Why Asymptotic Notation?

Learn why asymptotic notation is an essential tool for becoming an efficient programmer.

When writing programs, it's important to make smart programming choices so that code runs most efficiently. Computers seem to take no time evaluating programs, but when scaling programs to deal with massive amounts of data, writing efficient code becomes the difference between success and failure. In computer science, we define how efficient a program is by its **runtime**.

We can't just time the program, however, because different computers run at different speeds. My dusty old PC does not run as fast as your brand new laptop. Programming is also done in many different languages, how do we account for that in the runtime? We need a general way to define a program's runtime across these variable factors. We do this with **Asymptotic Notation**.

With asymptotic notation, we calculate a program's runtime by looking at how many instructions the computer has to perform based on the size of the program's input. For example, if I were calculating the maximum element in a collection, I would need to examine each element in the collection. That examining step is the same regardless of the language used, or the CPU that's performing the calculation. In asymptotic notation, we define the size of the input as N. I may be looking through a collection of 10 elements, or 100 elements, but we only need to know how many steps are performed *relative to the input* so N is used in place of a specific number. If there is a second input, we may define the size of that input as M.

There are varieties of asymptotic notation that focus on different concerns. Some will communicate the best case scenario for a program. For example, if we were searching for a value within a collection, the best case would be if we found that element in the first place we looked. Another type will focus on the worst case scenario, such as if we searched for a value, looked in the entire dataset and did not find it. Typically programmers will focus on the worst case scenario so there is an upper bound of runtime to communicate. It's a way of saying "things may get this bad, or slow, but they won't get worse!"

In this next module, we will learn more about asymptotic notation, how to properly analyze the runtime of a program through asymptotic notation, and how to take into consideration the runtime of different data structures and algorithms when creating programs. Learning these skills will change the way you think when you design programs and it will prepare you for the software engineering world where creating *efficient* programs is an essential skill.

Let's dive into the world of asymptotic notation!