Characteristics of Greedy

-> It works by making the most optimal decision at any moment of time

-> The objective of this method is to ~~make~~ choose the optimal solⁿ for every sub-problem so that you can get optimal solⁿ for the overall problem.

-> To solve a particular problem in an optimal way using greedy approach, there is a set or list of candidates C

-> Once a candidate is selected in the solⁿ, it is there forever, once a candidate is excluded from the solⁿ it is never considered

-> To construct the solⁿ in an optimal way, Greedy algorithm maintains two sets. One set contains candidates that have already been considered and chosen, while the other set contains candidates that have been considered but rejected.

Characteristics of Divide & Conquer

-> The divide and conquer breaks a problem into sub-problems that are similar to original problem

-> The sub-problems are then solved and the answers are combined

-> Each sub-problem should be smaller than the original problem

-> Divide & Conquer has 3 parts :-

1. Divide the problem into number of sub-problems that are smaller instances of the same problem.

(2) Conquer the subproblem by solving them recursively
(3) Combine the solutions to the subproblems into the soln for the original problem.

Characteristics of Dynamic Programming

(1) It is used in optimization problems.
(2) Dynamic Programming solves problems by combining the solutions of subproblems.
(3) Dynamic Programming solves each sub-problem just once and then saves its answer in a table, thereby avoiding the work of recomputing the answer every time.
(4) Two main properties of Dynamic Programming are as follow:-

(a) Overlapping Sub-Problems
It is mainly used where the soln of one sub-problem is needed repeatedly. The computed soln are stored in a table so that these don't have to be re-computed. Hence, this technique is needed where over-lapping sub-problem exists.

(b) Optimal Sub-Structure
A given problem has optimal substructure property, if the optimal soln of the given problem can be obtained using optimal soln of its sub-problems.

Differences

Divide & Conquer

① It involves three steps:
Divide the problem, Conquer the sub problem and Combine the soln
② They are recursive
③ It does more work on subproblems and hence has more time consumption
④ It is a top-down approach
⑤ In this subproblems are independent of each other
⑥ for e.g. Merge Sort & Binary Search etc.

Dynamic Programming

① It involves four steps:
Characterize the structure of optimal soln
Recursively define the value of optimal soln
Compute the value of optimal soln and
Construct an optimal ~~structure~~ soln from computed info.
② It is non Recursive
③ It solves subproblems only once and then stores it in table
④ It is a Bottom-up approach.
⑤ In this sub-problems are interdependent
⑥ for e.g. Matrix Multiplication.

Divide & Conquer

(1) optimises by breaking down a sub-problem into simpler versions of itself and using multi-threading & recursion to solve.

(2) Always finds the optimal soln but is slower than Greedy.

(3) Requires some memory to remember recursive calls.

Greedy

(1) Optimises by making the best choice at the moment.

(2) Doesn't always find the optimal soln, but is very fast.

(3) Requires almost no memory.

Greedy Method

1. In greedy we make whatever choice seems best at that moment and then solve the sub-problems arising after the choice is made.

2. In Greedy Method, sometimes there is no such guarantee of getting optimal sol^n.

3. It is more efficient in terms of memory as it never look back or revise previous choices.

4. Greedy Methods are generally faster.

5. Less efficient than Dynamic Programming
6. For e.g. Fractional Knapsack

Dynamic Programming

1. In Dynamic Programming, we choose at each step, but the choice may depend on the sol^n to sub-problems.
2. It is guaranteed that Dynamic Programming will generate an optimal sol^n as it generally considers all possible cases and then choose the best.
3. It require dp table for memorization and it increases its memory complexity
4. Dynamic Programming is generally slower
5. More efficient than Greedy method
6. For e.g. 0/1 Knapsack Problem

Tower of Hanoi

Proof of No. of moves by mathematical induction.

Prove :- The min number of moves needed to move n disks from one pin in ToH problem is $2^n - 1$ for all natural no.

I $P(1)$ is true.

II Given: $P(k)$ is true.
Prove: $P(k+1)$ is true.

Now 1 disk.

$$2^1 - 1 = 2 - 1 \Rightarrow 1$$

Now min num of moves required to move k disks from one point to another in ToH problem is $2^k - 1$

Assume we have $k+1$ disks
we first move the top k disks by $2^k - 1$ moves

Now move the largest disk.
∴ 1 more move needed.

Again to move ~~k+1~~ k disks from second to third pin $2^k - 1$ moves needed.

∴ Total no. of moves

$= 2^{k} - 1 + 1 + 2^{k} - 1$
$= 2^{k} + 2^{k} - 1 \Rightarrow 2(2^{k}) - 1$
$= 2^{k+1} - 1$

∴ The min no. of move needed for k+1 disks is $2^{k+1} - 1$

∴ Min num of moves needed for n disks is $2^{n} - 1$

⇒ Time complexity $T(n) \propto$ No. of moves.

For e.g. No. of moves needed for 3 disks will be

$2^{3} - 1 \Rightarrow 8 - 1 \Rightarrow 7$

So, No. of moves needed for 4 disks will be

$2^{4} - 1 \Rightarrow 15$
$15 = 7 + 1 + 7$
$T(4) = T(3) + 1 + T(3)$

$T(4) = 2T(3) + 1$

Now, Generalizing it,

$T(n) = 2T(n-1) + 1$

Now by using substitution method.

$$T(n) = 2T(n-1) + 1 \quad \text{—— } (1)$$

$$= 2(2T(n-2)+1)+1$$

$$= 2^2(T(n-2)) + (1+2) \quad \text{—— } (2)$$

$$= 2^2(2T(n-3)+1) + (1+2)$$

$$= 2^3(T(n-3)) + (1+2+4) \quad \text{—— } (3)$$

$$\Rightarrow 2^k T(n-k) + 2^k - 1$$

$$\Rightarrow 2^n T(0) + 2^n - 1$$

$$\Rightarrow 2^n + 2^n - 1$$

$$\boxed{\Rightarrow T(n) = O(2^n)}$$