Welcome to CSE 373!

CSE 373 Winter 2020

Instructor: Hannah C. Tang

Teaching Assistants:

Aaron Johnston	Ethan Knutson	Nathan Lipiarski
Amanda Park	Farrell Fileas	Sam Long
Anish Velagapudi	Howard Xiao	Yifan Bai
Brian Chan	Jade Watkins	Yuma Tou
Elena Spasova	Lea Quan	

Lecture Outline

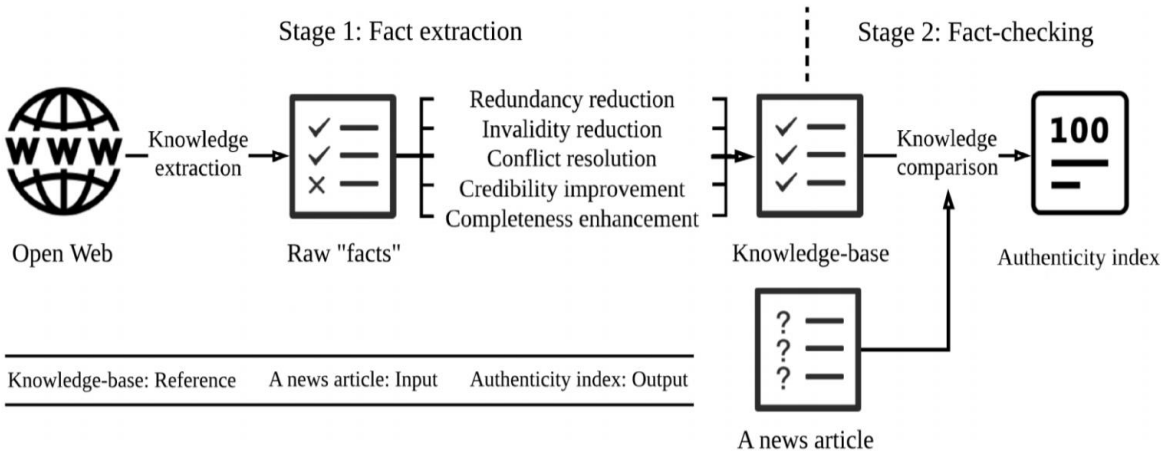
- **Introduction: Why Data Structures and Algorithms?**
- About This Course
  - Projects
  - People
  - Policies
  - Getting the Most out of This Course
- Abstract and Concrete Data Types

Data Structures and Algorithms

- **Data Structures**:
  - A way of organizing, storing, accessing, and updating a set of data
- **Algorithms**:
  - A series of precise instructions guaranteed to produce a certain answer

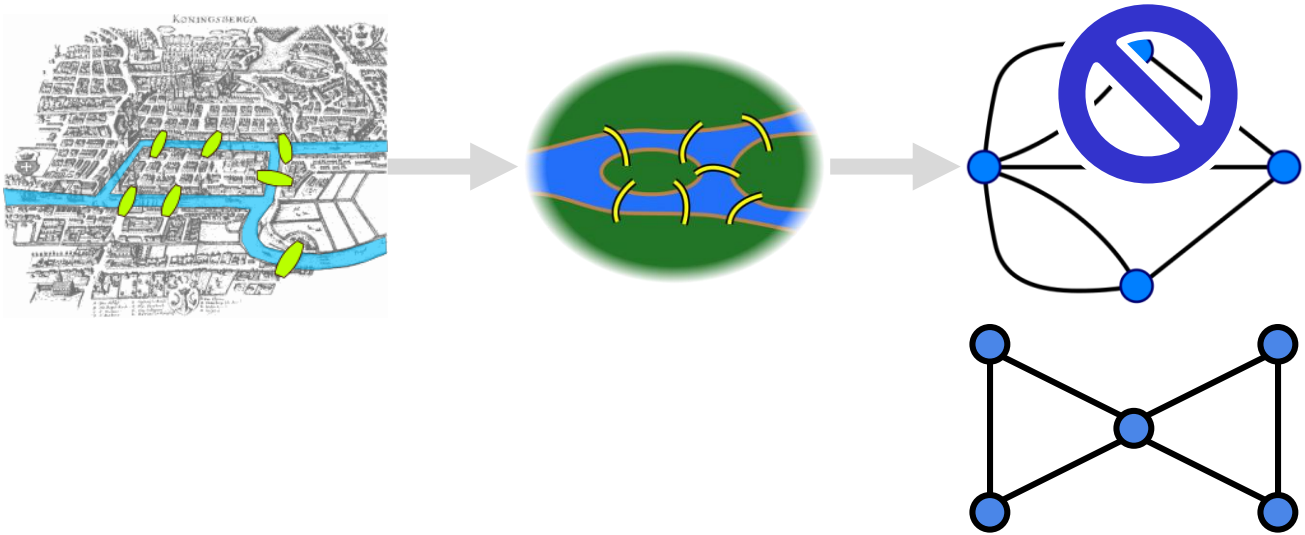
Why: Increase Progress (?) in Society

Why: Discover New Knowledge



Fake News: A Survey of Research, Detection Methods, and Opportunities (Xinyi Zhou, Reza Zafarani/arXiv:1812.00315)

Why: Understand Different Disciplines and Problems



Konigsberg Bridges (Bogdan Giuşcă/Wikimedia), Diagram of Seven Bridges (Chris Martin/Wikimedia), Konigsberg Graph (Riojajar~commonswiki/Wikimedia)

Why: Support Daily Life

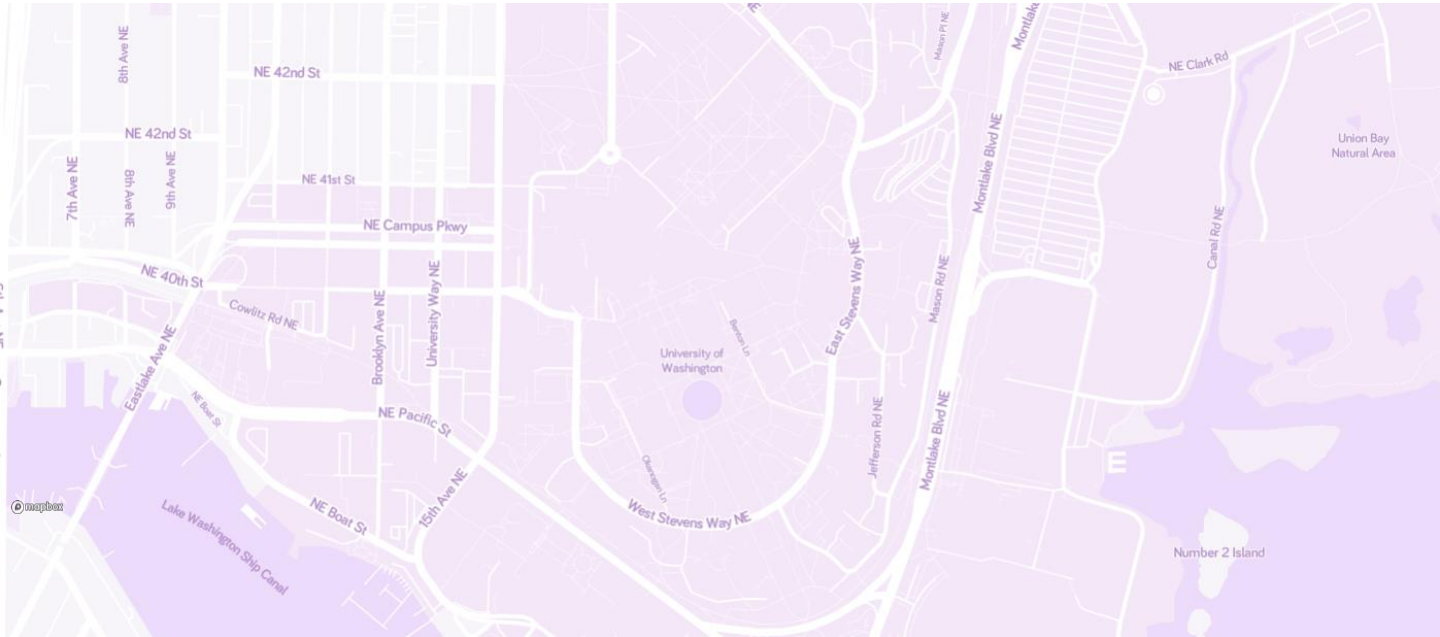
How to search the internet

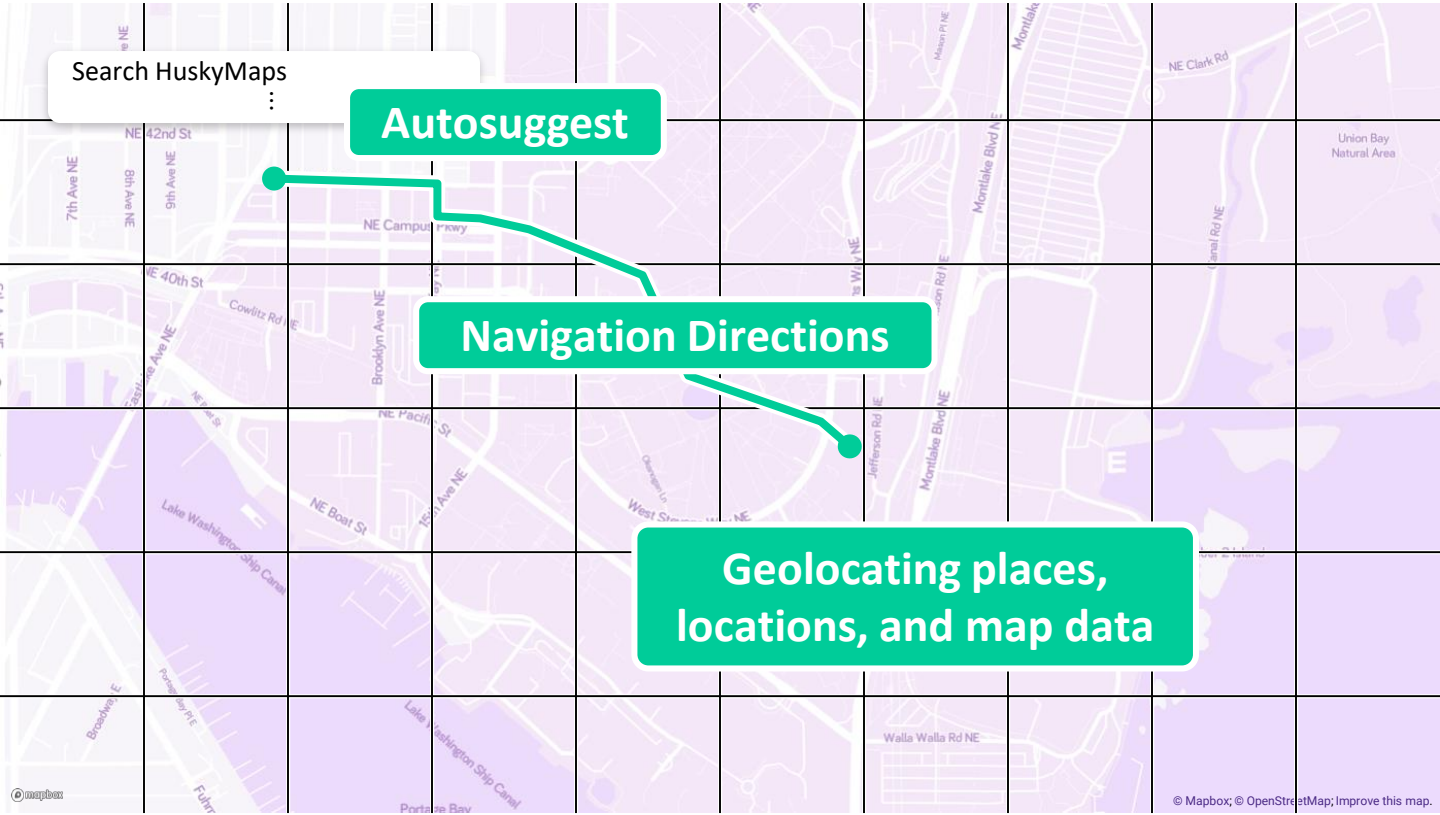
About 7,470,000,000 results (0.60 seconds)

Lecture Outline

- Introduction: Why Data Structures and Algorithms?
- About This Course
  - **Projects**
  - People
  - Policies
  - Getting the Most out of This Course
- Abstract and Concrete Data Types

Support Daily Life




Search HuskyMaps
Autosuggest
Navigation Directions
Geolocating places, locations, and map data
NE 42nd St
7th Ave NE
8th Ave NE
9th Ave NE
NE Campus Pkwy
NE 40th St
Cowlitz Rd NE
Brooklyn Ave NE
NE Pacific St
NE Boat St
Lake Washington Ship Canal
Montlake Blvd NE
Jefferson Rd NE
NE Clark Rd
Union Bay Natural Area
Walla Walla Rd NE
© Mapbox; © OpenStreetMap; Improve this map.

Lecture Outline

- Introduction: Why Data Structures and Algorithms?
- About This Course
  - Projects
  - **People**
  - Policies
  - Getting the Most out of This Course
- Abstract and Concrete Data Types

Introductions: Course Staff

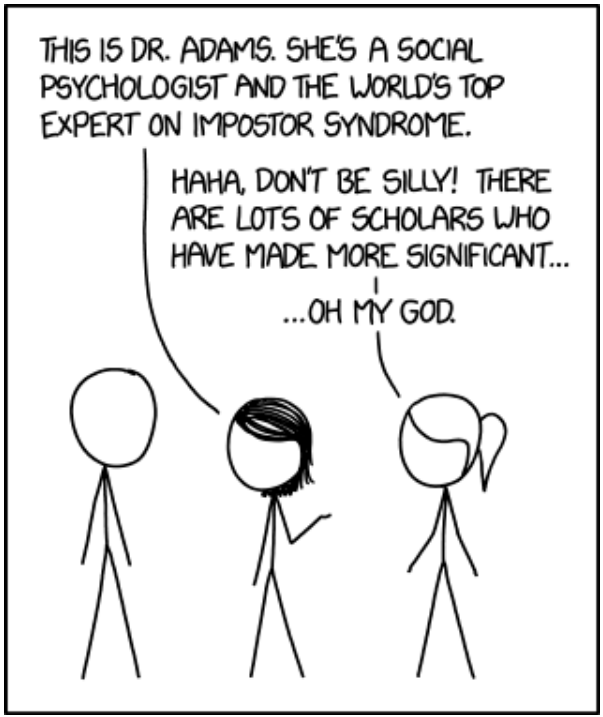
- Hannah C. Tang
  - UW CSE alumna with 17 years ~~of bugs~~ in industry
- TAs:
  - Aaron Johnston, Amanda Park, Anish Velagapudi, Brian Chan, Elena Spasova, Ethan Knutson, Farrell Fileas, Howard Xiao, Jade Watkins, Lea Quan, Nathan Lipiarski, Sam Long, Yifan Bai, Yuma Tou
  - Available in section, drop-in time, and discussion group
  - An invaluable source of information and help
- Get to know us
  - We are excited to help you succeed!

Introductions: Students

- ~230 students registered
  - Many students drop during the first week of the quarter; keep trying!
- In the meantime:
  - Attend lecture, pick a quiz section to attend, do the assignments
  - Email cse373-staff@cs to get added to repos, discussion boards, etc.
- Our course size is an asset!

Students and Imposter Syndrome

- It's easy to feel lost, as if everyone is "better" than you.
- "Nearly 70% of individuals will experience signs and symptoms of impostor phenomenon at least once in their life."
  - https://en.wikipedia.org/wiki/Impostor_syndrome
- Our course size can be an asset!



https://xkcd.com/1954

Lecture Outline

- Introduction: Why Data Structures and Algorithms?
- About This Course
  - Projects
  - People
  - **Policies**
  - Getting the Most out of This Course
- Abstract and Concrete Data Types

Communication

- Website: http://cs.uw.edu/373
  - Schedule, policies, materials, assignments, etc.
- Discussion: http://piazza.com/washington/winter2020/cse373
  - Announcements made here
  - Ask and answer questions – staff will monitor and contribute
- Drop-in Time ("office hours"): spread throughout the week
  - Can e-mail/private Piazza post to make individual appointments
- Anonymous feedback:
  - Comments about anything related to the course where you would feel better not attaching your name

Course Components

- Readings and Lectures
  - Pre-lecture reading is graded on *participation*, not correctness
  - Need more details? "Data Structures and Algorithm Analysis in Java" is available at the Engineering Library for checkout
  - Introduce the concepts; please take notes!!!
- Sections and QuickChecks
  - Pre-section QuickCheck is graded on *participation* and *correctness*
  - Apply the concepts, review materials
- Programming Homeworks
  - Approximately one per week
  - We have a late policy; don't fall behind!
- Exams
  - Midterm and Final; more details later in the quarter

Deadlines and Student Conduct

- Late policies
  - QuickChecks and Reading Quizzes: no late submissions accepted
  - Homeworks: Percentage deducted per day
    - One day late is "cheap"; can't submit after 4 days.

- Academic Conduct (**read** the full policy on the web)
  - In short: don't attempt to gain credit for something you didn't do and don't help others do so either
  - This does ***not*** mean suffer in silence!
    - Learn from the course staff and peers, talk, share ideas; *but* don't share or copy work that is supposed to be yours

Collaboration is Strongly Encouraged

- Discuss confusing points with each other
  - Organizing your thoughts is the best way to learn!
  - Piazza, study groups, the person sitting next to you, …
- Take initiative!
  - Form study groups with your peers in lecture and quiz section.
  - Review questions from previous quarters or other institutions.

Lecture Outline

- Introduction: Why Data Structures and Algorithms?
- About This Course
  - Projects
  - People
  - Policies
  - **Getting the Most out of This Course**
- Abstract and Concrete Data Types

Hooked on Gadgets

- Gadgets reduce focus and learning
  - Bursts of info (*e.g.* emails, IMs, etc.) are *addictive*
  - Heavy multitaskers have more trouble focusing and shutting out irrelevant information
    - http://www.npr.org/2016/04/17/474525392/attention-students-put-your-laptops-away
- Seriously, you will learn more if you use **paper** instead!!!
  - What types of activities do you do while taking notes?

Metacognition

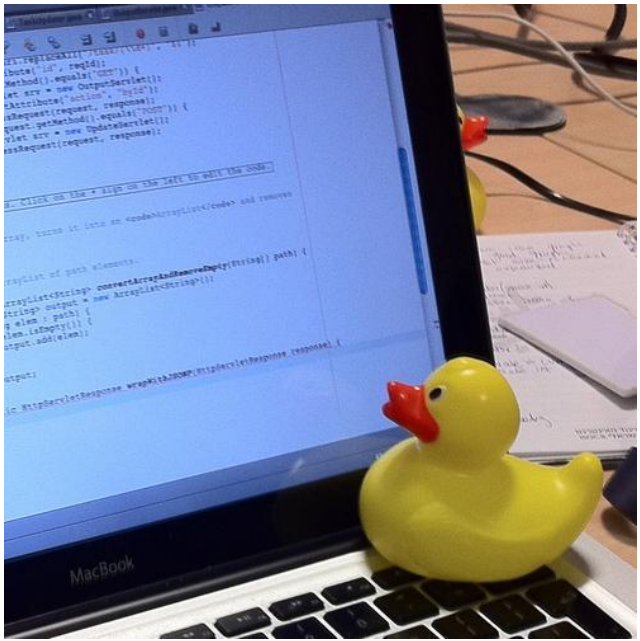
- **Metacognition**: asking questions about your solution process.

- Examples:
  - While debugging: explain to yourself why you're making this change to your program while debugging.
  - Before running your program: make an explicit prediction of what you expect to see.
  - When coding: be aware when you're not making progress, so you can take a break or try a different strategy.
  - When designing:
    - Explain the tradeoffs with using a different data structure or algorithm.
    - If one or more requirements change, how would the solution change as a result?
    - Reflect on how you ruled out alternative ideas along the way to a solution.
  - When studying: what is the relationship of this topic to other ideas in the course?

Real world analogues

Minimal working example

Rubber duck debugging



Rubber duck assisting with debugging (Tom Morris/Wikimedia)

Lecture Outline

- Introduction: Why Data Structures and Algorithms?
- About This Course
  - Projects
  - People
  - Policies
  - Getting the Most out of This Course
- **Abstract and Concrete Data Types**

Data Structures and Algorithms

- *Data Structures:*
  - *A way of organizing, storing, accessing, and updating a set of data*
  - Examples from 14X: arrays, linked lists, stacks, queues, trees

- *Algorithms:*
  - *A series of precise instructions guaranteed to produce a certain answer*
  - Examples from 14X: binary search, merge sort, recursive backtracking

Concrete Data Types

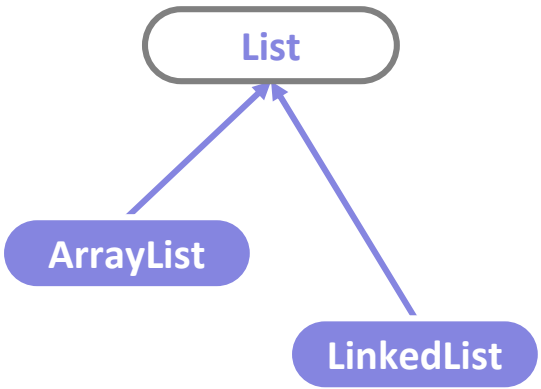
A variable's **data type** (or simply **type**) determines its possible values and operations.

```
int course;
course = 37;
course = -37;

course = 3.14;

(37 + 3) == 40;

course.equals(373);
```

```
String course;
course = "37";
course = "-37";

course = 3.14;

(37 + 3) == 40;

("37" + "4").equals("373");
```

Interfaces vs. Implementations

- In Java, an **interface** is a data type that specifies what to do but not how to do it.
  - **List**: an ordered sequence of elements.
- A **subtype** implements all methods required by the interface.
  - **ArrayList**: Resizable array implementation of the List interface.
  - **LinkedList**: Doubly-linked implementation of the List interface.



Abstract Data Types (ADTs)

- Java interfaces represent the concept of abstract data types.
- An **abstract data type** is a data type that does not specify any one implementation.
- **Data structures** implement ADTs.
  - **Resizable array** can implement List, Stack, Queue, Deque, PQ, etc.
  - **Linked nodes** can implement List, Stack, Queue, Deque, PQ, etc.

List ADT. A collection storing an ordered sequence of elements.

- Each element is accessible by a zero-based index.
- A list has a size defined as the number of elements in the list.
- Elements can be added to the front, back, or any index in the list.
- Optionally, elements can be removed.

Intuitively …

- Think of the ADTs and data structures you'll learn this quarter as a cookbook
  - ADTs are the chapters/category: Soups, Salads, Cookies, Cakes, etc
    - High-level descriptions of a category of functionality
    - You don't serve a soup when guests expect a cookie!
  - Data structures are the recipes: chocolate chip cookies, snickerdoodles, etc
    - Step-by-step, concrete descriptions of an item with specific characteristics
    - Understand your tradeoffs before replacing carrot cake with a wedding cake
- When you go out into the world …
  - Figure out which category is required
  - Choose the specific recipe that best fits the situation