Staff

• Instructor
  – Dan Suciu suciu@cs.washington.edu

• TAs
  – Sean Shih-Yen Liu syslu@cs.washington.edu
  – Saptarshi Bhattacharya saptarshi6@hotmail.com
  – Patrick Healy healy77@cs.washington.edu
Web Page

• All info is on the web

• Lots of stuff already there, including Homework 1, all midterm dates, and all homework due dates

• Future topics are tentative
Office Hours

• Dan Suciu – 662 CSE (Allen Center)
  – Monday 2pm-3pm, or by appointment
• Sean Shih-Yen Liu
  – Tues & Thurs 11am-12pm, CSE 220
• Saptarshi Bhattacharyya
  – Thurs 10-11am, CSE 220
• Patrick Healy
  – Wed 1:30-2:30pm, CSE 218
CSE 373 E-mail List

• If you are registered:
  – You will be automatically registered.

• Otherwise:
  – Subscribe by going to the class web page

• E-mail list is used for posting important announcements by instructor and TAs

• You may post too, but are responsible for anything sent here
CSE 373 Discussion Board

• There is a Catalyst e-post message board
• Use
  – General discussion of class contents
  – Hints and ideas about assignments (but not detailed code or solutions)
  – Other topics related to the course
Computer Lab

• College of Arts & Sciences Instructional Computing Lab

• Personal computer highly recommended

• Programming language: Java 5
  – Java 6 is also fine
  – Java 1.4 is ok for some things, but we will use generics which were introduced in Java 5.0
Programming Tools

• Eclipse, DrJava, Textpad, whatever…
  – Also may need JavaDoc, JUnit, which are easy to access from most tools

• We’re not religious about this as long as your code is standard Java
  – But stay away from code-generating “wizards”

• Sun Java for Windows/Linux, Java 5 for OS X, and most tools are freely available on the web – easy to set up at home
Textbook

Grading Breakdown

• Three midterms 50% (15%+15%+20%)
  – Oct.23, Nov.16, Dec.11

• Five assignments 50%
  – Three mini-projects, two write-ups, due on:
    – Oct. 8, Oct. 17, Nov. 5, Nov. 13, Dec. 3
  – Assignment 1 is posted!

• No final (replaced by the 3rd midterm)
Deadlines & Late Policy

• Exact times and dates will be given for each assignment

• Late policy: NONE

• Well, actually there is one, but you don’t want to use it:
  – 25% off 1\textsuperscript{st} day, 50% off 2\textsuperscript{nd} day, 75% off 3\textsuperscript{rd} day, 100% off 4\textsuperscript{th} day
Academic (Mis-)Conduct

• You are expected to do your own work
  – Group work, if any, will be clearly announced

• Sharing solutions, doing work for or accepting work from others will be penalized

• Integrity is a fundamental principle in the academic world (and elsewhere) – we and your classmates trust you; don’t abuse that trust
Homework for Friday !!

• Reading in Weiss (see next slide)
  – For Friday and Monday

• Assignment #1: (posted)
  – The sound blast problem (see second next slide)
Reading

• Reading in Data Structures and Algorithm Analysis in Java, by Weiss

• Read by Friday:
  – Chapters 3.1, 3.2, 3.3

• Read by Monday:
  – Chapter 1 – Mathematics and Java
  – Chapter 3 – (whole thing) Lists, Stacks, & Queues
  – Chapter 2 – Algorithm Analysis
Assignment 1 – Sound Blaster!

Play your favorite song in reverse!

**Aim:**
1. Implement stack ADT two different ways
2. Use to reverse a sound file

**Due:** Thursday, October 8, 11:45pm
Example

• Towers of Hanoi:

• Question to class: how long will it run for
  n =  3
  n = 10
  n = 20
  n = 50
  n = 100
Example

• How many moves do we need to solve it for some n?

\[ T(1) = 1 \]
\[ T(n) = T(n-1) + 1 + T(n-1) = 2 \cdot T(n-1) + 1 \]

**Proof:** let \( S(n) = T(n) + 1 \)

Then \( S(n) = 2 \cdot S(n-1) \)

Hence \( S(n) = 2^n \cdot S(0) = 2^n \)

(because \( T(0)=0 \), thus \( S(0)=1 \))

• We know \( T(3) = 7 \) runs in \( \approx 1 \text{ sec.} \), hence

<table>
<thead>
<tr>
<th>n</th>
<th>T(n)</th>
<th>Time (approx)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>( 2^{10} \approx 1000 )</td>
<td>140 seconds</td>
</tr>
<tr>
<td>20</td>
<td>( 2^{20} \approx 1 \text{M} )</td>
<td>38 hours</td>
</tr>
<tr>
<td>50</td>
<td>( 2^{50} \approx 10^{15} )</td>
<td>31 million years</td>
</tr>
<tr>
<td>100</td>
<td>( 2^{100} \approx 10^{30} )</td>
<td>( 3 \cdot 10^{22} ) years (only ( 10^{10} ) years since Big Bang)</td>
</tr>
</tbody>
</table>
Asymptotic Complexity

Consider a program that does something useful

• **Input:** n data items
  – a[0], a[1], …, a[n-1]

• **Computes:** crunch, crunch, crunch…
  – Time = T(n)

• **Output:** prints some result

Asymptotic complexity of T(n) (next slide).
Asymptotic Complexity

- This course is about $O(f(n))$

**DEFINITION**: The Big-O notation

$T(n) = O(f(n))$ if there exist constants $c$ and $n'$ such that: $T(n) \leq c f(n)$ for all $n \geq n'$

Example: Towers of Hanoi has $T(n) = O(2^n)$
Examples

Given n items a[0], a[1], . . . , a[n-1]

• Compute & print a[0]+a[1]+…+a[n-1]
  – T(n) = O(n)

• Compute & print (a[0] + a[n/2] + a[n-1]) / 3
  – T(n) = O(1)

• Print all permutations a[i[0]], a[i[1]],…a[i[n-1]]
  – T(n) = O(n!) = O(2^{n\cdot\log(n)}) (why?)
## Quiz: Match problem with Big O

<table>
<thead>
<tr>
<th>Problem Description</th>
<th>Big O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute a[0]<em>a[1]</em>…*a[n-1]</td>
<td>T(n) = O(1)</td>
</tr>
<tr>
<td>Remove duplicates from a[0], a[1], a[2], …, a[n-1]</td>
<td>T(n) = O(n)</td>
</tr>
<tr>
<td>Return a[n/3]</td>
<td>T(n) = O(n^2)</td>
</tr>
<tr>
<td>Find k ≥ 0, and positions i[0], i[1], …, i[k-1] such that: a[i[0]] + a[i[1]] +…+ a[i[k-1]] = n</td>
<td>T(n) = O(2^n)</td>
</tr>
</tbody>
</table>
Answers…

- Compute $a[0]*a[1]*…*a[n-1]$  \[ T(n) = O(1) \]
- Remove duplicates from $a[0], a[1], a[2], \ldots, a[n-1]$  \[ T(n) = O(n) \]
- Return $a[n/3]$  \[ T(n) = O(n^2) \]
- Find $k \geq 0$, and positions $i[0], i[1], \ldots, i[k-1]$ such that: $a[i[0]] + a[i[1]] + \ldots + a[i[k-1]] = n$  \[ T(n) = O(2^n) \]

Don’t worry yet! We will study the Big O in detail, starting on Monday.
The Brainy Hacker

• This course is not about better coding, why take it?

I don’t need 373 because:
• I’ll buy a faster laptop
• I’ll write clever code
The Apocalyptic Laptop

Seth Lloyd, SCIENCE, 31 Aug 2000

- A computer as powerful as the laws of physics will allow
- So energetic, like harnessing a thermonuclear reaction.
- Packed into so small a space that the whole thing would collapse and form a tiny black hole
The Apocalyptic Laptop

Seth Lloyd, SCIENCE, 31 Aug 2000

$5.4 \times 10^{50}$ operations per second

(Typical laptop today: $10^9$ operations per second)
What a Better Laptop Buys You

A lot for the good Big O’s, not much for the bad O’s
What Better Coding Buys You

1 year
1 second

Ultimate Laptop,

Since
Big Bang
1 day

Today’s laptop
(1000MIPS)

Ten times faster buys you...
...nothing (for a bad Big O)
A First Hurdle: Java

Public class Set_of ints {
    Public void insert( int x );
    Public void remove( int x ); ... }

Review the syntax
Review java, eclipse, java editor...

You’ll need all that for assignment 1, which is a mini-project, due in one week!
Java Resources

• See webpage for pointers

• Handy Library, from Weiss:
  http://www.cs.fiu.edu/~weiss/dsaajava/code/
Abstract Data Types

Abstract Data Type (ADT)
Mathematical description of an object and the set of operations on the object

Data Types
integer, array, pointers, ...

Algorithms
binary search, quicksort, ...

tradeoffs!
ADT Presentation Algorithm

• Present an ADT
• Motivate with some applications
• Repeat until it’s time to move on:
  – develop a data structure and algorithms for the ADT
  – analyze its properties
    • efficiency
    • correctness
    • limitations
    • ease of programming
• Contrast strengths and weaknesses
First Example: Queue ADT

• Queue operations
  – create
  – destroy
  – enqueue
  – dequeue
  – is_empty

• Queue property:
  – if x is enQed before y is enQed, then x will be deQed before y is deQed

• FIFO: First In First Out

Applications of the Q

• Hold jobs for a printer
• Store packets on network routers
• Make waitlists fair
• Breadth first search
Circular Array Q Data Structure

enqueue(Object x) {
    Q[back] = x ;
    back = (back + 1) % size 
}

dequeue() {
    x = Q[front] ;
    front = (front + 1) % size;
    return x ; }

How test for empty list?

How to find K-th element in the queue?

What is complexity of these operations?

Limitations of this structure?
enqueue(Object x) {
    back.next = new Node(x);
    back = back.next;
}

decqueue() {
    saved = front.data;
    temp = front;
    front = front.next;
    return saved;
}

What are tradeoffs?
- simplicity
- speed
- robustness
- memory usage