1: Introduction

CSE326 Spring 2002

April 1, 2002

--- Administrivia ---

- **Instructor:**
  Matthew Cary
  226c Sieg Hall
  cary@cs
  Office Hours: MW 11:30-12:30

- **TAs:**
  Nick Deibel  jdeibel@cs
  Kevin Chan   kachan@cs

- **Text:**
  *Data Structures and Their Algorithms*, Lewis and Denenberg.

--- What 326 is About ---

- **Formal Study of Algorithms and Data Structures**
  - ADT: Abstract Data Type
  - Asymptotic Running-time Analysis

- **Become familiar with UNIX development environment**
  - C++
  - Linux
  - g++, make, gdb, ddd, emacs
Policies

- Written homework due at start of class on the due date.

- Programming assignments
  - 1 day late: 10% off score

Grading

- Written homework: 15%
- Programming assignments: 30%
- Midterm: 20%, May 8
- Final: 30%, June 12
- etc.: 5%

Mechanics

- Home Page: www/326
  - Slides of current lectures

- Directory: /cse/courses/cse326/02sp
  - Reachable from IWS servers

- Mailing Lists: cse326ta@cs, cse326@cs
  - Important announcements on cse326ta@cs
  - Discussion on cse326@cs

- Lab: 329 Sieg
  - All programming assignments should be done on Linux
  - Windows Boxes with Xserver access to instructional workstation servers
  - IWS servers are: fiji, sumatra, ceylon, tahiti

CSE326 & The CSE Program

UW CSE Degree

UW CSE326 Sp ’02: 1—Intro
Writing computer programs is an *engineering* activity.

- Concerned about *efficiency*.
- Need to trade-off *detail* with *abstraction*.
- Concerned about *elegance*.

---

**Course Overview**

- **Efficiency**
  - *Asymptotic run-time analysis*
  - Quantify algorithm performance independent from the machine the algorithm runs on.
- **Abstraction**
  - Encapsulate bookkeeping details of program with *Abstract Data Type*
  - Stack, Queue, Dictionary, . . .
- **Elegance**
  - We'll study elegant implementations of ADTs

---

**Course Outline**

- **Review**
- **Analysis**
  - How to compare and evaluate algorithms
  - Will learn sorting algorithms in the process
- **ADTs**
  - The common ADTs that are used to solve most computer programming problems
    - Dictionary, Priority Queue, Set . . .
  - Efficient algorithms that implement these ADTs
    - Lists, Trees, Balanced Trees, Hashes, Tries . . .
- **Applications**
  - Graphs
  - Range Searching, Geographic Databases, Compression . . .
    - Depends on how much time we have
Prerequisites: Programming

- C++?
  - Recursion?
- Arrays?
- Lists?
- Trees?
  - Binary Trees?
- Searching?
  - Binary Search? Hashes?
- Sorting?
- ADTs?

Our Model of a Computer

- Computer memory is list of cells
- Each cell has an address
- Cells hold a number or an address (i.e., a pointer)

Von Neumann Definition of a Computer

Our Model of a Computer

- Draw pointers as arrows
- \( \Lambda \) denotes a NULL pointer

\( 
\begin{array}{|c|}
\hline
1240 & 2 \\
1244 & 1260 \\
1248 & 1 \\
1252 & 1240 \\
1256 & 873 \\
1260 & 3 \\
1264 & 0 \\
\hline
\end{array}
\)
Our Model of a Computer

Memory Viewed as List

Implementation Techniques You Know

- List

```c
struct ListNode {

};
```

- Array

```
int *array;  int n;
```

ADTs

Algorithm

An Abstract Data Type is:

- A Domain, and
- a set of Operations on the domain.

Algorithms use implementations of ADTs to solve problems
ADTs You Know

Stack (LIFO)

- Domain:
- Operations:
- Implementations:

Queue (FIFO)

- Domain:
- Operations:
- Implementations:

List (Array)

- Domain:
  - Numbers
  - Strings
  - Lists...
- Operations:
  - Access(i)
  - Length()
  - Concat(L1, L2)
  - MakeEmptyList()
  - IsEmptyList()
- Implementation:
Prerequisites: Mathematics

- Methods of proof?

- $\sum_{i=1}^{n} i = ?$

- $O(n^2)$ vs. $\Omega(n \log n)$?

- Probability?
  - Expectation?

Recurrence Relations

- $T(n) = c + T(n - 1), T(0) = a$

- $T(n) = c + 2T(n - 1), T(0) = a$

- $T(n) = c + T(n/2), T(0) = a$

Rules for Logarithms

- $\log a = x \iff 2^x = a$
- $\log a + \log b = \log ab$
- $\log a - \log b = \log \frac{a}{b}$
- $\log a^c = c \cdot \log a$
- $\log_b a = \frac{\log a}{\log b}$

See §1.3 in the textbook