What is Dynamic Programming?

In this lesson, we will learn the ideology behind dynamic programming.

We'll cover the following

- Why do we need to remember the past?
- Why is it called dynamic programming?

Chances are you've seen the following quote espoused to dynamic programming before:

> *Those who cannot remember the past are condemned to repeat it*
>
> **~ George Santayana**

Why do we need to remember the past?

Let's do a small exercise and see what it evaluates too.

```
x = 2*2*2*2*2*2*2
```

Now evaluate the following equation.

```
x = x*2
```

You had to count all the `2`s in the first equation, which took some time. But you were able to evaluate the second one much quicker because you remembered what `x` had evaluated to. Our brain works in a way where we tend to memorize things by default and we use these memories to make future decisions. Well, computers do not have this by default. As we saw in the Fibonacci numbers algorithm given in the last lesson, our algorithm was re-evaluating the same things again and again. If we could enable it to memorize older results that would save us the cost of so many re-evaluations. This is the concept of dynamic programming.

> *For the problem that depends on subproblems that are being repeatedly re-*

> *evaluated, we can store results of these subproblems to avoid re-evaluation.*

We will discuss some ways to store these results in the next lesson, but for this lesson let's look at the anatomy of the word **dynamic programming**

Why is it called dynamic programming?

The technique behind dynamic programming is pretty intuitive and straight-forward. If you want to get rid of the re-evaluation in a problem, then simply memorize its results. Why does it have such a fancy name?

The term *dynamic programming* was coined by *Dr. Richard Bellman* while he was working at RAND Corporation on multistage decision processes in the mid-1950s. He writes that his superior there was pretty hostile against the mathematical research. So, to hide the mathematical nature of the algorithm he was working on, Dr. Bellman chose a very clever name. He called it dynamic programming; since programming means finding an optimal solution or making an optimal decision. He called it dynamic to highlight the fact that his solution was multistage.

Turns out these two properties are very common in most of the recursive problems. As we observed in the last few lessons, recursion tends to be a repetitive step-by-step process, thus making it a multistage process. Also, since our end goal is to find an optimal solution, dynamic programming fits pretty well given the nature of recursive problems.

In theory, dynamic programming does not have a lot to do with actual memorization techniques but *it is the name of solving problems that depend on subproblems*. Memorization techniques are merely an addition to dynamic programming, a much-needed one.

In the next lesson, we will cover two approaches to dynamic programming.