

Dynamic Programming

In this tutorial, you will learn what dynamic programming is. Also, you will find the comparison between dynamic programming and greedy algorithms to solve problems.

Dynamic Programming is a technique in computer programming that helps to efficiently solve a class of problems that have overlapping subproblems and [optimal substructure](#) property.

If any problem can be divided into subproblems, which in turn are divided into smaller subproblems, and if there are overlapping among these subproblems, then the solutions to these subproblems can be saved for future reference. In this way, efficiency of the CPU can be enhanced. This method of solving a solution is referred to as dynamic programming.

Such problems involve repeatedly calculating the value of the same subproblems to find the optimum solution.

Dynamic Programming Example

Let's find the fibonacci sequence upto 5th term. A fibonacci series is the sequence of numbers in which each number is the sum of the two preceding ones. For example, `0,1,1, 2, 3`. Here, each number is the sum of the two preceding numbers.

Algorithm

Let n be the number of terms.

1. If $n \leq 1$, return 1.
2. Else, return the sum of two preceding numbers.

We are calculating the fibonacci sequence up to the 5th term.

1. The first term is 0.
2. The second term is 1.
3. The third term is sum of 0 (from step 1) and 1(from step 2), which is 1.
4. The fourth term is the sum of the third term (from step 3) and second term (from step 2)
i.e. `1 + 1 = 2`.

5. The fifth term is the sum of the fourth term (from step 4) and third term (from step 3) i.e.

$$2 + 1 = 3.$$

Hence, we have the sequence $0, 1, 1, 2, 3$. Here, we have used the results of the previous steps as shown below. This is called a **dynamic programming approach**.

```
F(0) = 0
F(1) = 1
F(2) = F(1) + F(0)
F(3) = F(2) + F(1)
F(4) = F(3) + F(2)
```

How Dynamic Programming Works

Dynamic programming works by storing the result of subproblems so that when their solutions are required, they are at hand and we do not need to recalculate them.

This technique of storing the value of subproblems is called memoization. By saving the values in the array, we save time for computations of sub-problems we have already come across.

```
var m = map(0 → 0, 1 → 1)
function fib(n)
  if key n is not in map m
    m[n] = fib(n - 1) + fib(n - 2)
  return m[n]
```

Dynamic programming by memoization is a top-down approach to dynamic programming. By reversing the direction in which the algorithm works i.e. by starting from the base case and working towards the solution, we can also implement dynamic programming in a bottom-up manner.

```
function fib(n)
  if n = 0
    return 0
  else
    var prevFib = 0, currFib = 1
    repeat n - 1 times
      var newFib = prevFib + currFib
      prevFib = currFib
      currFib = newFib
    return currFib
```

Recursion vs Dynamic Programming

Dynamic programming is mostly applied to recursive algorithms. This is not a coincidence, most optimization problems require recursion and dynamic programming is used for optimization.

But not all problems that use recursion can use Dynamic Programming. Unless there is a presence of overlapping subproblems like in the fibonacci sequence problem, a recursion can only reach the solution using a divide and conquer approach.

That is the reason why a recursive algorithm like Merge Sort cannot use Dynamic Programming, because the subproblems are not overlapping in any way.

Greedy Algorithms vs Dynamic Programming

[Greedy Algorithms](#) are similar to dynamic programming in the sense that they are both tools for optimization.

However, greedy algorithms look for locally optimum solutions or in other words, a greedy choice, in the hopes of finding a global optimum. Hence greedy algorithms can make a guess that looks optimum at the time but becomes costly down the line and do not guarantee a globally optimum.

Dynamic programming, on the other hand, finds the optimal solution to subproblems and then makes an informed choice to combine the results of those subproblems to find the most optimum solution.

Different Types of Dynamic Programming Algorithms

1. [Longest Common Subsequence](#)
2. [Floyd-Warshall Algorithm](#)