Unit 05: Asymptotic Analysis

Anthony Estey

CSC 225: Algorithms and Data Structures I

University of Victoria

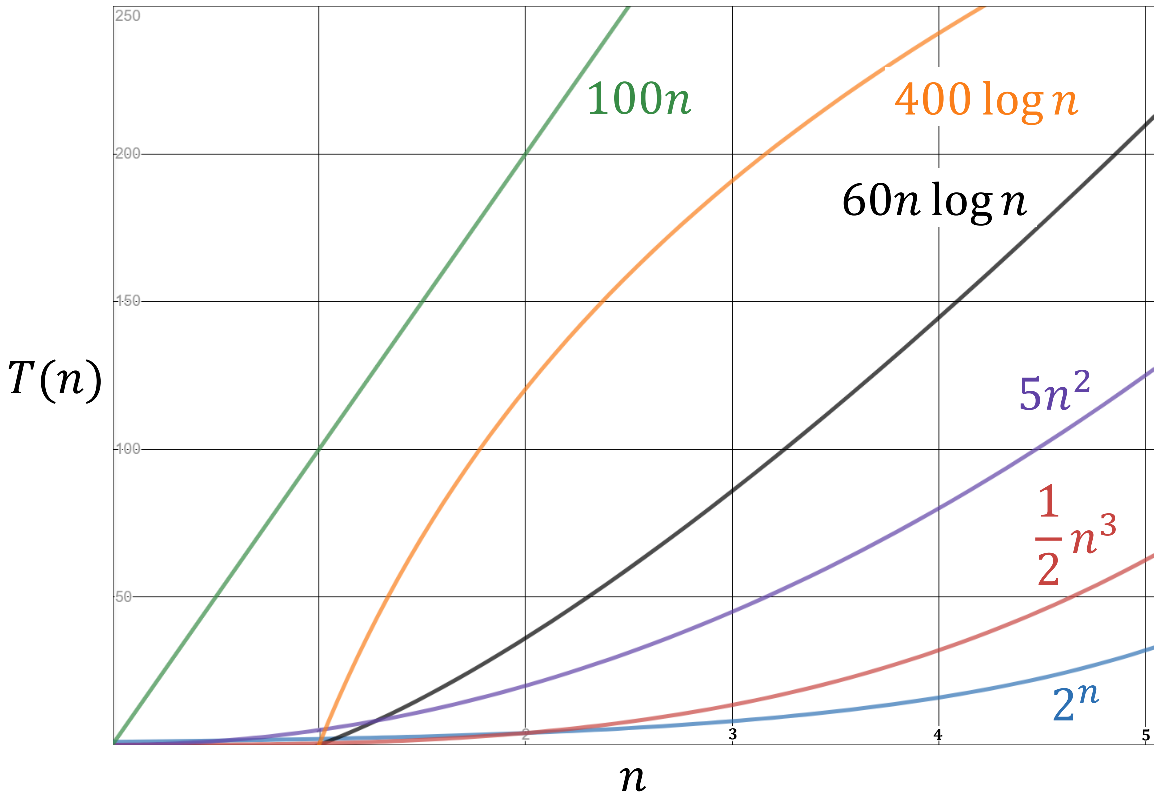
*Thanks to Dr. Rich Little (University of Victoria) and Dr. Ed Knorr (University of British Columbia) for providing some of the contents of these slides

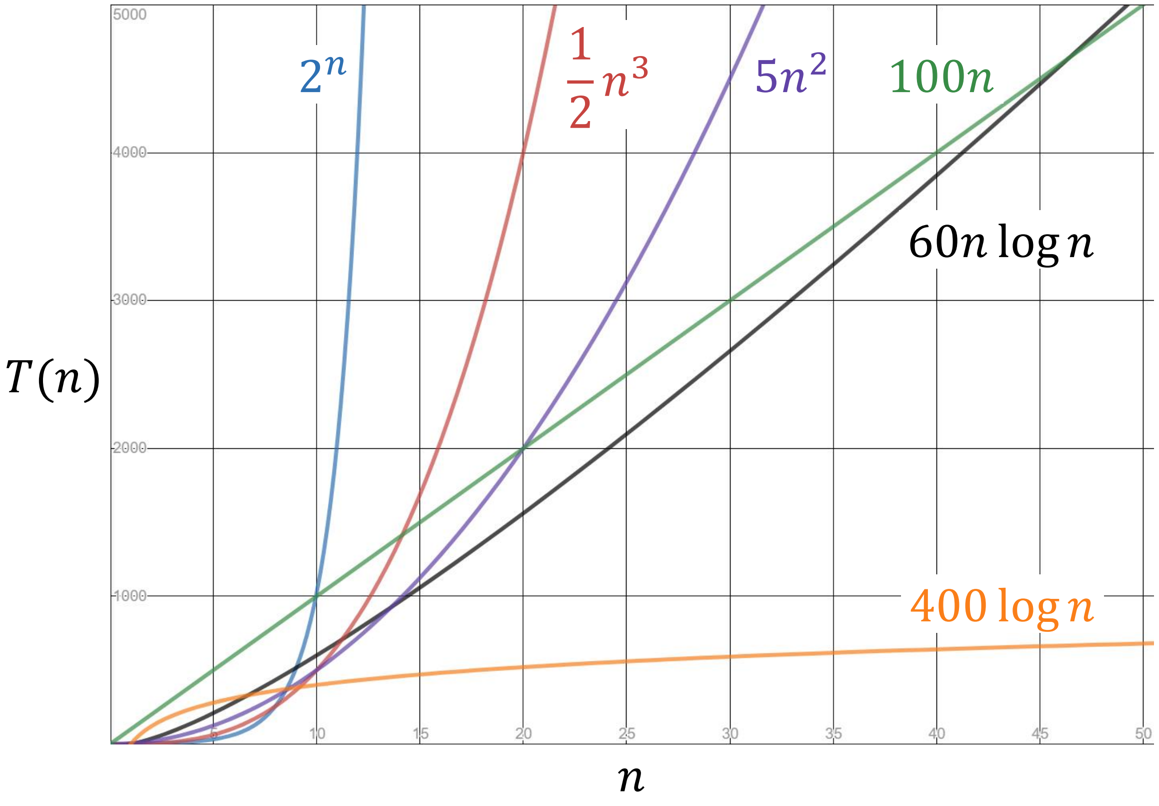
Unit 05 Overview

- Supplemental Reading:
  - Algorithm Design and Analysis. *Michael Goodrich and Roberto Tamassia*
    - Pages 11-18
- Learning Objectives: (You should be able to…)
  - use asymptotic notation to simplify functions and to express relations between functions
  - know and compare the asymptotic bounds of common functions
  - understand when and why to use worst-case, best-case, or average-case complexity measures

Asymptotic Analysis

- The running times of various algorithms are plotted below:

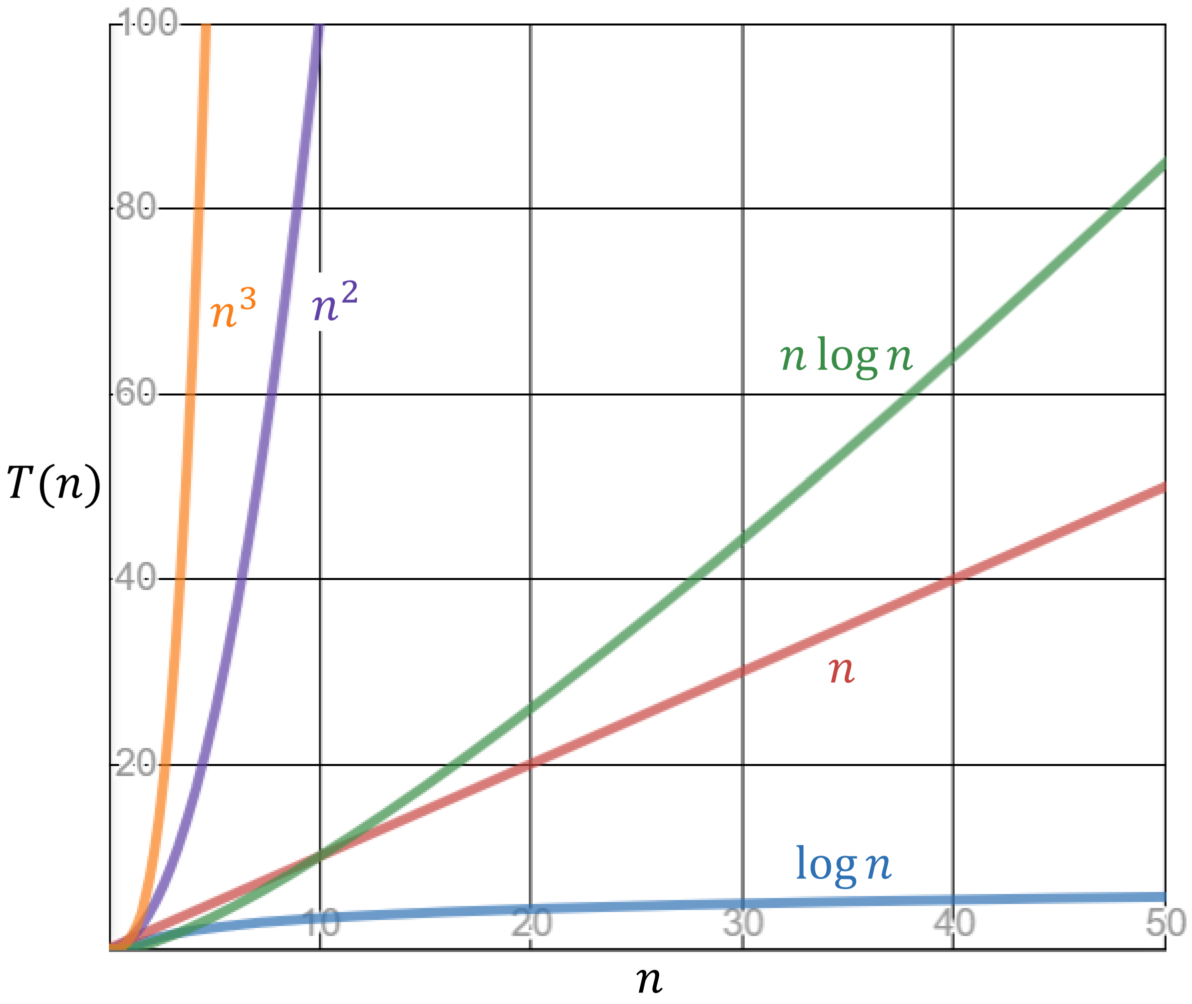




Graph images created using https://www.desmos.com/

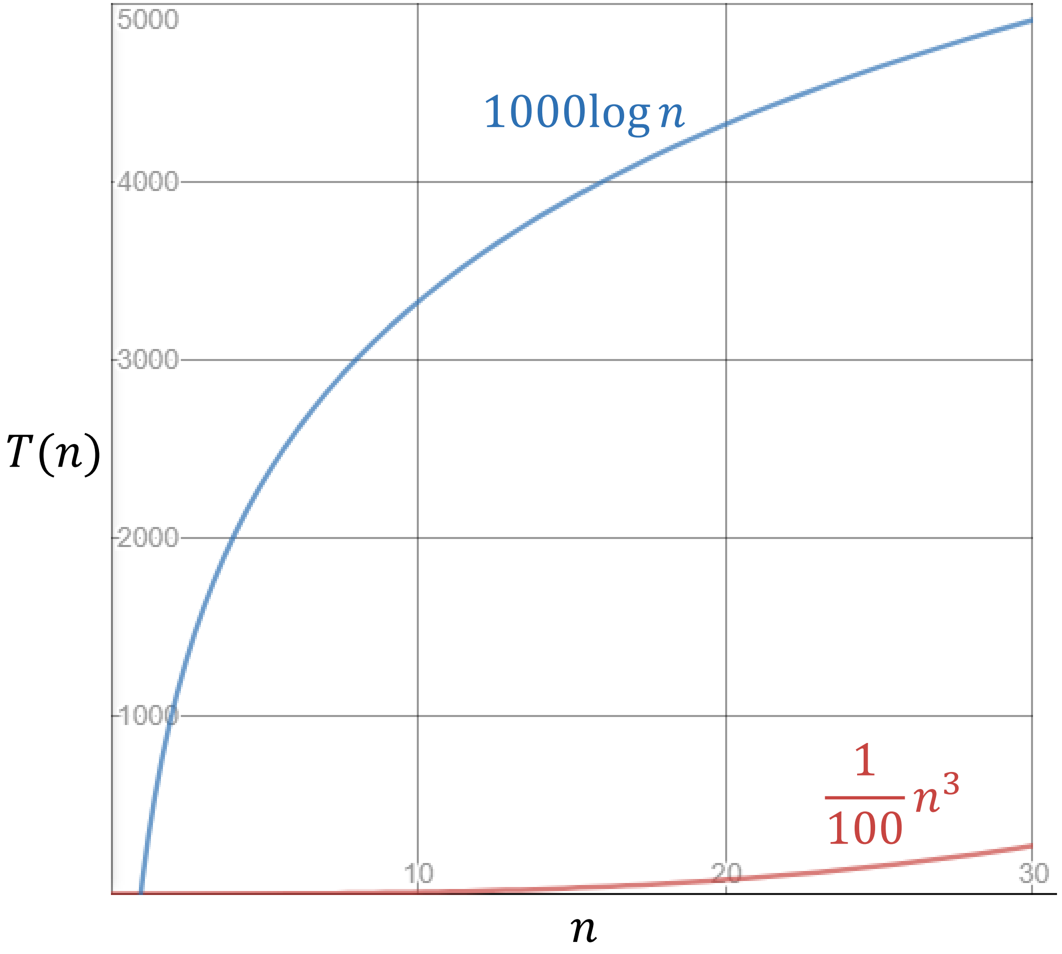
Asymptotic Analysis

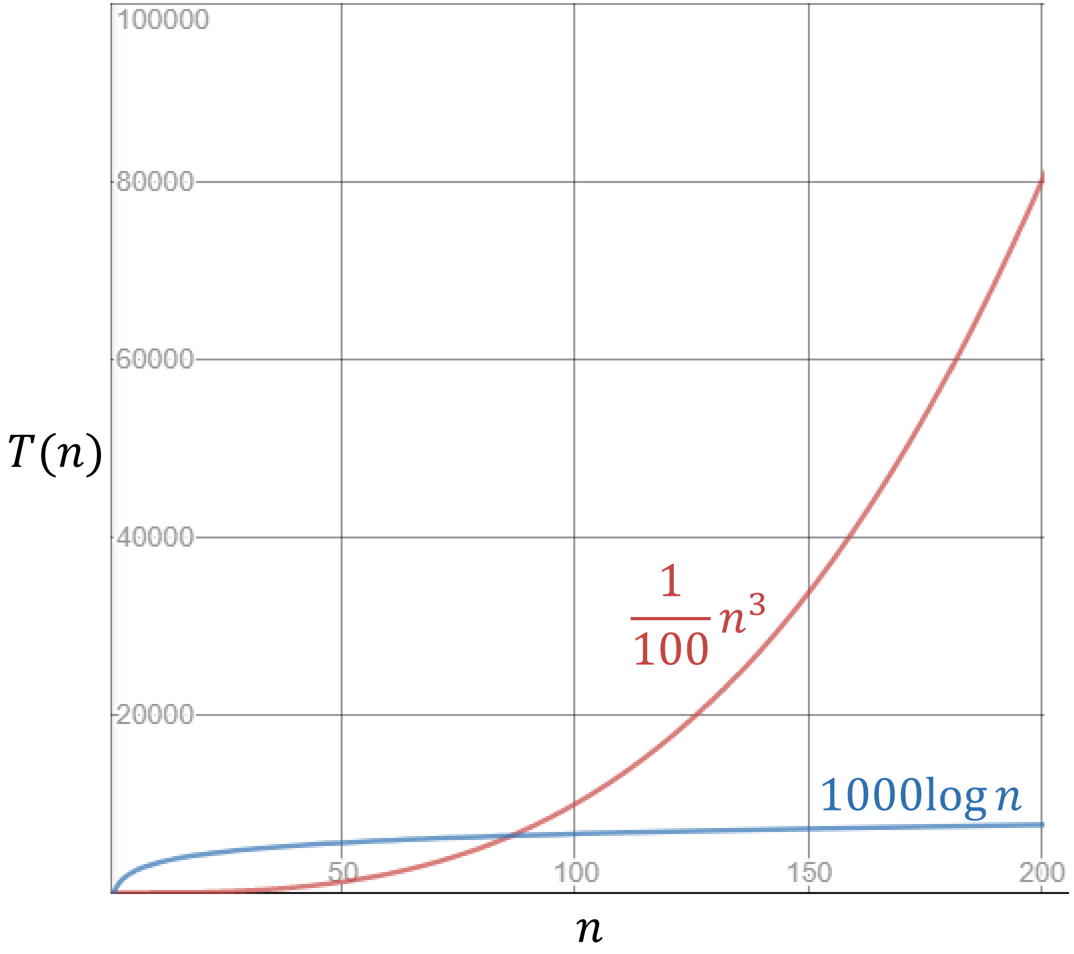
- In general, we will classify our algorithms into one of five categories:
  - Logarithmic -- O(log n)
  - Linear -- O(n)
  - Quadratic -- O(n^2)
  - Cubic -- O(n^3)
  - Exponential -- O(2^n)



Graph images created using https://www.desmos.com/

Asymptotic Analysis

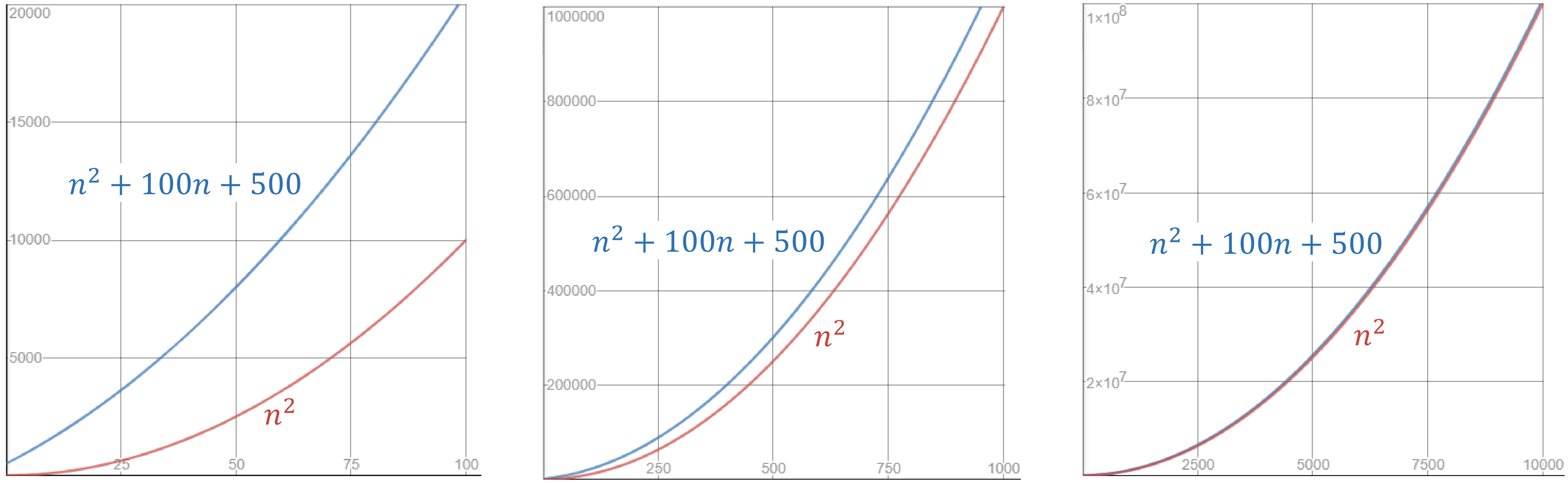




Graph images created using https://www.desmos.com/

Asymptotic Analysis

- Our runtime analysis has focused on counting operations
  - Big-Oh notation allows us to characterize the main factors affecting an algorithm's running time



Graph images created using https://www.desmos.com/

Rates of Growth

- Assume a device executes 1 trillion operations per second
- The following chart illustrates different execution time growth rates given an input size n

$\boldsymbol{n =}$	$\mathbf{10}$
$\log n$	$1ps$
n	$10ps$
$n \log n$	$10ps$
n^2	$100ps$
2^n	$1ns$

picosecond ($1ps = 10^{-12}s$) - one trillionth of a second

nanosecond ($1ns = 10^{-9}s$) - one billionth of a second

microsecond ($1\mu s = 10^{-6}s$) - one millionth of a second

Rates of Growth

- Assume a device executes 1 trillion operations per second
- The following chart illustrates different execution time growth rates given an input size n

$\boldsymbol{n =}$	$\mathbf{10}$	$\mathbf{100}$
$\log n$	$1ps$	$2ps$
n	$10ps$	$100ps$
$n \log n$	$10ps$	$200ps$
n^2	$100ps$	$10ns$
2^n	$1ns$	$10^{18}s$

32 billion years!

picosecond ($1ps = 10^{-12}s$) - one trillionth of a second

nanosecond ($1ns = 10^{-9}s$) - one billionth of a second

microsecond ($1\mu s = 10^{-6}s$) - one millionth of a second

Rates of Growth

- Assume a device executes 1 trillion operations per second
- The following chart illustrates different execution time growth rates given an input size n

$\boldsymbol{n =}$	**10**	**100**	**1000**
$\log n$	$1ps$	$2ps$	$3ps$
n	$10ps$	$100ps$	$1ns$
$n \log n$	$10ps$	$200ps$	$3ns$
n^2	$100ps$	$10ns$	$1\mu s$
2^n	$1ns$	$10^{18}s$	$10^{289}s$

picosecond ($1ps = 10^{-12}s$) - one trillionth of a second

nanosecond ($1ns = 10^{-9}s$) - one billionth of a second

microsecond ($1\mu s = 10^{-6}s$) - one millionth of a second

Rates of Growth

- Assume a device executes 1 trillion operations per second
- The following chart illustrates different execution time growth rates given an input size n

$\boldsymbol{n =}$	$\mathbf{10}$	$\mathbf{100}$	$\mathbf{1000}$	$\mathbf{10,000}$	$\mathbf{10^5}$	$\mathbf{10^6}$	$\mathbf{10^9}$
$\log n$	$1ps$	$2ps$	$3ps$	$4ps$	$5ps$	$6ps$	$9ps$
n	$10ps$	$100ps$	$1ns$	$10ns$	$100ns$	$1\mu s$	$1ms$
$n \log n$	$10ps$	$200ps$	$3ns$	$40ns$	$500ns$	$6\mu s$	$9ms$
n^2	$100ps$	$10ns$	$1\mu s$	$100\mu s$	$10ms$	$1s$	1 week
2^n	$1ns$	$10^{18}s$	$10^{289}s$				

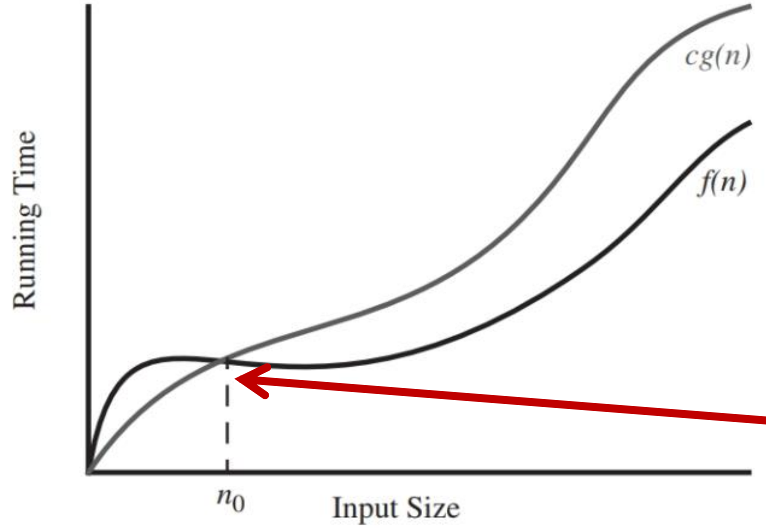
picosecond ($1ps = 10^{-12}s$) - one trillionth of a second

nanosecond ($1ns = 10^{-9}s$) - one billionth of a second

microsecond ($1\mu s = 10^{-6}s$) - one millionth of a second

Big-Oh Notation

- Formal definition:
  - Let $f(n)$ and $g(n)$ be functions mapping nonnegative integers to real numbers.
  - We say that $f(n)$ is $O(g(n))$ if there is a real constant $c > 0$ and an integer constant $n_0 \geq 1$ such that $f(n) \leq cg(n)$ for every integer $n \geq n_0$.



Visually, we see that the $f(n)$ curve fits under the $cg(n)$ curve.

For all inputs size n_0 and greater, $f(n)$ grows no faster than $cg(n)$

Figure 1.5: The function $f(n)$ is $O(g(n))$, for $f(n) \leq c \cdot g(n)$ when $n \geq n_0$.

Big-Oh terminology

- Big-Oh notation allows us to state that a function of n is less than or equal to another function, up to a constant factor (c), as n grows toward infinity (asymptotically)
- This allows us to characterize the execution time required for a function in the worst-case scenario!
- We often say:
  - "$f(n)$ is **big-Oh** of $g(n)$"
  - "$f(n)$ is **order** $g(n)$"
- And write it as:
  - $f(n) \in O(g(n))$

Big-Oh Example

- Show that $10{,}000n^2 + 25n$ is $O(n^2)$

- Approach:
  - find positive constants c and n_0 such that $T(n) \leq cf(n)$ for all $n \geq n_0$

- Solution

$$\begin{aligned} 10{,}000n^2 + 25n &\leq 10{,}000n^2 + 25n^2 \text{ when } n \geq 1 \\ &\leq 10{,}025n^2 \end{aligned}$$

$$\therefore T(n) \in O(n^2) \text{ with } c = 10025 \text{ and } n_0 = 1$$

Big-Oh Application

- It is not always necessary to apply the Big-Oh definition directly to obtain characterize a function as Big-Oh:

Theorem 1.7: *Let $d(n)$, $e(n)$, $f(n)$, and $g(n)$ be functions mapping nonnegative integers to nonnegative reals.*

1. If $d(n)$ is $O(f(n))$, then $ad(n)$ is $O(f(n))$, for any constant $a > 0$.
2. If $d(n)$ is $O(f(n))$ and $e(n)$ is $O(g(n))$, then $d(n)+e(n)$ is $O(f(n)+g(n))$.
3. If $d(n)$ is $O(f(n))$ and $e(n)$ is $O(g(n))$, then $d(n)e(n)$ is $O(f(n)g(n))$.
4. If $d(n)$ is $O(f(n))$ and $f(n)$ is $O(g(n))$, then $d(n)$ is $O(g(n))$.
5. If $f(n)$ is a polynomial of degree d (that is, $f(n) = a_0 + a_1 n + \cdots + a_d n^d$), then $f(n)$ is $O(n^d)$.
6. n^x is $O(a^n)$ for any fixed $x > 0$ and $a > 1$.
7. $\log n^x$ is $O(\log n)$ for any fixed $x > 0$.
8. $\log^x n$ is $O(n^y)$ for any fixed constants $x > 0$ and $y > 0$.

Asymptotic Analysis

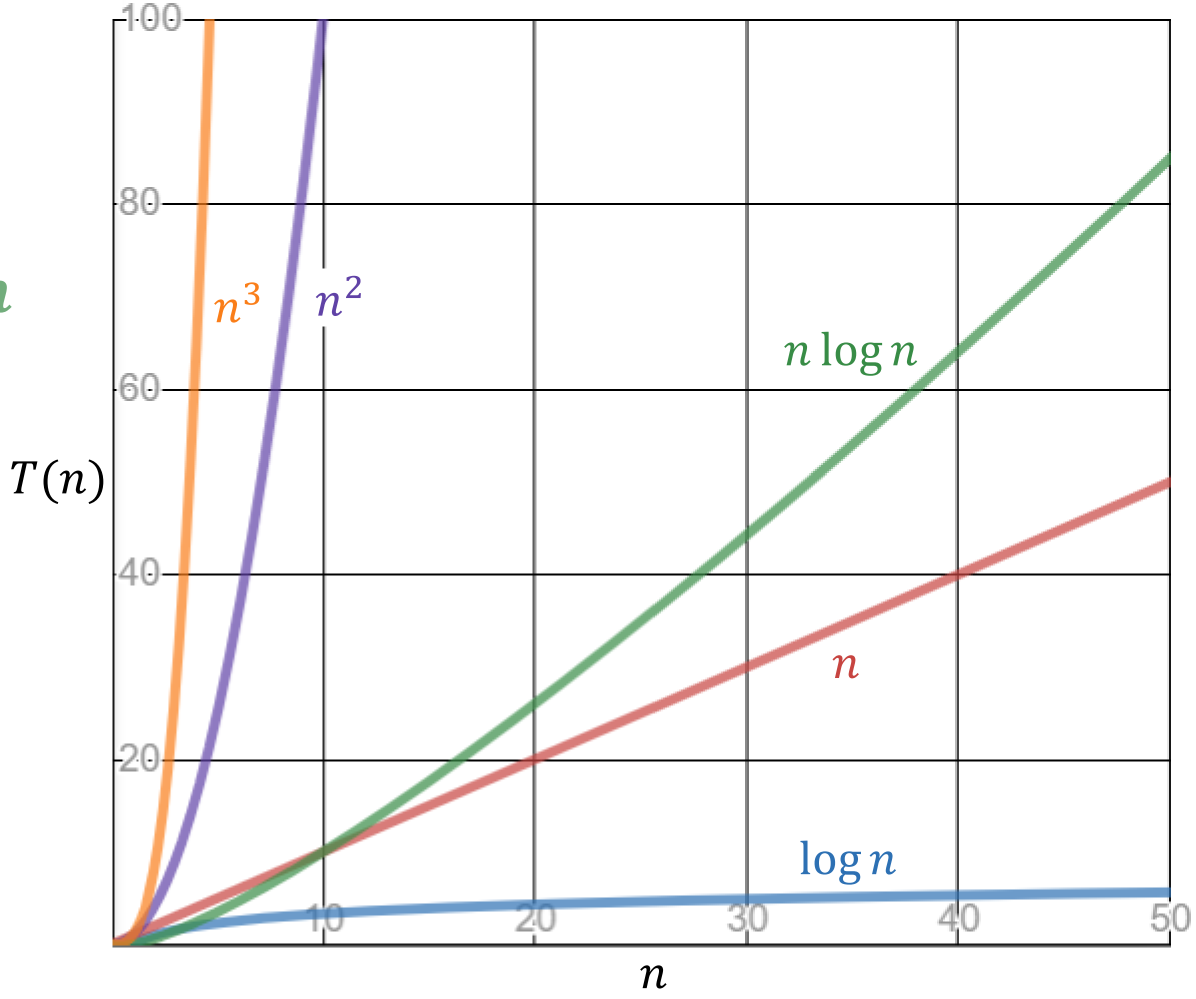
- We use Big-Oh notation for asymptotic upper bounds

$d(n) = \log n$ $\quad e(n) = n$ $\quad f(n) = n \log n$

$g(n) = n^2$ $\quad h(n) = n^3$

$$\boxed{\begin{array}{l} d(n) \in O(e(n)) \\ d(n) \in O(f(n)) \\ d(n) \in O(g(n)) \\ d(n) \in O(h(n)) \end{array}}$$

… with $c, n_0 = 1$



Graph images created using https://www.desmos.com/

Asymptotic Notation

- Big-Oh:
- $T(n) \in O(f(n))$ iff there are positive constants c and n_0 such that $T(n) \leq cf(n)$ for all $n \geq n_0$.

- Big-Omega:
- $T(n) \in \Omega(f(n))$ iff there are positive constants c and n_0 such that $T(n) \geq cf(n)$ for all $n \geq n_0$.

- Big-Theta:
- $T(n) \in \Theta(f(n))$ iff $T(n) \in O(f(n))$ and $T(n) \in \Omega(f(n))$

Big-Theta Example

- $T(n) = 2n + 1$
- Big-Oh:
  - $2n + 1 \leq 3n$ for all $n \geq 1$
  - $\therefore T(n) \in O(n)$ with $c = 3$ and $n_0 = 1$
- Big-Omega:
  - $2n + 1 \geq 2n$ for all $n \geq 1$
  - $\therefore T(n) \in \Omega(n)$ with $c = 2$ and $n_0 = 1$
- Big-Theta
  - Since $T(n) \in O(n)$ and $T(n) \in \Omega(n)$, $T(n) \in \Theta(n)$

Asymptotic Notation

- Little-Oh:
- $T(n) \in o(f(n))$ iff for **any** constant $c > 0$, there is a constant $n_0 > 0$ such that $T(n) < cf(n)$ for all $n \geq n_0$.
  - $\lim_{n \to \infty} \frac{T(n)}{f(n)} = 0$

- Little-Omega:
- $T(n) \in \omega(f(n))$ iff for **any** constant $c > 0$, there is a constant $n_0 > 0$ such that $T(n) > cf(n)$ for all $n \geq n_0$.
  - $\lim_{n \to \infty} \frac{T(n)}{f(n)} = \infty$

Asymptotic Notation - Recap

- What do these notations really tell us?

- Big-Oh: $T(n) \leq f(n)$
- Big-Omega: $T(n) \geq f(n)$
- Big-Theta: $T(n) = f(n)$
- Little-Oh: $T(n) < f(n)$
- Little-Omega: $T(n) > f(n)$

Typical Asymptotics

- Tractable:
  - Constant: $\Theta(1)$
  - Double logarithmic: $\Theta(\log\log n)$
  - Logarithmic: $\Theta(\log n) - (\log_b n\,, \log n^2 \in \Theta(logn))$
  - Poly-Log: $\Theta\left(log^k n\right) - (log^k n \equiv (\log n)^k)$
  - Fractional power: $\Theta(n^c)$, where $0 < c < 1$
  - Linear: $\Theta(n)$
  - Log-Linear: $\Theta(n \log n)$
  - Super-Linear: $\Theta(n^{1+c})$ (c is a constant > 0)
  - Quadratic: $\Theta(n^2)$
  - Cubic: $\Theta(n^3)$
- Intractable:
  - Exponential: $\Theta(c^n)$ (c is a constant > 1)

class of polynomial time algorithms

Recap: Asymptotic Notation

- Given what we have learned over the last few slides, we have:

- $O(n)$: $1, \log n, n^{0.9}, 100n$
- $\Omega(n)$: $n, n\log n, n^2, 2^n$
- $\Theta(n)$: $n, 100n, n + \log n$
- $o(n)$: $1, \log n, n^{0.9}$
- $\omega(n)$: $n\log n, n^2, 2^n$

- We will work through some examples to further illustrate this