CSE 373: Data Structures and Algorithms
Lecture 1: Introduction, ADTs, Stacks & Queues

Instructor: Lilian de Greef
Quarter: Summer 2017
Welcome!

Today’s Structure:
• Introductions and course mechanics
• Start material
  • Abstract Data Types (ADTs)
  • Stacks
  • Queues
Self Introductions

(Your homework 0!)
Lilian de Greef

• CSE PhD Student
• Working with Shwetak Patel on health applications of CS
• Interests & Hobbies
  • Ultimate Frisbee
  • Piano
  • Hiking / backpacking
  • Some TV shows

ldegreef@cs.washington.edu
cse373-staff@cs.washington.edu
Kyle Thayer
Ben Jones
Vlad Shamalov
Dorothy

- Senior (undergrad) in Computer Science and ACMS
- TAing this quarter because I loved this class both when I took it and when I’ve been a TA for 373 in the past
- Some of my hobbies are reading, exploring Seattle, and photography
Anupam Gupta

- Junior - Majoring in CSE and HCDE.
- Hobbies: Watching Movies, Sleeping.
- Interests: AI, Programming Languages, Data Mining.
- Why TA? Because it’s a lot of fun and also because I get to meet a lot of new, fun, people and talk to them about CS (which is awesome!!)
- See you all around!
Course Logistics
Classroom environment

- Laptop policy
- Lectures starting promptly at 10:50
- Will have discussions in class
  - With neighbors
  - With entire class
  - Hence, pack yourselves to the front and sit together
- Somewhere we can feel comfortable making mistakes
  - One of the best ways to learn!
General Logistics

• Website: http://cs.washington.edu/373
• Mailing list: cse373a_17su@uw.edu
• Piazza discussion board
• Textbook: Weiss 3rd Edition in Java
• Computers for homework assignments
  • College of Arts & Sciences Instructional Computing Lab: http://depts.washington.edu/aslab/
  • Or your own machine
• Java
  • Used for programming assignments
  • Recommended environment: Eclipse
Sections & Office Hours

• TBA by Tuesday, in class on Wednesday
• Lilian’s office hours (*for just today*):
  • 1:00 – 2:00pm
  • CSE 220
Contact

• Use Piazza!
  • https://piazza.com/washington/summer2017/cse373
  • Don’t post code or solutions publicly
  • For questions with code, solutions, grades, etc., make private posts to instructors
  • Can post anonymously

• Email me
  • For "Lilian's eyes only" concerns
  • I'll reply within 24 hours
  • Put [CSE 373] at beginning of subject
Collaboration and Academic Integrity

DON’T CHEAT!

Seriously, read the policy online.
Using PollEverywhere

• How:
  • You anonymously vote on multiple choice questions in lecture
  • Via text messaging (SMS) or web browser (don’t need to buy a clicker)*

• Why:
  • A way for me to check in
  • A way for you to check in
  • Research shows using Peer Instruction with polling improves learning!

* If access to SMS or a web browser in class is a challenge for you, please come talk to me
Using PollEverywhere: for Peer Instruction

• Format
  1. I'll pose a question
  2. Vote individually, invisible to class
  3. Discuss!
  4. Group vote

• Discussion is key!
  • "Just getting the right answer" is not enough - need to be able to explain/argue for it!
  • Testing yourself helpful ("right answer"), but learning happens during discussion
Take part in class-wide discussion!

• I know, can be intimidating
• Your questions and explanations are critical for fellow students' learning
• If you have a question, it’s likely that others have the same one. You're not alone!
Let’s get started with Data Structures!

Today: Abstract Data Types (ADTs), Stacks, Queues
Expectations: Basic Understanding of

• Conditionals
• Loops
• Methods
• Fundamentals of defining classes and inheritance
• Basic algorithm analysis (e.g. $O(n)$ vs $O(n^2)$ etc.)

• Arrays
• Singly linked lists
• Simple binary trees
• Recursion
• A few sorting and searching algorithms
What is a Data Structure?

• On super high level: a container for data
What is a Data Structure?

• On super high level: a container for data

• Real-world examples of containers:

What should I put my sandwich in?
The crux of this course

• Understanding your data structures and algorithms to choose the right one for the job.

• Fundamental CS skill

• After this course, I want you to be able to
  • Make good design choices
  • Justify and communicate design decisions
Terminology

• **Abstract Data Type (ADT)**
  - Mathematical description of “thing”
    - Meaning
    - Operations
  - No implementation details

• **Data structure**
  - Specific way to implement ADT
    (organization of data & family of algorithms)

---

e.g. bag:
Meaning of bag:
flexible container with an opening at the top
Some of its operations:
open, close, insert, take out

---

e.g. bag:
• Different kinds of bags: with handles, without handles, with clasp, with drawstring, with zip-lock, etc.
• Is one kind of bag the best?
Terminology

• **Algorithm**
  • Language-independent description of step-by-step process

• **Implementation** of a data structure
  • Specific implementation in a specific language

---

e.g. Algorithm for closing a zip-lock bag
1. Bring ends of opening together
2. Press one end
3. Run hand along top to seal

---

e.g. bag:
• Can implement in paper, plastic, canvas, leather, etc.
Terminology

Application Programmer Interface (API): Implementation of an ADT in particular language
Computer Science example: Stacks!
Stack ADT

• Meaning

LIFO (last-in-first-out)

• Operations

-push() - constructor
-pop() - size
-peek - isEmpty
Stack data structures

• Specific kinds of stacks:
  
  • Stacks using Arrays
  
  • Stacks using Linked-lists

• Example implementation: library “java.util.Stack”
Stack Practice!

1. new Stack
2. push(☺)
3. push(☆)
4. pop()

• As an array

• As a linked list
Stacks are used a lot!

- Undo / redo
- Back / forward on browsers
- Recursion
- Matching braces

\[
\{ ( (a + b) \times c - (d / (e + f)) ) \}
\]

- ... and much more!
Another example: Queues!
Queue ADT

• Meaning
  FIFO (first-in-first-out)

• Operations
  - enqueue
  - dequeue
  - isEmpty
  - size
Queue Data Structure: Linked List

enqueue(f)
dequeue()
Queue Data Structure: Linked List

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

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

• What if queue is empty?
  • Enqueue?
  • Dequeue?

• Can you find the k\text{th} element in the queue?

• Can list be full?

• How to test for empty?

• What is the complexity of the operations?
Queue Data Structure: Array

What happens when we dequeue several times, and *front* catches up to *rear*?
Queue Data Structure: Array

Hmmm...
How do we enqueue to the rear now?
Queue Data Structure: Circular Array!

View the array as *circular* and allow both *front* and *rear* to advance through (around) the array.

We wouldn’t need to move elements for enqueues and dequeues!
If we can assume the queue is not empty, how can we implement dequeue()? 

```java
public E dequeue() {
    size--;
    E e = array[front];
    //Your code here!
    return e;
}
```

A) `front++;`
   `if (front == array.length)`
   `front = 0;`

B) `rear = rear-1;`
   `if (rear < 0)`
   `rear = array.length-1;`

C) `for (int i = 0; i < rear; i++) {
    array[i] = array[i+1]
} 
front++;`
   `if (front == array.length)`
   `front = 0;`

D) None of these are correct