Tutorial-7

Q1) What is the Greedy algorithmic paradigm? When should you make use of greedy algorithms in problem solving?

→ Greedy is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. This means that it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution.

A problem must comprise these 3 components for a greedy algorithm to work:

① It has optimal substructures. The optimal solution for the problem contains optimal solutions to the sub-problems.

② It has a greedy property (hard to prove its correctness). If you make a choice that seems the best at that moment & solve the remaining sub-problems later, you still reach an optimal solution. You'll never have to reconsider your earlier choices.

Q2) Analyse the time & space complexity of the following algorithms:
(i) Activity Selection, (ii) Job sequencing, (iii) fractional knapsack, (iv) Huffman Encoding.

→

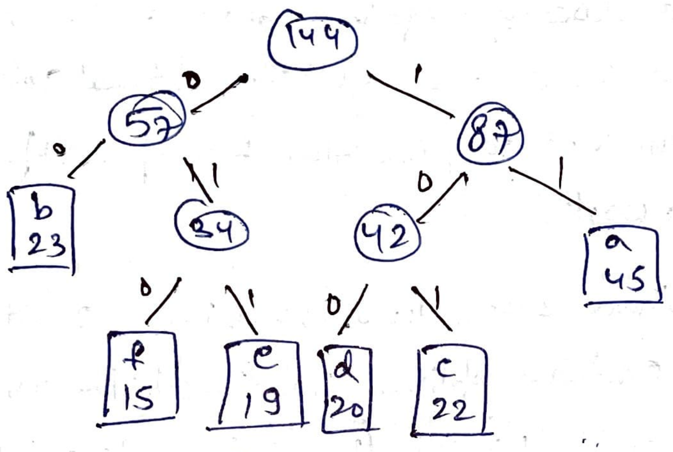
Algorithm	Time Complexity	Space Complexity
Activity Selection	$O(n \log n)$: Unsorted $O(n)$: Sorted	$O(1)$
Job sequencing	$O(n^2)$ $O(n \log n)$: Priority Queue	$O(n)$
fractional knapsack	$O(n \log n)$	$O(1)$
Huffman Encoding	$O(n \log n)$	$O(n)$

Q3) A file contains the following characters and their corresponding frequencies as shown below:

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

We use Huffman coding for data compression. Generate the encoding for a, b, c, d, e, f using Huffman encoding and find the average length of a character after compression.

→



Huffman's tree

Char	freq.	Huffman's Code	No of bits
a	45	11	2×45=90
b	23	00	2×23=46
c	22	101	3×22=66
d	20	100	3×20=60
e	19	011	3×19=57
f	15	010	3×15=45
Total	144	—	364

Average length of a character = $\frac{364}{144} \Rightarrow \boxed{2.52}$

Q4) What data structure is used while implementing Huffman Encoding? What are the application of Huffman Encoding?

→ Priority Queue is used for building the Huffman tree such that nodes with lowest frequency have the highest priority.

A min-heap data structure can be used to implement the functionality of a priority queue.

Applications

- Huffman Encoding is widely used in compression formats like GZIP, PKZIP (winzip) and BZIP2.
- Multimedia codes like JPEG, PNG & MP3 use Huffman Encoding.
- Huffman Encoding is used for transmitting fax and text.

Q5) Given weights and values of 7 items, put these items in a knapsack of capacity W = 15 such that you get the maximum total value in the knapsack:

Value	10	5	15	7	6	18	3
Weight	2	3	5	7	1	4	1

→ Applying fractional knapsack algorithm,

Value	Weight	V/W
6	1	6
10	2	5
18	4	4.5
15	5	3
3	1	3
5	3	1.66
7	7	1

Sorted in decreasing order of value per weight

$W = 15 - 1 = 14 - 2 = 12 - 4 = 8 - 5 = 3 - 1 = 2 - 2 = 0$

Max total Value = 0 + 6 + 10 + 18 + 15 + 3 + (1.66 * 2)

⇒ $\boxed{55.33 \text{ units}}$

Q6) Prove that fractional knapsack problem and Huffman Encoding has the greedy-choice property. (Why these algorithms are cat. as greedy ?)

→ • In fractional knapsack problem, the basic idea of the greedy approach is to calculate the ratio value/weight for each item & sort the items on the basis of this ratio. Then, take the item with the highst ratio & add them until we can't add the next item as a whole & at the end, add the next item as much as we can.

• In Huffman Encoding, the algorithm builds the tree 'T' analogous to the optimal code in a bottom up manner. It starts with a set of |c| leaves ('c' is the no. of characters) and performs (|c|-1) 'merging' operations to create the final tree. Huffman's greedy algorithm uses a table of the frequencies of each character to build up an optimal way of representing each character as a binary string.

Q7) Consider a set of activities given below, along with starting & finishing time of each activity. find the maximum number of activities performs by a single person assuming by a single person assuming that a person can work on a single activity at a time :-

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

→ After sorting activities in increasing order of end time,

Start time	1	6	9
End time	3	8	11
	0	1	2

for these 3 activities,
start_time [i] >= end_time [j]; (i>j)

∴ $\boxed{\text{Max Activities = 3}}$

Q8) Consider the following jobs where every job has a deadline and an associated profit if completed within deadline. It is given that every job takes a single unit of time. Perform Job sequencing such that you maximizing the total profit if only one job can be scheduled at a time:

	A	B	C	D	E
Profit	20	15	10	5	1
Deadline	2	2	1	3	3

→ After sorting jobs in decreasing order of their profits,

∵ Max. deadline = 3

∴ Max. array size = 3

Job Schedule

C	A	D

0 1 2 3

Deadline →

Max. total profit = 10 + 20 + 5

⇒ $\boxed{35 \text{ units}}$

Q9) When should we avoid making use of greedy approach in problem solving? Give examples to support your explanation.

→ Sometimes greedy algorithms fail to find the globally optimal solution because they Don't consider all the data. The choice made by a greedy algorithm may depend on choices it has made so far, but it is not aware of future choices it could make. e.g. let us consider that the capacity of a knapsack is 25 ($W = 25$) & the items are as shown in the following table:

	A	B	C	D
Profit	24	18	18	10
Weight	24	10	10	7

Without considering the profit per unit weight (p_i/w_i) if we apply greedy approach to solve this problem first item 'A' will be selected as it'll contribute maximum profit among all the elements.

After selecting 'A', no more ~~to~~ item will be selected.

Hence, for this given set of items, total profit is 24, whereas the optimal solution can be achieved by selecting items 'B' and 'C', where the total profit is 18+18=36.

Q10) How can you optimise the approach used to solve the Job sequencing problem? Write an algorithm for the same.

→ We can optimise the approach of solving Job sequencing problem by using Priority Queue (Max Heap).

Algorithm

1. Sort the jobs based on their deadlines.
2. Iterate from the end & calculate the available slots between every two consecutive deadlines. Include the profit, deadline, and job ID of i^{th} job in the max heap.
3. While the slots are available and there're jobs left in the max heap, include the job ID with maximum profit & deadline in the result.
4. Sort the result array based on their deadlines.