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.

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

Dynamic Programming Example

Take the case of generating the fibonacci sequence.

If the sequence is F(1) F(2) F(3)........F(50), it follows the rule F(n) = F(n-1) + F(n-2)

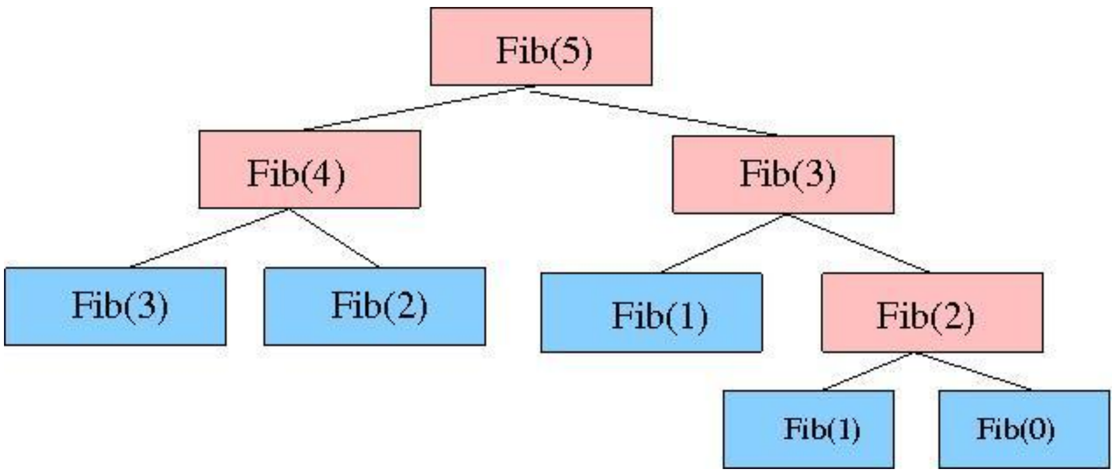
F(50) = F(49) + F(48)

F(49) = F(48) + F(47)

F(48) = F(47) + F(46)

...

Notice how there are overlapping subproblems, we need to calculate F(48) to calculate both F(50) and F(49). This is exactly the kind of algorithm where Dynamic Programming shines.



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 memorization. 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 memorization 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 currentFib
```

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