Today

- Quiz 5
- Algorithm Analysis (cont)

Assignments

- HW 5 out (due Thurs)
- Quiz 6 Thurs
- Midterm exam next Thurs
**Big-O Definition**

\[ T(n) \text{ is } O(f(n)) \iff \text{there exist positive constants } k \text{ and } n_0 \text{ such that:} \]

\[ T(n) \leq k \cdot f(n) \]

for all \( n \geq n_0 \)

**Comments:**

- showing \( T(n) \) is \( O(f(n)) \) involves finding a \( k \) and \( n_0 \)
- as an upper bound can be misleading ...
- e.g., \( T(n) \) being \( O(n) \) implies \( T(n) \) is also \( O(n^2) \)

**Exercise:**

- Let \( T(n) \leq 3n + 2 \) and \( f(n) = n \)
- \( 3n + 2 \leq 5n \) for \( n \geq 1 \)

\[ \text{... thus } k = 5 \text{ and } n_0 = 1 \]

- Which implies \( T(n) \) is \( O(n) \)
Big-$\Omega$

$T(n)$ is $\Omega(f(n))$ iff there exist positive constants $k$ and $n_0$ such that:

$$T(n) \geq k \cdot f(n)$$

for all $n \geq n_0$

Comments:

- showing $T(n)$ is $\Omega(f(n))$ involves finding a $k$ and $n_0$
- as a lower bound can be misleading ...
- e.g., $T(n)$ being $\Omega(n^2)$ implies $T(n)$ is also $\Omega(n)$

Exercise:

- Let $T(n) \geq 4n + 3$ and $f(n) = n$
- $4n + 3 \geq 4n$ for $n \geq 1$ \(\text{... thus } k = 4 \text{ and } n_0 = 1\)
- Which implies $T(n)$ is $\Omega(n)$
Big-$\Theta$

$T(n)$ is $\Theta(f(n))$ iff there exist positive constants $k_1$, $k_2$, and $n_0$ s.t.:

$$k_1 \cdot f(n) \leq T(n) \leq k_2 \cdot f(n)$$

for all $n \geq n_0$

Comments:

- showing $T(n)$ is $\Theta(f(n))$ often by showing $\Omega(f(n))$ and $O(f(n))$ for an $n_0$

Example:

- Let $4n + 3 \leq T(n) \leq 5n + 3$ and $f(n) = n$
- $T(n)$ is $\Omega(n)$ and $O(n)$ for $n_0 = 1$
- Which implies $T(n)$ is $\Theta(n)$
Some “Standard” Algorithm Growth Rates

\[ O(n) \text{ and even } O(n \log n) \text{ algorithms are often considered “for free primitives”} \]

- since they basically only require “reading all the input”
Properties of Big-O Notation

**Sum of two functions**

\[ O(f(n)) + O(g(n)) = O(\max(f(n), g(n))) \]

- e.g., duplicate check \( O(n^2) \) followed by membership check \( O(n) \) is \( O(n^2) \)

**Product of two functions**

\[ O(f(n)) \cdot O(g(n)) = O(f(n) \cdot g(n)) \]

- e.g., doing \( O(n) \) work \( O(\log n) \) times is \( O(n \log n) \)
- usually requires more careful analysis to ensure really a product (more later)
Things to watch-out for with Asymptotic Analysis

Algorithm versus Problem

- Bubble sort is an $O(n^2)$ (comparison) sorting algorithm
- But the problem of (comparison) sorting is an $O(n \log n)$ problem
- The latter is a theoretical lower bound on comparison sorting

Tight Upper and Lower Bounds

- An algorithm that is $O(n)$ is also $O(n^2)$
- Similarly an algorithm that is $\Omega(n)$ is also $\Omega(\log n)$
- However, we usually mean that the given bound is “tight”
- Note that algorithms that are $O(1)$ are also $\Theta(1)$ and $\Omega(1)$ ... why?