Name: Trevor Buck

ID: 109081318

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 7 **Spring 2019, CU-Boulder**

Explain dynamic programming to a biology major—what it is, how it works, and why it is valuable.

Dynamic programming is based on the idea that big problems can be broken down into more simple problems. Then, by solving and storing those solutions to the smaller and easier problems, the solution to the overall problem comes easy. The most common and straightforward example is trying to solve for large recursive algorithms like the Fibonacci numbers. If you needed to find the 100th number in this sequence, you would need to know the 99th and the 98th numbers. If you try to do this without dynamic programming, you would have to solve each of those numbers separately. Meaning that in order to find the 99th number, you would have to add the 98th and the 97th number in the sequence, and then you would have to go back and solve for the 98th number. This means that you would have to solve for the 98th number twice. If you follow the path a little longer, you see that you would need to solve the 97th number 3 times. And so on and so on. Basically, you're doing a lot of the same steps over and over again. Dynamic programming eliminates the overhead of this problem by storing those numbers in some sort of array. Most commonly, however, dynamic programming uses a bottom up approach. Meaning that it starts with the basics and works its way up to the harder problem. Let's dive back into the Fibonacci problem. When faced with the task of solving the 100th Fibonacci number, dynamic programming takes on the task by starting at the bottom. It stays well we know that eventually we are going have to solve for the first Fibonacci number so let's start there. It says the first number is just 1. Ok store that. Then the second number is also 1, and we can store that. Then when it gets to the third number, it can say, well I already know what the first and the second are, so I just have to do one thing and add those two numbers together. Then when it gets to the fourth number, it once again realizes that it already knows both the second and the third, so all it has to do is add those together. As you can see, now, at each step in the process of getting to 100, we only have one operation to do. So when we get to the 100th number in the sequence, we know we've done exactly 100 operations. Easy peasy and super fast.

To summarize I think it's easiest if we just write out the steps to dynamic programming:

1. Break the big problem up into smaller tasks
2. Start with the easiest problems
3. Store those short and easy solutions somewhere close
4. Use those solutions to work your way up
5. Stop when you get to the top

Name: Trevor Buck

ID: 109081318

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 7 **Spring 2019, CU-Boulder**

One thing to keep in mind, is that the Fibonacci sequence is only one example of dynamic programming. But the applications are nearly limitless. You are a biology major, so let me show you an example in your field. I know that you guys like to study genomes and how they are related to one another. Lots of these genomes are thousands of units long. Trying to compare these genomes at first glance could be rather difficult, but using dynamic programming makes life much easier. Just break it up into smaller tasks and work your way up. For example, you could start with simply comparing one tiny section of two genomes. Then store the similarities. Then use that to compare with two tiny sections of the same genomes. Then store those similarities. Then so on and so on until you have completely compared two full genomes. Then you can compare those solutions to other genomes. One last time, just start with the basics, store the results, and use that to help you move forward!

If you have any questions, let me know and I'll see what I can do to help!
Thanks, Trevor