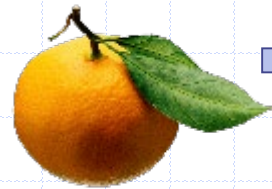
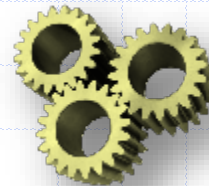


# Algorithm Analysis



**Input**



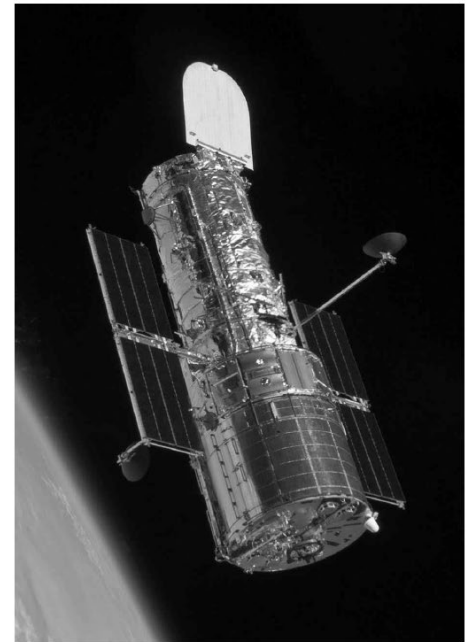
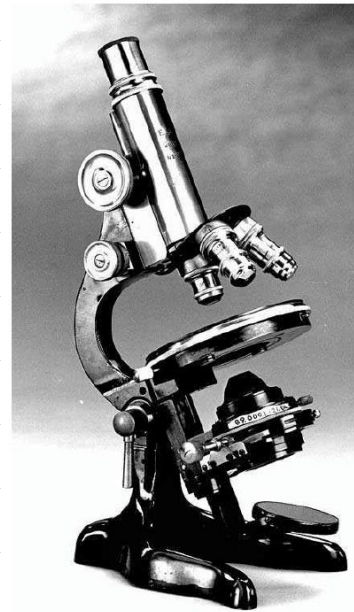
**Algorithm**



**Output**

# Scalability

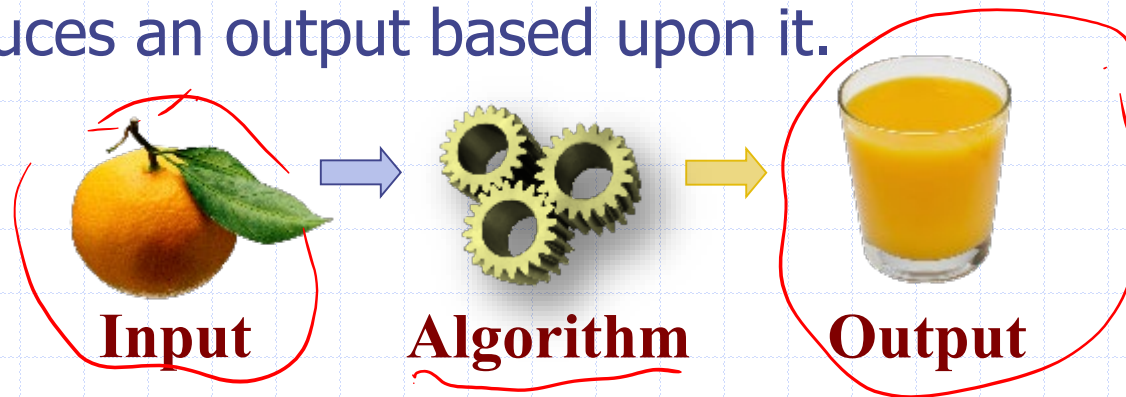
- ❑ Scientists often have to deal with differences in scale, from the microscopically small to the astronomically large.
- ❑ Computer scientists must also deal with scale, but they deal with it primarily in terms of data volume rather than physical object size.
- ❑ **Scalability** refers to the ability of a system to gracefully accommodate growing sizes of inputs or amounts of workload.



Microscope: U.S. government image, from the N.I.H. Medical Instrument Gallery, DeWitt Stetten, Jr., Museum of Medical Research. Hubble Space Telescope: U.S. government image, from NASA, STS-125 Crew, May 25, 2009.

# Algorithms and Data Structures

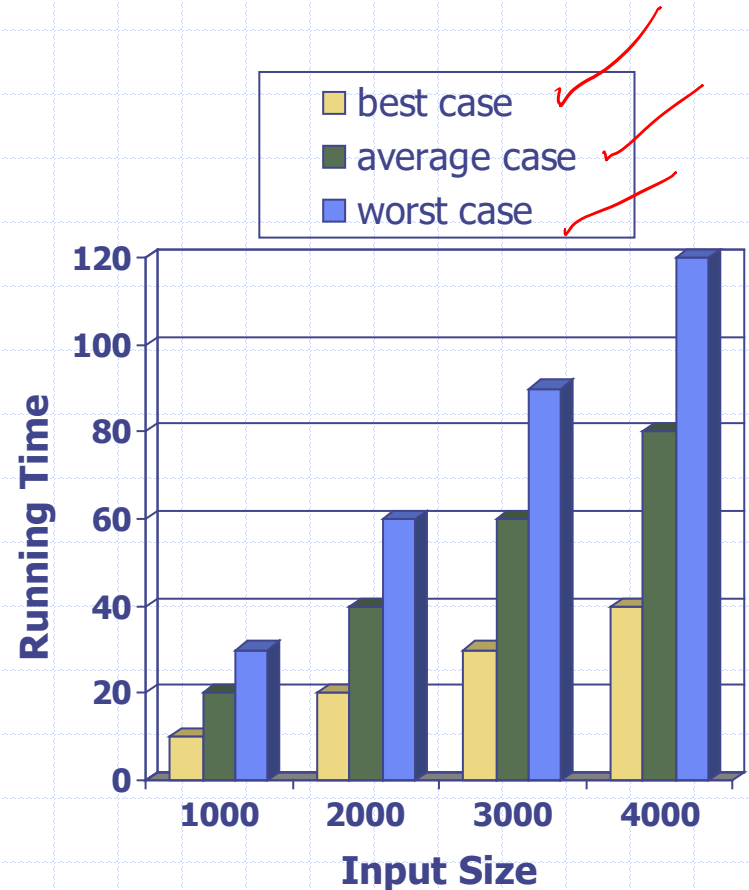
- An **algorithm** is a step-by-step procedure for performing some task in a finite amount of time.
  - Typically, an algorithm takes input data and produces an output based upon it.



- A **data structure** is a systematic way of organizing and accessing data.

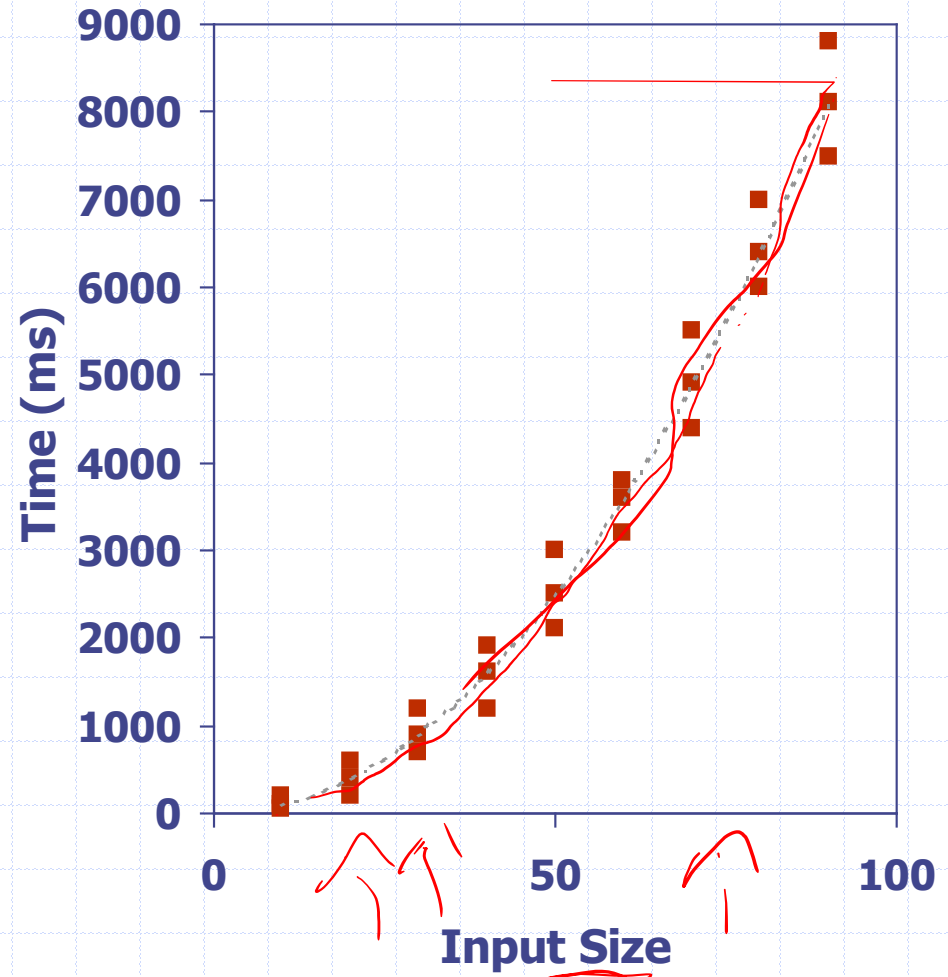
# Running Times

- Most algorithms transform input objects into output objects.
- The running time of an algorithm typically grows with the input size.
- Average case time is often difficult to determine.
- We focus primarily on the **worst case running time**.
  - Theoretical analysis ✓
  - Might not capture real-world performance



# Experimental Studies

- Write a program implementing the algorithm
- Run the program with inputs of varying size and composition, noting the time needed:
- Plot the results
- Try to match a curve to the times

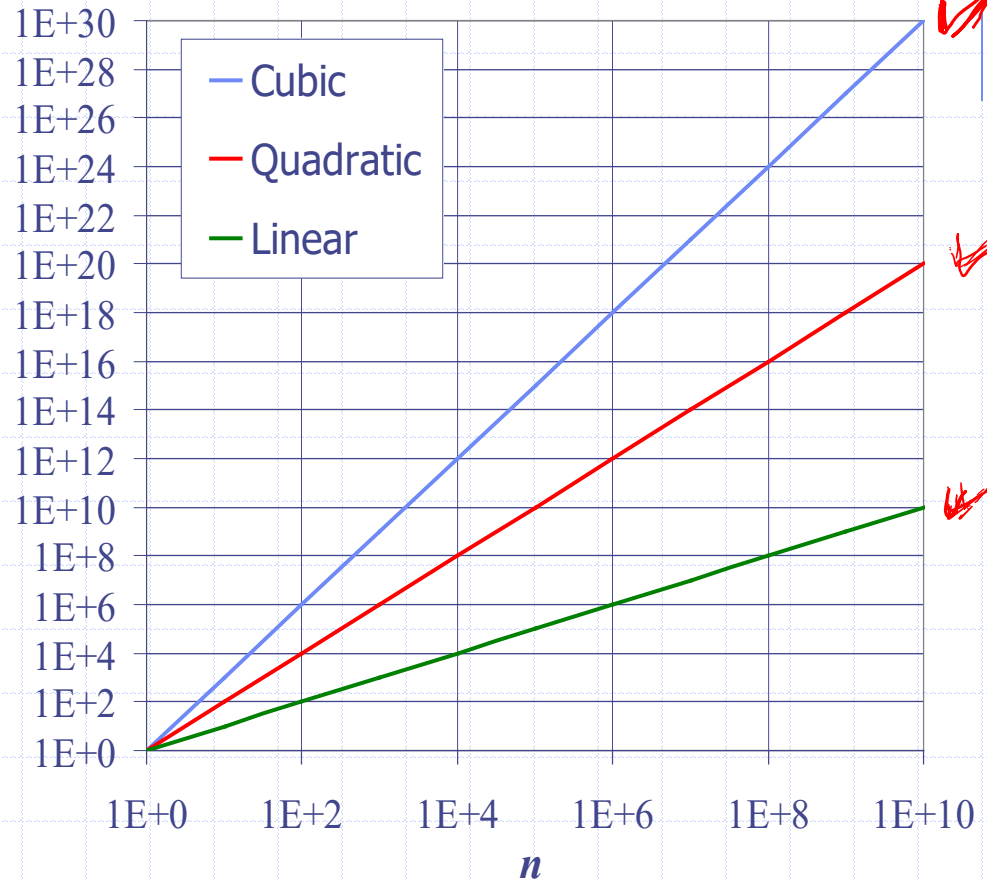


# Seven Important Functions

- Seven functions that often appear in algorithm analysis:

- Constant  $\approx 1$
- Logarithmic  $\approx \log n$
- Linear  $\approx n$
- N-Log-N  $\approx n \log n$
- Quadratic  $\approx n^2$
- Cubic  $\approx n^3$
- Exponential  $\approx 2^n$

- In a **log-log chart**, the slope of the line corresponds to the exponent in the growth rate



Log-Log

$$y = n^c$$

$$\log y = \log n^c$$

$$\log y = c \log n$$

$$y = c^x$$

# Why Growth Rate Matters

if runtime is...	time for $n + 1$	time for $2n$	time for $4n$
$c \lg n$	$c \lg (n + 1)$	$c (\lg n + 1)$	$c(\lg n + 2)$
$cn$	$c(n + 1)$	$2cn$	$4cn$
$cn \lg n$	$\sim cn \lg n + cn$	$2cn \lg n + 2cn$	$4cn \lg n + 4cn$
$cn^2$	$\sim cn^2 + 2cn$	<b><math>4cn^2</math></b>	$16cn^2$
$cn^3$	$\sim cn^3 + 3cn^2$	$8cn^3$	$64cn^3$
$c2^n$	$c2^{n+1}$	$c2^{2n}$	$c2^{4n}$

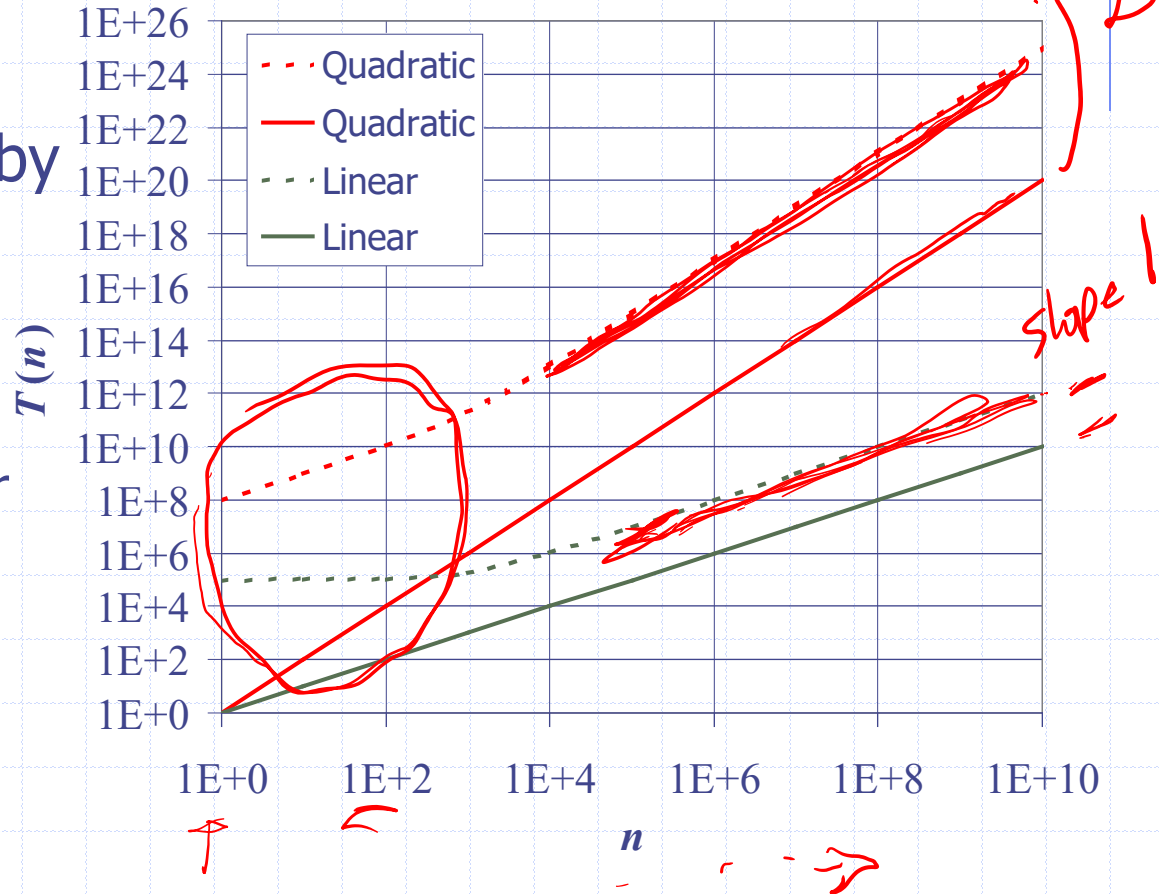
runtime quadruples when problem size doubles





# Constant Factors

- The growth rate is minimally affected by
  - constant factors or
  - lower-order terms
- Examples
  - $10^2n + 10^5$  is a linear function
  - $10^5n^2 + 10^8n$  is a quadratic function



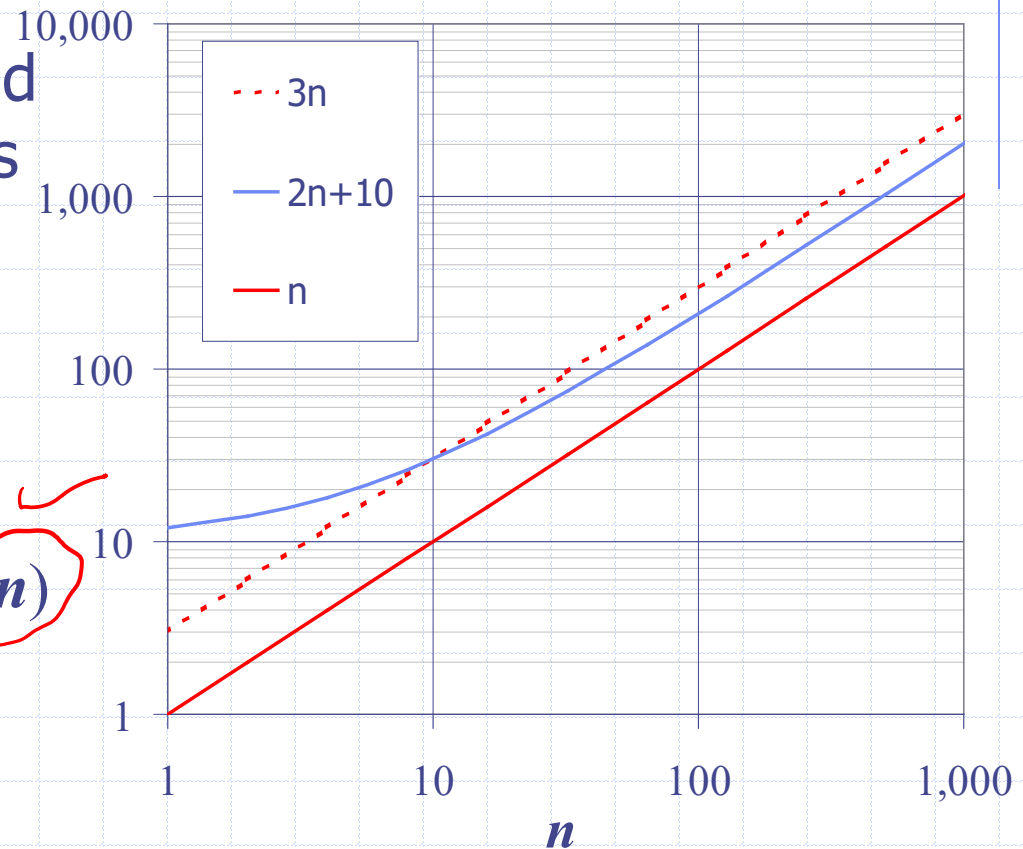
# Big-Oh Notation

- Given functions  $f(n)$  and  $g(n)$ , we say that  $f(n)$  is  $O(g(n))$  if there are positive constants  $c$  and  $n_0$  such that

$$f(n) \leq cg(n) \text{ for } n \geq n_0$$

- Example:  $2n + 10$  is  $O(n)$

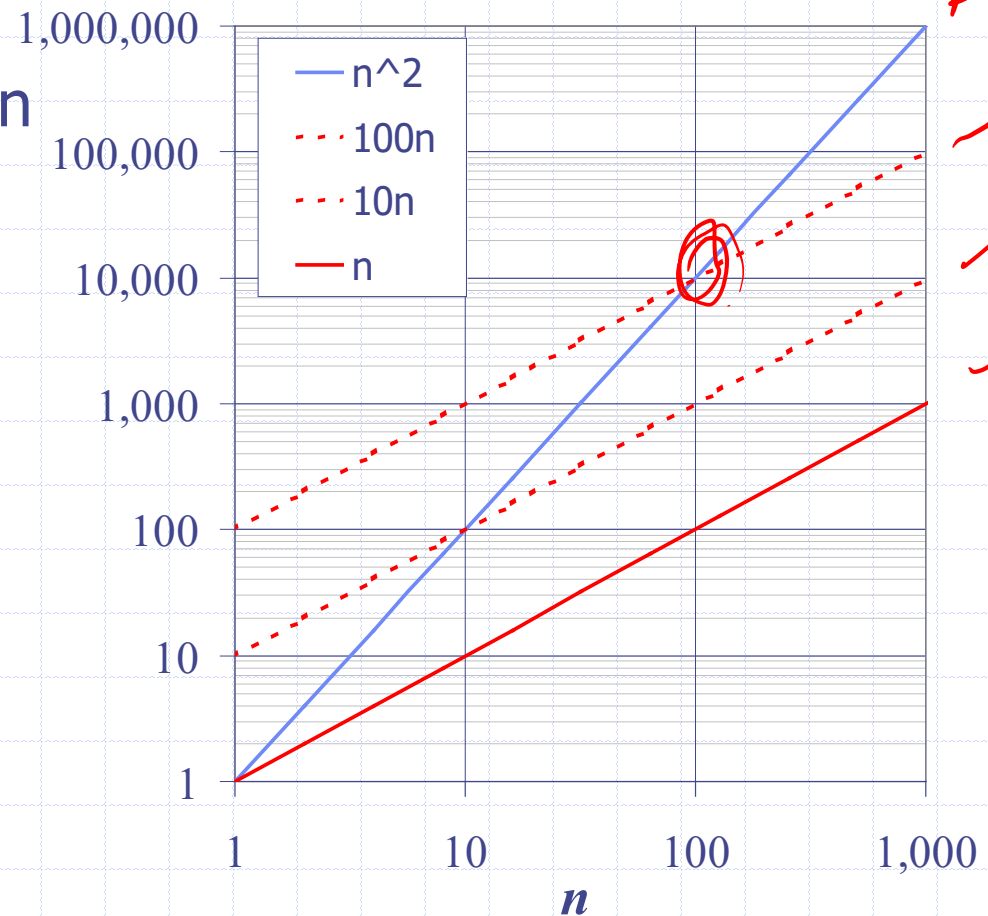
- $2n + 10 \leq cn$
- $(c - 2)n \geq 10$
- $n \geq 10/(c - 2)$
- Pick  $c = 3$  and  $n_0 = 10$



# Big-Oh Example

□ Example: the function  $n^2$  is not  $O(n)$

- $n^2 \leq cn$
- $n \leq c$
- The above inequality cannot be satisfied since  $c$  must be a constant



# Big-Oh Rules



- If  $f(n)$  is a polynomial of degree  $d$ , then  $f(n)$  is  $O(n^d)$ , i.e.,
  1. Drop lower-order terms ✓
  2. Drop constant factors ✓
- Use the smallest possible class of functions
  - Say “ $2n$  is  $O(n)$ ” instead of “ $2n$  is  $O(n^2)$ ”
- Use the simplest expression of the class
  - Say “ $3n + 5$  is  $O(n)$ ” instead of “ $3n + 5$  is  $O(3n)$ ”

# Relatives of Big-Oh



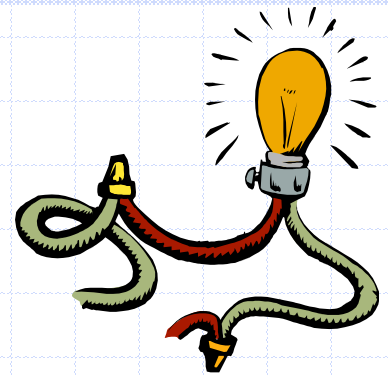
## big-Omega

- $f(n)$  is  $\Omega(g(n))$  if there is a constant  $c > 0$  and an integer constant  $n_0 \geq 1$  such that
$$f(n) \geq c g(n) \text{ for } n \geq n_0$$

## big-Theta

- $f(n)$  is  $\Theta(g(n))$  if there are constants  $c' > 0$  and  $c'' > 0$  and an integer constant  $n_0 \geq 1$  such that
$$c'g(n) \leq f(n) \leq c''g(n) \text{ for } n \geq n_0 .$$

# Intuition for Asymptotic Notation



## big-Oh

- $f(n)$  is  $O(g(n))$  if  $f(n)$  is asymptotically less than or equal to  $g(n)$

} upper

## big-Omega

- $f(n)$  is  $\Omega(g(n))$  if  $f(n)$  is asymptotically greater than or equal to  $g(n)$

} lower

## big-Theta

- $f(n)$  is  $\Theta(g(n))$  if  $f(n)$  is asymptotically equal to  $g(n)$

} equal