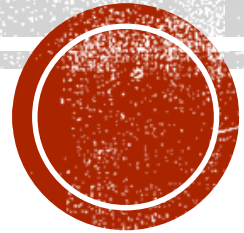
M0: SYLLABUS

M1: DATA STRUCTURES, ALGORITHMS ANALYSIS, BIG OH

Summer 2019 – CSC 395 – ST: Algorithm & Data Structure Concepts



CONTACT INFO

- **Instructor** Dr. Razvan Alex. Mezei
- **Email address** rmezei@stmartin.edu
- **Office** Panowicz Hall, Room 111
- **Office Phone** 360-688-2748

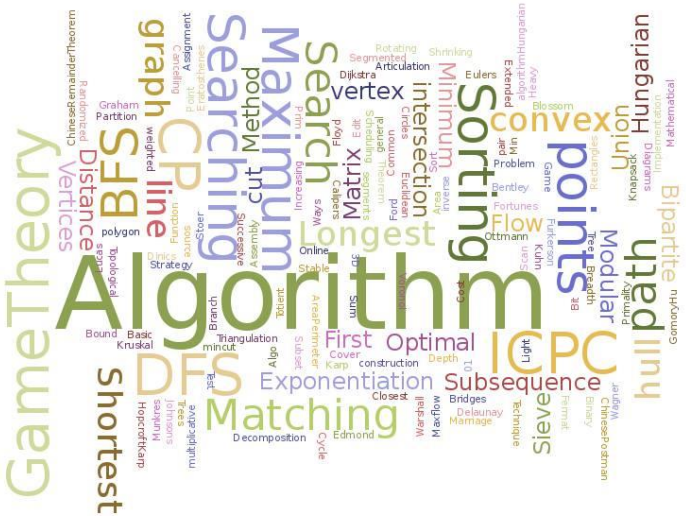
OFFICE HOURS

- **Drop-In** (no appointment needed!)**:**
  - TBD
- **By Appointment:**
  - Email me in advance to set up an appointment
- **By Zoom**
  - https://zoom.us/j/5517425424

Course Description:

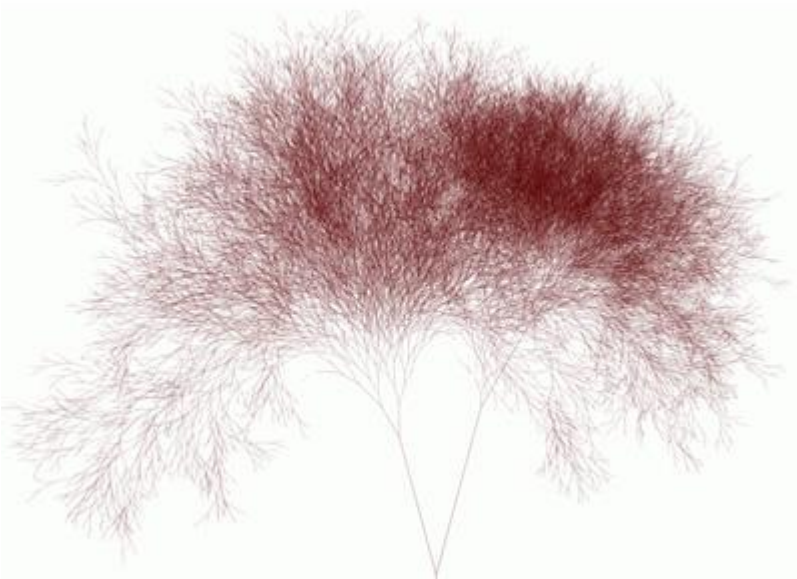
- A study of data types, abstract data types, data structures and associated algorithms. Use of lists, trees and graphs will be studied. Different searching and sorting algorithms will be examined..

- Note: this is similar to the regular CSC 340.

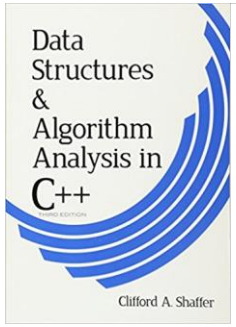


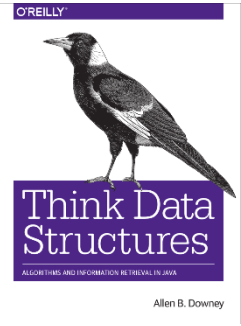
Required Textbook:

- None

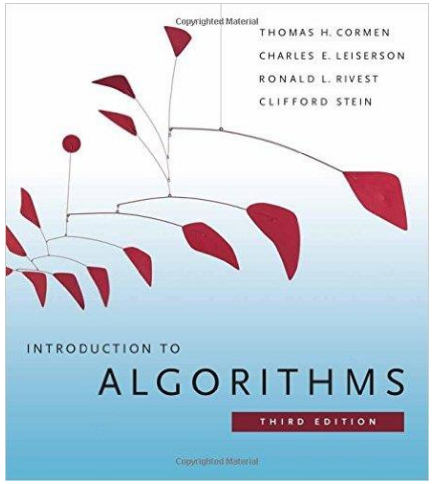


Other Recommended Textbook(s):





- http://people.cs.vt.edu/~shaffer/Book/C++3e20130328.pdf
- https://legacy.gitbook.com/book/cathyatseneca/data-structures-and-algorithms/details
- Cracking the coding interview – 189 programming questions and solutions, Gayle Laakmann McDowell, 6th edition
- http://greenteapress.com/thinkdast/thinkdast.pdf
- Introduction to Algorithms, Third Edition, by Comen, Leiserson, Rivest, Stein, ISBN: 978-0262033848
- http://www.aupress.ca/books/120226/ebook/99Z_Morin_2013-Open_Data_Structures.pdf
- https://mva.microsoft.com/en-US/training-courses/c-fundamentals-for-absolute-beginners-16169
- https://www.edx.org/course/algorithms-and-data-structures-3



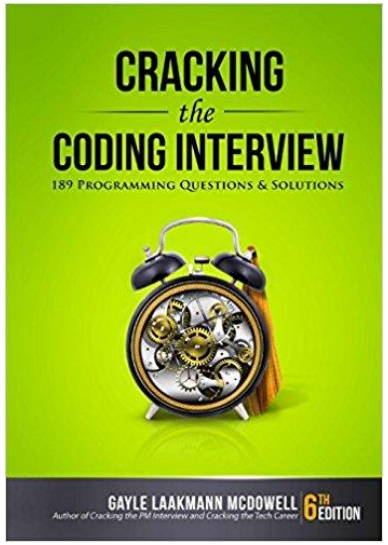
Download Microsoft Visual Studio

- We will be using Microsoft Visual Studio on the computers in the classroom. If you wish to install Visual Studio on your personal computer or laptop, you can download it for free. https://www.visualstudio.com/en-us/visual-studio-homepage-vs.aspx Select Community Edition

Algorithms and Data Structures

Learn how to write faster and more efficient code against the backdrop of famous algorithms.

Microsoft



Course Objectives

- Comprehend the concept of data structure
- Comprehend the techniques needed for algorithm analysis
- Apply algorithm analysis skills to analyze the performance of various algorithms
- Implement searching algorithms such as: sequential searching and binary searching
- Implement data structures such as: lists, stack, and queues
- Implement sorting algorithms such as: bubble sort, selection sort, insertion sort, merge sort, heap sort, and quick sort.
- Implement binary trees, binary search trees, graphs, graph algorithms
- Comprehend the concepts of hash tables, collision, iterators.
- Compare between different data structures. Pick an appropriate data structure for a design situation.

Learning Objectives

- **Module 1: Syllabus, C# quick review, Data Structures and Algorithms Basics**
  - Comprehend the concept of data structure
  - Quick review of C#. Implement simple algorithms in C#.
  - Comprehend the techniques of algorithm analysis.
  - Apply algorithm analysis techniques to various algorithms
- **Module 2: Searching and Sorting Algorithms**
  - Comprehend the working process of sequential search and binary search
  - Apply sequential search and binary search to find the target element in C#
  - Comprehend the working process of bubble sort, selection sort, insertion sort, merge sort, and quicksort.
  - Implement bubble sort, selection sort, insertion sort, merge sort, and quicksort in C#
- **Module 3: Lists, Sack, and Queues**
  - Comprehend the basic idea of lists, stack, and queues
  - Implement lists, stack, and queues in C#
- **Module 4: Binary Trees and Binary Search Trees**
  - Comprehend the basic idea of binary tree and the working process of build a binary tree
  - Implement binary trees and binary search trees in C#
- **Module 5: Graphs and Graph Algorithms**
  - Comprehend the concept of graphs
  - Comprehend how to represent graphs
  - Implement graphs and graph algorithms in C#
- **Module 6: Hash tables and Array Lists**
  - Comprehend the concept of array lists, hash tables, and collision.
  - Implement array lists in C#
  - Apply hash tables to specific applications in C#

Methods of Assessment

In addition to the ones below, there may be some unannounced pop-quizzes.

- **Unexcused absences may lower your grade!**

HW Programming assignments	60% ← open notes
Mid-term Exam (June 5th, 2019)	20% ← closed notes
Final Exam (July 3rd, 2019)	20% ← closed notes

Grading Standards:

A-: 90-92	A: 93-96	A+:97-100
B-: 80-82	B: 83-86	B+:87-89
C-: 70-72	C: 73-76	C+:77-79
D-: 60-62	D: 63-66	D+:67-69
F : 0–59		

If your code does not compile, crashes at start, or contains no meaningful comments, it will automatically be graded with 0!

Homeworks are usually due on Tuesdays (typically the week after they were assigned).

Course Schedule (tentative)

- The dates, schedule, requirements etc. are tentative. The instructor reserves the right to make changes to the course schedule during the duration of the course. Students will be informed, in advance, of any changes and an updated schedule will be distributed and posted to Moodle. Students are responsible for checking Moodle and completing any requirements posted there.

Syllabus Updates and Moodle Listings

- The dates, schedule, requirements etc. are tentative. The instructor reserves the right to make changes to the syllabus during the duration of the course. Students will be informed, in advance, of any changes and an updated syllabus will be distributed and posted to Moodle. Students are responsible for checking Moodle and completing any requirements posted there.

- **Any kind of disruptive behavior will not be tolerated**

- **Attendance policy**:
  - You are expected to attend each class meeting.
    - Every two incidents of coming to class late by 5 minutes or more, or leaving early, equals one absence.
      - If you come late to class, you will not be given any extra time for tests or quizzes (unless you contacted me ahead of time and I accepted your excuse as reasonable)
    - For each 4 unexcused absences (that is two weeks' worth of classes) your grade may decrease by a letter grade.
  - Students absent from an announced quiz or examination, unless excused, may not be permitted to make up quizzes or examinations.
  - Pop-quizzes cannot be made-up.

- During my office hours don't expect me to teach you everything you may have missed in class. I am expecting you to at least read from the textbook before you stop by my office. Come with questions. I will do my best to answer as many questions as you have.

- You are expected to do your own work on the homework and tests.
  - Penalties for turning in work which is not your own will be SEVERE.

- **Late Submission Policy:** Unless otherwise specified, the homework/project is due 11:59 pm, on the given due date. Late work submission will not be accepted (unless you are excused for class) and will get a 0 grade. Do not submit code that does not compile or contains no comments. Such code will receive 0 points.
  - If your code does not compile, crashes at start, or contains no meaningful comments, it will automatically be graded with 0!
  - You are expected to keep the instructor informed of any changes or issues that may prevent you from completing your assignments or this course.

- **Cell Phones and other electronic devices (including iPods, iPhones, etc):** these devices are to be put on silent or turned off before entering the class and shall remain this way for the duration of the class. Cell phones are not to be visible or used at any time, especially not during quizzes or exams. If your device is disrupting the class and I need to ask you to leave early, then you will also get an UNEXCUSED absence.

- You are expected to **check your SMU email at least once a day**. This will be the way I will try to reach you (in case I need to inform you of any schedule changes, send you more course related materials, etc.)

- Besides my office hours, there is free tutoring available on campus. Check the following link for more details: www.stmartin.edu/academics/academic-resources/center-student-learning-writing-and-advising/peer-tutoring

How to Get Started with Moodle

- Visit: http://moodle.stmartin.edu
- See "Getting Started with Moodle," "Tutorials for Students," and "Frequently Asked Questions" in the pane entitled "NAVIGATION" in the left margin.
- LOGIN with your Saint Martin's username and password when ready. For username and password help, see: https://www.stmartin.edu/directory/integrated-technology-services/technology-help

Standard St. Martin's information

- Helpful Links
  - Link to Saint Martin's undergraduate Academic Catalog 2018-2019
    https://www.stmartin.edu/sites/default/files/smu-files/registrar/ug_1819_final_noindex.pdf
  - Link to Student Handbook currently in use (2018-2019):
    https://www.stmartin.edu/sites/default/files/smu-files/about/student-handbook-2018-2019.pdf
  - Link to emergency/weather information:
    http://www.stmartin.edu/directory/office-public-safety
  - Link to sign up for e2campus emergency alert text messaging:
    https://www.e2campus.net/my/stmartin/signup.htm
  - Link to Office of Registrar forms:
    http://www.stmartin.edu/forms?topic=Registrar

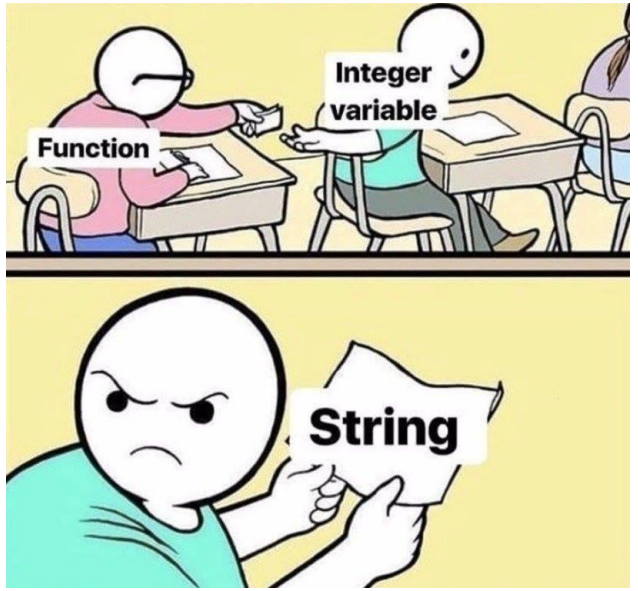
Standard St. Martin's information

- Saint Martin's University and the O'Grady Library
  - The O'Grady Library collections and services are available to all Saint Martin's University students, regardless of physical location. To explore the collections, services and research tools available, visit the O'Grady Library homepage: https://www.stmartin.edu/academics/academic-resources/ogrady-library

- Research assistance
  - **Appointments**: Librarians are available to make either in-person or phone appointments with individuals and/or groups working on research projects. The librarians can do a general tour of the library, orient you to the services and collections available or provide in-depth instruction on using the research tools available for your class. To request a consultation, please use the form: https://www.stmartin.edu/academics/academic-resources/ogrady-library/research/consultations
  - **Chat service**: The library also partners with a nationwide group of librarians to offer a chat service 24 hours a day, 7 days a week. Working late on a paper and not sure where to start? The chat service librarians can access almost all of the library's resources and can often help you get started on a project or help you figure out if something is available. To start a chat, go to: http://tinyurl.com/dyaew23
  - **Research guides**: The library also has a series of research guides that pull together key databases, reference sources and websites that are particularly useful in a subject area. To explore the research guides, go to: http://stmartin.libguides.com/
  - **How do I…? tutorials**: The O'Grady Library has a collection of "how to" tutorials designed to help you get started on research at the library. To explore the tutorials, go to http://stmartin.libguides.com/tutorials/

- **Administrative Policies/Support**
  - Access and Accommodations
    - Your experience in this class is important to me. If you have already established accommodations with Disability Support Services for Students (DSS), please communicate your approved accommodations to me at your earliest convenience so we can discuss your needs in this course.
    - If you have not yet established services through DSS, but have a temporary health condition or permanent disability that requires accommodations (conditions include but not limited to; mental health, attention-related, learning, vision, hearing, physical or health impacts), you are welcome to contact DSS at 360-438-4580 or dss.testing@stmartin.edu or smu.dss@stmartin.edu DSS offers resources and coordinates reasonable accommodations for students with disabilities and/or temporary health conditions. Reasonable accommodations are established through an interactive process between you, your instructor(s) and DSS. It is the policy and practice of the Saint Martin's University to create inclusive and accessible learning environments consistent with federal and state law

- Academic Honesty/Professionalism
  - **WHAT IS ACADEMIC INTEGRITY?** Saint Martin's University is a community of faculty, students and staff engaged in the exchange of ideas in the ongoing pursuit of academic excellence. Essential to our mission is a focused commitment to scholarly values, intellectual integrity and a respect for the ideas, beliefs and work of others. This commitment extends to all aspects of academic performance. All members are expected to abide by ethical standards both in their conduct and their exercise of responsibility to themselves and toward other members of the community. As an expression of our shared belief in the Benedictine tradition, we support the intellectual, social, emotional, physical and spiritual nurturing of students.
  - **WHAT IS ACADEMIC DISHONESTY?** Saint Martin's University defines Academic Dishonesty as violating the academic integrity of an assignment, test and/or evaluation of any coursework. This dishonest practice occurs when you seek to gain for yourself or another, an academic advantage by deception or other dishonest means. You have a responsibility to understand the requirements that apply to particular assessments and to be aware of acceptable academic practice regarding the use of material prepared by others. Therefore, it is your responsibility to be familiar with the policies surrounding Academic Dishonesty as these may differ from other institutions.



- University Sanctioned Activities
  - If you are absent from class due to university sanctioned activities, such as sports, it is your responsibility to request that the absence be excused, otherwise, the absence will be recorded as unexcused. Absent students are responsible for catching up with the class, and if any assignments are due on the day of the absence, it is your responsibility to turn in the assignments on time (prior to class). Assignments may be submitted as an attachment to email: rmezei@stmartin.edu. Please request the policy handout, "Requirement for receiving Excused Absence" on the first day of the class if you think this policy might apply to you.

- Center for Learning, Writing, and Advising
  - The Center for Student Learning, Writing and Advising offers free academic services for all Saint Martin's students at all levels of achievement in pursuit of intellectual growth and academic excellence. The Learning Center is home to the STEM Study Center which provides subject area peer tutoring (science, technology, engineering, and math as well as business/ accounting/economics, and world languages). At the Writing Center, students meet with trained peer readers to discuss their academic, personal and professional writing. The Advising Center works with students with academic advising, connecting with campus support resources, transition and self-exploration guidance, personalized academic improvement plans, learning workshops, and support major change. The Advising Center staff also works closely with the University's Early Alert Program — a referral system that supports student success. Saint Martin's Disability Support Services is located in the Center for any student with a disability who is interested in using their accommodations. These students can connect with the Disability Support Services Coordinator who will evaluate the documentation, determine appropriate accommodations, and serve as a learning resource and advocate with assisting students in meeting their academic goals. https://www.stmartin.edu/academics/academic-resources/center-student-learning-writing-and-advising

- Counselling and Wellness Center
  - The Counseling and Wellness Center (CWC) is committed to helping you meet the challenges of life you may experience during college. The CWC promotes and enhances the health and development of students through professional mental health services, education and training. Integrating faith, reason and service, we empower you to develop self-awareness, knowledge and skills, necessary to make healthy choices and build relationships in a multicultural world. Integrating faith, reason and service, the CWC empowers students to develop self-awareness, knowledge and the skills necessary to make healthy choices and build relationships in a multicultural world. https://www.stmartin.edu/directory/counseling-and-wellness-center
- Sexual Misconduct/Sexual Harrassment Reporting
  - Saint Martin's University is committed to providing an environment free from sex discrimination, including sexual harassment and sexual violence. There are Title IX/sexual harassment posters around campus that include the contact information for confidential reporting and formal reporting. Confidential reporting is where you can talk about incidents of sexual harassment and gender-based crimes including sexual assault, stalking, and domestic/relationship violence. This confidential resource can help you without having to report your situation to the formal reporting process through the Dean of Students – Ms. Melanie Richardson, Associate VP of Human Resources – Ms. Cynthia Johnson, Public Safety – Mr. Will Stakelin, or the Office of the Provost – unless you request that they make a report. Please be aware that in compliance with Title IX and under the Saint Martin's University policies, educators must report incidents of sexual harassment and gender-based crimes including sexual assault, stalking, and domestic/relationship violence.
  - If you disclose any of these situations in class, on papers, or to me personally, I am required to report it.

YOU ARE EXPECTED TO DO A LOT OF READING ON YOUR OWN!

PLEASE STUDY THE ASSIGNED TEXTBOOK AND VIDEO RESOURCES!

VERY TENTATIVE SCHEDULE

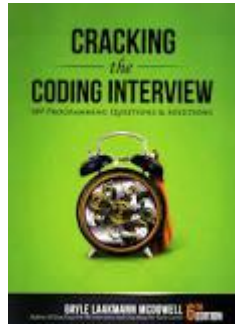
CSC 395 - SUMMER 2019 - 08									
W	15-May	syllabus, big-Oh data structures	M1		W	12-Jun	binary tree	M4	
F	17-May	C#, arrays, files, searching, lib.	M2		F	14-Jun	binary tree	M4	
W	22-May	sorting, sorting lib	M2		W	19-Jun	POSSE	?	
F	24-May	quicksort, linked list	M3		F	21-Jun	graphs, graph ADT	M5	
W	29-May	linked list	M3		W	26-Jun	graph algorithms	M5	
F	31-May	arrayList(?), stack, queues, dll (if time)	M3		F	28-Jun	hash tables and array lists	M6	
W	5-Jun	MIDE TERM EXAM - AP CS ???			W	3-Jul	FINAL EXAM		
F	7-Jun	AP CS - Kansas			F	5-Jul	FINAL EXAM - last day of classes - skip		

notice: I need to find someone to proctor the mid-term exam.

WHY THIS COURSE IS USEFUL

- will focus on the basic concepts, job-interview questions, as well as some programming competition questions related to the topics mentioned above.

AN EMAIL ...FOR A POSITION AT GOOGLE



- As a reminder, you'll be interviewing for the Software Engineer role in the Bay Area.
- Prep Material and Links:
- **Cracking the Coding Interview by Gayle Laakmann McDowell comes highly recommended as an initial resource from which to study
- **Your coding competency will be evaluated. Function names will also be noted, the code needs to be almost compilable. I had an engineer say this to me once:
- "It's fine if you don't remember the name of a function in the standard library, or if you write a function signature and say "I'll fill this in later", as long as I'm confident that you actually can write it. But otherwise, I expect it to be pretty close to compilable, I need to know you can actually write in a language."
- *Your interviews will cover data structures and algorithms (i.e. Big-O Notation). I'd focus on repetition of algorithmic based problems- see below websites with which to practice.
- *Be sure to talk through your thought process about the questions you are asked as well as your approach to problems and solutions. Ask specific questions if you need more clarification. Also, don't forget to think about ways to improve the initial solution you present.

AN EMAIL ...FOR A POSITION AT GOOGLE (2)

- *Feel free to check out algorithm tutorials and other refresher tools here: http://community.topcoder.com/ tc?module=Static
- *Visit www.interviewcake.com for practice problems
- *We've also received great feedback on http://www.lintcode.com/en/
- *check out www.leetcode.com for more practice problems
- Life at Google: Interview Info (YouTube Videos)
- How to: Work at Google — How We Hire: https://www.youtube.com/watch?v=k-baHBzWe4k
- How to: Prepare for a Google Engineering Interview: https://www.youtube.com/watch?v=ko-KkSmp-Lk
- How to: Work at Google — Example Coding/Engineering Interview: https://www.youtube.com/watch?v=XKu_SEDAykw&t

SOME MOTIVATION – AN ONLINE DISCUSSION

- It's always hard to know what to expect when going in for that interview -- and preparation can make all the difference. What is the interview process like at Microsoft?
- Any advice on striking the right note?
- […]
- Refresh your algorithm and data structure knowledge (binary trees, hashes, doubly linked stuff etc).
- […]
- You need to be good on data structure and C++ classes
- […]
- You need to be refresh the data structure and algorithm concepts and basic algorithm related to linked lists, tree, stack, queue. Need to know about Big O time notation. What ever algo you write most certainly they will ask for time complexity so be prepared for that, also with some list of basic test cases for any algorithm.

Source: https://www.indeed.com/forum/cmp/Microsoft/Microsoft-Interview-Questions/t9502

C# QUICK OVERVIEW

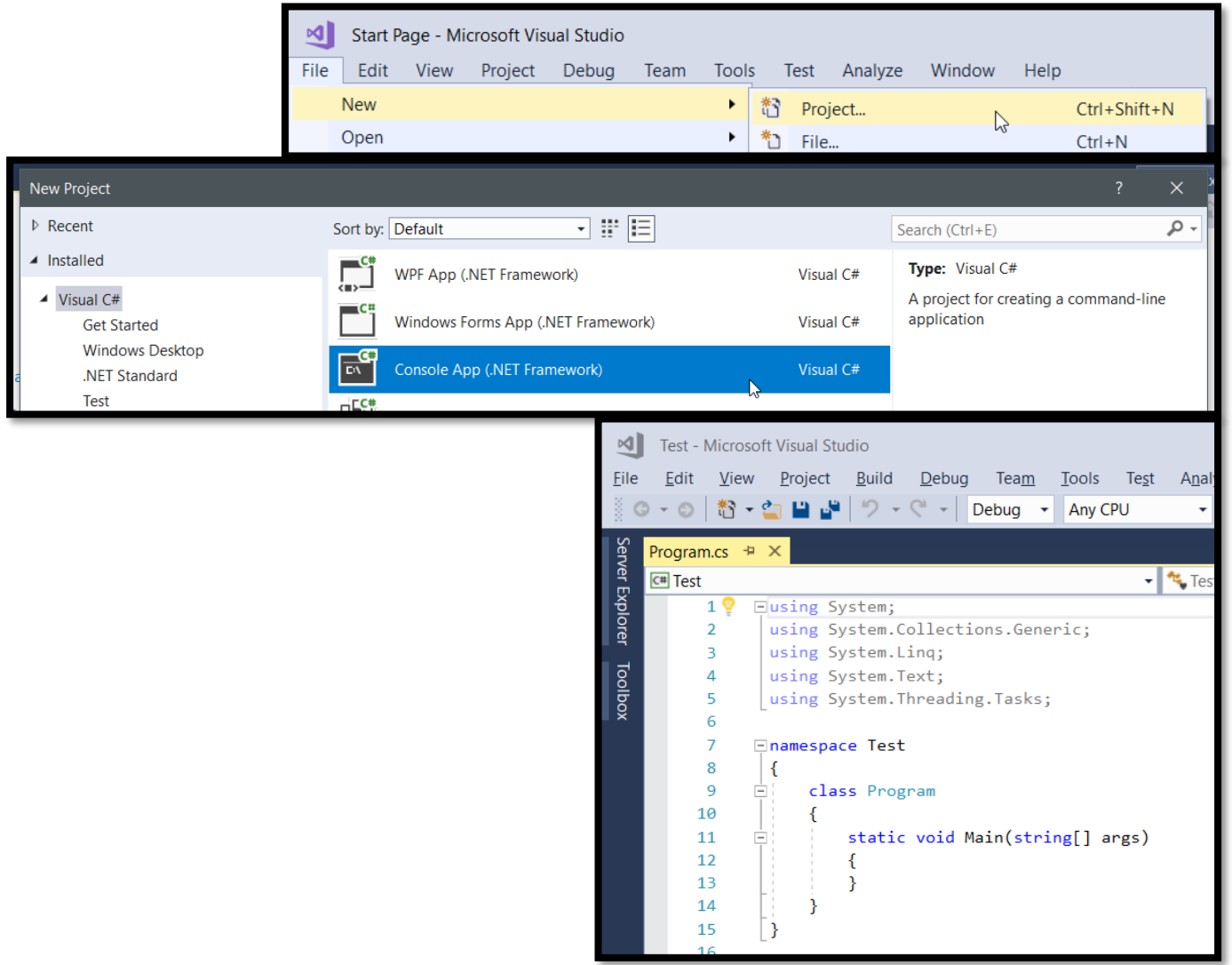
- Please review the following topics and make sure you master them before we go more deep into this course
- I am here to help so please stop use my office hours whenever you have any questions or want to just talk about CS

INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

- We will be using Microsoft Visual Studio on the computers in the classroom. If you wish to install Visual Studio on your personal computer or laptop, you can download it for free.
  - https://www.visualstudio.com/en-us/visual-studio-homepage-vs.aspx
    - Select Community Edition
  - Quick overview on how to install Visual Studio … https://mva.microsoft.com/en-US/training-courses/c-fundamentals-for-absolute-beginners-16169?l=4RDmRvOZD_5401937557
  - Creating your first C# application … https://mva.microsoft.com/en-US/training-courses/c-fundamentals-for-absolute-beginners-16169?l=p90QdGQIC_7106218949

CREATING A C# PROJECT

- Open Visual Studio
- File > New > Project …
- Visual C# > Console App



A C# PROGRAM CONSISTS OF THE FOLLOWING PARTS - SKIP:

- Namespace declaration
- A class
- Class methods
- Class attributes
- A Main method
- Statements and Expressions
- Comments

```
using System;

namespace HelloWorldApplication {
   class HelloWorld {
      static void Main(string[] args) {
         /* my first program in C# */
         Console.WriteLine("Hello World");
         Console.ReadKey();
      }
   }
}
```

DATA TYPES – REVIEW ON YOUR OWN!

The variables in C#, are categorized into the following types:

- Value types: int, byte, char, double, bool, …
- Reference types: object, dynamic, string, …
- Pointer types: int* iptr;

VARIABLES – REVIEW ON YOUR OWN!

SYNTAX

- DataType variableName = value;

EXAMPLE

- string Name = "thecodingguys";
- int Year = 2013;

ARRAYS – REVIEW ON YOUR OWN!

SYNTAX

- DataType[ ] ArrayName = { Comma Separated Values } // Array of any size
- DataType[] ArrayName = new DataType[3] {Command Separated Values } //Expects 3 values

EXAMPLE

- string[] MyGamesOf2013 = {"GTAV", "Battlefield3"};
- string[] MyMoveisOf2013 = new string[3] {"The Amazing Spiderman", "The Expendables 2", "Rise of the planet of the apes"};

YOU SHOULD ALSO MASTER:

Conditional statements

Loops

Methods & classes

Working with files (read from/write to)

Classes and objects

*If you don't master these, please stop by my office so I can help you get up to speed!

EXCEPTIONS

SYNTAX

```
try {
        /***/
}
catch (Exception) {
        throw;
}
```

EXAMPLE

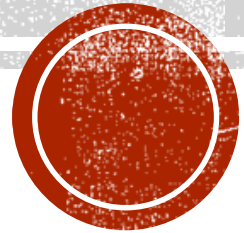
```
try {
        string result = "k";
        Console.WriteLine(Convert.ToInt32(result) + 10);
}
catch (Exception ex) {
        Console.WriteLine(ex.Message);
}
```

OTHER USEFUL LINKS

- suggested reading:
  - https://www.tutorialspoint.com/csharp/index.htm
  - https://www.tutorialspoint.com/csharp/csharp_tutorial.pdf
  - https://www.thecodingguys.net/resources/cs-cheat-sheet.pdf
- suggested videos:
  - https://mva.microsoft.com/en-US/training-courses/c-fundamentals-for-absolute-beginners-16169
  - https://courses.edx.org/courses/course-v1:Microsoft+DEV285x+1T2018a/course/
- links:
  - https://www.visualstudio.com/en-us/visual-studio-homepage-vs.aspx <--Select Community Edition
  - https://www.tutorialspoint.com/csharp/csharp_tutorial.pdf

M1: DATA STRUCTURES, ALGRITHMS ANALYSIS, BIG OH

Summer 2019 – CSC 395 – Data Structures and Algorithms



MAIN TEXTBOOK SOURCE

- See chapters 1 & 3 from http://people.cs.vt.edu/~shaffer/Book/C++3e20130328.pdf

- Note: this is a little more mathematical chapter, but it will be useful for the remaining of this course. Do not panic!
  - All other chapters will focus on coding, and we'll only use the knowledge from this chapter to analyze the complexity of the algorithms we'll develop next.

INTRODUCTION

- Suppose you have to solve problems like:
  - Find the shortest route from downtown Lacey, WA to downtown Seattle, WA
  - Find the quickest route from downtown Lacey, WA to downtown Seattle, WA
  - How many cities with more than 10,000 people lie within 500 miles of Lacey, WA?
  - Find all patients of Dr. Who that didn't have an flu shot in the past 12 months
  - Find all providers that have seen patient John Jay in the last three years

- To answer questions like these, it is **not enough to have** the necessary information. **We must organize that information** in a way that allows us to find the answers in time to satisfy our needs.
- The primary purpose of most computer programs is not to perform calculations, but to **store and retrieve information** — usually **as fast as possible**

THREE PRIMARY GOALS FOR THE COURSE

- The first is to **present the commonly used data structures**. These form a programmer's basic data structure "toolkit." For many problems, some data structure in the toolkit provides a good solution.

- The second goal is to **introduce the idea of tradeoffs** and reinforce the concept that there are costs and benefits associated with every data structure. This is done by describing, for each data structure, the **amount of space and time required for typical operations**.

- The third goal is to teach how to **measure the effectiveness of a data structure or algorithm**. Only through such measurement can you determine which data structure in your toolkit is most appropriate for a new problem.
  - We'll learn about **asymptotic analysis**

THE NEED FOR DATA STRUCTURES

- Processor speed and memory size still continue to improve. Won't any efficiency problem we might have today be solved by tomorrow's hardware?
  - We solve more and more complex problems
  - We work with larger and larger data sets … big data
  - We also need efficiency in order to minimize the power consumption our smartphones

DATA TYPES

- Type is a collection of values.
  - For example, the bool type consists of the values true and false.
  - An integer is a simple type because its values contain no subparts. A bank account record will typically contain several pieces of information such as name, address, account number, and account balance. Such a record is an example of an aggregate type or composite type.

- Data type = is a type together with a collection of operations to manipulate the type.
  - an integer variable is a member of the integer data type.
  - Addition is an example of an operation on the integer data type.

- Two data types
  - Built-in Data Type: bool, byte, int, string, char…
    - data types for which a language has built-in support
  - Derived Data Type: Student, List, SortedList, Queue, Dictionary
    - data types built by combination of built-in data types and associated operations on them.

DATA STRUCTURES AND ADT

- An abstract data type (ADT) is a mathematical model for data types, where a data type is defined by its behavior from the point of view of a user of the data
  - specifically in terms of possible values, possible operations on data of this type, and the behavior of these operations.
  - The concept of an ADT can help us to focus on key issues even in non-computing applications.
  - An ADT does not specify how the data type is implemented.
    - **Examples: linked lists, stacks, queues, binary trees, graphs …**

- A data structure is the implementation for an ADT.
  - In an object-oriented language such as C#, an ADT and its implementation together make up a class. Each operation associated with the ADT is implemented by a member function or method.
  - Example … a queue is … (discussion – ADT, implementation,…)
  - See the book … file structure vs data structure …

COMMON OPERATIONS

- Examples of data structures we'll see in this course:
  - Array lists, Linked lists, Stacks, Queues, Binary Trees, Graphs

- The following are common operations that can be found in collection data structures
  - Traversing
  - Searching
  - Insertion
  - Deletion
  - Sorting
  - Merging

- However, using the proper data structure can make the difference between a program running in a few seconds and one requiring many days.

- We'll see several ADTs in the next chapters. This is just an introduction!

SELECTING A DATA STRUCTURE – SKIP

1. Analyze your problem to determine the basic operations that must be supported.
   - Examples of basic operations include inserting a data item into the data structure, deleting a data item from the data structure, and finding a specified data item.
2. Quantify the resource constraints for each operation.
3. Select the data structure that best meets these requirements.

- Each data structure has associated costs and benefits.
  - it is hardly ever true that one data structure is better than another for use in all situations.
  - If one data structure or algorithm is superior to another in all respects, the inferior one will usually have long been forgotten.

ALGORITHMS BASICS – SKIP

- What is an algorithm
  - It is a step by step procedure, which defines a set of instructions to be executed in certain order to get the desired output.

- By definition, something can only be called an algorithm if it has all of:
  - It must be correct.
  - It is composed of a series of concrete steps.
  - There can be no ambiguity as to which step will be performed next.
  - It must be composed of a finite number of steps.
  - It must terminate. In other words, it may not go into an infinite loop.

ALGORITHM ANALYSIS

- How do you compare two algorithms for solving some problem in terms of efficiency?
  1. One could implement both algorithms as computer programs and then run them …
  2. Asymptotic analysis

- Asymptotic analysis **measures the efficiency of an algorithm as a function of the input size**.
  - For the remaining of this lecture we'll focus on this topic (asymptotic analysis)
  - Typically you will analyze:
    - the time required for an algorithm (or the instantiation of an algorithm in the form of a program), and
    - the space required for a data structure.

OTHER USEFUL LINKS

- Big-O notation in 5 minutes — The basics: https://www.youtube.com/watch?v=__vX2sjlpXU

- http://bigocheatsheet.com/pdf/big-o-cheatsheet.pdf

ASYMPTOTIC ANALYSIS – EXAMPLE

- Consider a simple algorithm to solve the problem of finding the largest value in an array of n integers. The algorithm looks at each integer in turn, saving the position of the largest value seen so far.
  - This algorithm is called the largest-value sequential search.

- What affects how long it takes?
  - Size? The numbers to search through? The speed of the computer (CPU, ram, etc)

- Our focus will be on the size of the array, n.

- Let c (some contant) denote the time it takes to compare two numbers.
  - To simplify, we will also include here the amount it takes to increment the counter

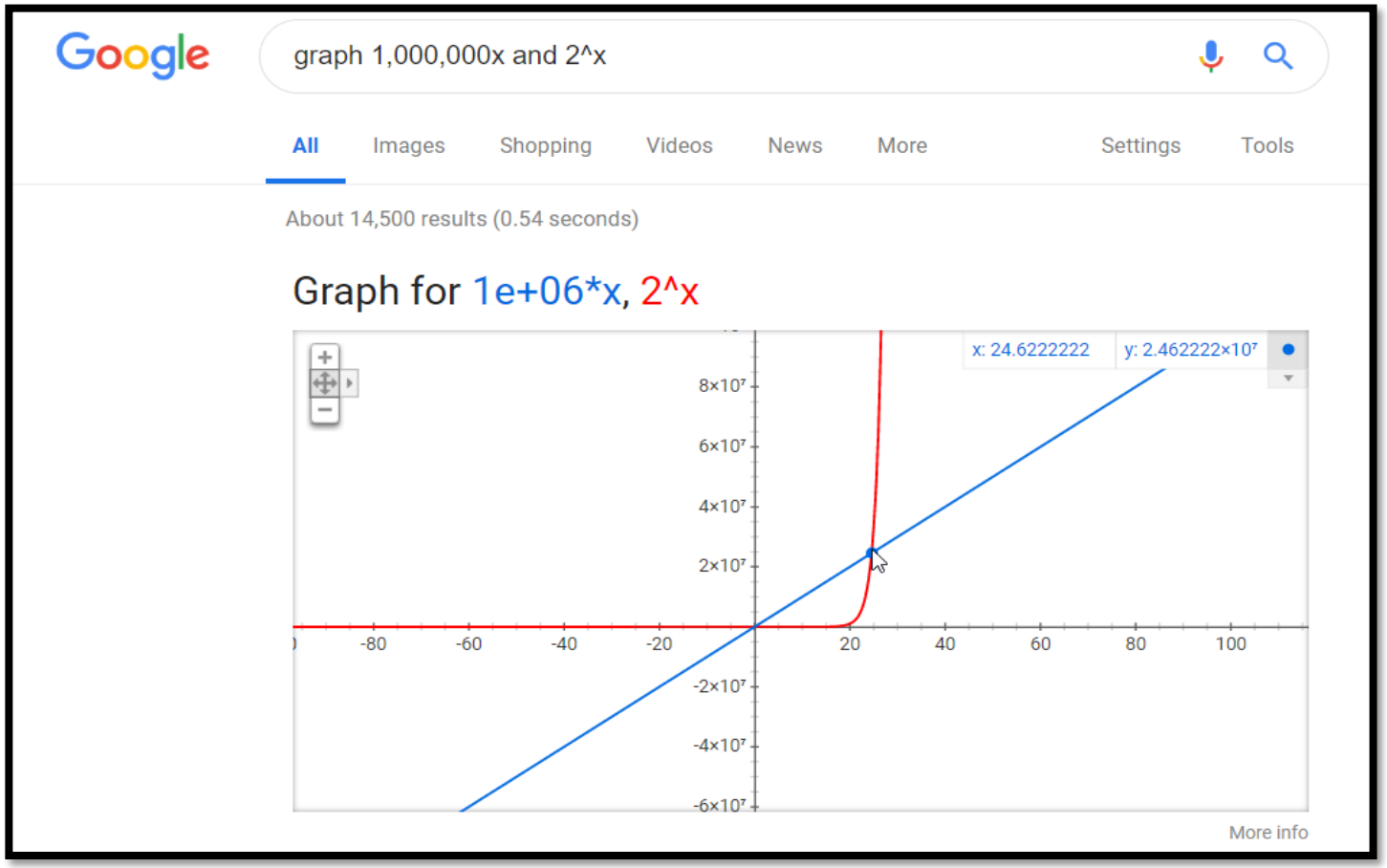
- What is the running time for our algorithm?
  - T(n)=cn
  - we say the running time is O(n) (read it big oh n)
  - Or, we can say that the time complexity for this algorithm is O(n)

ASYMPTOTIC ANALYSIS – EXAMPLE(2)

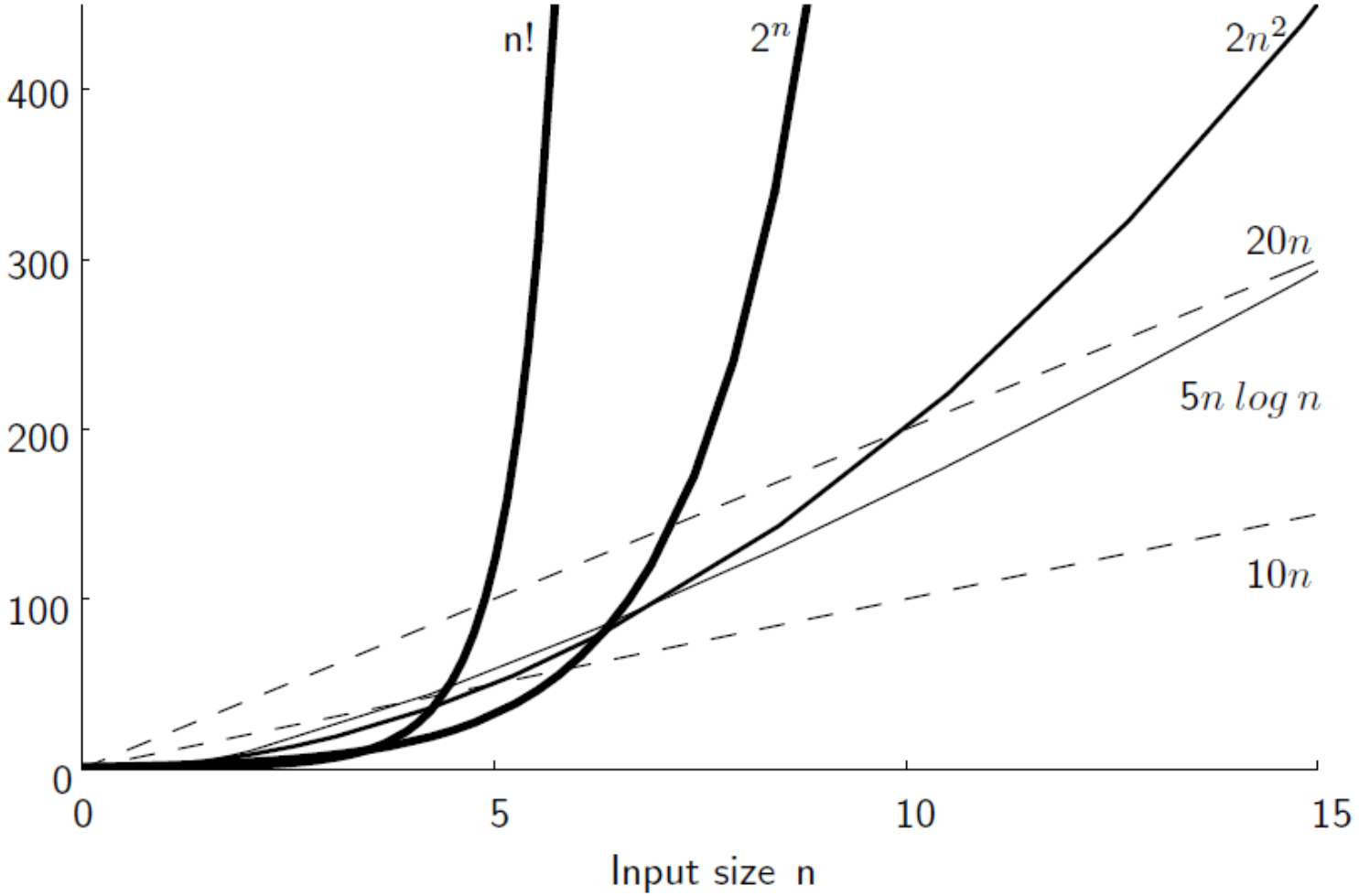
- Here is another simple example: given a sorted array, from smallest to largest
- What would be its running time (of finding the largest value)?
- Does the size of the array affect the running time?
- Congratulations, you just performed Asymptotic Analysis on this algorithm
- We'll see many more examples in this course. You should master this technique, most employers love testing you on this and it is a common language used among developers.

WHY WE USUALLY DON'T COUNT THE CONSTANT

- To simplify our notation and computation we don't count the constants. For large values of n, the constant is really irrelevant. We just want to know the "class" of the functions.
- Compare for example the following functions:
  - f(x)=1,000,000x which is a O(n) function ←note: we prefer using O(n) instead of O(x)
  - g(x)=2^x, which is a O(2^n) function



GROWTH RATES



The horizontal axis represents input size.

The vertical axis can represent time, space, or any other measure of cost.

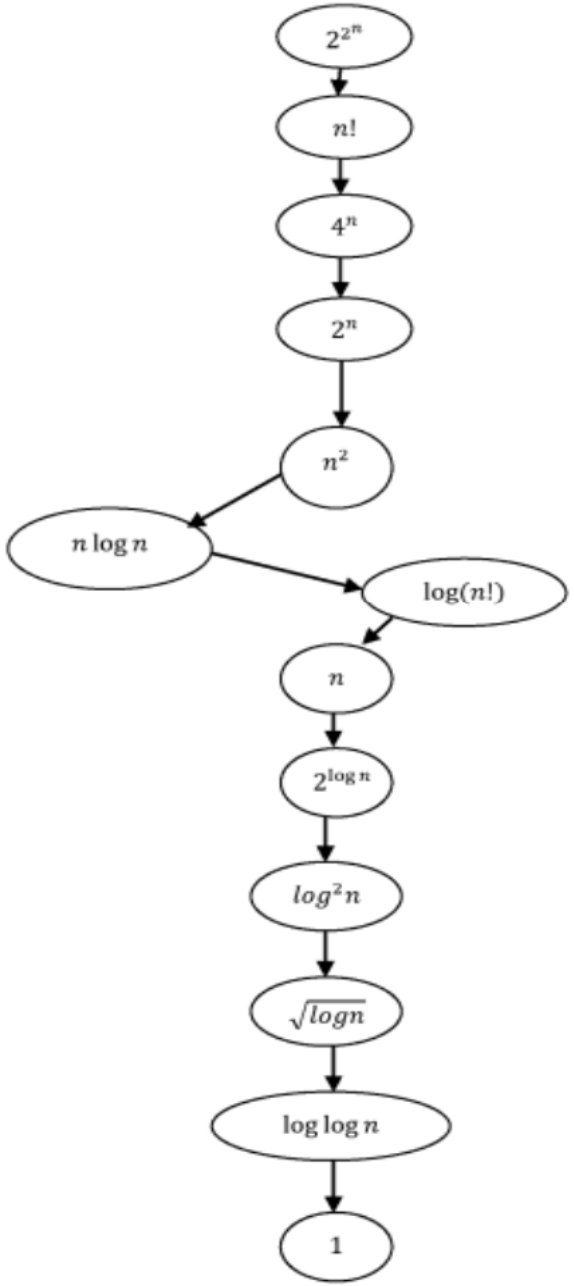
COSTS FOR GROWTH RATES - SKIP

n	log log n	log n	n	n log n	n^2	n^3	2^n
16	2	4	2^4	$4 \cdot 2^4 = 2^6$	2^8	2^{12}	2^{16}
256	3	8	2^8	$8 \cdot 2^8 = 2^{11}$	2^{16}	2^{24}	2^{256}
1024	≈ 3.3	10	2^{10}	$10 \cdot 2^{10} \approx 2^{13}$	2^{20}	2^{30}	2^{1024}
64K	4	16	2^{16}	$16 \cdot 2^{16} = 2^{20}$	2^{32}	2^{48}	2^{64K}
1M	≈ 4.3	20	2^{20}	$20 \cdot 2^{20} \approx 2^{24}$	2^{40}	2^{60}	2^{1M}
1G	≈ 4.9	30	2^{30}	$30 \cdot 2^{30} \approx 2^{35}$	2^{60}	2^{90}	2^{1G}

log could means log base 10, or log base e.

For us, it means log base 2.

2^30=1073741824
2^35=34359738368
2^60=1152921504606846976
2^90=1237940039285380274899124224
2^1G=...



Decreasing Rates Of Growth

Time Complexity	Name	Example
1	Constant	Adding an element to the front of a linked list
$logn$	Logarithmic	Finding an element in a sorted array
n	Linear	Finding an element in an unsorted array
$nlogn$	Linear Logarithmic	Sorting n items by 'divide-and-conquer' - Mergesort
n^2	Quadratic	Shortest path between two nodes in a graph
n^3	Cubic	Matrix Multiplication
2^n	Exponential	The Towers of Hanoi problem

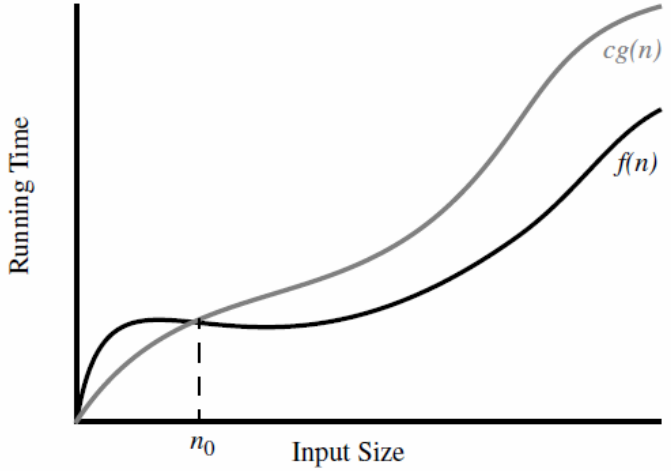
- The growth rate for an algorithm is the rate at which the cost of the algorithm grows as the size of its input grows. The most commonly used ones in this course are:
  - $O(1)$ - Constant: it takes the same amount of time, regardless of the input.
    Even doubling the value of n, it takes roughly the same time to run.
    - Examples: 10, 1024, $\log(2018)$, etc.
  - $O(\log n)$ - Logarithmic: An input a thousand times larger roughly takes just about ten extra steps longer to run.
    - Examples: $log(x)$, $123 \cdot log(x) + 1022$, etc.
  - $O(n)$ - Linear: as the value of n grows, the running time of the algorithm grows in the same proportion. An input a thousand times larger roughly takes about a thousand times longer to run.
    - Examples: $20x$, $5x, 10x + 1022$, $10x + 1024\log(x)$, etc.
  - $O(n \log n)$
  - $O(n^2)$ - Quadratic: An input a thousand times larger roughly takes about a million times longer to run.
    - Examples: $2x^2$, $5x^2 + 10x + 1000$, $10x^2 + 1022x\log(x) + 12$, etc.
  - $O(2^n)$ - Exponential: An input a thousand times larger roughly takes about a ??? times longer to run.
    An input ten times larger roughly takes about a thousand times longer to run.
    - Examples: 2^x, $2^x + x^{1,000,000}$, $10 \cdot 2^x + x^3 + x\log(x)$, etc.

BEST, WORST, AND AVERAGE CASES – SKIP

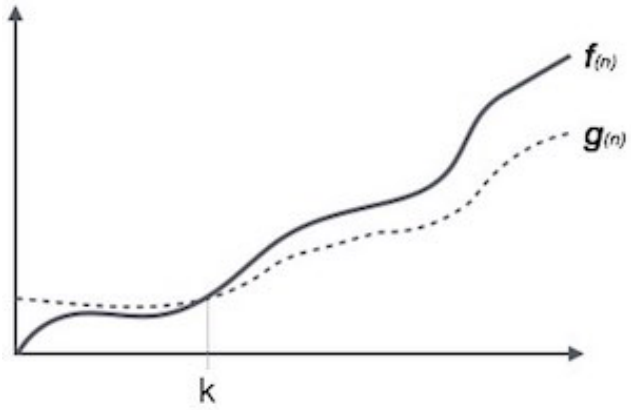
- Following are commonly used asymptotic notations used in calculating running time complexity of an algorithm.
  - **O Notation (big oh)** – for the worst case (used for **upper bound**) Takes at most …≤
    - For example, what would be the running time for searching for a value in a list?
  - **Ω Notation (big omega)** – for the best case (used for **lower bound**) Takes at least … ≥
    - For example, what would be the running time for searching for a value in a list?
  - **Θ Notation (big theta)** – the worst case ≈ the best case Takes exactly …=
    - For example, what would be the running time for displaying n values?
  - We will sometimes consider the "average" case. Especially when comparing two algorithms that have the same worst case running time.
- For a given input size n we often express the time T to run the algorithm as a function of n, written as f(n).
- We will always assume f(n) is a non-negative value.

THE MATH … – SKIP Ω

- Let f (n) and g(n) be functions mapping positive integers to positive real numbers. The function f(n) is O(g(n)) if there is a real constant c > 0 and an integer constant n0 ≥1 such that f (n) ≤ cg(n), for n ≥n0.
  - In particular, prove that $2014n^3 + 23n + 77$ is $O(n^3)$.
  - In particular, prove that $2014n^3 + 23n + 77$ is $O(n^4)$.



- Let f (n) and g(n) be functions mapping positive integers to positive real numbers. The function f(n) is Ω(g(n)) if there is a real constant c > 0 and an integer constant n0 ≥1 such that f (n) ≥ cg(n), for n ≥n0.
  - In particular, prove that $2014n^3 + 23n + 77$ is $\Omega(n^3)$.
  - In particular, prove that $2014n^3 + 23n + 77$ is $\Omega(n^2)$.

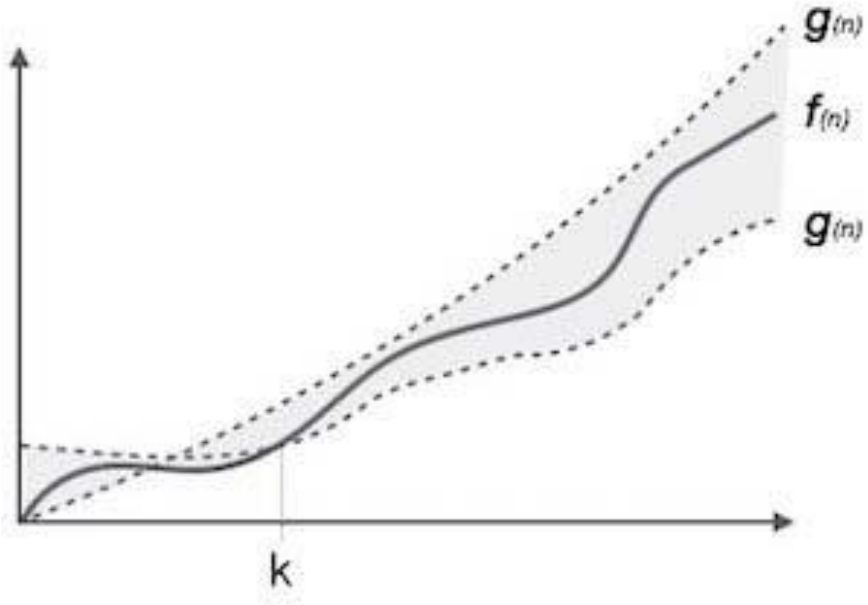


- In this course we'll mostly skip the Ω (big Omega). Not on the test.

THE MATH … VERY BRIEFLY …

- Let f (n) and g(n) be functions mapping positive integers to positive real numbers.
- The function f(n) is Θ(g(n)) if f(n) is O(g(n)) and f(n) is Ω(g(n)), that is there are two real constants $c_1 > 0$ and $c_2 > 0$ and integer constants $n_1 \geq 1$ and $n_2 \geq 1$ such that $f(n) \leq c_1 g(n)$, for $n \geq n_1$ and $f(n) \geq c_2 g(n)$, for $n \geq n_2$.

  - In particular, prove that $2014n^3 + 23n + 77$ is $\Theta(n^3)$.



QUICK REVIEW

- constant multipliers do not affect Big-Oh
  - $10000n^2$ and $0.00005n^2$ are both $O(n^2)$

- lower order terms do not affect Big-Oh
  - $2^n + n^{1000}$ is still $O(2^n)$ ←**we say here that 2^n is the dominant term**

- General Ordering: $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(c^n)$

Source: https://sites.fas.harvard.edu/~cscie119/sections/section3_solutions.pdf

QUICK REVIEW (2)

- Tips on Analyzing the Time Complexity of Iterative Algorithms
  - If you have nested loops, and the outer loop iterates i times and the inner loop iterates j times, the statements inside the inner loop will be executed a total of i * j times.
    This is because the inner loop will iterate j times for EACH of the i iterations of the outer loop.
  - This means that if both the outer and inner loop are dependent on the problem size n, the statements in the inner loop will be executed O(n^2) times:

```
for ( int i = 0; i < n; ++i ) {
    for ( int j = 0; j < n; ++j ) {
        // these statements are executed O(n^2) times
    }
}
```

Source: https://sites.fas.harvard.edu/~cscie119/sections/section3_solutions.pdf

QUICK REVIEW (3)

- Tips on Analyzing the Time Complexity of Iterative Algorithms
  - Likewise, if you have triply-nested loops, all of which are dependent on the problem size n, the statements in the innermost loop will be executed O(n^3) times:

```
for ( int i = 0; i < n; ++i ) {
    for ( int j = 0; j < n; ++j ) {
        for ( int k = 0; k < n; ++k ) {
            // these statements are executed O(n^3) times
        }
    }
}
```

Source: https://sites.fas.harvard.edu/~cscie119/sections/section3_solutions.pdf

QUICK REVIEW (4)

- Tips on Analyzing the Time Complexity of Iterative Algorithms
  - However, imagine a case with doubly-nested loops where only the outer loop is dependent on the problem size n, and the inner loop always executes a constant number of times, say 3 times:

```
for ( int i = 0; i < n; ++i ) {
    for ( int j = 0; j < 3; ++j ) {
        // these statements are executed O(n) times
    }
}
```

  - In this particular case, the inner loop will execute exactly 3 times for each of the n iterations of the outer loop, and so the total number of times the statements in the innermost loop will be executed is 3n or O(n) times, NOT O(n^2) times.

Source: https://sites.fas.harvard.edu/~cscie119/sections/section3_solutions.pdf

QUICK REVIEW (5)

- Tips on Analyzing the Time Complexity of Iterative Algorithms
  - Now, imagine a third case: you have doubly nested loops, and the outer loop is dependent on the problem size n, but the inner loop is dependent on the current value of the index variable of the outer loop:

```
for ( int i = 0; i < n; ++i ) {
    for ( int j = 0; j < i; ++j ) {
        // these statements are executed O(n^2) times
    }
}
```

  - the total number of times the statements in the inner loop will be executed will be equal to the sum of the integers from 1 to n - 1, which is ((n - 1)*n)/2 = n^2/2 - n/2 = O(n^2) times

Source: https://sites.fas.harvard.edu/~cscie119/sections/section3_solutions.pdf

MORE EXAMPLES ...

- **Example**: We begin with an analysis of a simple assignment to an integer variable.
```
a = b;
```
  - Find the running times in terms of big-oh~~, big-omega, big-theta~~

- **Example**: Consider a simple for loop.
```
sum = 0;
for (i=1; i<=n; i++)
        sum += n;
```
  - Find the running times in terms of big-oh~~, big-omega, big-theta~~

- **Example**: Analyze a code fragment with several for loops, some of which are nested.
```
sum = 0;
for (i=1; i<=n; i++) // First for loop
        for (j=1; j<=i; j++) // is a double loop
                sum++;
for (k=0; k<n; k++) // Second for loop
        A[k] = k;
```

SPACE BOUNDS

- Example: What are the space requirements for an array of n integers?
- If each integer requires c bytes, then the array requires cn bytes, which is $\Theta(n)$.

SPACE/TIME TRADEOFF PRINCIPLE

- One important aspect of algorithm design is referred to as the **space/time tradeoff principle.** The space/time tradeoff principle says that one can often achieve a reduction in time if one is willing to sacrifice space or vice versa.

- Many programs can be modified to **reduce storage** requirements by **"packing" or encoding information**. "Unpacking" or decoding the information requires additional time. Thus, the resulting program uses less space but runs slower.

- Conversely, many programs can be modified to pre-store results or reorganize information to allow faster running time at the expense of greater storage requirements.

RECURSIVE FUNCTIONS ... REMINDER

Recursive Function

- A function which calls itself

```
int factorial ( int n ) {
   if ( n == 0)      // base case
      return 1;
   else              // general/ recursive case
      return n * factorial ( n - 1 );
}
```

Note: although I will appreciate if you can find running times for recursive functions, these are not mandatory for this course. Finding their running time is not as straight forward as the previous examples.

EXAMPLE – SKIP

- **What is the time, space complexity of following code?**

```
7   namespace Strings
8   {
9       class Program
10      {
11          static void Main(string[] args)
12          {
13              string str = "Saint Martin's";
14
15              //version 1
16              for (int i = 0; i < str.Length; i++)
17                  if(i%2==1)
18                      Console.WriteLine(str[i]);
19
20              //version 2
21              for (int i = 1; i < str.Length; i+=2)
22                      Console.WriteLine(str[i]);
23          }
24      }
25  }
```

EXERCISES

- Arrange the following expressions by growth rate from slowest to fastest.
  - $4n^2,\ \log_3 n,\ n!,\ 3^n,\ 20n,\ 2, n^{2/3}$

- Do exercise 3.11/page 87: which one is the dominant term?
  ~~For each of the following pairs of functions, either f(n) is in O(g(n)), f(n) is in Ω(g(n)), or f(n) = Θ(g(n)). For each pair, determine which relationship is correct.~~

(a) $f(n) = \log n^2;\quad g(n) = \log n + 5.$
(b) $f(n) = \sqrt{n};\quad g(n) = \log n^2.$
(c) $f(n) = \log^2 n;\quad g(n) = \log n.$
(d) $f(n) = n;\quad g(n) = log^2 n.$
(e) $f(n) = n \log n + n;\quad g(n) = \log n.$
(f) $f(n) = \log n^2;\quad g(n) = (\log n)^2.$
(g) $f(n) = 10;\quad g(n) = \log 10.$
(h) $f(n) = 2^n;\quad g(n) = 10n^2.$

EXERCISES(2)

- Determine running time for the following code fragments:

(a)
```
a = b + c;
d = a + e;
```

(b)
```
sum = 0;
for (i=0; i<3; i++)
   for (j=0; j<n; j++)
      sum++;
```

(c)
```
sum=0;
for (i=0; i<n*n; i++)
   sum++;
```

(d)
```
for (i=0; i < n-1; i++)
  for (j=i+1; j < n; j++) {
    tmp = A[i][j];
    A[i][j] = A[j][i];
    A[j][i] = tmp;
  }
```

(e)
```
sum = 0;
for (i=1; i<=n; i++)
  for (j=1; j<=n; j*=2)
    sum++;
```

EXERCISES(3) – IF TIME

- **What is the time, space complexity of following code:**

```
int a = 0, b = 0;
for (i = 0; i < N; i++) {
    a = a + rand();
}
for (j = 0; j < M; j++) {
    b = b + rand();
}
```

```
int a = 0;
for (i = 0; i < N; i++) {
    for (j = N; j > i; j--) {
        a = a + i + j;
    }
}
```

```
int a = 0, i = N;
while (i > 0) {
    a += i;
    i /= 2;
}
```

Source: https://www.geeksforgeeks.org/practice-questions-time-complexity-analysis/

EXERCISES(4) – IF TIME

- For each of the following pairs of functions $T_1(n)$ and $T_2(n)$ find the dominant term ~~clearly answer the following questions: Is $T_1(n) = O(T_2(n))$?, Is $T_1(n) = \Omega(T_2(n))$?, Is $T_1(n) = \Theta(T_2(n))$?~~

  - $T_1(n) = 6n^2$ $T_2(n) = n^2 \log n$
  - $T_1(n) = 3/2\ n^2 + 7n - 4$ $T_2(n) = 8n^2$
  - $T_1(n) = n^4$ $T_2(n) = n^3 \log n$

Source: https://www.cse.wustl.edu/~sg/CS241_FL99/hw1-practice.html

INTERVIEW PROBLEMS (JUST FOR PRACTICE!)

- Write a function that returns the nth prime number. For example, nthPrime(1) should return 2 since 2 is the first prime number, nthPrime(2) should return 3, nthPrime(3) should return 5, and so on. [Glassdoor.com: Microsoft interview questions]

- Reverse a String ~~in any language you would like~~. Reverse the words in a string. [Glassdoor.com: Microsoft interview questions]

- Given a sentence, write an algorithm to reverse each of the words of the sentence while preserving the order of the words. [Glassdoor.com: Microsoft interview questions]

- Source: https://www.glassdoor.com/Interview/Microsoft-Software-Developer-Interview-Questions-EI_IE1651.0,9_KO10,28_IP9.htm

INTERVIEW QUESTIONS (JUST FOR PRACTICE)

- What is a data structure?
- Differentiate between file and structure storage structure.

- Source:
  - https://career.guru99.com/top-50-data-structure-interview-questions/
  - https://www.geeksforgeeks.org/commonly-asked-data-structure-interview-questions-set-1/

HOMEWORK FOR MODULE 1 (PAGE 1)

DUE: see moodle

This part is worth 60% of homework 1 assignment

Write separate C# programs for each problem below

- **Problem 1(20 points):**
  Ask the user to enter a string (e.g. "Welcome to Saint Martin's U!").
  Your program should count and display the total number of vowels (A, E, I, O, U) – count both uppercase and lowercase vowels – in that given input. In the example above, the output should 9.

- **Problem 2(20 points):**
  Ask the user to enter a positive integer **b**. Validate the input (throw an exception if the input is not positive).
  Then output whether or not the number **b** is divisible by 3.

- **Problem 3(20 points):** given a file **input.txt** count the total number of vowels (A, E, I, O, U) in it.
  - **Create your own input.txt testing file**

- **If your code does not compile, crashes at start, or contains no meaningful comments, it will automatically be graded with 0!**

HOMEWORK FOR MODULE 1 (PAGE 2)

DUE: see moodle, 11:59 PM

This part is worth 40% of the homework 1 assignment

Assume that each of the expressions below gives the processing time $T(n)$ spent by an algorithm for solving a problem of size n. Select the dominant term(s) having the steepest increase in n and specify the lowest Big-Oh complexity of each algorithm.

Expression	Dominant term(s)	$O(\ldots)$
$5 + 0.001n^3 + 0.025n$		
$500n + 100n^{1.5} + 50n \log_{10} n$		
$0.3n + 5n^{1.5} + 2.5 \cdot n^{1.75}$		
$n^2 \log_2 n + n(\log_2 n)^2$		
$n \log_3 n + n \log_2 n$		
$3 \log_8 n + \log_2 \log_2 \log_2 n$		
$100n + 0.01n^2$		
$0.01n + 100n^2$		
$2n + n^{0.5} + 0.5n^{1.25}$		
$0.01n \log_2 n + n(\log_2 n)^2$		
$100n \log_3 n + n^3 + 100n$		
$0.003 \log_4 n + \log_2 \log_2 n$		