# CSC212: Data Structures and Abstractions

Instructor: Krishna Venkatasubramanian

# About Me

- Joined URI: Fall 2019
- PhD: Arizona State U
- Most recently at: Worcester Polytechnic Institute
- Research interests:
  - Cybersecurity
  - Human-computer interaction
  - Accessibility  for folks with physical/intellectual disabilities
  - Tech for marginalized population
  - Wearable technologies
- Pronouns: He/Him/They

# Course Meeting times

- Class Sessions
    - Tuesday & Thursday, 11:00 AM–12:15 PM
  - Location: White Hall 205

- Lab Sessions
    - Lab 1: Wednesday, 12PM- 1:45PM
    - Lab 2: Friday, 10AM- 11:45AM
  - Location: Library 166

# Prerequisites

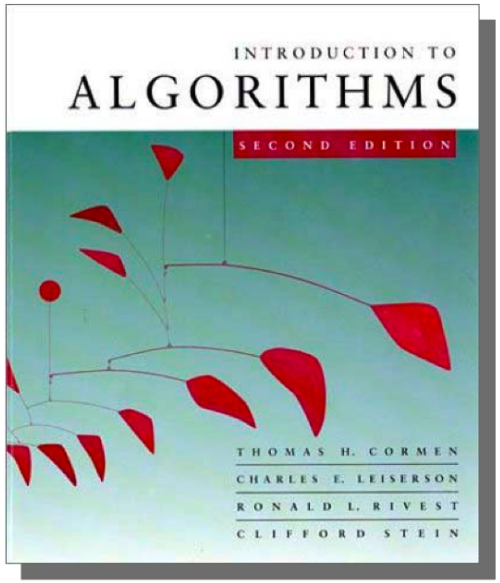
- C-or better CSC 211, MTH 180
- Intended for computer science and computer engineering majors.
- *Knowledge of Python beneficial (unofficial)*

# Logistics

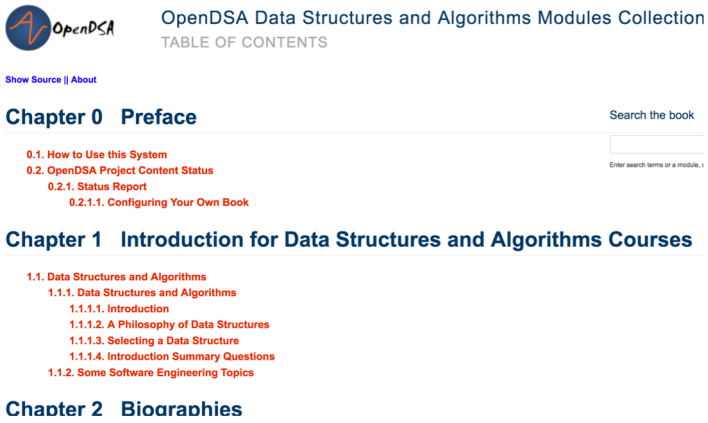
- Course web site
  - https://calhobbes.github.io/csc212-f19/
- Contacts
  - **krish@uri.edu**
  - **Tyler 131**
- Teaching assistants
  - Tom Howard
  - Olivia Coffey
  - Mary Wishart
- Office Hours
  - Check course webpage

# Textbook

- Selected chapters and sections "Introduction to Algorithms, 3rd Edition"!
  - **Older versions of the book are absolutely fine** to have.




- OpenDSA online open textbook
  - Link on course webpage!



# Learning Objectives

- This course aims to teach you the one of the **most important elements** of Computer Science

- Upon successful completion of this course, each student will be able to:
  - **choose appropriate data structures and algorithms** to solve a problem;
  - **compare different algorithms and data structures** based on efficiency, using empirical and theoretical algorithm analysis techniques;
  - **implement solutions** using **recursive functions**;
  - **implement** basic algorithmic tasks for **sorting and searching and analyzing them**
  - **implement** and use **basic data structures**, including linked lists, stacks, queues, priority queues, hash tables, trees, and graphs;

# About this course

- This can be a **challenging** course
- But it can also be very **rewarding** course
- It **opens you up to do bigger and fancier things** in Computer Science
- **Teaches you, in some sense, what we use computers for**
  - You **will use topics from this course** in some form every time you program something even a little complicated
- **Side note:** Job interviews of top firms --- often don't care about your programming language knowledge as much as the concepts you will learn in this class.

# Elements of the Course

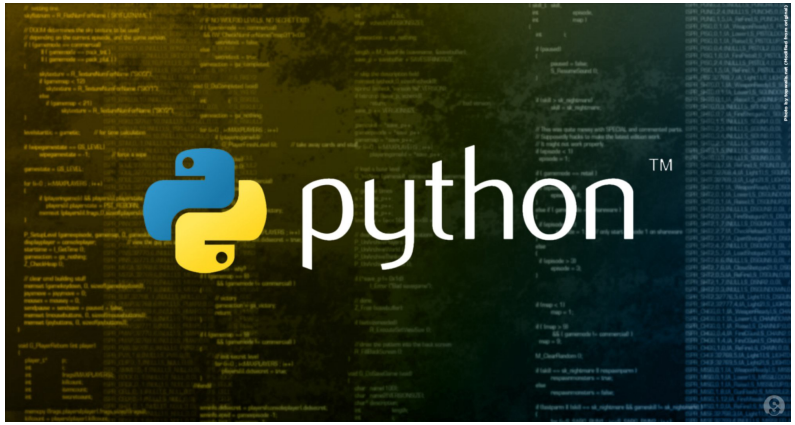
- **Lectures**
  - Cover data structures and algorithm concepts in the class
- **Programming Assignments**
  - Write programs to exercise various concepts we learned in class
  - All assignments are individual assignments
  - *Mandatory for passing this course*
- **Lab Sessions**
  - Helps you improve your Python programming capabilities
  - Hands-on experience with concepts covered in class
  - Practical help from TA staff.
- **Quizzes**
  - One quiz about every other week (**7 in total) --- 2 lowest quizzes will be dropped**
  - Improves theoretical understanding of the subject
  - *Mandatory for passing this course*
- **Final Exam**
  - Basically a longer quiz at the end of semester
  - *Mandatory for passing this course*

**NO WRITTEN HOMEWORKS**

FINAL EXAM: Closed book, one 8½-by-11 sheet of prepared notes, no internet-connected calculators or other electronics

# Python

- **We will use Python** for labs and assignments in the class
- **We use it because the syntax is simpler and closer to pseudocode** that we shall often see in the class and in the text-book
- **It's easy to learn**, if you know other programming languages
- **Python will NOT be taught in this class.**
- **I have provided a bunch of self-study links on course webpage.**



# Python Workshop @ URI Library

- Instructor: Prof. Indrani Mandal
- Venue: Library 130
- Time: Fridays 2pm-4pm **(Sept 6-Sept 27)**
- How to sign-up: Walk-in for the first week
  - Limited seats

- Materials covered:
  - Python intro (Sept 6)
  - Strings, lists, dictionaries, loops & functions (Sept 13)
  - File operations & Exception Handling (Sept 27)
  - Classes and objects (Sept 27)

# Ground Rules for the Course

# Ground Rule #1: You are Grownups!

- **You will be treated like adults.**

- **There will be no handholding** on the part of the teaching staff.

- **That means,** you are responsible for:
  - keeping up the with the reading
  - doing your assignments, labs etc.
  - asking questions and seeking help

# Ground Rules # 2: Self Advocacy

- **This is a very large course!**
  - >100 students
  - It's too difficult for the teaching staff to pay attention to everyone, unless you bring up an issue

- **It is up to you to ask questions** about topics you don't understand

- **It is up to you to let me know if you are falling behind** so that we can work on strategies to help

- **If you are seeking help, do so as early as possible!** Coming to me in November or December may be too late!

- **GO TO OFFICE HOURS!!**

# Ground Rule #3: Questions

- **There are no "stupid" questions.**
- It is a waste of your time and the class's time to proceed when you don't understand basic terms and concepts.
- If you don't understand it, someone else probably doesn't it, either.
- **BY ALL MEANS ASK QUESTIONS!!!**

# Ground Rule #3: Air Space

- **I would like this course to be interactive** and would like as many students and possible to participate.
- Please **feel free to speak/ask questions/state opinions** in class.
- **If you have spoken about 2-3 times in the class already, please let others** in the class to **talk** and not jump in.
- **Also I might have to stop you** *if the discussion goes on for too long*, in the interest of time
  - This is not intended to be disrespectful
  - You can always complete your point to me after the class

# Ground Rule #4: Interact

- **Help each other!**
- Even when a project or assignment is specified as ***individual***, ask your friends or classmates about stuff you don't understand.
- It is a waste of your time try to figure out some obscure detail on your own when there are lots of resources around.
- **When you have the answer, *write it in your* coding style.**

# Ground Rule # 5: Practice

- **Practice, Practice, Practice**. There is no other alternative to mastering the topics covered in this class. You have to make mistakes and learn.

- **If you are NOT serious about this course, you will not do well in this class!**

- **Assignments and labs in this course may be time consuming for people who haven't programmed much before**

# Names and Faces

- Ideally I would like to know all of you. Difficult in a such large class. But, try we must
- **So, when speaking in class, please identify yourselves (every time)**
  - Please use your preferred name
- If I use the wrong pronoun to address you:
  - You can correct me then and there
  - Or, you come up to me after class and let me know one-on-one if that is more comfortable

# Grading Policy

- Final grades will be computed as follows:
  - 25% - Programming Assignments
  - 25% - Final Exam
  - 40% - Quizzes
  - 10% - Lab

- Only exceptional work in the class gets an 'A'.
- I **typically** curve the grades (unless this is a class full of geniuses).

- So, just focus on learning
  - Don't worry if your scores are in the 70% range, as that may not mean a C-range final grade.

- **Also, I calculate grades only toward end of the semester**. So if you ask me "what my grade is" in the middle of the semester, I usually don't have an answer for you – unless you have been doing very well or very poorly!

# In Class Behavior

- The classroom is a place for learning. While you are in class:
  - I expect you to remain focused on the course material
  - **Laptops and tablets are for taking notes only**. Please do not use them for surfing the web, watching inappropriate content etc.
  - Please have **cellphones muted** in class
  - **No unnecessary, off-topic chatter**
  - Please **be respectful of everyone**

- If I **find you using laptops/tablets/cellphones** inappropriately
  - **First strike:** an immediate **lowering** of the final grade **by one full letter grade** (e.g., a B+ becomes C+)
  - **Second strike:** you **get an F** in the class, immediately.

# Office Hours Etiquette

- I **encourage you to go to office hours** for this class
- **Office hours are not for TA or ME to solve your assignments/problems,** but to guide you to solve the problem yourself.
- **If you cannot make it to MY office hours, please feel free to email me and we can setup a time to meet**
- **I DO NOT accept walk-ins**. Send me a note to setup an appointment please!

# Disability Accommodations

- Any student with a documented disability is welcome to contact me as early in the semester as possible, so that we may arrange reasonable accommodations.

- As part of this process, please be in touch with Disability Services for Students Office (http://web.uri.edu/disability/aboutdss/).

# Academic Honesty

- Discussions with others to understand general assignment and class-related concepts are strongly encouraged (OUTSIDE THE CLASS, OF COURSE).

- However, **when working on assignments/labs/exams/quizzes, all written work and source code must be your own**.

- Students are prohibited from accessing or comparing assignments/quizzes/exams with those of other students prior to submitting each assignment.

- **Copying another individual solution is plagiarism, a serious offense**, and the one most common in computer science courses.
  - Anyone that provides homework answers, program code for a programming assignment to another individual is also guilty of academic dishonesty.
  - Both will be prosecuted in accordance with the University's Policy of Academic Honesty.
  - **If you do not have sufficient time to complete an assignment, then submit a partial solution.**

**Oh yes, if I catch you cheating, you will get an instant 'F' in the course**

# Late Submissions

- **No late submissions are allowed**.

- Exceptions include extenuating circumstance and religious observance.

- If you are going to miss deadline for an assignment, quiz etc., **and you have valid reason**, **please inform me in writing (i.e., email) ahead of schedule!**

# Piazza

- We will be using the Piazza forum tool for outside-the-class discussions and in-class communication
- A great tool to help each other!
- You can post questions and comments on topics covered in the class on Piazza
- You can comment on other people's questions and comments about topics
- Accounts on Piazza will be created for you very soon. You should be able to login with your URI emails

https://piazza.com/uri/fall2019/csc212/home

# Gradescope

- Gradescope is a online grading tool that we will use for this class
- **Gradescope will be used by you to submit yours lab and programming assignments**
- **Your quizzes will also be graded on Gradescope**
- **All your grades for this class will be visible on Gradescope**
- Accounts on Gradescope will be created for you very soon. You should be able to login with your URI emails

# Emails

- All official communication for you will come through **Piazza**
  - **Except first week (this week), when I will email you all.**
  - **Please check your URI emails!**
- If you email me **please include [CSC212] in the subject line, otherwise, I cannot guarantee reading your email**
- If you email me, **I may take up to 24 hours to respond**. If you email me on Friday, 24 hours is defined as Monday!
- Please don't email me a few hours before assignment deadlines to ask for extension or expect me to respond immediately.

# URI Religious Holidays

- It is the policy of the University of Rhode Island to accord students, on an individual basis, the opportunity to observe their traditional religious holidays.

- Students desiring to observe a holiday of special importance must **provide written notification to each instructor.**

- **Complete list of holidays observed can be found here:** http://www.interfaith-calendar.org/

# Academic Enhancement Center

- Nearly all students recognize that regardless of how well or poorly they are doing in a given class, there are ways to improve their learning and studying.

- The Academic Enhancement Center (AEC) and Writing Center (WC), located in Roosevelt Hall, offers several kinds of support that help students improve their learning and academic performance in this class as well as other classes.
  - For information on any of these programs, visit the AEC website or call the AEC's main number at (401) 874-2367.

- **Subject Specific Tutoring**, located on the fourth floor of Roosevelt Hall, helps students navigate 100 and 200 level math, chemistry, physics, biology, and other select STEM courses (**includes CSC 212**).
  - Information on what these programs offer, when they are available, and how to utilize them are available at AEC tutoring.

# Summary of Things to Remember:

- This *can* be a **time-consuming** class.
  - Concepts are abstract and can take time to understand
  - Practice is essential

- There are **LOT of QUIZZES** in this class.
  - Keeping up with your reading is essential
  - Quizzes are closed everything!

- Focus of the course on **theory + programming**

# Introduction to CSC212

# Problems!

- Problems are basically questions for which we need answers
  - Example: **Can we list people who commented on a person's given Instagram picture?**

- A problem definition should not include any constraints on *how* the problem is to be solved.

- A problem can be thought of as a **function** in the mathematical sense
  - A *function* is a matching between inputs (the *domain*) and outputs (the *range*).
  - What are Input and Output for the Q above?

- Problems can be broad and typically you solve a specific **instance** of it:
  - Example: **Who commented on Jane's selfie from last nights party?**

# Algorithm

- An **algorithm** is a process by which we solve the problem given problem

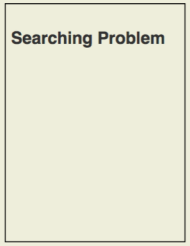
- An algorithm must by definition
  - Given a **correct answer!**
  - It is composed of a series of ***concrete steps*** --- completely understood and doable steps. ***A recipe!***
  - There can be *no ambiguity* as to which step will be performed next.
  - It must be composed of a *finite* number of steps
  - It must *terminate – no infinite loops! (though this is **impossible** to determine in the general case --- check out **halting problem***
    [https://en.wikipedia.org/wiki/Halting_problem])

# Programs

- A program is a **concrete representation of an algorithm**
- Algorithms are usually presented in terms of programs, or parts of programs
- Programs are specified in programming languages (i.e., Python, C/C++ etc.)
- Therefore **algorithms can be written in many different programming languages**

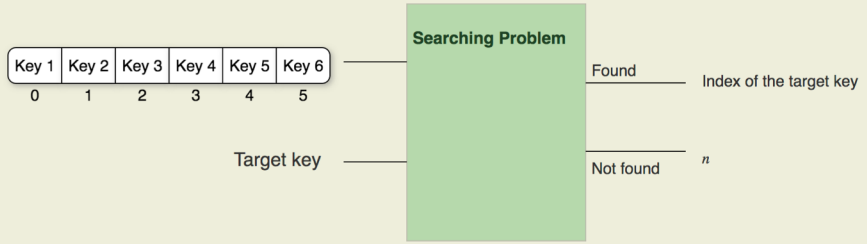
# Example – A Problem

A problem is a task that matches input to output. Consider the problem of searching for an element in an unsorted array.




Here, we have:
**Input:** An array, and the target key.
**Output:** The index of the target element if it is found, or the size of the array if not found.

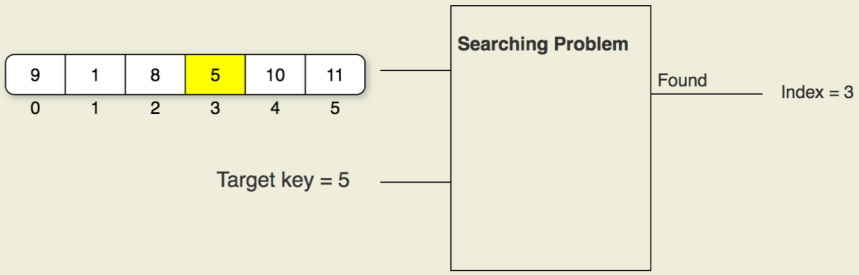
We show the searching problem as a 'black box'. In this context, we don't need to know how a search is actually performed.



# Example – Problem Instance + Algorithm

A problem instance is a specific selection of values for the problem input.

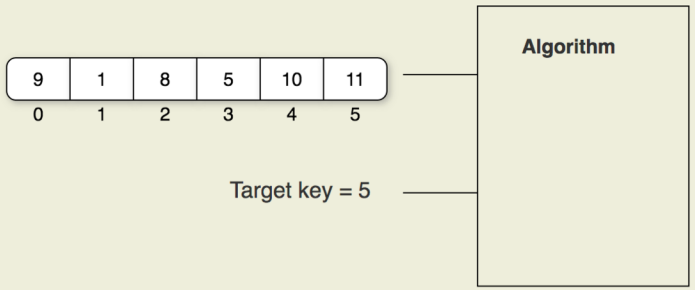
Here we see an example of a searching problem instance in which we have initialized the array and we have a value for the target key.




An algorithm is a recipe or a specific way of mapping problem input to output.

An algorithm takes a problem instance as its input.

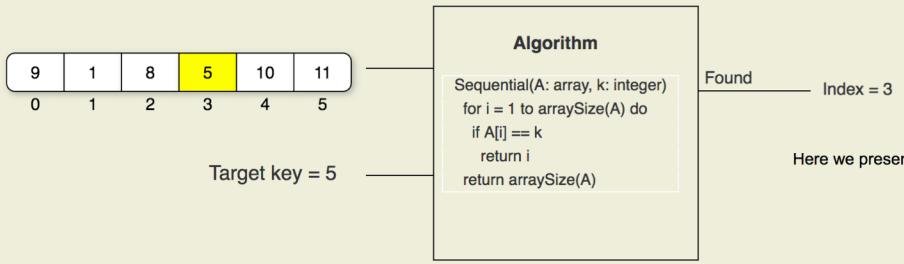
Then a series of steps is performed to generate the output.



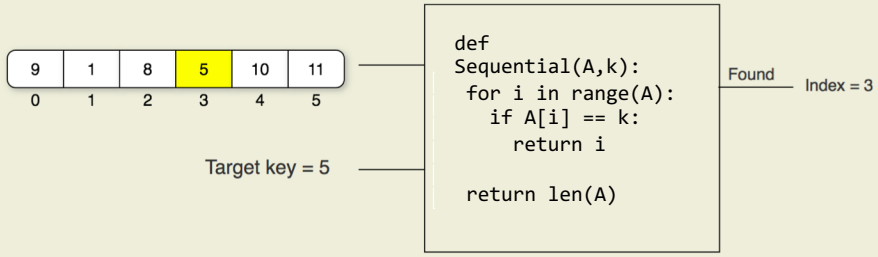
# Example – Algorithm + Program

There are several different algorithms that can solve a particular problem. For the searching problem, here we present pseudocode for the sequential search algorithm.

The sequential search algorithm simply loops through all the keys in the array until the target key is found in (which case the index is returned). If its not there, then the array size is returned.




Here we present the sequential search algorithm implemented as a Java function.



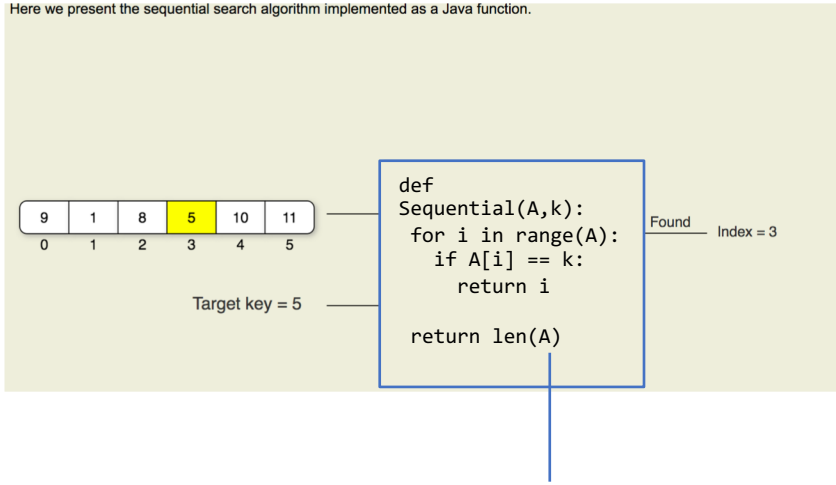
# Data Structures

- **Algorithms use data structures** to store information when needed

- Representing information is fundamental to computer science.

- One of the most important task of computers is to
  - Store information
  - Retrieve information

- Data Structures as the name suggests ***are ways in which information is store and retrieved --- that's it!***
  - These form a programmer's basic "toolkit".




**Array** data structure used here

It's a simple data structure
In this class we shall study several more complex ones

# What You Will Learn!

- **Problem Types/Families**
  - Sorting, Searching, Graph and Tree Algorithms, String Processing
- **Algorithms on Different Data Structures**
  - Arrays, Trees, Queues, Graphs, Strings, Hash Tables, Linked Lists, Stacks
- **Analysis and Evaluation**
  - Coding and running algorithms,
  - Analytical analysis using big-O notation
- **Algorithmic Strategies and Methodologies**
  - Tree and graph traversal, Greedy Algorithms, Divide and Conquer…


That's all Folks!
Any Question ?