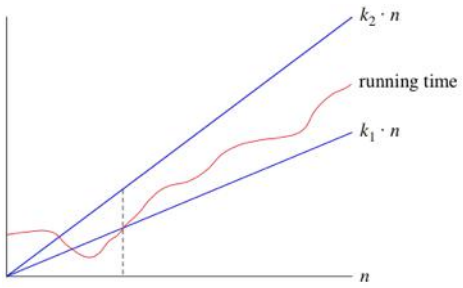
Asymptotic notation

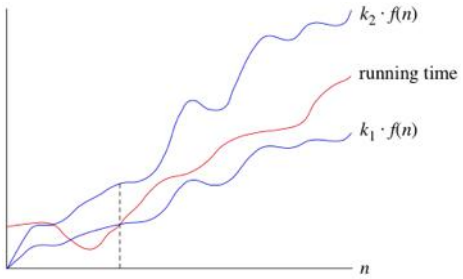
By dropping the less significant terms and the constant coefficients, we can focus on the important part of an algorithm's running time—its rate of growth—without getting mired in details that complicate our understanding. When we drop the constant coefficients and the less significant terms, we use **asymptotic notation**. We'll see three forms of it: big-Θ notation, big-O notation, and big-Ω notation.

Big-θ (Big-Theta) notation

When we say that a particular running time is Theta(n) -> Θ(n) we're saying that once n gets large enough, the running time is at least $k_1 \cdot n$, and at most $k_2 \cdot n$, for some constants k_1 and k_2. Here's how to think of Theta(n) -> Θ(n)



We are not restricted to just n in big-Θ notation. We can use any function, such as n^2,nlogn or any other function of n. Here's how to think of a running time that is Theta(f(n)) -> Θ($f(n)$) for some function $f(n)$:

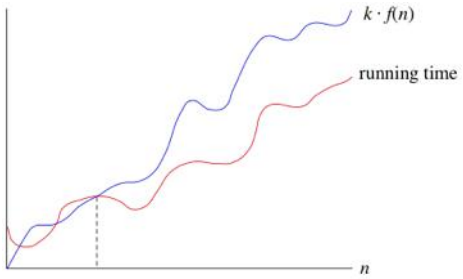


Big-O notation

We use big-Θ notation to asymptotically bound the growth of a running time to within constant factors above and below. Sometimes we want to bound from only above -> Big(O)

It would be convenient to have a form of asymptotic notation that means "the running time grows at most this much, but it could grow more slowly." We use "big-O" notation for just such occasions.

If a running time is $O(f(n))$, then for large enough n, the running time is at most $k \cdot f(n)$, for some constant k. Here's how to think of a running time that is $O(f(n))$:

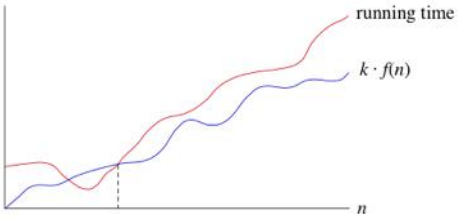


We use big-O notation for **asymptotic upper bounds**, since it bounds the growth of the running time from above for large enough input sizes.

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.

If a running time is Omega(f(n)) -> Ω($f(n)$) , then for large enough n, the running time is at least $k \cdot f(n)$, for some constant k. Here's how to think of a running time that is Omega(f(n)) -> Ω($f(n)$):



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($f(n)$).

If we established function is O(n) and Ω(n) ,then we can conclude that it is also Θ(n).