Tutorial - 7

① What is the Greedy algorithmic Paradigm? When should you make use of Greedy Algorithms in Problem solving?

Ans ① Greedy algorithmic Paradigm that builds up a solution Piece by Piece, always choosing the next Piece that offers the most obvious and immediate benefit.

② So the Problems where choosing locally optimal also leads to global solution are best fit for Greedy. For example Consider the Fractional Knapsack Problem.

③ Greedy algorithms are simple instinctive algorithms used for optimization, Problems.

② Analyse the time and space complexity of the following algorithms :-

		Time complexity	Space complexity
①	Activity selection →	O(n log n)	O(n)
②	Job sequencing →	O(n log n)	O(n).
③	Fractional Knapsack →	O(n log n)	O(n)
④	Huffman Encoding →	O(n log K)	O(K)

③ Which data structure is used while implementing Huffman Encoding? What are the applications of Huffman Encoding?

Ans: A min Heap data structure can be used to implement the functionality of a Priority queue.

Huffman is widely used in all the mainstream compression formats that you might encounter - from GZIP, PKZIP and BZIP2 to image formats such as JPEG and PNG

AltGr Ctrl Home

④ Prove that Fractional Knapsack Problem and Huffman Encoding has the greedy-choice Property.

Ans/ Huffman encoding is an algorithm that follows the problem solving mechanism of making the local optimal solution at each stage. with thinking of finding a global optimum solution for the problem and In Huffman coding in every stage we try to find the prefix free binary code and try to minimize expected code word for optimum solution for compress the data.

Fractional Knapsack has the greedy property. let j be the item with maximum vi/wi. Then there exists an optimal solution in which you take as much of item j as possible. we can therefore take a piece of k, with ∈ weight out of the knapsack and put a piece of j with ∈ weight in.

⑤ when should we avoid making use of greedy approach in problem solving? Give examples to support your explanation.

Ans/ ① one of the biggest issue I can think of off-hand with employing a greedy algorithm is it is very easy for a hypothetical adversary to create instances that will lead with a carrot and a stick.

② The greedy algorithm to produce arbitrarily bad TSP tour.

Ex-1 suppose you may start with the MST, but then you need to add constraints that guarantee that the solution is a travelling salesperson tour. That's what the tree search does, basically.

Scanned with CamScanner

⑯ How can you optimize the approach used to solve the job sequencing Problem? Write an algorithm for the same.

Ans:- To optimize the approach used to solve the job sequencing is to generate all subsets of a given set of jobs and check individual subsets for the feasibility of jobs in that subset. Keep track of maximum Profit among all feasible subsets. The Time complexity of this solution is exponential. This is a standard Greedy Algorithm Problem.

Algorithm

- sort the jobs based on decreasing order of Profit.
- Create two variables, total job. = 0, maxProfit = 0.
- Also, find the maximum deadline among all the jobs.
- Initalise a set storing all the jobs in decreasing order.
- Iterate through the jobs and Perform the following -
- If the set is empty or the deadline of the current job is less than the last element of the set, ignore the job
- Else, apply binary search and find the nearest slot i, such that i< deadline and add the Profit.
- Increment total jobs by 1 and ~~ete~~ remove the ith element from the set.
- Return the maximum Profit.