CSE 250

Data Structures

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

Lec 01: Course Overview

Course Staff

Eric Mikida

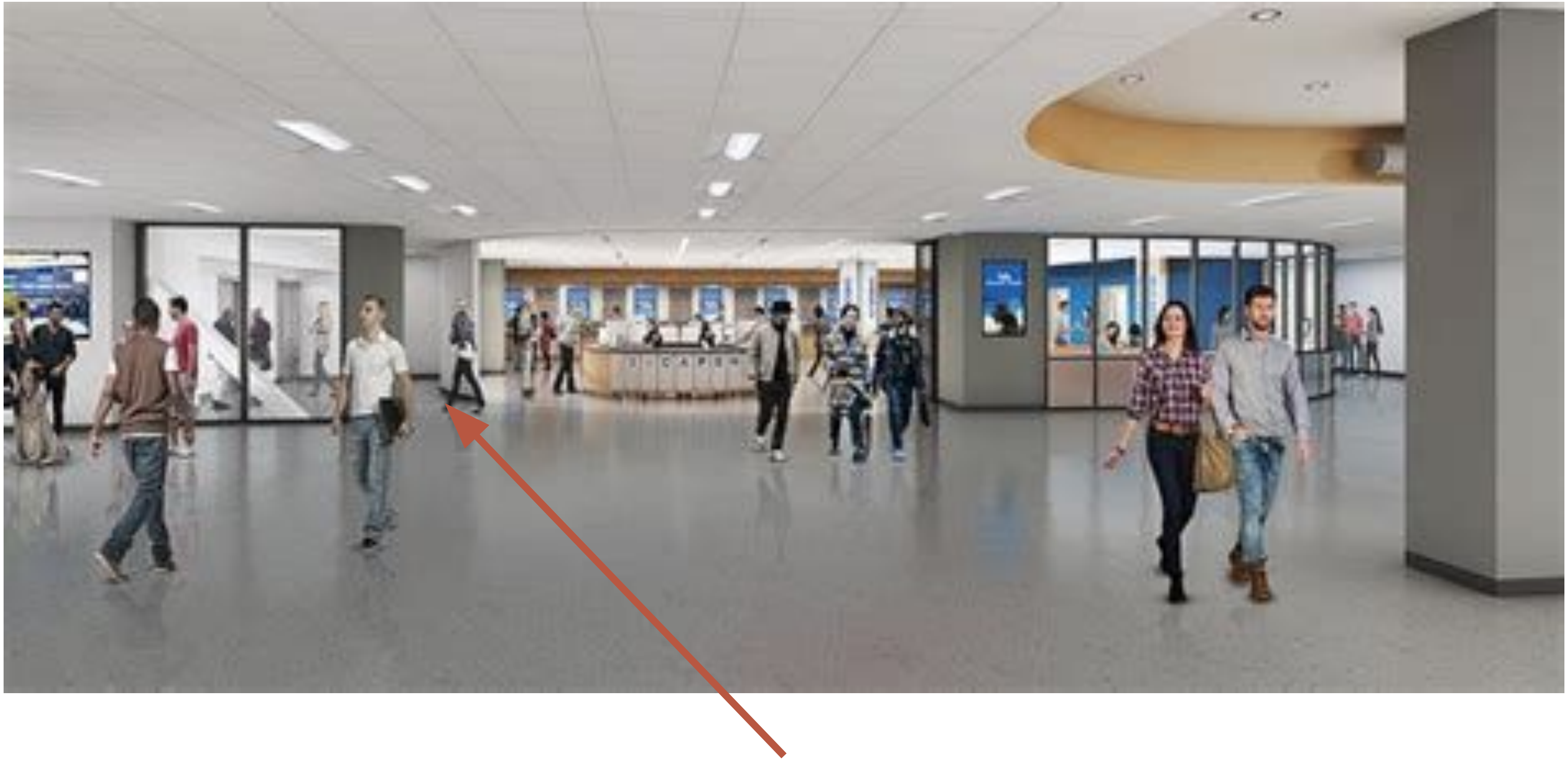
Email: epmikida@buffalo.edu

Office: Capen 208 (inside of 212 Capen)

Course TAs

Amelia Graca
Dikshit Khandelwal
Kartike Chaurasia
Riad Mukhtarov
Alexander Terry
Brendan O'Connell
Chris Dearing
Derek Gage
Doniyor Ismatilloev
Ethan Phan
Evan Jiang
Jonathan Guzman
Jordan Wang
Kiki Tran
Marian Huynh
Marvin Pierre-Pierre
Milos Petrovic
Morgan Li
Eric Xie
Ronan Kasmier
Shreyas Narayanan Sridhar
Wonwoo Jeong
Joy Lee

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



Take the elevators next to 1Capen to 2, then turn right.

Logistics

- **Course Website**
  - https://cse.buffalo.edu/courses/cse250
  - All course materials, links, schedule, extra resources
- **Course Forum (Piazza)**
  - https://piazza.com/buffalo/spring2024/cse250
  - All discussion for the course is hosted here – check regularly
- **AutoLab**
  - https://autolab.cse.buffalo.edu/courses/cse250-s24
  - Assignment submission, grades

Logistics

- **Course Website**
  - https://cse.buffalo.edu/courses/cse250
  - All course materials, links, schedule, extra resources
- **Course Forum (Piazza)**
  - https://piazza.com/buffalo/spring2024/cse250
  - All discussion for the course is hosted here – check regularly
- **AutoLab**
  - https://autolab.cse.buffalo.edu/courses/cse250-s24
  - 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: 40%
- Participation: 10%
- Midterms: 15% x 2 = 30%
- Final Exam: 20%

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

Written Assignments

~Bi-Weekly Written Assignemnts

- 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

Grading for most 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

You have 2-3 weeks per assignment

- Plan to start early and work throughout
- 25% penalty per day late, up to 48 hours

3 ‘grace days’ for the semester

- Applied automatically, even if your score does not increase

Exams

Two In-Class Midterms (Fri 3/1 and Fri 4/12, in class)

- More details as exams approach

One Final Exam (5/13/24 @ 3:30PM, Knox 110)

- 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

- Recitations start Mon, Jan 29th
- Attendance is mandatory (starting after add/drop)
- Next week recitations will focus on setting up your dev environment

Collaboration, AI, Extra Resources

Do…

- Work together to brainstorm ideas
- Explain concepts to each other
- Include a list of your collaborators on all submitted work

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!

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

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

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
  - Citing outside work will help you avoid AI repercussions
  - (we grade you on the work you did, but you won’t get an AI violation)

Other Ways to Get an F

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

Ways to Avoid an F

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

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.

Now…What even is “Data Structures!?

What is a Data Structure?

Data

Container

What is a Data Structure?

SameD
ata

Different Container

more defensible

What is a Data Structure?

SameD
ata

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

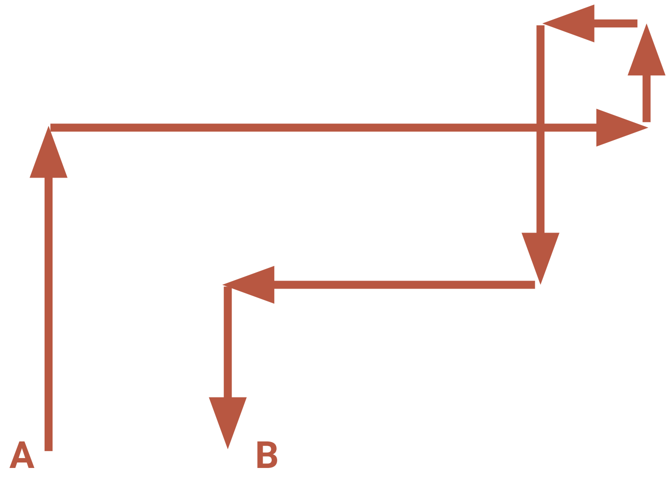
Focus on "complexity"

Strategic Programming

Look at the big picture

Design (not just implement) an algorithm

Focus on "complexity"

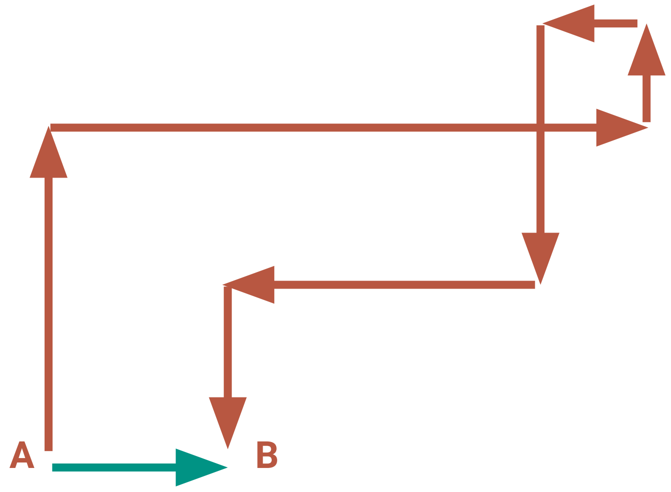


Strategic Programming

Look at the big picture

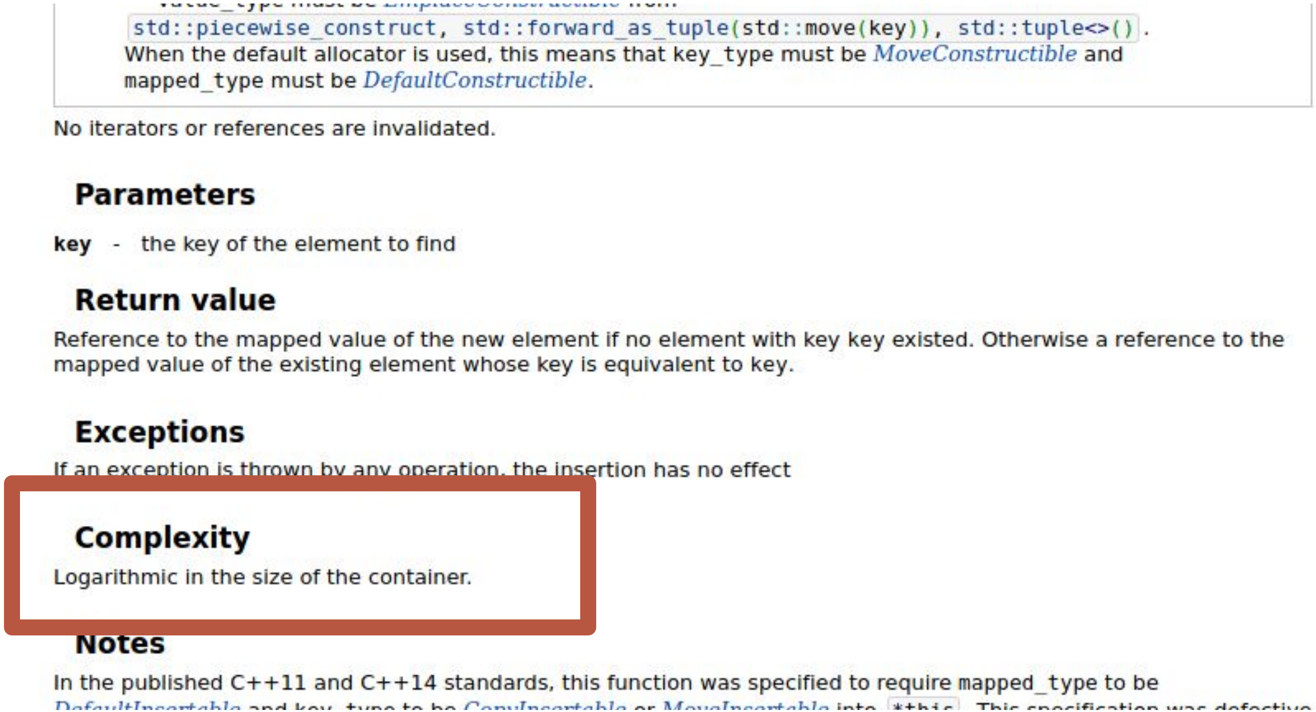
Design (not just implement) an algorithm

Focus on "complexity"



Why not just move east 30 feet...

What is “Complexity”?



value_type must be *EmplaceConstructible* 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

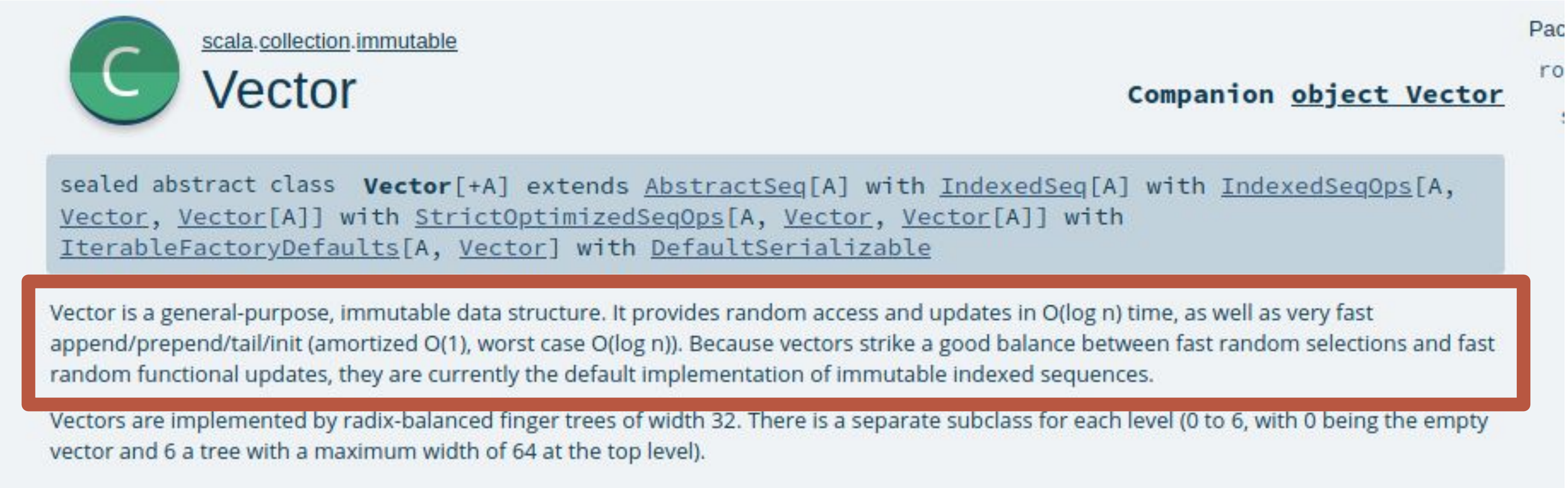
Logarithmic in the size of the container.

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

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).

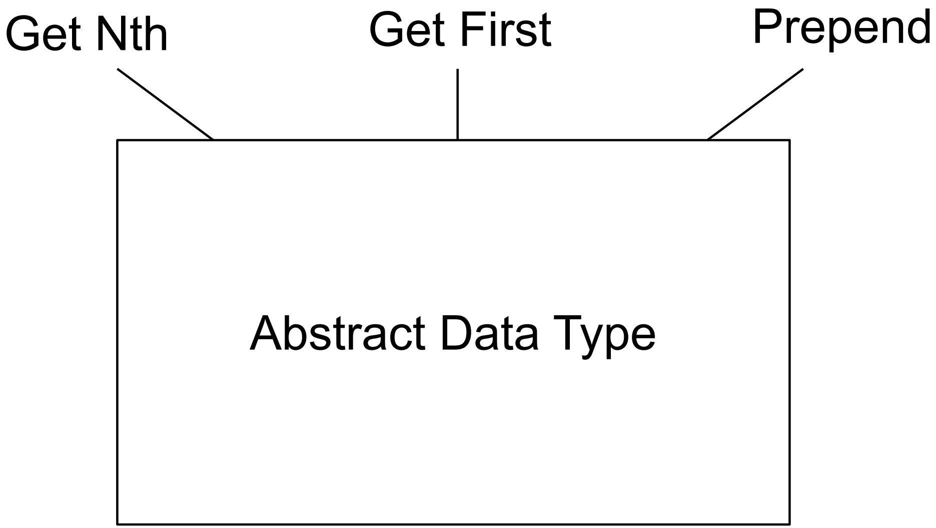
(screenshot: https://www.scala-lang.org/api)

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.

Analyzing Solutions



Analyzing Solutions

Option 1

- Very fast Prepend, Get First
- Very slow Get Nth

Option 2

- Very fast Get Nth, Get First
- Very slow Prepend

Option 3

- Very fast Get Nth, Get First
- Occasionally slow Prepend

Which is better?

Analyzing Solutions

Option 1 (Linked List)

- Very fast Prepend, Get First
- Very slow Get Nth

Option 2 (Array)

- Very fast Get Nth, Get First
- Very slow Prepend

Option 3 (ArrayList…in reverse)

- Very fast Get Nth, Get First
- Occasionally slow Prepend

Which is better?

IT DEPENDS!

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	List	ArrayList, LinkedList
Recursive analysis, divide and conquer, Average/Expected Runtime		
Midterm #1		

Course Roadmap

Analysis Tools/Techniques	ADTs	Data Structures
	Stack, Queue	ArrayList, LinkedList
Review recursive analysis	Graphs, PriorityQueue	EdgeList, AdjacencyList, AdjacencyMatrix
	Trees	BST, AVL Tree, Red-Black Tree, Heaps
Midterm #2		
Review expected runtime	HashTables	Chaining, OpenAdressing, Cuckoo Hashing
Miscellaneous		

First Assignments

Academic Integrity Quiz

- Posted on AutoLab
- Should take < 10 minutes, unlimited attempts
- Due Sun Feb 4 @ 11:59PM
- **YOU MUST GET 100% TO PASS THE COURSE**

PA0

- Posted to course website (submission on AutoLab)
- Walks through setup of IntelliJ and GitHub
- Also covered in next weeks recitations
- Due Sun Feb 4 @ 11:59PM
- **YOU MUST GET 100% TO PASS THE COURSE**

Join Piazza

- Accept invites sent via email to join the course Piazza

Questions?