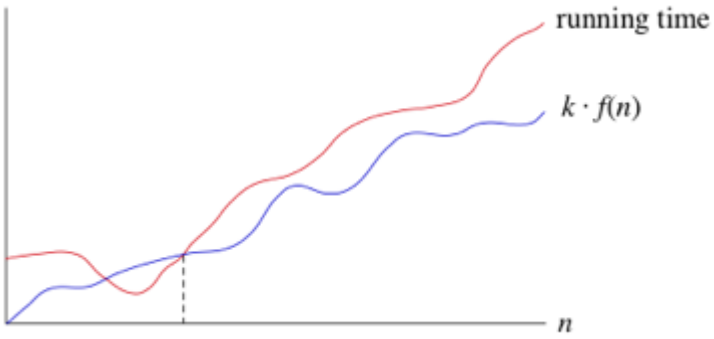
Big-Ω (Big-Omega) notation

Sometimes, we want to say that an algorithm takes at least a certain amount of time, without providing an upper bound. We use big-Ω notation; that's the Greek letter "omega."

If a running time is **Ω(f(n))**, then for large enough **n**, the running time is at least **k·f(n)** for some constant **k**. Here's how to think of a running time that is **Ω(f(n))**:



We say that the running time is "big-Ω of **f(n)**." We use big-Ω notation for **asymptotic lower bounds**, since it bounds the growth of the running time from below for large enough input sizes.

Just as **Θ(f(n))** automatically implies **O(f(n))**, it also automatically implies **Ω(f(n))**. So we can say that the worst-case running time of binary search is **Ω(lgn)**. We can also make correct, but imprecise, statements using big-Ω notation. For example, just as if you really do have a million dollars in your pocket, you can truthfully say "I have an amount of money in my pocket, and it's at least 10 dollars," you can also say that the worst-case running time of binary search is **Ω(1)**, because it takes at least constant time.