Lecture I1: Introduction

Adam Finkelstein
Kevin Wayne

Overview

What is COS 126?

- Broad introductory survey course.
  - no prerequisites (although previous programming helpful in beginning)
- Basic principles of computer science.
  - hardware and software systems
  - programming in C and other languages
  - algorithms and data structures
  - theory of computation
  - applications to solving scientific problems

What isn't COS 126?

- A programming course.

The Usual Suspects

Lectures: (Adam Finkelstein, Kevin Wayne)

- Tuesday, Thursday 9:00 - 9:50, McCosh 46.

Precepts: (Andrew Appel, Adam Finkelstein, Jason Perry, Kevin Wayne, Lisa Worthington)

- Friday - tips on assignments, clarify lecture material.
- Monday - review exercises, clarify lecture material.

All pre-registered students are assigned to a precept at the time requested.

- Note: there are 2 precepts at 11am, and 3 at 1:30pm.
- Check the Web page to see which one you are in and where it meets.
- We will only allow switching to under-subscribed precepts.
- See Kevin (Room 207) if you are not in a precept.

Required Readings

Course packet.

- Pequod copy.

Kernighan and Ritchie.

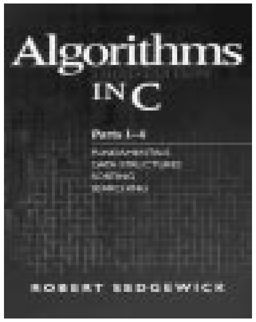
- Bible for C programmers.

Deitel and Deitel.

- "Required" C text for beginners.

Sedgewick.

- Algorithms and data structures.
- Also required in COS 226.



Lecture Outline

Programming Fundamentals (6 lectures).

Machine architecture (5 lectures).

Advanced programming (2 lectures).

Theory of computation (6 lectures).

Java (2 lectures).

Perspective (1 lecture).
- R1. Course summary.

Programming Assignments

Programming assignments (designed to illustrate scientific applications):
- 0. Hello world

Hello World!

Getting started in C and Unix.
Due Thursday, February 3 at 11:59pm.
Sign up for CS101 lab reservation in class today.
(this assignment only).

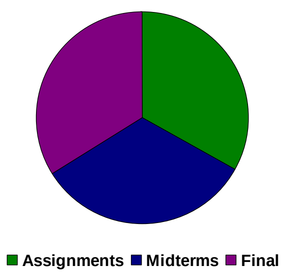
Grading

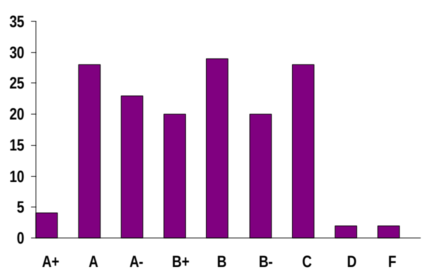
Assignments (33%).
- 9 programming assignments.
- Exercises (solutions provided).

Midterms (33%).
- 2 midterms (33% total).
- Many questions drawn from exercises.

Final (34%).

Course grades.
- No set curve.
- Last year's breakdown.





Survival Guide

Keep up with the course material.
- Attend lectures and precepts.
- Do readings when assigned.
- Do exercises and understand solutions.
- Start on programming assignments early.
- Think before you write code; compose first, then write code.

Visit course home page regularly for announcements and supplemental information:

```
courseinfo.Princeton.EDU/courses/COS126_S2000
www.Princeton.EDU/~cs126
```

- Contact Kevin (Room 207) if you aren't on course list.
- Contact CIT if you don't remember your Unix password.

Survival Guide

Keep in touch.

- Email: your preceptor, lecturers.
- Office hours: your preceptors, other preceptors, instructors.
- Discussion group on course web page.

Ask for help when you need it!

- Preceptors, instructors, lab TA's.

What Is Computer Science?

What is computer science?

1. The science of manipulation "information."
2. Designing and building systems that do (1).

Why we learn CS.

- Appreciate underlying principles.
- Understand fundamental limitations.

An example: linear feedback shift register machine.

- How to make a simple machine.
- What we can do with it.
- Science behind it.

Encryption Machine

Goal: design a machine to encrypt and decrypt data.

S	E	N	D	M	O	N	E	Y

encrypt

W	?	M	R	E	A	F	B	Z

decrypt

S	E	N	D	M	O	N	E	Y

Simple Encryption Scheme

1. Convert text input to N digit binary number.
2. Generate N random bits.
3. Take bitwise XOR of two strings.
4. Convert binary back into text.

Conversion

char	dec	binary
A	1	00001
B	2	00010
C	3	00011
. . .		
Z	26	11010

S	E	N	D	M	O	N	E	Y	message
10010	00101	01100	00100	01101	01110	01100	00101	11001	binary
00100	11001	00001	10101	01000	01111	01010	00111	00101	random bits
10110	11100	01101	10001	00101	00001	00110	00010	11100	XOR
W	?	M	R	E	A	F	B	?	send

Decryption Scheme

1. Convert encrypted message to binary.
2. Use same N random bits.
3. Take bitwise XOR of two strings.
4. Convert binary back into text.

W	?	M	R	E	A	F	B	?	message
10110	11100	01101	10001	00101	00001	00110	00010	11100	binary
00100	11001	00001	10101	01000	01111	01010	00111	00101	random bits
10010	00101	01100	00100	01101	01110	01100	00101	11001	XOR
S	E	N	D	M	O	N	E	Y	send

Why Does It Work?

Notation:

a	original message
b	random bits
^	XOR operation
a ^ b	encrypted message
(a ^ b) ^ b	decrypted message

Crucial property: (a ^ b) ^ b = a.

- Decrypted message = original message.

Why is crucial property true?

- b ^ b = 0
- a ^ 0 = a
- (x ^ y) ^ z = x ^ (y ^ z)
- (a ^ b) ^ b = a ^ (b ^ b) = a ^ 0 = a

Random Numbers

Are these 2000 numbers random?

```
0100110010000001100010001011101010111100100111100111010010000110111111110010110100110110
1100101001111110101001011000110001110110111000001101000100001010100010011100110111100111
1100000011110111011010000110101110110110000101111011100101100001001111111101101001011110
1010000111001000101100100011010100010111110111110000101101010100001010000000010101010101
1111010111100000101101110110000011100010001111111011110000110100110010100011101011101001
1100000100101010100011011000100100100011000101010011000110111101111010010010111111011000
0111110011000011101000100101000101011111110101101001001111101100101101101111100101001011
1001000000111010101011011000110101100111110100011110010001111011001111001010110101000011
0110011000000110000000000110011001101010011010111111001001011011111110100001111010101001010
0000100111011100111101001101001110100000011100110011110110111100011100001000011001110111
1011010110101100011100101010010000010101110111110100111100011000011001010101100100111000
1010110111001001001010111001110010111000000101100110010010000100110111010100111101011100
0010100100010100011001001100000010010001000000100000000000100010001001100010011010101011100
0110110101011000001010011001110011111100010110100010100111011000101100000000101110111010 1
1011010111101000001111101110100101001001101100100001011100110001111110011010010101100010
1000010000010001100110111000101111001101101000111000110010111010001101000000011011101110
0011111000100101100111010110010111110011100001100001000111011111110001111000000001111111
1110000111100010000111011001101000010010011001000000110001000101110101011110010011110011
1010010000110111111110010110100110110110010100111111010100101100011000111011011100000110
1000100001010100010011100110111100111110000000111101110110100001101011101101100001 0111101
1100101100001001111111101101001011110101000011100100010110010001101010001011111011111000
0101101010100001010000000010101010101111101011110000010110111011000000111000100011111101
11100001101001100101000111010111010011100000100101010100011011 00...
```

If not, what is the pattern?

Linear Feedback Shift Register

How might the "random number machine" be built?

- We'll investigate a simple machine called a "linear feedback shift register".

Some terminology

- Bit: a student who is either male or female (0 or 1).
- Cell (storage element): a seat that holds one student.
- Register: whole row of students.
- Shift register: when clock strikes, stand up and take seat to left.

Linear Feedback Shift Register

Linear feedback shift register.

- Machine consists of 11 bits.
- Bit values change at discrete time points.
- Bit values at time T+1 completely determined by bit values at time T.
  - new bits 1 - 10 are old bits 0 - 9
  - new bit 0 is XOR of previous bits 3 and 10
  - output bit 0

a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Time T
a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	a_3^a_{10}	Time T+1

LFBSR Demo

The Science Behind It

Are the bits really random?

How did the computer scientist die in the shower?

How long will it take the bit pattern to repeat itself.

Will the machine work equally well if we XOR bits 4 and 10?

How many cells do I need to guarantee a certain level of security?

Properties of Shift Register "Machine"

Clocked.

Control: start, stop, load.

Data: initial values of bits (fill).

Built from simple components.

- "Clock" (regular electrical pulse).
- Shift register cell remembers value until clock "ticks."
- Some wires "input", some "output."

Scales to handle huge problems.

- 10 cells yields 1 thousand "random" bits.
- 20 cells yields 1 million "random" bits.
- 30 cells yields 1 billion "random" bits.
- BUT, need to understand abstract machine!
  - higher math needed to know XOR taps

Properties of Computers

Same basic principles as LFBSR:

- Clocked.
- Control: start, stop, load.
- Data: initial values of bits.
- Built from simple components.
- Scales to handle huge problems.

Abstraction aids in understanding.

Simulating The Abstract Machine

C program to produce “random” bits.

sum.c

```
#include <stdio.h>
#define N 100

int main(void) {
  int i, new, fill = 01502;
  for (i = 0; i < N; i++) {
    new = ((fill >> 10) & 1) ^ ((fill >> 3) & 1);
    fill = (fill << 1) + new;
    printf(“%d\n”, new);
  }
  return 0;
}
```

You’ll understand this program by next week!

>>	shift right	&	“and” (1 if both bits 1, 0 otherwise)
<<	shift left	^	“exclusive or” (1 if both bits are different)

Simulating The Abstract Machine

C program to produce “random” bits.

Any “general purpose” machine can be used to simulate any abstract machine. Implications are:

- Test out new programs.
- Use old programs.
- Understand fundamental limitations of computers.

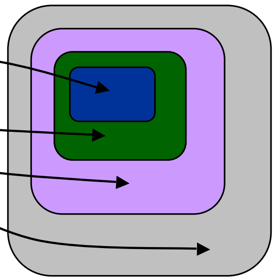
Layers of Abstraction

Layers of abstraction (recurring theme).

- Precisely defined for simple machine.
- Use it to build more complex one.
- Develop complex systems by building increasingly more complicated machines.
- Improve systems by substituting new (better) implementations of abstract machines at any level.

LFBSR layers of abstraction.

- Simple piece of hardware.
- Generate “random” bits.
- Use “random” bits for encryption.
- Use encryption for Internet commerce.



Layers of Abstraction

“Computer” layers of abstraction.

- Complex piece of hardware.
  - CPU, keyboard, printer, storage devices
- Machine language programming.
  - 0’s and 1’s
- Software systems.
  - editor (emacs): create, modify files
  - compiler (lcc): transform program to machine instruction
  - operating system (Unix): invoke programs
- Windowing system (X).
  - illusion of multiple computer systems