Module - 5

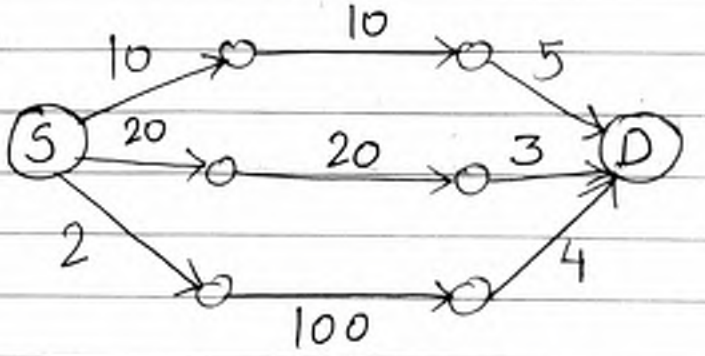
Dynamic Algorithms And NP-Hard and NP-Complete

Introduction to dynamic Algorithms :-

Dynamic programming is used to solve optimization problem. Optimization is finding either minimum or maximum answer. For example, if we need to find out profit we try to find maximum profit with given resources. If we have to find out cost we try to find out minimum cost with given resources.

Greedy algorithms are also used to find the optimal solution. But the approach of greedy algorithm & dynamic programming ~~approach~~ algorithms are completely different.

Example,



For finding the optimal soln for travelling from S to D, the greedy approach chooses the path with cost 2 in step 1.

But if we continue with the path with cost 2. It will not give the optimal solution.

In dynamic programming, we traverse through all the sequence of ~~dici~~ decisions and then give the optimal solution.

So the answer of dynamic programming algorithm is always optimal which is not the case in terms of greedy approach.

* Dynamic programming divide the problem into series of overlapping sub-problems.

Two features
1. optimal substructure
2. overlapping subproblems

In overlapping subproblems, we try to solve same problem again & again to get the optimal solution.

To avoid solving same subproblem again & again the solution of subproblem can be stored in a table & can be retrieved whenever required.

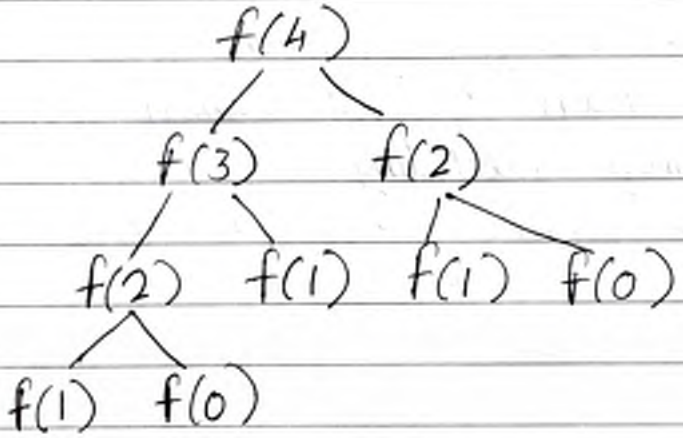
For example, to solve the fibonacci series we get recurrence relation as

$$f(n) = f(n-1) + f(n-2) \quad \text{if } n>1$$
$$= 1 \quad \text{if } n==1$$
$$= 0 \quad \text{if } n==0$$

if n =	0	1	2	3	4	5	6	7
f(n) =	0	1	1	2	3	5	8	13

For f(4)

Step1 optimal substructure
(Divide the problem into subproblems)



Table

f(1)
f(0)
f(2)
f(3)

So, we solve f(1) then f(0) then f(2) & so on. In the right subtree we need to solve f(2), f(1) & f(0) once again. so we store them in a table so than second time we dont need to calculate it once again.

If we will not store the answer in the table for the subproblems then the time complexity ~~of dyp with~~ will go to $O(2^n)$ i.e. exponentially time complexity will grow.

But if the results are stored in tables then we need to calculate only f(0), f(1), f(2) & f(3).

Applications of Dynamic Programming Approach

1) All pair shortest path
2) 0/1 Knapsack
3) Travelling Salesman problem
4) Coin Changing problem
5) Matrix chain multiplication
6) Flow shop scheduling
7) Optimal Binary Search Tree (OBST)