THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON DC

INTRODUCTION AND ASYMPTOTIC NOTATION

CS 6212 – Design and Analysis of Algorithms

CLASS & TEACHING STYLE

- Active Class
  - Frequent 1-3 minute discussion sessions
  - Talking/discussing/explaining concepts helps
- Programming Projects
  - Code style matters, long methods, variable naming, all are subject to criticism
- Frequent HWs and Quizzes
  - **No makeup for anything missed**
  - Do not hesitate to interrupt!
- Teaching Style (Criticism ☺)
  - I may not always give direct answers

LOGISTICS

- **Instructor**

  Prof. Amrinder Arora

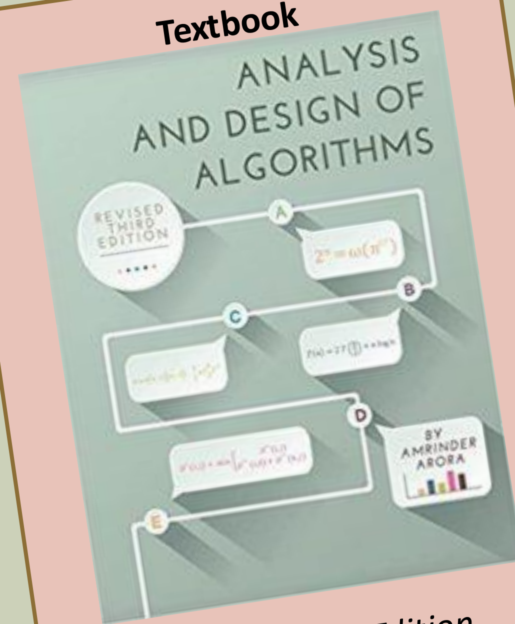
  amrinder@gwu.edu

  Please copy TA on emails

  Please feel free to call as well
  ☺

- Available for study sessions
  Science and Engineering Hall
  GWU

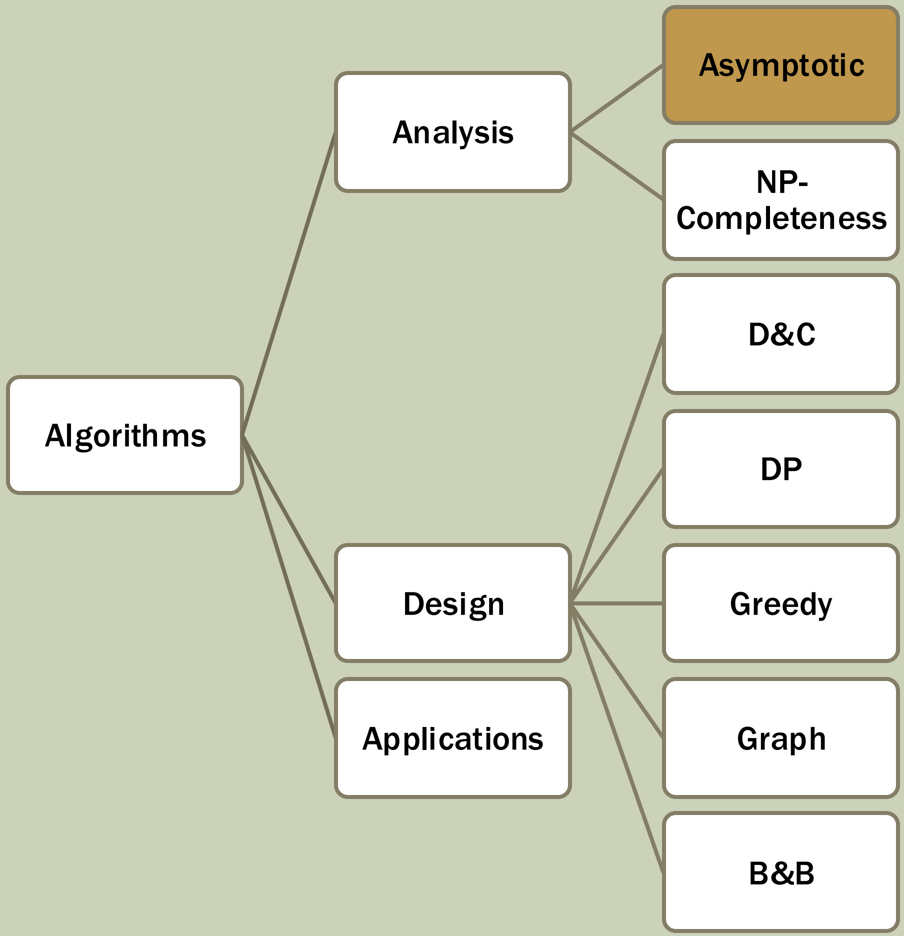
Textbook



Revised Third Edition
Cognella Academic Publisher
www.cognella.com

ISBN: **978-1793549952**

COURSE OUTLINE



PURPOSE OF THIS CLASS

Design and **Analysis** of **Algorithms**

- Designing – Algorithmic Techniques
- Analyzing – how much time an algorithm takes
- Proving inherent complexity of problems

"ALGORITHM" – DEFINITIONS

- A precise statement to solve a problem on a computer
- A sequence of definite instructions to do a certain job

REPRESENTING AN ALGORITHM

Pseudo-code consisting of:

- Variables and assignments
- Data Structures (Arrays, objects, etc.)
- Loops
- If Else/Switch/Case
- Function/Procedure

EXAMPLE 1: INSERTION SORT

Given: An array A of n numbers

Purpose: To sort the array

Algorithm:

```
for j = 1 to n-1
    key = A[j]
    // A[j] is added in the sorted sequence A[0..j-1]
    i = j - 1
    while i >= 0 and A [i] > key
        A[i + 1] = A[i]
        i = i - 1
    A[i + 1] = key
```

EXAMPLE 2: EUCLID'S ALGORITHM

- Best case running time
- Worst case running time
- Average case running time

```
gcd(n,m) {
   r = n%m
   if r == 0 return m

   // else
   return gcd(m, r)
}
```

Fresh in the market.
Cutting edge material!

EXAMPLE 3: BINARY SEARCH

```
binarySearch(A[0..N-1], value, low, high)
{
  if (high < low)
      return -1           // not found
  mid = (low + high) / 2
  if (A[mid] > value)
      return BinarySearch(A, value, low, mid-1)
  else if (A[mid] < value)
      return BinarySearch(A, value, mid+1, high)
  else return mid                // found
}
```

ANALYZING AN ALGORITHM

- How long will the algorithm take?
- How much memory will it require?

For Example:

```
function sum (array a) {
   sum = 0;
   for (int j : a)  {
      sum = sum + j
   }
   return sum;
}
```

ANALYZING AN ALGORITHM

- Why to do it?
  - A priori estimation of performance
  - To compare algorithms on a level playing field
- How to do it? (What is the model?)
  - Random access memory model
  - Math operations take constant time
  - Read/write operations take constant time
- What do we compute?
  - Time complexity: # of operations as a function of input size
  - Space complexity: # of bits used

ANALYZING ALGORITHMS

How to Analyze a Given Algorithm (Program)

Some tips:

- When analyzing an if then else condition, consider the arm that takes the longest time
- When considering a loop, take the sum
- When considering a nested loop, …

WHAT IS THE TIME COMPLEXITY OF THIS PROGRAM?

Possible Quiz Questions

<table>
<tr>
<td>

```
j = 1
while (j < n) {
  k = 2
  while (k < n) {
    Sum += a[j]*b[k]
    k = k * k
  }
  j++
}
```

Θ(n log log n)

</td>
<td>

```
for (int j = 1 to n) {
  for (int k = j to n) {
    x = k
    while (x < n) {
      Sum +=
a[j]*b[k]*c[x]
      If (x % 3 == 0) {
        x = n + 1
      }
      x = x + 1
    }
  }
}
```

</td>
<td>

```
for (int j = 1 to n) {
  k = j
  while (k < n) {
    Sum += a[k]*b[k]
    k += log n
  }
}
```

</td>
</tr>
</table>

REQUIRED MATH CONSTRUCTS

- Sets, functions
- Logs and Exponents
  - Taking log to the base 2 or the base 10 of random numbers (without using a calculator)

- Recurrence Relations
- Sums of series
  - Arithmetic Progression: $1 + 2 + 3 + \dots + n$
  - Geometric Progress: $1 + 3 + 9 + \dots 3^k$
  - AGP: $1 + 2.3 + 3.9 + \dots (k+1)\, 3^k$
  - Others: $1^2 + 2^2 + 3^2 + \dots + n^2$, Harmonic Series

ASYMPTOTIC NOTATION

- Big O notation
  - $f(n) = O(g(n))$ if there exist constants n_0 and c such that $f(n) \le c\, g(n)$ for all $n \ge n_0$.

  For example, consider $f(n) = n$, and $g(n) = n^2$. Then, $f(n) = O(g(n))$

  If $f(n) = a_0 n^0 + a_1 n^1 + \ldots + a_m n^m$,
  then $f(n) = O(n^m)$
- Big Omega notation
  - $f(n) = \Omega(g(n))$ if there exist constants n_0 and c such that $f(n) \ge c\, g(n)$ for all $n \ge n_0$.

ASYMPTOTIC NOTATIONS (CONT.)

- Small o notation
  - $f(n) = o(g(n))$ if for any constant $c > 0$, there exists n_0 such that $0 \le f(n) < c\,g(n)$ for all $n \ge n_0$.

  For example, $n = o(n^2)$
- Small omega (ω) notation
  - $f(n) = \omega(g(n))$ if for any constant $c > 0$, there exists n_0 such that $f(n) \ge c\,g(n) \ge 0$, for all $n \ge n_0$

  For example, $n^3 = \omega(n^2)$

ASYMPTOTIC NOTATIONS (CONT.)

- $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$
- If $f(n) = O(g(n))$ and $g(n) = O(f(n))$, then $f(n) = \Theta(g(n))$
- $f(n) = o(g(n))$ if and only if $g(n) = \omega(f(n))$
- $f(n) = o(g(n))$ implies $\lim_{n\to\infty} f(n)/g(n) = 0$

PROVING SMALL OH USING L'HOPITAL'S RULE

- In some cases, we need to use the L'Hopital's rule to prove the small oh notation. L'Hopital's rule states that assuming certain conditions hold, $\lim_{x \to c} \frac{f(x)}{g(x)} = \lim_{x \to c} \frac{f'(x)}{g'(x)}$

- For example, suppose we want to prove the following:

  $(\log n)^3 + 3 (\log n)^2 = o(\sqrt{n})$

- We can find the limit of these functions as follows:

  $\lim_{n\to\infty}(\log n)^3 + 3 (\log n)^2 / \sqrt{n}$

  $= \lim_{n\to\infty} 6 (\log n)^2 + 12 \log n / n^{1/2}$ // Using L'Hopital's rule

  $= \lim_{n\to\infty} 24 \log n + 24/ n^{1/2}$ // Using L'Hopital's rule

  $= \lim_{n\to\infty} 48/ n^{1/2}$ // Using L'Hopital's rule

  $= 0$

ASYMPTOTIC NOTATIONS (CONT.)

Analogy with real numbers

O	o	Θ	ω	Ω
$\leq$	$<$	$=$	$>$	$\geq$

ASYMPTOTIC NOTATIONS (CONT.)

Which properties apply to which (of 5) asymptotic notations?

- Transitivity
- Reflexivity
- Symmetry
- Transpose Symmetry
- Trichotomy

ASYMPTOTIC NOTATIONS (CONT.)

Which properties apply to which (of 5) asymptotic notations?

- Transitivity: O, o, Θ, ω, Ω
- Reflexivity: O, Θ, Ω
- Symmetry: Θ
- Transpose Symmetry: (O with Ω, o with ω)
- Trichotomy: Does not hold. For real numbers x and y, we can always say that either x < y or x = y or x > y. For functions, we may not be able to say that. For example, if f(n) = sin(n) and g(n)=cos(n)

LOGISTICS

- Rigor required by this class
- Saturday study sessions (Optional)
- Study sessions on other days – You need to coordinate
- Grading – Review course information sheet in Blackboard

TO DOS

- Review Project P1
- Review Course Outline
- Lecture 2 – I will begin the slides at Graph (~slide 19)

HOMEWORK ASSIGNMENTS

- Via Blackboard
- Due one week from when they are given

HELPFUL LINKS

- Core Concepts on Wikipedia
  - http://en.wikipedia.org/wiki/Arithmetic_progression
  - http://en.wikipedia.org/wiki/Geometric_series
  - http://en.wikipedia.org/wiki/L%27H%C3%B4pital%27s_rule

- Videos
  - Class Prelim video: https://www.youtube.com/watch?v=50BuWtYyPk8
  - Limits and L'Hopitals Rule https://www.youtube.com/watch?v=PdSzruR5OeE

SOME LESSONS FROM PREVIOUS CLASSES

- Almost no correlation between grades and background
- Correlation between grades and number of classes attended
- Correlation between grades and time spent on course
- Strong correlation between grades and homeworks/projects

80% of success is showing up!