Asymptotics Continued

Review

Last class we covered 3 important ideas in asymptotics:

- **Big Oh**: \( f = O(g) \) means that \( g \) provides an upper bound on \( f \). This bounds \( \left| \frac{f}{g} \right| \).
- **Big Omega**: \( f = \Omega(g) \) means that \( g \) provides a lower bound on \( f \). This bounds \( \left| \frac{g}{f} \right| \).
- **Big Theta**: \( f = \Theta(g) \) means that \( f = O(g) \) and \( f = \Omega(g) \)

**Notation**: \( f(x) = x^4 + O(x) \) means that \( \exists \ g(x) = O(x) \) such that \( f(x) = x^4 + g(x) \).

Asymptotic Equivalence

**Little Oh**: We say that \( f \) is \( o(g) \) (read “little oh of \( g \)”) if

\[
\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0
\]

**Asymptotic Equivalence**: Two functions \( f \) and \( g \) are *asymptotically equal* (denoted \( f \sim g \)) if

\[
\lim_{n \to \infty} \frac{f(n)}{g(n)} = 1
\]

How can we use the idea of asymptotic equivalence in practice? Let’s consider the prime number theorem. In the number theory unit, we saw that the prime number theorem states

\[
\lim_{x \to \infty} \frac{\pi(x)}{x/\log(x)} = 1
\]

where \( \pi(x) \) is the number of primes that are less than or equal to \( x \). We can use the notion of asymptotic equivalence to say \( \pi(x) \sim \frac{x}{\log(x)} \).

**Thm**: Stirling’s Formula:

\[
n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n
\]

To provide some intuition about the size of \( n! \), it has been shown that if everyone on Earth shuffled cards “until the end of the universe” we still would not have all possible permutations.
Basic Orders of Growth

The basic orders of growth are:

$$1, \log n, n \log n, n^k, k^n, n!, n^n$$

$n^k$ represents polynomials and $k^n$ represents exponentials. The following image provides a visual of the relative orders of growth:

![Big-O Complexity Graph](image)

Analyzing Recursive Functions

Example: Mergesort

Let’s consider the mergesort algorithm, which sorts $n$ items by recursively dividing the set of items in half. The number of operations required to execute the mergesort algorithm on $n$ items is

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \tag{1}$$

Claim: $T(n)$ is $O(n \log n)$.

Proof. We prove this claim by strong induction.

Base Case: $n = 2$.

$T(2)$ is constant, so we have

$$T(2) = \alpha \leq d \cdot 2 \log 2 \Rightarrow d \geq \frac{\alpha}{2 \log 2}$$

Inductive Step:
Let the strong inductive hypothesis be that our claim is true for $\frac{n}{2}$. Equation (1) allows us to say that

$$T(n) \leq 2 \cdot T\left(\frac{n}{2}\right) + cn$$

We then invoke the SIH:

$$T(n) \leq 2 \cdot T\left(\frac{n}{2}\right) + cn$$
$$\leq 2 \cdot d \cdot \left(\frac{n}{2} \cdot \log \frac{n}{2}\right) + cn$$
$$\leq d \cdot n \cdot \log \left(\frac{n}{2}\right) + cn$$
$$\leq d \cdot n \cdot (\log n - \log 2) + cn$$
$$\leq dn \log n - dn \log 2 + cn$$
$$\leq dn \log n - dn + cn$$
$$\leq dn \log n$$

$\blacksquare$