# Midterm prep – CS 325 Analysis of Algorithms

Table of Contents

[Midterm prep – CS 325 Analysis of Algorithms 1](#_Toc95424495)

[Algorithms / Strategies: 1](#_Toc95424496)

[Time complexity (mod1) 1](#_Toc95424497)

[Big O (mod1) 2](#_Toc95424498)

[Theta (mod1) 2](#_Toc95424499)

[Omega (mod1) 3](#_Toc95424500)

[Compare order of Growth 4](#_Toc95424501)

Mid term does not include Greedy technique topic.  
  
You might find it helpful to refer to the Week 5 & Midterm webinar slide deck to get guidance on preparation.  
  
There will not be any coding questions. The questions will be similar to the ones that you have seen in the quizzes and there would be a couple of essay pattern questions where you will describe the algorithm strategy or write the pseudocode.

Hints:  
1- Make sure to know the difference between Big O, theta, and omega.  
2- Know the growth rate and what functions are growing faster  
3- Determine the growth rate of an expression or a pseudocode  
4- Know the properties of each of the algorithm techniques (divide and conquer, dynamic programming, backtracking, etc.)  
5- Make sure to read the questions carefully.

# Algorithms / Strategies:

## Time complexity (mod1)

* Also known as time efficiency – how fast an algorithm runs
* Table

  Description automatically generated
* Table

  Description automatically generated

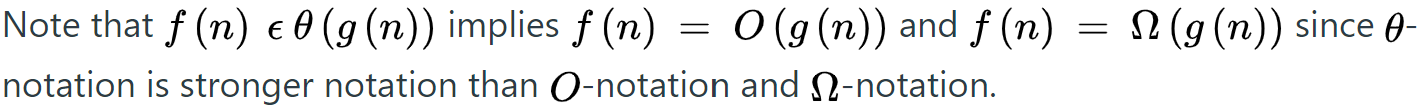
## Big O (mod1)

* T(n) “is O(g(n))” if T(n) ∈ O(g(n))
* T(n) is all functions with lower or same order of growth as g(n)
* O(g(n)) means T(n) falls withing g(n) to c(g(n)), for positive value c.
* Prove f(n) ≤ c(g(n))
* Formally: T(n) is in O(g(n)) if T(n) ≤ c(g(n)) for all n ≥ n0, for some positive constant c

Table

Description automatically generated

## Theta (mod1)

* 
* Use L’Hopital’s Rule or Stirling’s Formula (large values of n)
* T(n) “is Θ(g(n))” if T(n) ∈ Θ(g(n))
* Formally: T(n) is in Θ(g(n)), if c1(g(n)) ≤ T(n) ≤ c2(g(n)) for all n ≥ n0, for some positive constants c1 and c2.

Text

Description automatically generated

## Omega (mod1)

* T(n) “is Ω(g(n))” if T(n) ∈ Ω(g(n))
* Formally: T(n) is in Ω(g(n)), it T(n) ≥ c(g(n)) for all n ≥ n0, for some positive constant c.

Text

Description automatically generated

## Compare order of Growth

* Text

  Description automatically generated
* A picture containing table

  Description automatically generated
* Table

  Description automatically generated

**Divide and Conquer (mod2)**

**Function Growth rates**

Recursion (mod2)

Recurrence relation (mod2)

Recurrence formula (mod2)

**Dynamic Programming (mod3, mod4)**

**Backtracking (mod4, mod5)**