CSE332: Data Abstractions

Lecture 1: Intro; ADTs; Stacks/Queues

James Fogarty

Winter 2012

Today's Outline

- Introductions
- Homework 0
- Course Administrivia
- Project 1
- What is this course about?
- ADTs; Stacks/Queues
- What is this course *really* about?

CSE 332 Team

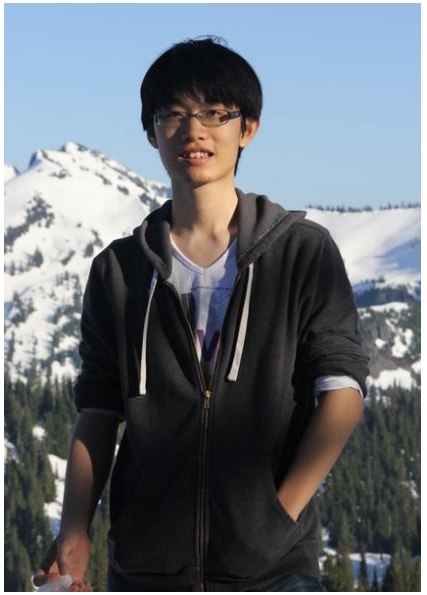
- Instructor:
  - James Fogarty, CSE 666
- TAs:
  - Tyler Robison
  - Haochen Wei
  - ??
- Email:
  - cse332-staff@cs
- Web:
  - http://www.cs.washington.edu/332

CSE 332 Team

- Instructor:
  - James Fogarty, CSE 666
- TAs:
  - Tyler Robison
  - Haochen Wei
  - ??
- Email:
  - cse332-staff@cs
- Web:
  - http://www.cs.washington.edu/332

CSE 332 Team

- Instructor:
  - James Fogarty, CSE 666
- TAs:
  - Tyler Robison
  - Haochen Wei
  - ??
- Email:
  - cse332-staff@cs
- Web:
  - http://www.cs.washington.edu/332



Section

- Section AC: Th, 12:30 - 1:20 p.m. **EEB 037**
- Section AA: Th, 1:30 - 2:20 p.m., EEB 037
- Section AB: Th, 2:30 - 3:20 p.m., EEB 031

Note room change for Section AC. Should be formalized today.

Homework 0:

- Name
- Year (1,2,3,4,5,6,??)
- Hometown

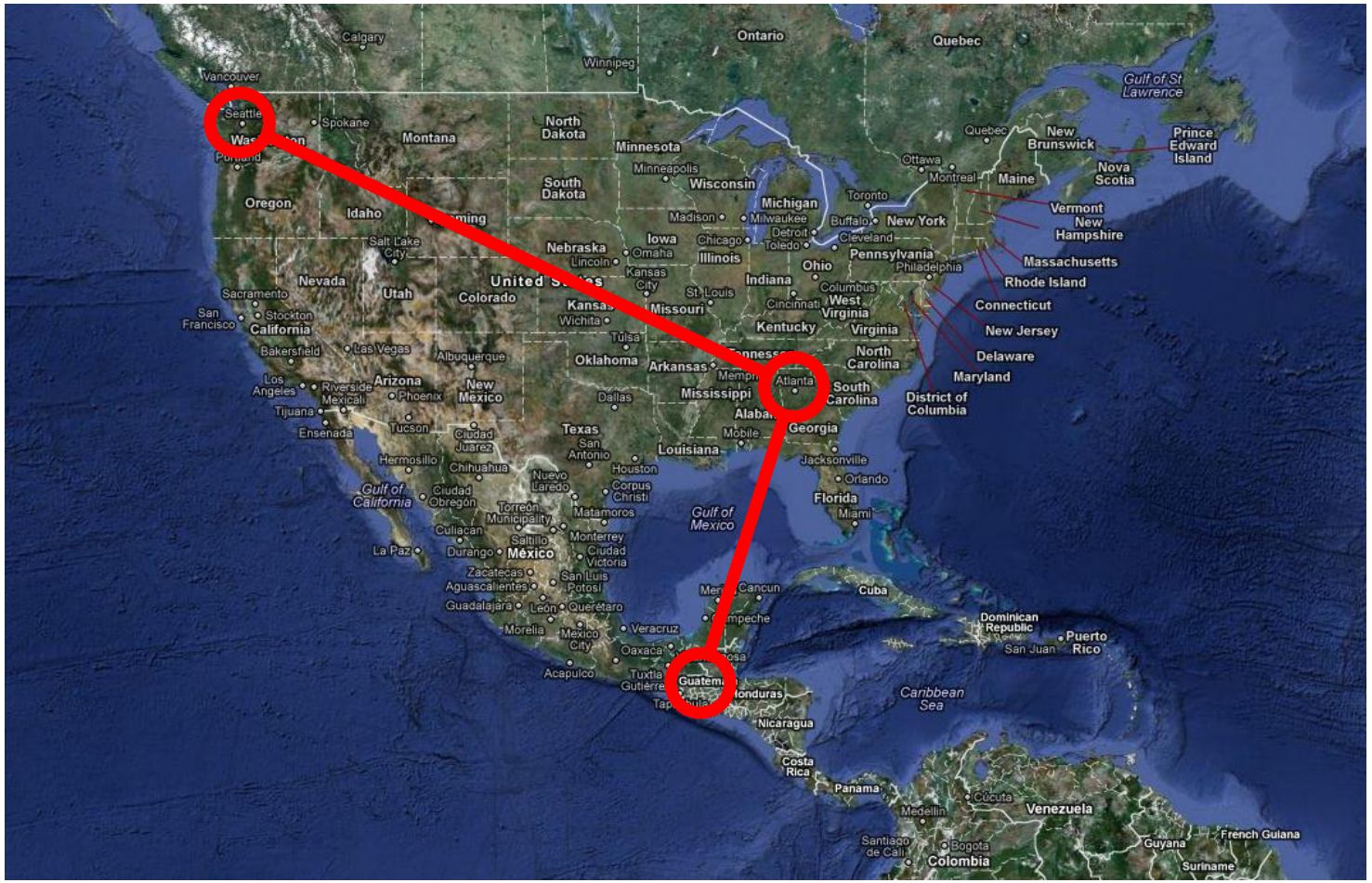
- Interesting Fact or
  "What I did on winter break"

- Submit as PDF via Catalyst

Today's Outline

- Introductions
- Homework 0
- Course Administrivia
- Project 1
- What is this course about?
- ADTs; Stacks/Queues
- What is this course *really* about?

Overloads, Lecture, and Geography



Overloads, Lecture, and Geography

- Tyler will give Friday and Monday's lectures
- I will coordinate with the advising staff via email
  - We should make the normal Monday schedule for final decisions on overload requests
- Ask me administrative questions today after class
  - Or send email, but do not expect immediate response
- TAs available for project and homework questions

Communication

Instructors

- cse332-staff@cs
- Office Hours TBA, or by appointment

Announcements

- cse332a_wi12@u.washington.edu
- You are automatically subscribed via @u.washington.edu
- You are responsible for traffic on this list

Monitored Discussion Forum

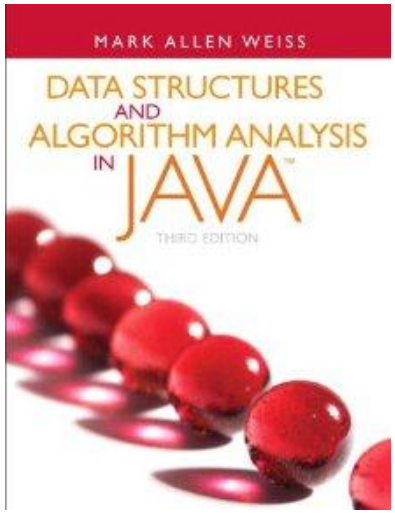
- Linked from website
- Please use real name and provide a picture
- We will sign up for email notifications, you can too

Anonymous Feedback

- Linked from Website

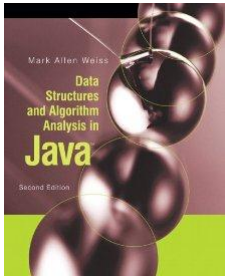
Textbook

Primary Textbook



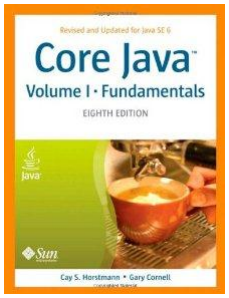
http://www.openisbn.org/price/0132576279/

Old version that we will try to support



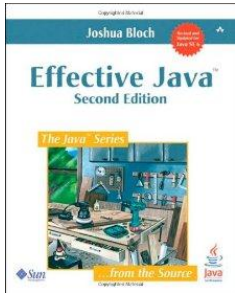
http://www.openisbn.com/price/0321370139/

People I trust say this is helpful



http://openisbn.com/price/0132354764/

I think everybody should read this



http://openisbn.com/price/9780321356680/

Course Calendar and Structure

- Holidays have a significant calendar impact this quarter
- Written homeworks assigned and due on Fridays
  - 7 total, first assigned this Friday, see “Written HW Guidelines”
  - Each homework covers through the preceding Monday
- Major programming projects
  - 3 total, each with multiple submissions
  - Project 1 is individual, posted and assigned beginning today
  - Projects 2 and 3 will allow you to work with a partner
  - Will post all projects as soon as they are prepared
- Midterm exam and final exam

Grading Mechanics

- Approximate Grading
  - 20% - Written Homework Assignments
  - 30% - Programming Projects
  - 20% - Midterm Exam
  - 25% - Final Exam
  - 5% - Best of the Above
- Drop lowest homework
- Compilation and correctness is only 40% of project grade
  - This course as a transition to the 400-level courses
- See “Grading Policies” and “Programming Guidelines”

Submission Mechanics

- Homeworks
  - Physical hand-in Friday at beginning of class
    - Late homework extremely penalized or not accepted
- Projects
  - In Java (required), using Eclipse (suggested)
  - Staged to ensure minimal progress
  - Submission via Catalyst upload
  - One project "late day" for use on any project
    - You must email cse332-staff before the deadline
- See "Grading Policies", "Programming Guidelines", and "Written HW Guidelines"

Collaboration and Academic Integrity

- Carefully read the course "Collaboration Policy"
  - Explains quite clearly how you can and cannot get or provide help on homework and projects
  - Understand the spirit of the "Gillian's Island rule"

- Always explain any unconventional action on your part
  - When it happens, when you submit, not after we ask

- I will promote an environment of great trust
  - But I will have little sympathy for violations

Project 1

- Sound Blaster!
  - Program for reversing the samples in an audio file
- Intellectual core is the implementation of a four stacks
  - Array and List implementations
  - Double and Generic implementations
- Read the website, get started immediately
  - Ask questions on forum or in section tomorrow
  - Milestone due next Wednesday, Jan 11
  - Full project due Wednesday, Jan 18

Today's Outline

- Introductions
- Homework 0
- Course Administrivia
- Project 1
- What is this course about?
- ADTs; Stacks/Queues
- What is this course *really* about?

What is 332 About

- Introduction to some of the basic structures used in all software
  - Understand the data structures and their tradeoffs
  - Rigorously analyze the algorithms that use them
  - Learn how to pick “the right thing for the job”
    - Time vs. space
    - One operation more efficient if another less efficient
    - Generality vs. simplicity vs. performance
  - Learn to justify and communicate your decisions

- Practice design, analysis, and implementation

- Experience the joy and the agony of multithreading

Today's Outline

- Introductions
- Homework 0
- Course Administrivia
- Project 1
- What is this course about?
- ADTs; Stacks/Queues
- A minor detail regarding geography
- What is this course *really* about?

Terminology

- Abstract Data Type (ADT)
  - Mathematical description of a “thing” with set of operations
- Algorithm
  - A high level and language-independent description of a step-by-step process
- Data Structure
  - A specific family of algorithms for implementing an ADT
- Implementation
  - A specific instantiation in a specific language

Example: Stacks

- The ***Stack*** ADT supports operations:
  - **`isEmpty`**: have there been same number of pops as pushes
  - **`push`**: takes an item
  - **`pop`**: raises an error if isEmpty, else returns most-recently pushed item not yet returned by a pop
  - Often some more operations

- A Stack data structure could use a linked-list or an array or something else, with associated algorithms for the operations

- One implementation is in the library **`java.util.Stack`**

Why is a Stack Useful

The Stack ADT is a useful abstraction because:

- It arises all the time in programming (see Weiss 3.6.3)
  - Recursive function calls
  - Balancing symbols (parentheses)
  - Evaluating postfix notation: 3 4 + 5 *
  - Infix ((3+4) * 5) to postfix conversion
- We can code up a reusable library
- We can communicate in high-level terms
  - "Use a stack and push numbers, popping for operators…"
  - Rather than, "create a linked list and add a node when…"

The Queue ADT

- Operations

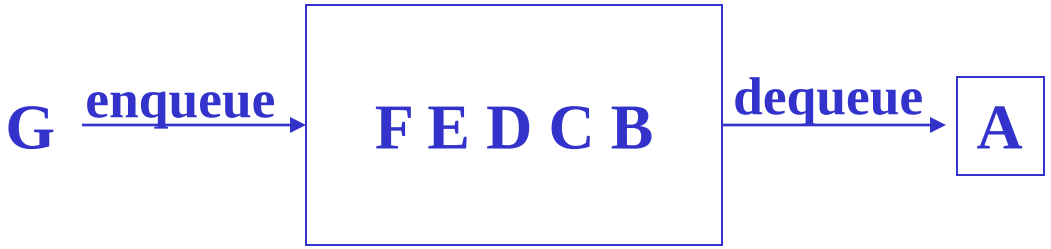
  `create`

  `destroy`

  `enqueue`

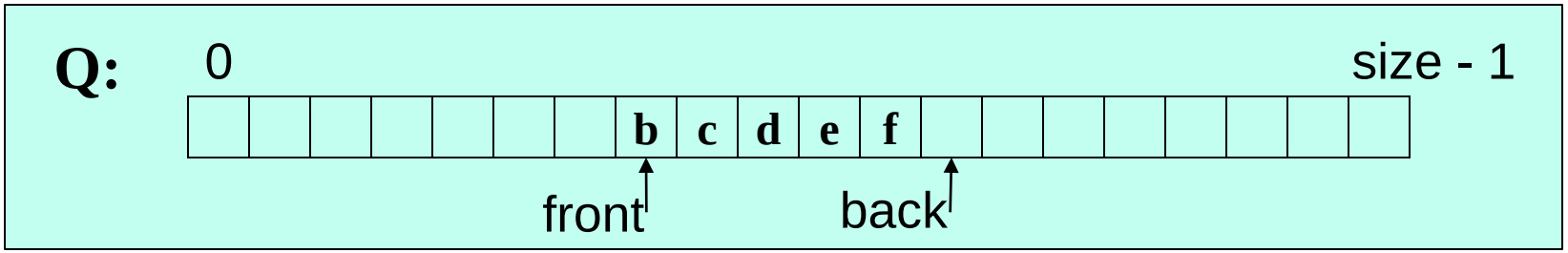
  `dequeue`

  `is_empty`



- Just like a stack except:
  - Stack: LIFO (last-in-first-out)
  - Queue: FIFO (first-in-first-out)
- Just as useful and ubiquitous

Circular Array Queue Data Structure

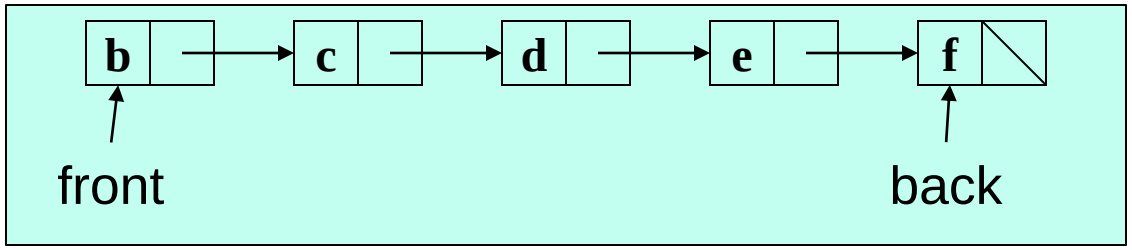



```
// Basic idea only!
enqueue(x) {
  Q[back] = x;
  back = (back + 1) % size
}
```

```
// Basic idea only!
dequeue() {
  x = Q[front];
  front = (front + 1) % size;
  return x;
}
```

- What if ***queue*** is empty?
  - Enqueue?
  - Dequeue?
- What if ***array*** is full?
- How to *test* for empty?
- What is the *complexity* of the operations?
- Can you find the k^{th} element in the queue?

Linked List Queue Data Structure




```
// Basic idea only!
enqueue(x) {
  back.next = new Node(x);
  back = back.next;
}
```

```
// Basic idea only!
dequeue() {
  x = front.item;
  front = front.next;
  return x;
}
```

- What if ***queue*** is empty?
  - Enqueue?
  - Dequeue?
- Can ***list*** be full?
- How to *test* for empty?
- What is the *complexity* of the operations?
- Can you find the k^{th} element in the queue?

The Stack ADT

- Operations

  `create`

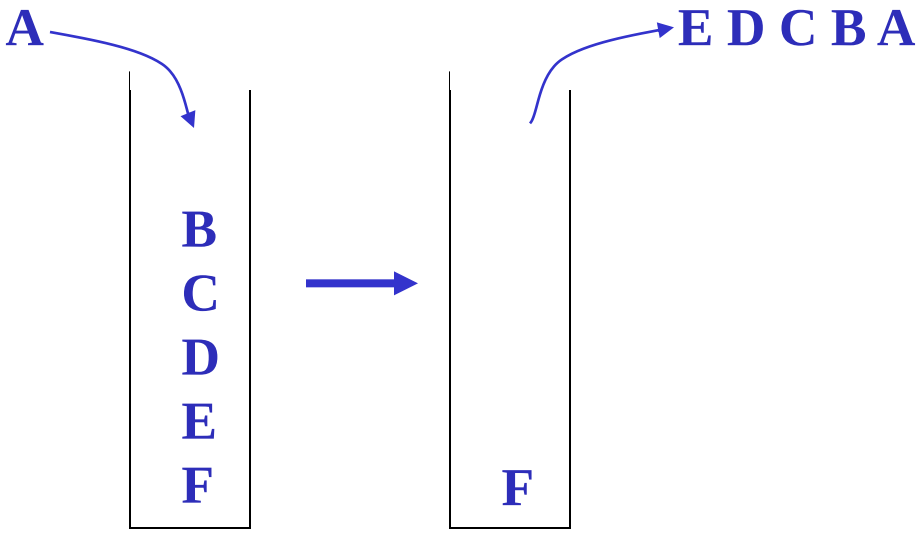
  `destroy`

  `push`

  `pop`

  `top`

  `is_empty`



- Can also be implemented with an array or a linked list
  - This is Project 1!
  - As with queues, type of elements is irrelevant
    - Ideal for Java's generic types (Project 1B)

Array vs. Linked List Implementations

Array:

- May waste unneeded space or run out of space
- Space per element excellent
- Operations very simple / fast
- Constant-time access to k^{th} element
- For operation insertAtPosition, must shift elements
  - But not part of these ADTs

List:

- Always just enough space
- But more space per element
- Operations very simple / fast
- No constant-time access to k^{th} element
- For operation insertAtPosition must traverse elements
  - But not part of these ADTs

This is something every trained computer scientist knows in their sleep. It's like knowing how to do arithmetic or ride a bike.

Today's Outline

- Introductions
- Homework 0
- Course Administrivia
- Project 1
- What is this course about?
- ADTs; Stacks/Queues
- What is this course *really* about?

What is CSE 332 Really About

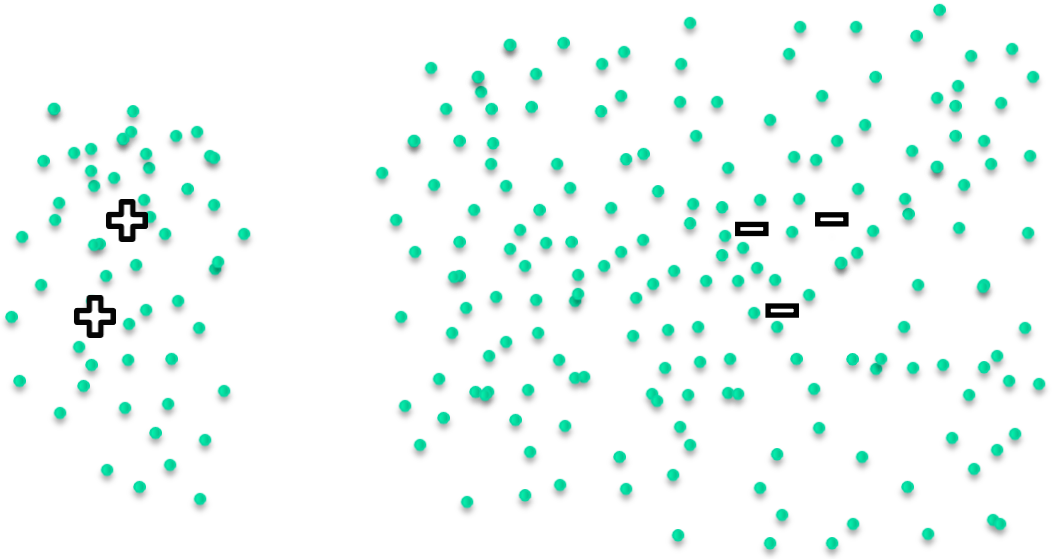
- Steve Seitz says:
  - 100 level and some 300 level courses teach how to do stuff
  - 332 teaches ***really cool*** ways to do stuff
  - 400 level courses teach how to do ***really cool*** stuff

- Dan Grossman says:
  - Three years from now, this course will seem like it was a waste of your time because you cannot imagine not "just knowing" every main concept in it
  - Key abstractions computer scientists use almost every day
  - A major aspect of what separates us from others who program

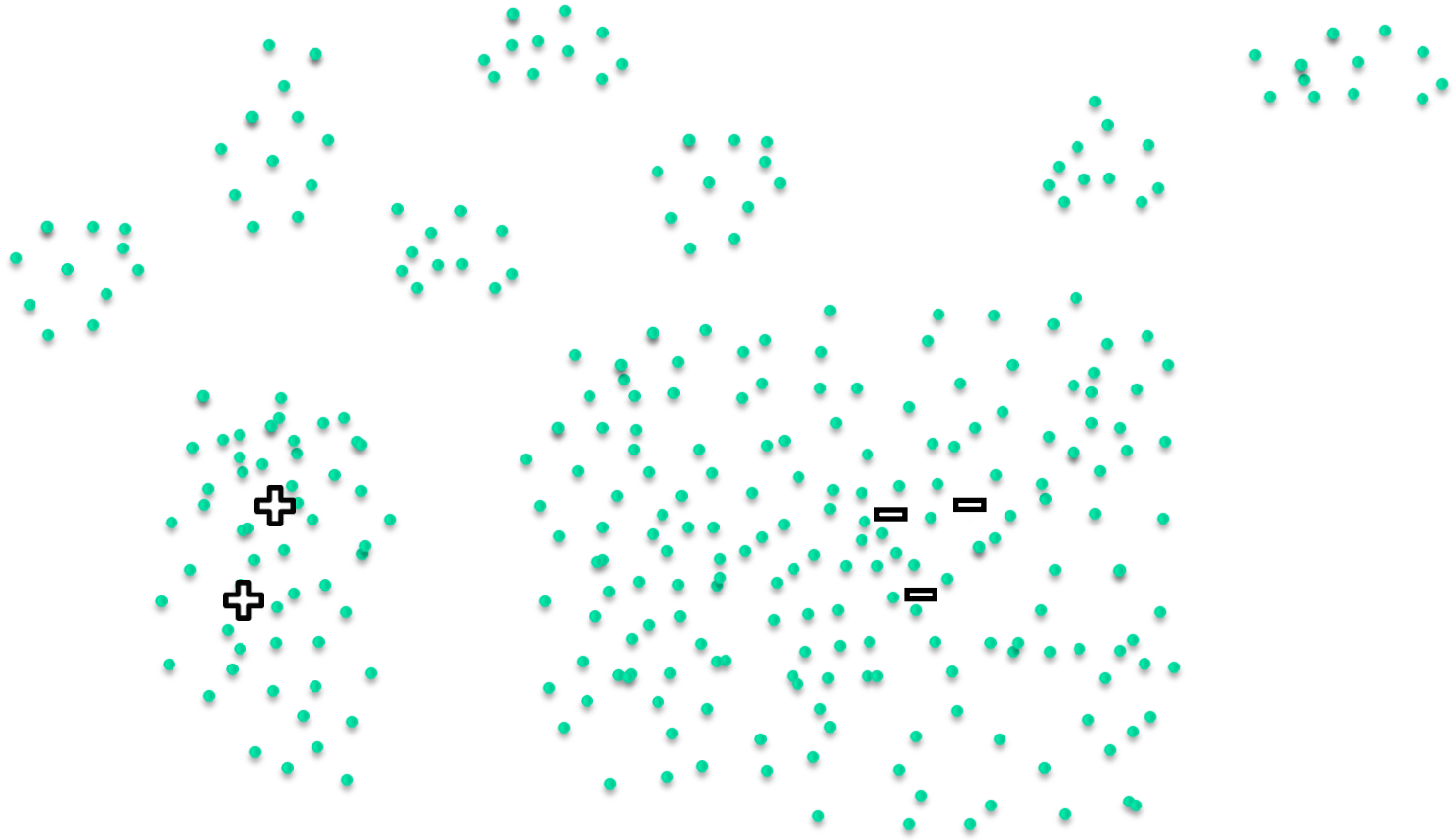
What is CSE 332 Really About

- James Fogarty says:
  - Computers are fricking insane
    - Raw power can enable bad solutions to many problems
  - This course is about how to attack non-trivial problems
    - Problems where it actually matters how you do it

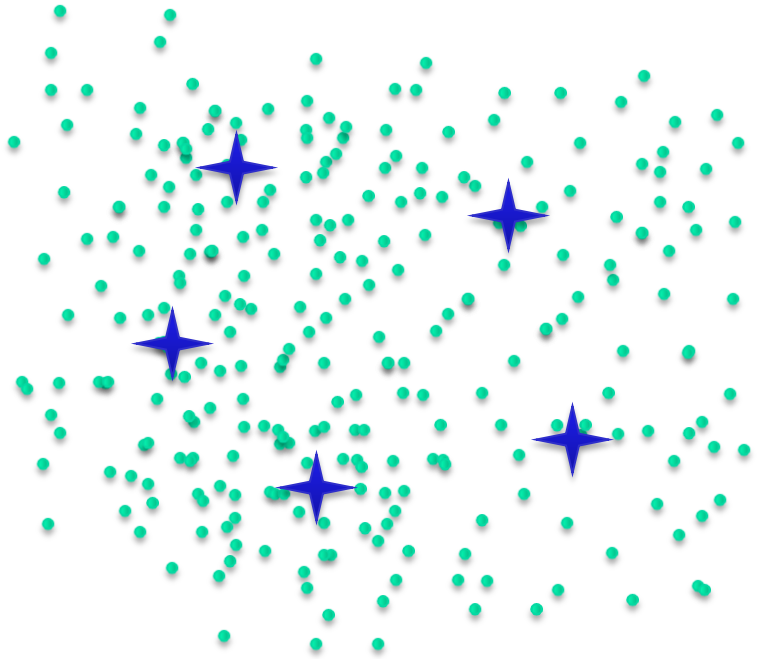
How Do We Decide Which are Positive?



How About Now?



How Do We Choose a Representative Set?



Things to Do

- Read the webpage and course policies
- Read and get started on the project
- Readings in Weiss
  - Chapter 1
  - Chapter 2
  - Chapter 3