Tutorial 7

Ayank Gupta

Section B

Ans 1 ⇒ Greedy Algo Paradigm ⇒ Greedy is an algorithm paradigm that builds up a solution piece by piece, always choosing the next piece that offers most obvious and immediate benefits.

- Applications of Greedy ⇒
  1. CPU scheduling
  2. minimum spannig trees.

Ans 2 ⇒

	Activity Selection	Job Sequencing	Fractional Knapsack	Huffman Coding
Time Complexity	$O(n\log n)$	$O(n^2)$	$O(n\log n)$	$O(n\log n)$
Space complexity	$O(n)$	$O(n)$	$O(n)$	$O(n)$

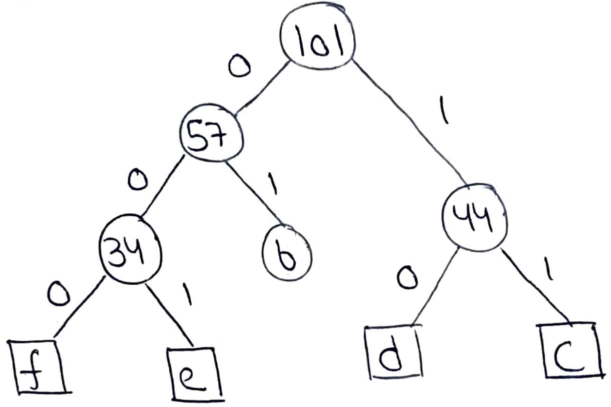
Ans 3 ⇒ Huffman Coding ⇒

char	freq.
a	45
b	23
c	22
d	20
e	19
f	16

* Sorted freq. ⇒

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

* Huffman Tree ⇒



f = 000

e = 001

b = 01

d = 10

c = 11

Ans4 ⇒ DS used for Huffman Coding ⇒ Binary tree is used for building Huffman coding and its also used for Huffman Encoding.

⊙ Applications ⇒ (1) Huffman code is used to convert fixed length codes into variable length codes which result in lossless compression.

(2) Compressed codes may be further compressed using JPEG and MPEG.

Ans5 ⇒ weight = 15

value	10	5	15	7	6	18	3
wt	2	3	5	7	1	4	1

value/wt ⇒ 5 1.6 3 1 6 4.5 3

- choose highest v/w ratio for which wt $w \leq W$.

- let current weight = c

x =

0	0	0	0	1	0	0	c=1

x =

1	0	0	0	1	0	0	c=3

x =

1	0	0	1	1	0	0	c=7

⋮

⇒ next highest wt. is 1.6 which has 3 unit wt., but hence we will fraction it as per requirement.

- req. wt. = W - C ⇒ 15 - 13 = 2

hence wt. added = 2/3

xw ⇒

2	2	5	0	1	4	1

vx ⇒

10	3.3	15	0	6	18	3

- maxProfit / maxValue ⇒ 55

Ans6 => KnapSack Algo. => To solve the problem we take the value/wt ratio and on the basis of this ratio a wt. i, which has highest v/w ratio, added to knapsack. until we can't add the next wt. as a whole and that point of time we take the req. fraction of wt. and it to knapsack. This is nothing but greedy approach of taking highest ratio everytime.

○ Huffman Coding => It is based on the freq. of the character. we assign the var. length code. to i/p char. length of the assigned codes are based on the freq. of corresponding characters.

Ans7 =>

	a	b	c	d	e	f
startTime	1	2	0	6	9	10
endTime	3	5	7	8	11	12

○ included process => a, d, e

max. no. of process = 3

Ans9 => Greedy Algos are not suitable for problem where a soln. is req. for every subproblem like sorting. In such problms greedy problems can be wrong.