CSE 250: Course Overview, Logistics

Lecture 1

Aug 26, 2023

Who are we?

Hi, I'm Oliver.
okennedy@buffalo.edu

- I study how people interact with data (spreadsheets).
- I make data go vroom (compilers, data structures).
- I've been teaching data structures for 4 years.
- I do western martial arts ⚔️.
- I bike … a lot 🚲.

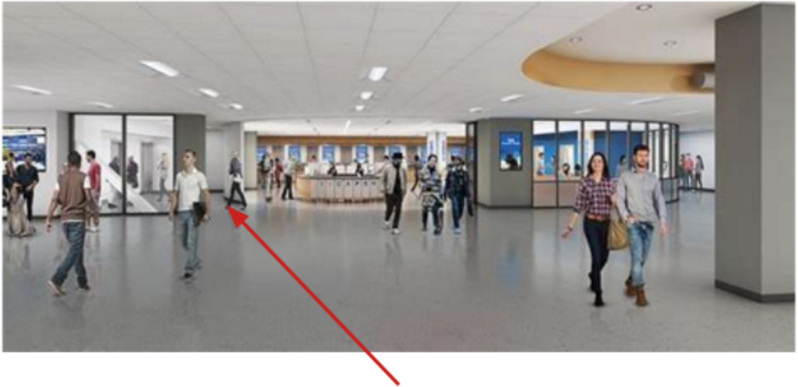
Who are we?

Oliver Kennedy
okennedy@buffalo.edu
Where: Capen 211 (in Capen 212)
Office Hours: Fri 10:00-11:30

Eric Mikida
epmikida@buffalo.edu
Where: Capen 208 (in Capen 212)
Office Hours: Tue 3:30-5 PM;
Wed 11:00-1:00

- Alex Kim
- Alex Terry
- Brendan O'Connell
- Chris Dearing
- Derek Gage
- Doniyor Ismatilloev
- Emilie Griffin
- Eric Xie
- Ethan Phan
- Eugenia Vance
- Evan Jiang
- Faizaan Mohammed Ali
- Isabel Kimos
- Jennifer Furdzik
- Jonathan Guzman
- Jordan Wang
- Joy Lee
- Julia Joseph
- Marian Huynh
- Matthew Bieniak
- Milos Petrovic
- Robby Pruzan
- Ronan Kasmier
- Shreyas Narayanan Sridhar
- Vipassana Khandare
- Vrushaali Nagaraj
- Wonwoo Jeong

Finding Capen 212



212 Capen: Take these elevators, then turn right.

Logistics

- **Course Forums**: Piazza
  https://piazza.com/buffalo/fall2024/cse250
- **Course Website / Syllabus**:
  https://cse.buffalo.edu/courses/cse250/2024-fa
- **Assignment Submission**: Autolab
  https://autolab.cse.buffalo.edu/courses/cse250-f24/
- **Assignment Distribution**: Github Classroom

Logistics

- **Course Forums**: Piazza
  https://piazza.com/buffalo/fall2024/cse250
- **Course Website / Syllabus**:
  https://cse.buffalo.edu/courses/cse250/2024-fa
- **Assignment Submission**: Autolab
  https://autolab.cse.buffalo.edu/courses/cse250-f24/
- **Assignment Distribution**: Github Classroom

Please keep discussions on Piazza (use private posts if necessary)

Logistics

- **Course Forums**: Piazza
  `https://piazza.com/buffalo/fall2024/cse250`
- **Course Website / Syllabus**:
  `https://cse.buffalo.edu/courses/cse250/2024-fa`
- **Assignment Submission**: Autolab
  `https://autolab.cse.buffalo.edu/courses/cse250-f24/`
- **Assignment Distribution**: Github Classroom

Please keep discussions on Piazza (use private posts if necessary)
Always include `[CSE-250]` on the subject line when emailing

Development Environment

Supported Operating Systems

- MacOS
- Ubuntu Linux
- Windows + WSL/Ubuntu

Supported Development Environments

- IntelliJ

Development Environment

Supported Operating Systems

- MacOS
- Ubuntu Linux
- Windows + WSL/Ubuntu

Supported Development Environments

- IntelliJ

Other setups are ok, but the more your setup differs, the less we'll be able to help you.

Syllabus

https://cse.buffalo.edu/courses/cse250/2024-fa

Grading

Grade Breakdown

- Assignments 40% (5%. each)
- Participation 10%
- 2 Midterms 15% each
- Final Exam 20%

Score (x)	Letter Grade	Quality Points
$90\% \leq x \leq 100\%$	A	4
$85\% \leq x < 90\%$	A-	3.67
$80\% \leq x < 85\%$	B+	3.33
$75\% \leq x < 80\%$	B	3
$70\% \leq x < 75\%$	B-	2.67
$65\% \leq x < 70\%$	C+	2.33
$60\% \leq x < 65\%$	C	2
$55\% \leq x < 60\%$	C-	1.67
$50\% \leq x < 55\%$	D	1
$0\% \leq x < 50\%$	F	0

Written Assignments

- **∼Bi-Weekly Written Assignments**
  Expect to spend about a week working on it
  Submissions allowed up to a day late (50% penalty)
- **You are responsible for submission format**
  Submit **only** PDFs
  Submissions that can't be read will receive a 0
- **We recommend writing solutions by hand**
  Handwritten work is retained more effectively
  It's easier to write out math by hand

Programming Assignments

Typical Programming assignment

- Write Test Cases (5/30 points)
  Due before implementation
- Implementation Correctness (20/30 points)
- Implementation Efficiency (5/30 points)

Programming Assignments

Typical Programming assignment

- Write Test Cases (5/30 points)
  Due before implementation
- Implementation Correctness (20/30 points)
- Implementation Efficiency (5/30 points)

Your grade is based on the <u>last</u> submission you make.

Assignments

You have 2-3 weeks per assignment

- Plan to start early and work throughout
- 25% penalty per day late
- Assignment schedule on course site
    - Let us know **early** if you have conflicts.

You 3 'grace days' for the semester

- Applied automatically, even if your score does not increase

Assignments

You have 2-3 weeks per assignment

- Plan to start early and work throughout
- 25% penalty per day late
- Assignment schedule on course site
  - Let us know **early** if you have conflicts.

You 3 'grace days' for the semester

- Applied automatically, even if your score does not increase

Course staff have lives (yep, it's true).
Do not expect help after 5 PM on the Friday before it is due.

Exams

- In-Class Midterms (Fri 10/4 and Friday 11/8, in class)
  - More details as exams approach
- One Final Exam (**12/13 at 8 AM**)
  - Comprehensive exam (all topics are fair game)
  - Check for conflicts ASAP
  - If HUB updates, trust the date in HUB

Please contact **Accessibility Resources** for accommodations

https://www.buffalo.edu/studentlife/who-we-are/departments/accessibility.html

Attendance / Participation

- **Lecture**
  - No recorded attendance
  - We're here to answer questions (use the opportunity!)
- **Recitation**
  - Attendance is mandatory after add/drop
  - Recitation next week is optional for tech support
  - Normal recitations begin the week of 9/9

Collaboration

- Do...
  - ... work together to brainstorm ideas
  - ... explain concepts to each other
  - ... include a list of collaborators on all submitted work
- Do **not**...
  - ... write solutions while working together
  - ... describe the details of solutions or code
  - ... leave your code in a place where someone else can see it

If in doubt, ask a member of the course staff.

Resource Policy

- Do...
  - ... use materials provided by course staff (Piazza, Class, OH)
  - ... use materials provided by textbooks, readings
  - ... cite materials you reference for written work
  - ... cite sources for all code you reference/copy

Resource Policy

- Do **not**...
  - ... reference random videos that "helped you solve the problem"
  - ... reference exact solutions found online
  - ... use LLMs/Chatbots/etc...
  - ... hire "private tutors"

Resource Policy

- Do **not**...
  - ... reference random videos that “helped you solve the problem”
  - ... reference exact solutions found online
  - ... use LLMs/Chatbots/etc...
  - ... hire “private tutors”
    - If you have an actual tutor, please contact course staff.

Why?

- This is an intro-level course. You're learning.
- If you don't understand, you will struggle with later courses (e.g., 331).
- If someone else does the work, you're not the one that understands.
- We want you to understand the pieces, so that you can (eventually) start fitting them together in clever ways.

Why?

- This is an intro-level course. You're learning.
- If you don't understand, you will struggle with later courses (e.g., 331).
- If someone else does the work, you're not the one that understands.
- We want you to understand the pieces, so that you can (eventually) start fitting them together in clever ways.
- If we catch you cheating, **you get an F**.

Other Ways to Get an F

- Work in a group by assigning each person to a problem.
- Copy a friend's homework because you forgot (∼1% of your grade is not worth it)
- Share your homework with your friend (I can't tell who copied)
- Submit work without citations (Cited work included in your project is not an AI violation)

Other Ways to Get an F

- Work in a group by assigning each person to a problem.
- Copy a friend's homework because you forgot ($\sim$1% of your grade is not worth it)
- Share your homework with your friend (I can't tell who copied)
- Submit work without citations (Cited work included in your project is not an AI violation)
  (Although we will grade you on the work you did)

You are liable if someone else submits your work as their own.

Amnesty Policy

Don't cheat!

Amnesty Policy

Don't cheat! ... but we understand that mistakes happen.

Amnesty Policy

Don't cheat! ... but we understand that mistakes happen.

You may retract any work you submit, at any time **before** we discover that it contains an AI violation.

Amnesty Policy

Don't cheat! ... but we understand that mistakes happen.
You may retract any work you submit, at any time **before** we discover that it contains an AI violation.

Dear Dr. Kennedy,

In order to preserve academic integrity in CSE 250, I would like to withdraw my submission for Project/Homework XXX.

Sincerely,
Name (ubname@buffalo.edu)

Life Lesson

If ChatGPT can do your work… you will not be employable for very long.

When to Ask Questions

- In Class (raise your hand[1])
- Piazza (Ask anytime!)
- Office Hours (All the TAs have been where you've been)
- Recitations (... if you prefer smaller, less intimidating settings)

[1] Oliver's slides always have small mistakes 'to make sure that you're paying attention'.

How to Ask Questions

To ask a question...

- Check if the answer already exists
  (syllabus, Piazza, course site)
- Frame your question carefully.
  (Avoid phrases like "it's not working")
- Let the SA/Instructor know what you tried.
  (Try to solve the problem yourself before asking)
- Copy and paste text error messages, and not screenshots
  (Piazza can't search images)

How to Ask Questions

To ask a question…

- Check if the answer already exists
  (syllabus, Piazza, course site)
- Frame your question carefully.
  (Avoid phrases like "it's not working")
- Let the SA/Instructor know what you tried.
  (Try to solve the problem yourself before asking)
- Copy and paste text error messages, and not screenshots
  (Piazza can't search images)

Often, if you follow the steps above, you'll find that you can answer your own question in the process!

Data

Data

Container

Different Container (More Defensible)

Same Data

Different Container (More efficient `skriches()`)

Same Data

So what is a data structure?

A **thing** to put your **things** in.

Why?

1. Which is easier to find stuff in: an organized or a messy room?

So what is a data structure?

A **thing** to put your **things** in.

Why?

1. Which is easier to find stuff in: an organized or a messy room?
2. Which is easier to maintain?

Examples of Data Structures

- Store a list of things in some order (”List”)
  - Array
  - Linked List
  - ArrayBuffer
- Store things organized by an Attribute (”Map”, ”Dictionary”)
  - Hash Table
  - Binary Search Tree
  - Red-Black Tree

How do I make my code efficient?

How do I make my code efficient?

- **Tactical**: Optimize your Code
  - Understand the memory hierarchy
  - Understand the CPU/OS

How do I make my code efficient?

- **Tactical**: Optimize your Code
  - Understand the memory hierarchy
  - Understand the CPU/OS
- **Strategic**: Optimize your Algorithm
  - Understand how your algorithm scales
  - Avoid repetition in your code

How do I make my code efficient?

- **Tactical**: Optimize your Code
  - Understand the memory hierarchy
  - Understand the CPU/OS
- **Strategic**: Optimize your Algorithm
  - Understand how your algorithm scales
  - Avoid repetition in your code

CSE 250 focuses on optimizing algorithms

Tactical Programming

Go from point A to point B

1. Move up 100 feet
2. Turn right, move forward 200 feet
3. Move north 10 feet then turn left
4. Move forward 20 feet
5. Move south 50 feet
6. Move west 150 feet, then turn left
7. Move forward 60 feet

Tactical Programming

Go from point A to point B

1. Move up 100 feet
2. Turn right, move forward 200 feet
3. Move north 10 feet then turn left
4. Move forward 20 feet
5. Move south 50 feet
6. Move west 150 feet, then turn left
7. Move forward 60 feet

We can optimize each individual step

Tactical Programming

Go from point A to point B

1. Move up 100 feet
2. Turn right, move forward 200 feet
3. Move north 10 feet then turn left
4. Move forward 20 feet
5. Move south 50 feet
6. Move west 150 feet, then turn left
7. Move forward 60 feet

We can optimize each individual step

- Taking a bike will speed up step 2 compared to walking.

Tactical Programming

Go from point A to point B

1. Move up 100 feet
2. Turn right, move forward 200 feet
3. Move north 10 feet then turn left
4. Move forward 20 feet
5. Move south 50 feet
6. Move west 150 feet, then turn left
7. Move forward 60 feet

We can optimize each individual step

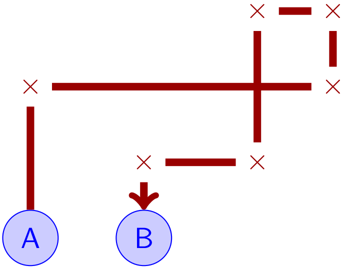
- Taking a bike will speed up step 2 compared to walking.
- Installing an east/west slip-and-slide will speed up step 6.

Strategic Programming

- Look at the big picture
- Design an algorithm
- Focus on "complexity"

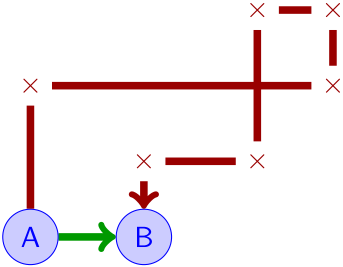
Strategic Programming

- Look at the big picture
- Design an algorithm
- Focus on “complexity”



Strategic Programming

- Look at the big picture
- Design an algorithm
- Focus on "complexity"



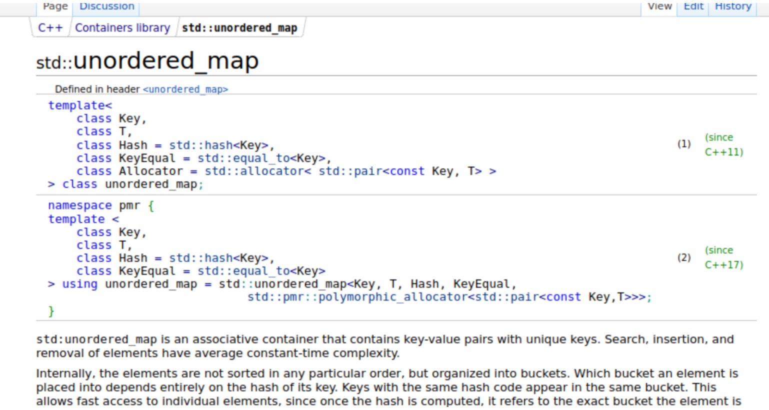
Why not just move east 30 feet...

Demo

Live Demo

(thanks to Prakshal Jain; 2021 TA for the suggestion/prototype)

C++ Standard Library



std::unordered_map

Defined in header <unordered_map>

```
template<
    class Key,
    class T,
    class Hash = std::hash<Key>,                                  (1)  (since
    class KeyEqual = std::equal_to<Key>,                               C++11)
    class Allocator = std::allocator< std::pair<const Key, T> >
> class unordered_map;
```

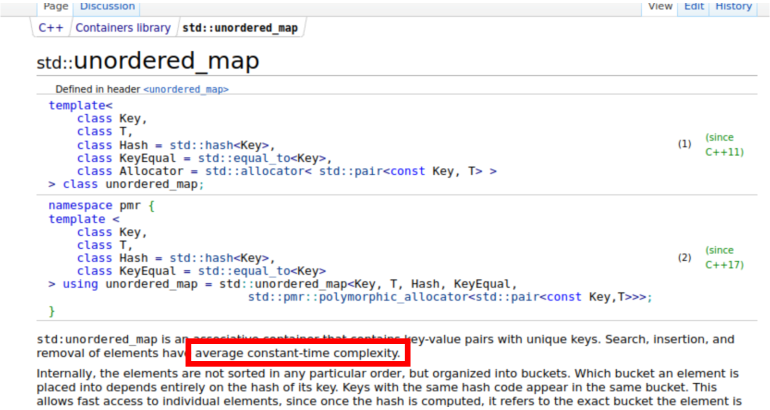
```
namespace pmr {
template <
    class Key,
    class T,
    class Hash = std::hash<Key>,                                  (2)  (since
    class KeyEqual = std::equal_to<Key>                                C++17)
> using unordered_map = std::unordered_map<Key, T, Hash, KeyEqual,
                            std::pmr::polymorphic_allocator<std::pair<const Key,T>>>;
}
```

std:unordered_map is an associative container that contains key-value pairs with unique keys. Search, insertion, and removal of elements have average constant-time complexity.

Internally, the elements are not sorted in any particular order, but organized into buckets. Which bucket an element is placed into depends entirely on the hash of its key. Keys with the same hash code appear in the same bucket. This allows fast access to individual elements, since once the hash is computed, it refers to the exact bucket the element is

https://en.cppreference.com/w/cpp/container/unordered_map

C++ Standard Library



Page | Discussion | View | Edit | History

C++ / Containers library / **std::unordered_map**

std::unordered_map

Defined in header <unordered_map>

```
template<
    class Key,
    class T,
    class Hash = std::hash<Key>,
    class KeyEqual = std::equal_to<Key>,
    class Allocator = std::allocator< std::pair<const Key, T> >
> class unordered_map;
```

(1) (since C++11)

```
namespace pmr {
template <
    class Key,
    class T,
    class Hash = std::hash<Key>,
    class KeyEqual = std::equal_to<Key>
> using unordered_map = std::unordered_map<Key, T, Hash, KeyEqual,
                            std::pmr::polymorphic_allocator<std::pair<const Key,T>>>;
}
```

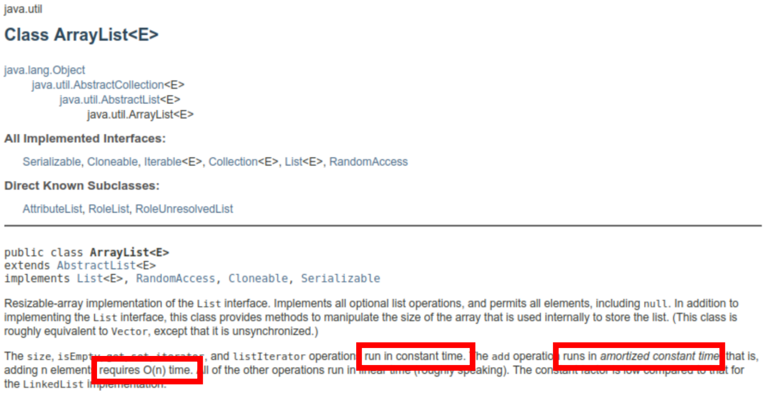
(2) (since C++17)

std:unordered_map is an associative container that contains key-value pairs with unique keys. Search, insertion, and removal of elements have average constant-time complexity.

Internally, the elements are not sorted in any particular order, but organized into buckets. Which bucket an element is placed into depends entirely on the hash of its key. Keys with the same hash code appear in the same bucket. This allows fast access to individual elements, since once the hash is computed, it refers to the exact bucket the element is

https://en.cppreference.com/w/cpp/container/unordered_map

Java’s Util



java.util

Class ArrayList<E>

java.lang.Object
  java.util.AbstractCollection<E>
    java.util.AbstractList<E>
      java.util.ArrayList<E>

All Implemented Interfaces:

Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

Direct Known Subclasses:

AttributeList, RoleList, RoleUnresolvedList

```
public class ArrayList<E>
extends AbstractList<E>
implements List<E>, RandomAccess, Cloneable, Serializable
```

Resizable-array implementation of the `List` interface. Implements all optional list operations, and permits all elements, including `null`. In addition to implementing the `List` interface, this class provides methods to manipulate the size of the array that is used internally to store the list. (This class is roughly equivalent to `Vector`, except that it is unsynchronized.)

The `size`, `isEmpty`, `get`, `set`, `iterator`, and `listIterator` operations run in constant time. The `add` operation runs in *amortized constant time*, that is, adding n elements requires O(n) time. All of the other operations run in linear time (roughly speaking). The constant factor is low compared to that for the `LinkedList` implementation.

https://docs.oracle.com/javase/7/docs/api/

Complexity Lore

Every (good) standard data structure library provides guarantees on the complexity of its data structures' operations

Complexity Lore

Every (good) standard data structure library provides guarantees on the complexity of its data structures' operations

Understanding complexity can be the difference between code that runs in 6 hours vs code that runs in 8 seconds.

Containers

We have

- A list of cats

We want

- To get the first cat in the list
- To add a new cat to the front of the list
- To get the nth cat in the list

Containers

We have

- A list of cats

We want

- To get the first cat in the list
- To add a new cat to the front of the list
- To get the nth cat in the list

This is an abstract data type

Abstract Datatypes

Stuff you store (data)

- A list of cats

Operations you can perform on the stored stuff

- To get the first cat in the list
- To add a new cat to the front of the list
- To get the nth cat in the list

So how do we store our list of cats?

Options

1. **Very Fast**: Prepend, Get First
   Very Slow: Get Nth

Options

1. **Very Fast**: Prepend, Get First
   Very Slow: Get Nth
2. **Very Fast**: Get Nth, Get First
   Very Slow: Prepend

Options

1. **Very Fast**: Prepend, Get First
   Very Slow: Get Nth
2. **Very Fast**: Get Nth, Get First
   Very Slow: Prepend
3. **Very Fast**: Prepend, Get First
   Sometimes Slow: Prepend

Options

1. **Very Fast**: Prepend, Get First
   Very Slow: Get Nth
2. **Very Fast**: Get Nth, Get First
   Very Slow: Prepend
3. **Very Fast**: Prepend, Get First
   Sometimes Slow: Prepend

Which is best?

Options

1. **Very Fast**: Prepend, Get First `Linked List`
   Very Slow: Get Nth
2. **Very Fast**: Get Nth, Get First `Array`
   Very Slow: Prepend
3. **Very Fast**: Prepend, Get First `ArrayBuffer (reversed)`
   Sometimes Slow: Prepend

Which is best?

It Depends!
No one option is always best!

Some Common Ideas

More work now

vs

More work later

Storing Data

vs

Computing Data

Course Roadmap - Part 1

Analysis Tools/Techniques	**ADTs**	**Data Structures**
Asymptotic Analysis (Unqualified) Runtime Bounds		
	Sequence	Array, Linked List
Amortized Runtime	List	ArrayList, LinkedList
Recursive Analysis Divide and Conquer Average/Expected Runtime		
	Stack, Queue	ArrayList, LinkedList
	Midterm 1	

Course Roadmap - Part 2

Analysis Tools/Techniques	**ADTs**	**Data Structures**
Revisit Recursive Analysis	Graphs PriorityQueue	Edge List, Adjacency List, Adjacency Matrix
	Trees	BST, AVL Tree, Red-Black Tree, Heaps
Midterm 2		
Review Expected Runtime	Hash Tables	Chaining, Open Addressing, Cuckoo Hashing
Misc Topics/Buffer		
Final Exam		

AI Quiz

Log into autolab; it will take you under 10 minutes.

Due Sun, Sept 8 at 11:59 PM

Successfully completing the AI exam with a passing grade is mandatory. If you don't get 100% by the deadline, you get an 'F'.

Java Hello World Project

Posted on course website; Submit a java program that prints out your github username.

Due Sun, Sept 8 at 11:59 PM

This project does not count for a grade, but you must get a 100% to pass the class.

Join Piazza

- Accept emailed invites to join the course Piazza
- Read over @6 and @7

Questions?