W11L1: $P$ and big $O$ notation

Monday, April 13, 2020  9:21

Announcements:
- quiz this week on gradescope
- no homework this week
- midterm 2 next week "in lab" (on gradescope)

We now want to discuss runtime: how long does it take a Turing machine to reach an answer?

Definition:
Let $f, g: \mathbb{N} \to \mathbb{R}^+$ be two functions. We say that $f(n)$ is $O(g(n))$ if there exists a constant $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, $f(n) \leq c \cdot g(n)$. If $f$ is $O(g)$ then we say that "$f$ is asymptotically upper-bounded by $g$" and "$g$ is an asymptotic upper bound for $f$.

Problem 1: Prove that $f(n) = 2n^2 + 7n + 16$ is $O(g(n))$ where $g(n) = n^2$.

Pick $c = 25$ and $n_0 = 1$.
Want to show that $2n^2 + 7n + 16 \leq 25 \cdot (n^2)$ for $n \geq n_0 = 1$.

$$c \cdot g(n) = 2n^2 + 7n + 16 \geq 2n^2 + 7n + 16 \frac{b/c n^2}{n^2} \geq 2n^2 + 7n + 16 \frac{b/c n^2}{n^2} = f(n)$$

$$\Rightarrow f(n) \text{ is } O(g(n)).$$

Problem 2: Order these 3 functions by increasing $O(\cdot)$ asymptotic growth:
- $f(n) = 100n$
- $g(n) = n^{10}$
- $h(n) = \frac{1}{n^{10}}$
Problem 2: Order these 3 functions by increasing \( O(\cdot) \) asymptotic growth:

- \( f(n) = 100n \)
- \( g(n) = \frac{n^{10}}{6} \)
- \( h(n) = 4n^2 \)

**Claim:** \( f \) is \( O(h) \) and \( h \) is \( O(g) \).

**Heuristic:** look at the highest-order term

**Theorem:** If \( f, g, \) and \( h \) are all functions and \( f \in O(g) \) and \( g \in O(h) \) then \( f \in O(h) \).

**Idea:** \( O(\cdot) \) is transitive

We are going to use \( O(\cdot) \) to compare the efficiency of different Turing machines to solve the same problem.

The "speed" of a Turing machine is the number of steps used in computation (up until it halts). This runtime is a function of input size: \( f(n): \mathbb{N} \to \mathbb{N} \) will be the number of steps a Turing machine takes (until it halts) when given an input of length \( n \).

If we have a Turing machine \( M \) which decides \( L \subseteq \Sigma^* \) then we can define a function

\[ T_M(w) = \text{the number of steps } M \text{ takes on input } w \]

We are interested in the runtime of \( M \), which is the **worst-case** runtime, defined as:

\[ T_M(n) = \text{maximum number of steps } M \text{ takes on any input } w \text{ where } |w| = n. \]