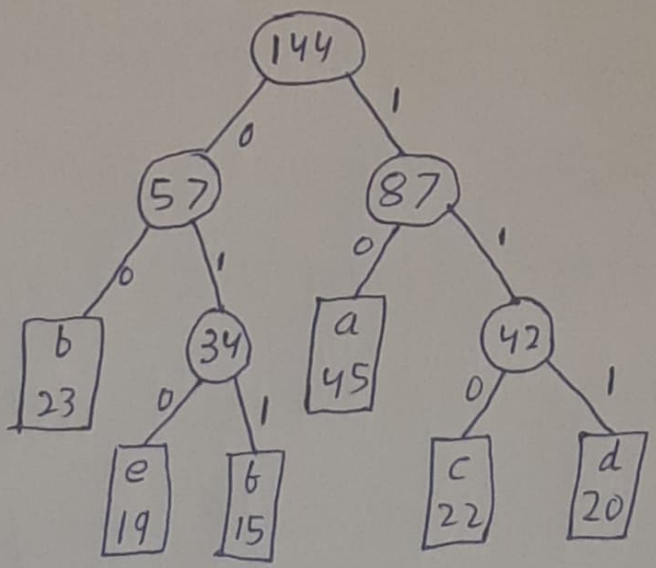
(2)







Final Huffman Tree =>



a = 10
b = 00
c = 110
d = 111
e = 010
b = 111

Total bits Used => (45 x 2) + (23 x 2) + (22 x 3) + (20 x 3)
+ (19 x 3) + (15 x 3)

= 364 bits

④ 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 route with traffic magnitude

Ans-5

V	10	5	15	7	6	18	3
w	2	3	5	7	1	4	1
V/w	5	1·67	3	1	6	4·5	3

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

$\text{Profit} = 30 + 10 + 18 + 15 + 3 + 3 \cdot 34$

$= 79 \cdot 34$

V	6	10	18	15	3	5
w	1	2	4	5	1	3
V/w	6	5	4·5	3	3	1·67

Ans-6 Fractional knapsack: It is using a 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.

(4)

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

Ans-7

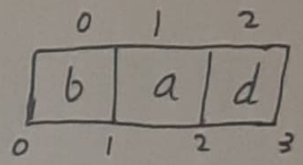
Start	1	2	0	6	9	10
End	3	5	7	8	11	12
Index	0	1	2	3	4	5

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

i.e. => Max = 4

Ans - 8

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

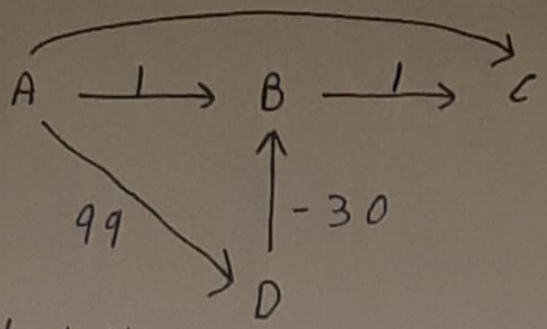


Profit = 20 + 15 + 5
= 40

Ans-9 Times when not to use greedy approach

- When the approach involves a lot of consumptions, a such as "pick always the ------"
- We should avoid greedy approach on complex implementation.
- When we are making performance-critical applications.

Eg => Dijkstra's algorithm is very unoptimized for graphs with negative edges.



We can't find the distance of the pair [A, C] → it gives 0, though it is -200

Ans - 10 Normally, the time complexity of Job sequencing is $O(n^2)$ but we can improve it using a Priority Queue, made of Max Heap.

Algorithm :-

1. Sort the job based on deadlines
2. Iterate the ord & calculate the available slots b/w 2 consecutive deadlines include all data in Max-Heap
3. If there are slots available & there are jobs in the max heap, include the job ID with max profit & deadlines in the result.

Sort the array based on deadlines

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

Space Complexity => $O(n)$