Dynamic Programming

Dynamic programming approach is similar to divide and conquer in breaking down the problem in smaller and yet smaller possible sub-problems. But unlike, divide and conquer, these sub-problems are not solved independently. Rather, results of these smaller sub-problems are remembered and used for similar or overlapping sub-problems.

Dynamic Programming

Dynamic programming is used where we have problems which can be divided in similar sub-problems, so that their results can be re-used. Mostly, these algorithms are used for optimization. Before solving the in-hand sub-problem, dynamic algorithm will try to examine the results of previously solved sub-problems. The solutions of sub-problems are combined in order to achieve the best solution.
So we can say that
The problem should be able to be divided in to smaller overlapping sub-problem.
The optimum solution can be achieved by using optimum solution of smaller sub-problems.

Dynamic Programming

Dynamic algorithms use **memoization**.
Comparison In contrast to greedy algorithms, where local optimization is addressed, dynamic algorithms are motivated for overall optimization of the problem.
In contrast to divide and conquer algorithms, where solutions are combined to achieve overall solution, dynamic algorithms uses the output of smaller sub-problem and then try to optimize bigger sub-problem. Dynamic algorithms uses memoization to remember the output of already solved sub-problems.

Dynamic Programming

Example The following computer problems can be solved using dynamic programming approach

- Fibonacci number series
- Knapsack problem
- Tower of Hanoi
- All pair shortest path by Floyd-Warshall
- Shortest path by Dijkstra
- Project scheduling

Dynamic programming can be used in both top-down and bottom-up manner. And ofcourse, most of the times, referring to previous solution output is cheaper than re-computing in terms of CPU cycles.