CSE 332: Data Structures

Spring 2016
Richard Anderson
Lecture 1

CSE 332 Team

• Instructors: Richard Anderson
  – anderson at cs
• TAs: Hunter Zahn, Andrew Li
  – hzahn93 at cs
  – lia4 at cs

Today’s Outline

• Introductions
• Administrative Info
• What is this course about?
• Review: queues and stacks

Course Information

http://www.cs.washington.edu/332
(or buy 2nd edition—1/3 price on Amazon!)

Written homeworaks

Written homeworks (8 total)
  – Assigned weekly
  – Due at the start of class on due date
  – No late homeworks accepted

Communication

Staff
  – cse332-staff@cs.washington.edu
  – (or our individual addresses)

Announcements
  – cse332a_sp16@u
  – (you are automatically subscribed @u)
Projects

- Programming projects (3 total, some with phases)
  - In Java
  - Eclipse encouraged
  - Turned in electronically
  - Work on individually
  - Start work early
    - You have two to three weeks on the projects
    - They are going to be very hard to get done in two to three days
  - Issue to watch out for: Java generics

Overall grading

Grading

- 20% - Written Homework Assignments
- 30% - Programming Assignments
- 20% - Midterm Exam (Apr 29)
- 30% - Final Exam (June 6, 2:30-4:20 pm)

Collaboration

- HWs and Projects must be done solo
  - But you can discuss problems with others as long as you follow the Gilligan’s island rule

Section

Meet on Thursdays
What happens there?
- Answer questions about current homework
- Previous homeworks returned and discussed
- Discuss the project (getting started, getting through it, answering questions)
- Finer points of Java, eclipse, etc.
- Reinforce lecture material

Homework for Today!!

Reading in Weiss
Chapter 1 – (Review) Mathematics and Java
Chapter 2 – (Next lecture) Algorithm Analysis
Chapter 3 – (Project #1) Lists, Stacks, & Queues
Today's Outline

• Introductions
• Administrative Info
• What is this course about?
• Review: Queues and stacks

Common tasks

• Many possible solutions
  – Choice of algorithm, data structures matters
  – What properties do we want?

Why should we care?

• Computers are getting faster
  › No need to optimize

• Libraries: experts have done it for you

Program Abstraction

Problem defn:

Algorithm:

Implementation:

Data Abstraction

Abstract Data Type (ADT):

Data Structure:

Implementation:

Terminology

• Abstract Data Type (ADT)
  – Mathematical description of an object with set of operations on the object. Useful building block.
• Algorithm
  – A high level, language-independent, description of a step-by-step process.
• Data structure
  – A specific organization of the data to accompany algorithms for an abstract data type.
• Implementation of data structure
  – A specific implementation in a specific language.
A starting problem: Prefix Sum

• Input: Array arr of size n
• Methods:
  – arr.sum(i) – find the sum of arr[0]...arr[i]
  – arr.update(i, value) – update arr[i] to value

Solutions

• Naïve
  – arr.sum(i): Loop through and add values
  – arr.update(i, value): arr[i] = value;

• Prefix array
  – Compute pre[i] = arr[0] + ... + arr[i] for all i
  – arr.sum(i): return pre[i]
  – arr.update(i, value): recompute prefix array

Examples

• Naïve:
  
  | 1 | 3 | -4 | 6 | 3 | 5 | -4 | 9 |

• Prefix Array:
  
  | 1 | 3 | -4 | 6 | 3 | 5 | -4 | 9 |
  | 1 | 4 | 0 | 6 | 8 | 14 | 10 | 19 |

Better solution: Tree of partial sums

| 1 | 3 | -4 | 6 | 3 | 5 | -4 | 9 |

Sum and Update in O(log n) time

| 1 | 3 | -4 | 6 | 3 | 5 | -4 | 9 |

Today’s Outline

• Introductions
• Administrative Info
• What is this course about?
• Review: queues and stacks
First Example: Queue ADT

- FIFO: First In First Out
- Queue operations
  - create
  - destroy
  - enqueue
  - dequeue
  - is_empty

```
G enque F E D C B
dequeue A
d
```

Queues in practice

- Print jobs
- File serving
- Phone calls and operators

(Later, we will consider "priority queues.")

Array Queue Data Structure

```
Q back

enqueue(Object x) {
    Q[back] = x
    back = (back + 1)
}
dequeue() {
    x = Q[0]
    shiftLeftOne()
    back = (back - 1)
    return x
}
```

Circular Array Queue Data Structure

```
Q size - 1
front back
enqueue(Object x) {
    assert(!is_full())
    Q[back] = x
    back = (back + 1)
}
dequeue() {
    assert(!is_empty())
    x = Q[front]
    front = (front + 1)
    return x
}
```

Linked List Queue Data Structure

```
front back
void enqueue(Object x) {
    if (is_empty())
        front = back = new Node(x)
    else {
        back->next = new Node(x)
        back = back->next
    }
}
bool is_empty() {
    return front == null
}
```

Circular Array vs. Linked List

- Advantages of circular array?

- Advantages of linked list?
Second Example: Stack ADT

- LIFO: Last In First Out
- Stack operations
  - create
  - destroy
  - push
  - pop
  - top
  - is_empty

Stacks in Practice

- Function call stack
- Removing recursion
- Balancing symbols (parentheses)
- Evaluating postfix or “reverse Polish” notation

Assigned readings

**Reading** in Weiss
Chapter 1 – (Review) Mathematics and Java
Chapter 2 – (Next lecture) Algorithm Analysis
Chapter 3 – (Project #1) Lists, Stacks, & Queues