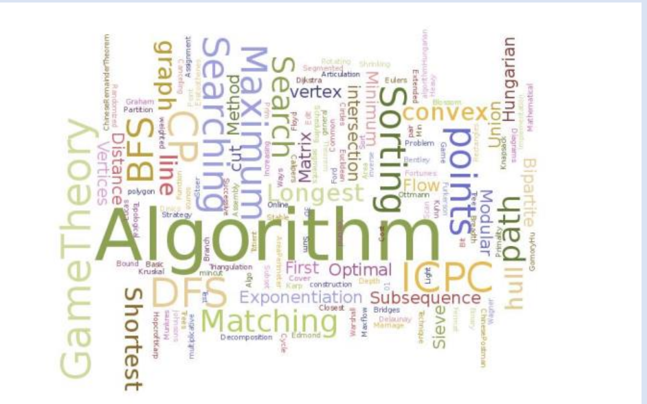
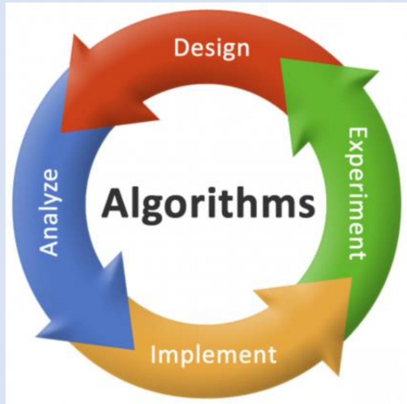
# Growth of Functions

COP 3503
Fall 2021
Department of Computer Science
University of Central Florida
Dr. Steinberg

## Why talk about Growth of Functions?

- We have observed that algorithms have different running times.
- These running times are represented as functions.
- As input size grows, so does the function output.

## Asymptotic Notation

- Asymptotic Notation describes the behavior  of algorithms.
- Last class when we used big-Theta to describe the number of times a statement is executed, that was asymptotic notation.
- There are a total of 5 notations that can be used to describe algorithms. In this class we are primarily interested in running time, however asymptotic notation can help understand other aspects such as algorithm's space requirement.
- In this course, we are primarily interested in describing the running time.
- The 5 notations are O, Θ, Ω, ω, o

## O-Notation (big-Oh)

- We use O-notation to give an upper bound (asymptotic upper bound) on a function, to within a constant factor.
- $f(n) = O\big(g(n)\big)$
- Formal Definition:
  - $O\big(g(n)\big) = \{f(n)$: there exist a positive constant $c$ $(c > 0)$ and $n_0 \geq 0$ such that $0 \leq f(n) \leq c * g(n)$ for all $n \geq n_0\}$

## Ω-Notation (big-Omega)

- We use Ω-notation to give a lower bound (asymptotic lower bound) on a function, within a constant factor.
- $f(n) = \Omega\big(g(n)\big)$
- Formal Definition:
  - $\Omega\big(g(n)\big) = \{f(n)$: there exist a positive constants $c$ $(c > 0)$ and $n_0 \geq 0$ such that $0 \leq c * g(n) \leq f(n)$ for all $n \geq n_0\}$

## Θ-Notation (big Theta)

- We use Θ -notation to give a tight bound (asymptotic tight bound) on a function, to within a constant factors.
- Formal Definition:
  - $\Theta(g(n)) = \{f(n)$: there exist positive constants $c_1$ $(c_1 > 0), c_2$ $(c_2 > 0)$ and $n_0 \geq 0$ such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0\}$
- Utilized when f(n) and g(n) have the same <u>order of growth</u>
- Theorem

$$f(n) = \Theta(g(n)) \; iff \; \begin{cases} f(n) = O(g(n)) \\ f(n) = \Omega(g(n)) \end{cases}$$

## o-Notation (little oh)

- Used to represent an upperbound, which is not asymptotically tight
- Formal Definition
  - $o\big(g(n)\big) = \{f(n){:}\, for\ any\ positive\ constant\ c > 0$, there exists a positive constant $n_0 > 0$ such that $0 \leq f(n) < cg(n)$ for all $n \geq n_0\}$
- Quick Definition
  - $f(n) = o\big(g(n)\big)\ iff\quad \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

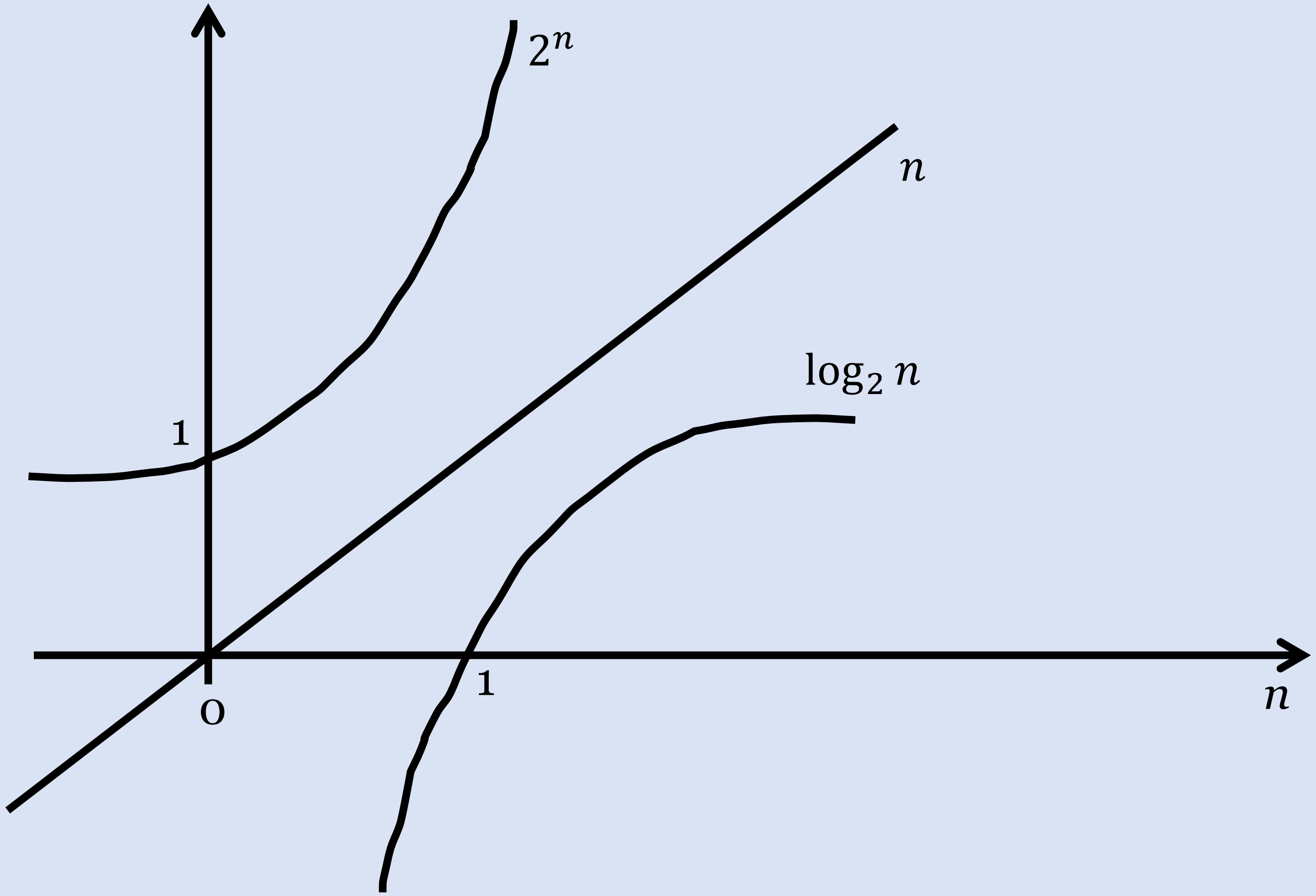
## $\omega$-Notation

- Used to represent a lower bound, which is not asymptotically tight
- Formal Definition
  - $\omega(g(n)) = \{f(n): for\ any\ positive\ constant\ c > 0$ , there exists a positive constant $n_0$ such that $0 \leq cg(n) < f(n)$ for all $n \geq n_0\}$
- Quick Definition
  - $f(n) = \omega(g(n))\ iff \quad \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$

## Interesting Analogies

- $f = O(g) \quad f \leq g$
- $f = \Omega(g) \quad f \geq g$
- $f = \Theta(g) \quad f = g$
- $f = \mathrm{o}(g) \quad f < g$
- $f = \omega(g) \quad f > g$

## Rates of growth between polynomial, exponential, and polylogarithmic functions


$2^n$
$n$
$\log_2 n$
$n$