

CSE 250

Data Structures

Dr. Eric Mikida
epmikida@buffalo.edu
208 Capen Hall

Lec 01: Course Overview

Course Staff

Eric Mikida

Email: epmikida@buffalo.edu

Office: Capen 208 (inside of 212 Capen, inside of Silverman)

Office hours don't start until next week and will be posted to course website

Course USAs (see [Piazza](#))

Aria Orni

Dorian Trotman

Eugenia Vance

Faizaan Mohammed Ali

Hulaye Dialo

Isabel Kimos

Julia Joseph

Justin Rimpel

Mafruha Manal

Matthew Bieniak

Sam Carrillo

Sean Gaygen

Logistics

- **Course Website**
 - <https://cse.buffalo.edu/courses/cse250/2026-sp>
 - All course materials, links, schedule, extra resources
- **Course Forum (Piazza)**
 - <https://piazza.com/buffalo/spring2026/cse250>
 - All discussion for the course is hosted here – check regularly
- **AutoLab**
 - <https://autolab.cse.buffalo.edu/courses/cse250-s26>
 - Assignment submission, grades

**Please keep class discussions on Piazza (private/anonymous posts exist)
Always include [CSE 250] in the subject line when emailing**

Development Environment

- **Supported Operating Systems**
 - MacOS
 - Ubuntu Linux
 - Windows + WSL/Ubuntu
- **Supported Dev Environments**
 - IntelliJ (Community Edition is Free)
 - PA0 walks you through the setup process

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

Course Syllabus

Grading

Grade Breakdown:

- Assignments: 30%
- Participation: 10%
- Midterms: 15% x 2 = 30%
- Final Exam: 30%

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 (5% each)

~Bi-Weekly Written Assignments

- Expect to spend about a week per assignment
- Submit up to 24hrs after deadline with a 50% penalty

You are responsible for submission formatting

- Submit only PDFs
- Submissions that do not load will receive a 0

We recommend writing solutions by hand

- Better retention of what you have written
- Easier to write out math by hand than on a computer

Programming Assignments (5% each)

Grading for programming assignments will be as follows:

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

Grades will always be based on the **LAST** submission you make

Programming Assignments (5% each)

You have 2-3 weeks per assignment

- Plan to start early and work throughout
- Testing phase is due after the first week, **no late submission allowed**
- Implementation phase stays open 2 extra days
 - 25% penalty per day late, up to 48 hours
 - Grace days negate late penalties

3 'grace days' for the semester

- Applied automatically, even if your score does not increase

Exams

Two In-Class Midterms (Wed 2/25 and Fri 4/10, in class)

- More details as exams approach

One Final Exam (Thu 5/7 @ 3:30PM)

- Comprehensive, covering any topics from throughout the semester
- Check for conflicts ASAP
- If HUB changes the date/location...trust the HUB

If you need accommodations, contact [Accessibility Resources](#) ASAP

Class Participation

Lecture

- No recorded attendance
- Easy access to ask questions live (use it)

Recitation

- Attendance is mandatory
- **NO RECITATION THIS WEEK**
- Recitations begin next week

Academic Integrity

Collaboration

Do...

- Work together to brainstorm ideas
- Explain concepts to each other

Do Not...

- Write solutions when working together
- Describe the details of solutions to problems or code
- Leave your code in a place where it is accessible to another student

When in doubt, ask a member of the course staff!

Collaboration

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

Resource Policy

Do...

- Use materials provided by course staff (Piazza, Class, OH)
- Use materials from the course lectures / recitations
- Cite all materials you reference for written work
- **Cite sources** for all code you reference / copy

Resource Policy

Do NOT...

- Reference random videos on YouTube that “helped you solve the problem”
- Hire “private tutors”
 - Save the money from Chegg
 - If you’re not doing the work yourself, you’re not learning
 - If you have an actual tutor, contact course staff
- Reference exact solutions found online
- Use ChatGPT or other generative AI to write your code for you

If you are caught using unauthorized resources, you get an F

A Note About ChatGPT (and other LLMs)

- Good at writing code, not good at design
- Helpful when you have the fundamentals...harmful when you don't
- Being a computer scientist is not about writing code it's about designing solutions to problems

(As a reminder, you are liable if someone submits your work as their own...or you both submit the same solution...it all looks the same to us)

Other Ways to Get an F

- Work in a group by assigning each person to a problem
- Copying your friend's homework because you forgot
 - Each homework is not worth a lot on its own
- Sharing your homework with your friend
 - I have no way to know who did the work and who shared
- Submitting work without citations

Amnesty Policy

Don't Cheat...but we understand mistakes are made.

We will grant amnesty for any AI violation **IF** you tell us about it **BEFORE**
we discover it

How to be Successful in 250

How to be Successful in 250

- Start assignments early!
 - Assignments are designed to be completable in the given timeframe...
 - But technical difficulties **will occur**
 - Bugs in your code **will occur**
 - Real life curveballs **will occur**
- Ask questions!
 - Ask them early
 - Ask them often
 - Ask in lecture, recitation, piazza and office hours!

Asking Questions

First...**check if the answer exists** (syllabus, Piazza, course website)

Then...

Ask in lecture, recitation, Piazza, or office hours

Come prepared, form the question carefully, many times you will answer your own question in the process!

Thinking through your question is a great first step.

Still worried?

When in doubt, come talk to us

- We **WANT** you to succeed
- We **WANT** to help you

If you are worried about coming to talk to the professor, talk to the TAs!

- They took this course recently, they know what it's like

**Now...What even is
"Data Structures"!?**

What is a Data Structure?

Data



Container

What is a Data Structure?

Same Data



Different Container

more defensible

What is a Data Structure?

Same Data



more efficient access to
skritches()

Different
Container

What is a Data Structure?

- Store a list of things in some order (“List”)
 - Array
 - LinkedList
 - ArrayList
- Store things organized by an attribute (“Map”, “Dictionary”)
 - Hash Table
 - Binary Search Tree
 - Red-Black Tree

Why should you care?

- **Tactical:** Optimize your Code (“reducing the constants”)
 - Understand the memory hierarchy
 - Understand the CPU / OS
- **Strategic:** Optimize your Design (“reducing the complexity”)
 - Understand how your algorithm scales
 - Understand repetition in your code

CSE 250

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

- For example, taking a bike will speed up step 2 compared to walking

Strategic Programming

Look at the big picture

Design (not just implement) an algorithm

Focus on "complexity"

What is “Complexity”?

`mapped_type` must be *emplace_constructible* from `std::piecewise_construct`, `std::forward_as_tuple(std::move(key))`, `std::tuple<>()`.
When the default allocator is used, this means that `key_type` must be *MoveConstructible* and `mapped_type` must be *DefaultConstructible*.

No iterators or references are invalidated.

Parameters

key - the key of the element to find

Return value

Reference to the mapped value of the new element if no element with key `key` existed. Otherwise a reference to the mapped value of the existing element whose key is equivalent to `key`.

Exceptions

If an exception is thrown by any operation, the insertion has no effect

Complexity

Logarithmic in the size of the container.

Notes

In the published C++11 and C++14 standards, this function was specified to require `mapped_type` to be *DefaultInsertable* and `key_type` to be *CopyInsertable* or *MoveInsertable* into `*this`. This specification was defective

(screenshot: cppreference.com)

What is “Complexity”?



[scala.collection.immutable](https://scala-lang.org/api/scala/collection/immutable)

Vector

Companion object **Vector**

```
sealed abstract class Vector[+A] extends AbstractSeq[A] with IndexedSeq[A] with IndexedSeqOps[A, Vector, Vector[A]] with StrictOptimizedSeqOps[A, Vector, Vector[A]] with IterableFactoryDefaults[A, Vector] with DefaultSerializable
```

Vector is a general-purpose, immutable data structure. It provides random access and updates in $O(\log n)$ time, as well as very fast append/prepend/tail/init (amortized $O(1)$, worst case $O(\log n)$). Because vectors strike a good balance between fast random selections and fast random functional updates, they are currently the default implementation of immutable indexed sequences.

Vectors are implemented by radix-balanced finger trees of width 32. There is a separate subclass for each level (0 to 6, with 0 being the empty vector and 6 a tree with a maximum width of 64 at the top level).

What is “Complexity”?

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

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

Some Common Ideas

More work now

VS

More work later

Storing Data

VS

Computing Data

Course Roadmap

Analysis Tools/Techniques	ADTs	Data Structures
Asymptotic Analysis, (Unqualified) Runtime Bounds	Sequence	Array, LinkedList
Amortized Runtime, Average/Expected Runtime	List	ArrayList, LinkedList
	Set, Stack, Queue	ArrayList, LinkedList
Midterm #1		

Course Roadmap

Analysis Tools/Techniques	ADTs	Data Structures
	Graphs	EdgeList, AdjacencyList, AdjacencyMatrix
Recursive analysis and Induction	Sets, PriorityQueue	BST, AVL Tree, Red-Black Tree, Heaps
Midterm #2		
Expected runtime	Sets, Maps	HashTables
Miscellaneous		

First Assignments

First Assignments

Academic Integrity Quiz

- Posted on AutoLab, due Sun Feb 1 @ 11:59PM
- **YOU MUST GET 100% TO PASS THE COURSE**

PA0

- Posted to course website, due Sun Feb 1 @ 11:59PM
- **YOU MUST GET 100% TO PASS THE COURSE**

Join Piazza

- Accept invites sent via email to join the course Piazza
- Read over @6 and @7

Questions?