# Intro: Big-$O$ Notation

Daniel Kane

Department of Computer Science and Engineering
University of California, San Diego

## Data Structures and Algorithms
## Algorithmic Toolbox

# Learning Objectives

- Understand the meaning of Big-$O$ notation.
- Describe some of the advantages and disadvantages of using Big-$O$ notation.

# Big-$O$ Notation

## Definition

$f(n) = O(g(n))$ ($f$ is Big-$O$ of $g$) or $f \preceq g$ if there exist constants $N$ and $c$ so that for all $n \geq N$, $f(n) \leq c \cdot g(n)$.

$f$ is bounded above by some constant multiple of $g$.

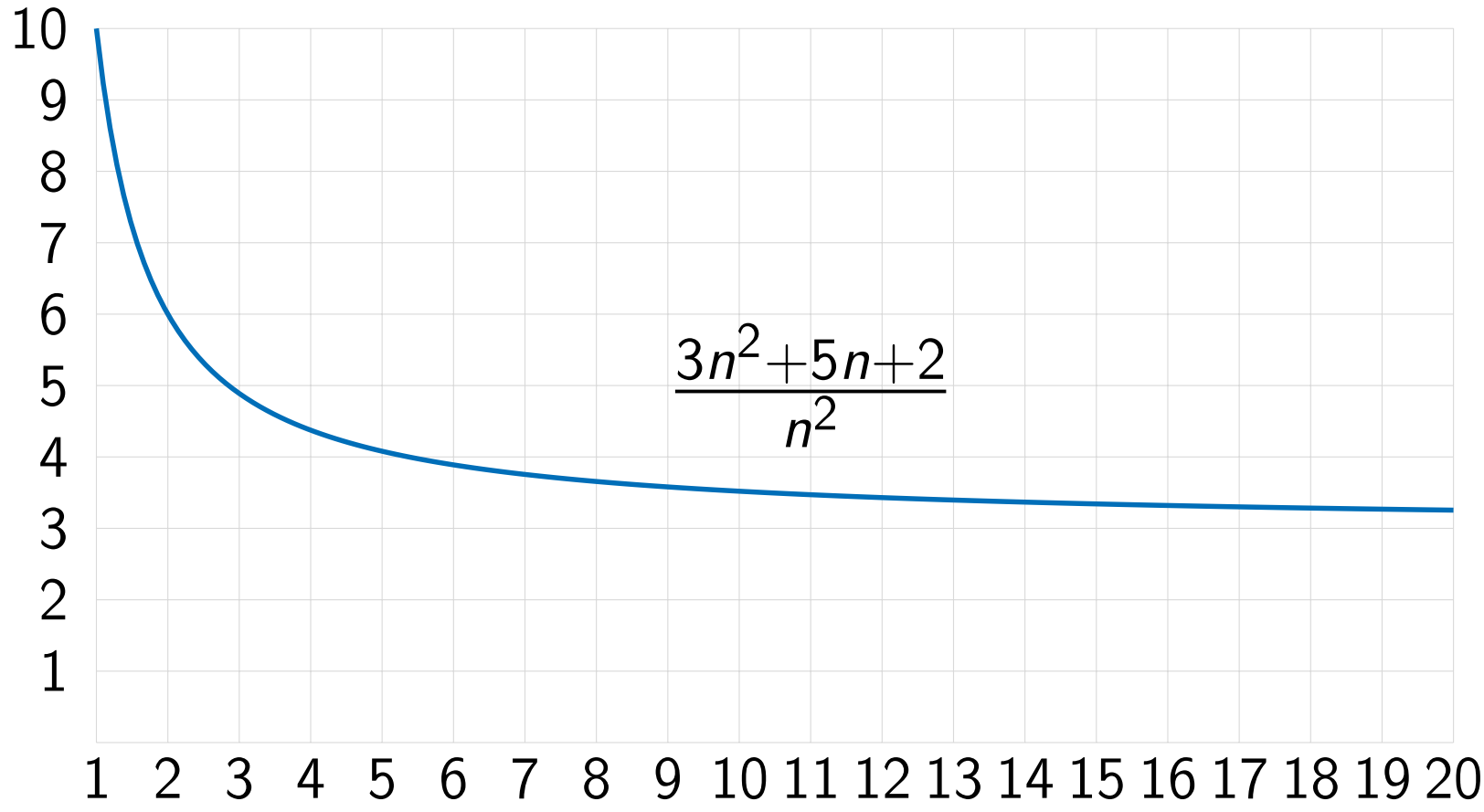
# Big-$O$ Notation

## Example

$3n^2 + 5n + 2 = O(n^2)$ since if $n \geq 1$,
$3n^2 + 5n + 2 \leq 3n^2 + 5n^2 + 2n^2 = 10n^2$.

# Growth Rate

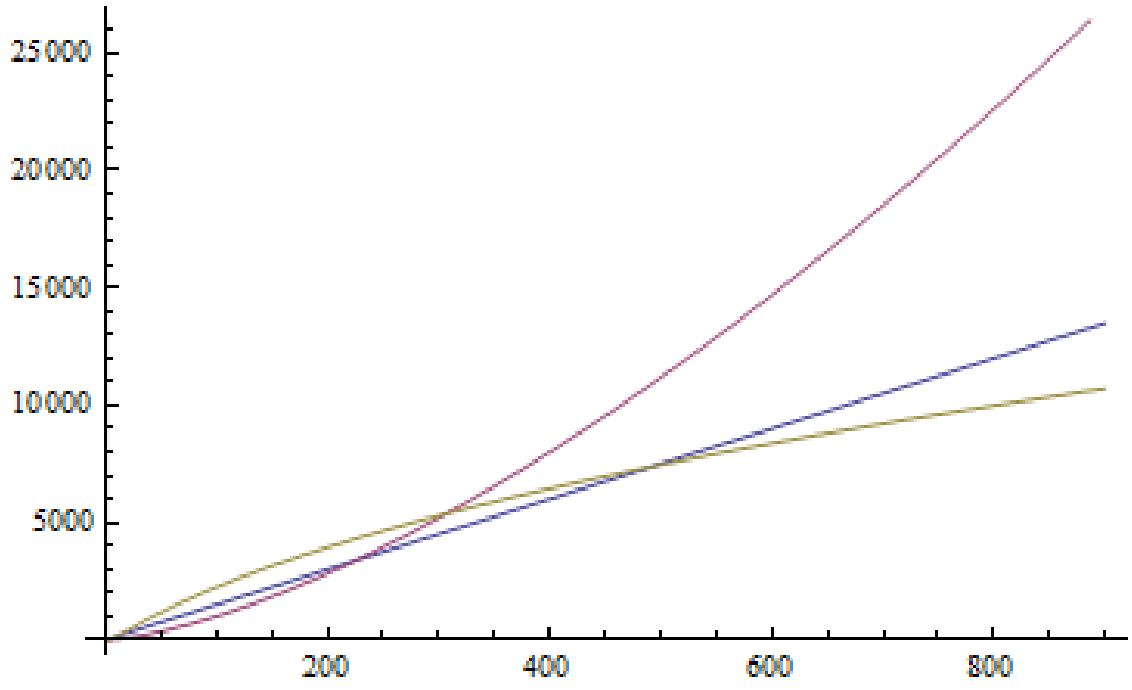
$3n^2 + 5n + 2$ has the same growth rate as $n^2$



# Using Big-$O$

We will use Big-$O$ notation to report algorithm runtimes. This has several advantages.

# Clarifies Growth Rate

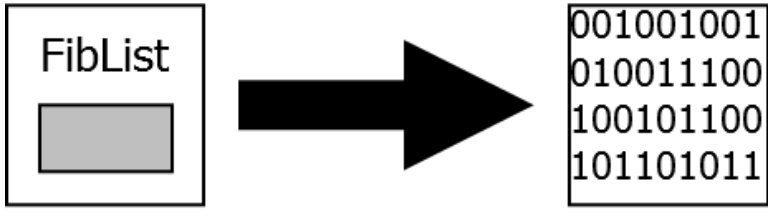


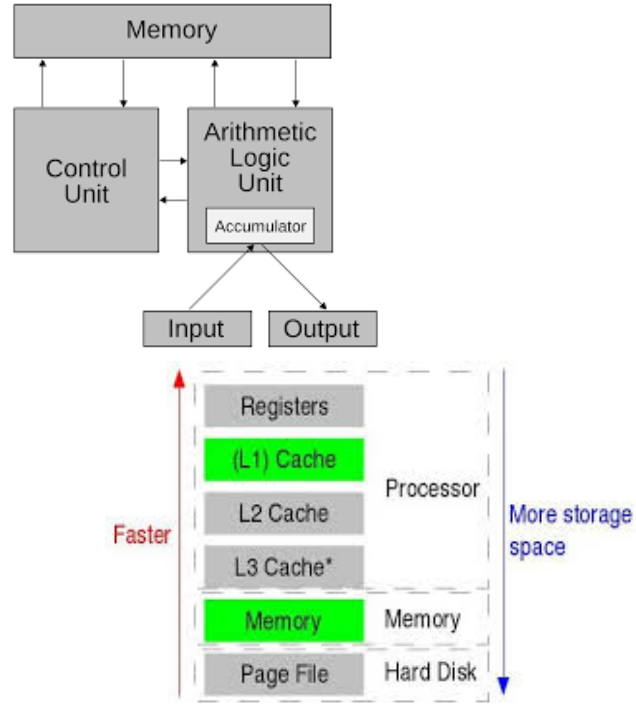
# Cleans up Notation

- $O(n^2)$ vs. $3n^2 + 5n + 2$.
- $O(n)$ vs. $n + \log_2(n) + \sin(n)$.
- $O(n \log(n))$ vs. $4n \log_2(n) + 7$.
  - Note: $\log_2(n), \log_3(n), \log_x(n)$ differ by constant multiples, don't need to specify which.
- Makes algebra easier.

# Can Ignore Complicated Details

No longer need to worry about:



# Warning

- Using Big-$O$ loses important information about constant multiples.
- Big-$O$ is *only* asymptotic.