# Tutorial - 7

1. Greedy algorithm Paradigm - It builds up a $sol^n$ piece by piece, always choosing the next piece that offers most obvious and immediate benefit. So the problems were choosing locally optimal also leads to good solution are best fit for greedy.
eg:- Fractional Knapsack.

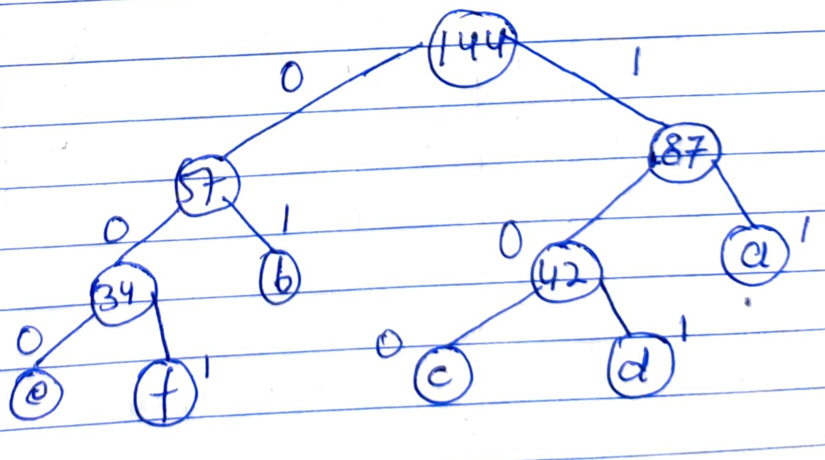
2.

| | Time Complexity | Space Complexity |
|---|---|---|
| Activity Selection | $O(n)$ | $O(1)$ |
| Job Sequencing | $O(n^2)$ | $O(n)$ |
| Fractional Knapsack | $O(n \log n)$ | $O(n)$ |
| Huffmann encoding | $O(n \log n)$ | $O(n)$ |

3.

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

a → 45 x
b → 23 x
c → 22 x
d → 20 x
e → 19 x
f → 15 x
ef → 34 x
cd → 42 x
bef → ~~5~~ 57
acd → 87
abcdef




a → 11 → 90
b → 01 → 46
c → 100 → 66
d → 101 → 60

e → 000 = 3×19 = 57

f → 011 = 3×15 = 45

Total = 364

4) Full binary tree is used while implementing Huffman encoding.

Application of Huffmann Encoding

1) They are used for transmitting text.

2) They are used by conventional compression formats like PKZIP, GZIP etc.

5.

| index | v | w | v/w |
|---|---|---|---|
| 1 | 10 | 2 | 5 |
| 2 | 5 | 3 | 1.6 |
| 3 | 15 | 5 | 3 |
| 4 | 7 | 7 | 1 |
| 5 | 6 | 1 | 6 |
| 6 | 18 | 4 | 4.5 |
| 7 | 3 | 1 | 3 |

| Sorted | v | w | v/w |
|---|---|---|---|
| 5 | 6 | 1 | 6 |
| 7 | 3 | 1 | 3 |
| 1 | 10 | 2 | 5 |
| 2 | 5 | 3 | 1.6 |
| 6 | 18 | 4 | 4.5 |
| 3 | 15 | 5 | 3 |
| 4 | 7 | 7 | 1 |

K = 15, (15 − 1 − 1 − 2 − 3 − 4 − 4) = 0

Profit = 6 + 3 + 10 + 4.8 + 4×3

= 35.8

6. Fractional knapsack has greedy choice property which states that an optimal solution to a problem can be obtained by making local best choices at each step of the algorithm.

Now my proof assumes that there's an optimal sol$^n$ to the fractional knapsack problem that does not include a greedy choice and then tries to a contradiction.

Proof:- Assume there's an optimal sol$^n$ $A = \{a_1, a_2 \ldots a_n\}$ to the problem(E) that does not include item i, with greatest value per weight (V/W) ratio of all initial items. Suppose $a_1$ is the item in sol$^n$ A with the greatest value per weight ratio.

$$\frac{Vi}{Wi} > \frac{Wa_1}{da_1}$$

Now suppose we remove $a_1$ from A and we obtain a sol$^n$ A' to sub problem F'

$$A' = A - a_1$$

If we combine sol$^n$ A' with greedy choice, we will obtain a greater or equal valuable sol$^n$ B, since $\frac{Vi}{Wi} \geq \frac{Va_1}{Wa_1}$

If B is greater than A, then this is a contradiction and thus i must be included, if B = A then we have shown that greedy choice is included anyway. Since that would mean that

$$\frac{Vi}{Wi} = \frac{Va_1}{Wa_1}$$

8. 

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

JOBID / Profit / Deadline

~~a~~
~~b~~
~~c~~
~~d~~
~~e~~

| 0 | 1 | 2 |
|---|---|---|
| b | a | d |

0 1 2 3

20 + 15 + 15 = 40

Profit = 40

9) In Dijkshtra's algorithm, greedy approach doesn't work work in graphs with -ve edges.
Similarly we can't break objects in knapsack problem (0/1) the soln we obtain when using greedy can be pretty bad. We can always make algorithm fail badly.
In travelling salesman problem, we can greedly approach the problem by always going to the nearest possible city. We select any of the cities as the first one and apply that strategy.

10) Generate all the subsets of a given set of jobs and check individual subsets for the feasibility of jobs in that subset. keep track of ~~minimum~~ maximum profit among all feasible subsets.
Algorithm :-
1) Sort all jobs in decreasing order of profit.
2) Iterate on jobs in decreasing order of profit for each job, do the following :-

Date.
Page No.

a) Find a time slot i, such that it is empty and i < deadline and i is greatest. Put the job in the ~~so~~ slot. and mark this slot filled.

b) In no such job exists, then ignore job.