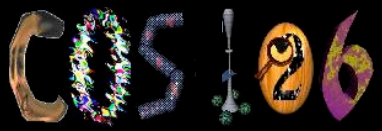


General Computer Science †
Princeton University
Fall 2008

Douglas Clark

† Some lectures overlap with ISC/CHM/COS/MOL/PHY 231/2

Overview

What is COS 126?
- Broad, but technical, intro to CS.
- No prerequisites, intended for novices.

Goals.
- Demystify computer systems.
- Empower you to exploit available technology.
- Build awareness of substantial intellectual underpinnings.

Topics.
- Programming in Java.
- Machine architecture.
- Theory of computation.
- Applications to science, engineering, and commercial computing.

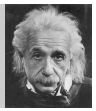
Overview

What is COS 126? Broad, but technical, intro to CS.

Goals.
- Demystify computer systems.
- Empower you to exploit available technology.
- Build awareness of substantial intellectual underpinnings.

Topics.
- Programming in Java.
- Machine architecture.
- Theory of computation.
- Applications to science, engineering, and commercial computing.

> *" Computers are incredibly fast, accurate, and stupid; Humans are incredibly slow, inaccurate, and brilliant; together they are powerful beyond imagination. "*



The Usual Suspects

Lectures. [Doug Clark]
- Tuesdays and Thursdays, Friend 101.

Precepts. [David Blei · Will Clarkson · Tom Funkhouser · Donna Gabai · Maia Ginsburg · Michael Golightly · Wyatt Lloyd · Anna Pop · Sid Sen · Jeff Terrace · Tao Yue] (lead preceptors)
- COS 126: Tue+Thu or Wed+Fri, various times (and places).
- ISC/CHM/COS/MOL/PHY 231/2: Wed or Th, 1:30, Icahn Lab 200.
  - (Wed people please follow instructions from HQ this week)
- Tips on assignments, worked examples, clarify lecture material.

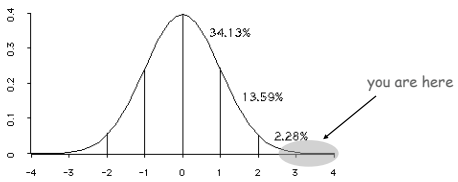
Friend 016/017 lab. [Undergrad lab assistants]
- Weekdays 7-11pm, some weekend hours.
- Full schedule on Web.

For full details: See **www.princeton.edu/~cos126**

Grades

Course grades. No preset curve or quota.

9 programming assignments. 40%.
2 exams. 50%. ← can drop lowest one
Final programming project. 10%.
Extra credit and staff discretion. Adjust borderline cases. ← participation helps, frequent absences hurts



Course Materials

Course website. [www.princeton.edu/~cos126]

- Submit assignments, check grades.
- Programming assignments.
- Lecture notes. ← print slides before lecture; annotate during lecture

at least skim before lecture; read thoroughly afterwards

Required readings. Sedgewick and Wayne. *Intro to Programming in Java: An Interdisciplinary Approach.* [Labyrinth Books]

Recommended. Harel. *Computers Ltd.: What they really can't do.* [Labyrinth]

Programming Assignments

Desiderata.

- Address an important scientific or commercial problem.
- Illustrate the importance of a fundamental CS concept.

Examples.

- N-body simulation.
- Pluck a guitar string.
- DNA sequence alignment.
- Estimate Avogadro's number.

} you solve scientific problems from scratch!

Due. (Usually) Mondays 11 pm via Web submission.

Computing equipment.

- Your laptop. [OS X, Windows, Linux, iPhone, …]
- OIT desktop. [Friend 016 and 017 labs]

Survival Guide

Keep up with the course material.

- Attend lectures and precepts.
- Take notes.
- *At least* skim readings *before* lecture; read more closely after.
- Plan multiple sessions for programming assignments.
- Visit course home page regularly.

Ask for help when you need it!

- Preceptors / instructor.
  - email, office hours, precepts
  - concepts, programming assignments
- Undergrad Lab TAs.
  - OS support, help with minor debugging

What's Ahead?

Tuesday. Lecture 2: Intro to Java.

COS 126 Precept 1. Meets today or tomorrow.
ICS/CHM/COS/MOL/PHY 231/2. Meets today and/or tomorrow

Not registered? Go to any precept today; officially register ASAP.
Change precepts? Use SCORE.

see Donna O'Leary in CS 410
if the only precept you can attend is closed

ASAP! Read sections 1.1 – 1.2 of Intro to Programming.

Assignment 0. Due Monday 9/15, 11 pm.

- Install Java programming environment + a few exercises.
- Lots of help available, don't be bashful.

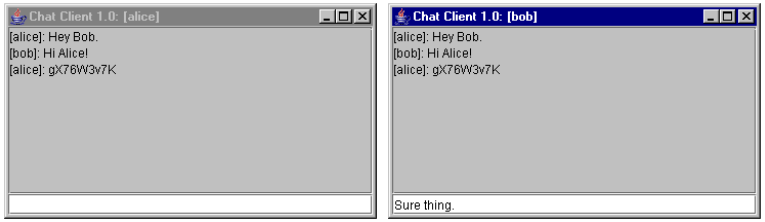
END OF ADMINISTRATIVE STUFF

Prologue: A Simple Machine

Secure Chat

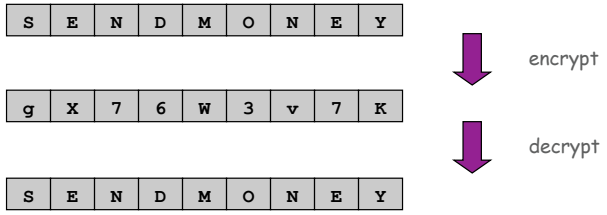
Alice wants to send a secret message to Bob?

- Can you read the secret message `gX76W3v7K` ?
- But Bob can. How?



Encryption Machine

Goal. Design a machine to encrypt and decrypt data.



Encryption Machine

Goal. Design a machine to encrypt and decrypt data.

S	E	N	D	M	O	N	E	Y

encrypt

g	X	7	6	W	3	v	7	K

decrypt

S	E	N	D	M	O	N	E	Y

Enigma encryption machine.

- "Unbreakable" German code during WWII.
- Broken by Turing bombe.
- One of first uses of computers.
- Helped win Battle of Atlantic by locating U-boats.

A Digital World

Data is a sequence of bits. (0 or 1)

- Text.
- Documents, pictures, sounds, movies, ...
- Programs, executables.

File formats. txt, pdf, doc, ppt, jpeg, mp3, divx, java, exe, ...

computer with a lens

computer with earbuds

computer with a radio

A Digital World

Data is a sequence of bits. (0 or 1)

- Text.
- Documents, pictures, sounds, movies, ...
- Programs, executables.

File formats. txt, pdf, doc, ppt, jpeg, mp3, divx, java, exe, ...

computer with a cash dispenser

computer with a ballot box

computer with a heating element

A Digital World

Data is a sequence of bits. (0 or 1)

- Text.
- Documents, pictures, sounds, movies, ...
- Programs, executables.

Base64 encoding. Use 6 bits to represent each alphanumeric symbol.

Binary	Char	Binary	Char	Binary	Char	Binary	Char	Binary	Char	Binary	Char
000000	A	001011	L	010110	W	100001	h	101100	s	110111	3
000001	B	001100	M	010111	X	100010	i	101101	t	111000	4
000010	C	001101	N	011000	Y	100011	j	101110	u	111001	5
000011	D	001110	O	011001	Z	100100	k	101111	v	111010	6
000100	E	001111	P	011010	a	100101	l	110000	w	111011	7
000101	F	010000	Q	011011	b	100110	m	110001	x	111100	8
000110	G	010001	R	011100	c	100111	n	110010	y	111101	9
000111	H	010010	S	011101	d	101000	o	110011	z	111110	+
001000	I	010011	T	011110	e	101001	p	110100	0	111111	/
001001	J	010100	U	011111	f	101010	q	110101	1		
001010	K	010101	V	100000	g	101011	r	110110	2		

One-Time Pad Encryption

Encryption.

- Convert text message to N bits.

0 or 1

Base64 Encoding

char	dec	binary
A	0	000000
B	1	000001
...	...	...
Y	24	011000
...	...	...

S	E	N	D	M	O	N	E	Y	
010010	000100	001101	000011	001100	001110	001101	000100	011000	base64

message

One-Time Pad Encryption

Encryption.

- Convert text message to N bits.
- Generate N random bits (one-time pad).

S	E	N	D	M	O	N	E	Y	
010010	000100	001101	000011	001100	001110	001101	000100	011000	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits

message

One-Time Pad Encryption

Encryption.

- Convert text message to N bits.
- Generate N random bits (one-time pad).
- Take bitwise XOR of two bitstrings.

sum corresponding pair of bits: 1 if sum is odd, 0 if even

XOR Truth Table

x	y	x ^ y
0	0	0
0	1	1
1	0	1
1	1	0

S	E	N	D	M	O	N	E	Y	
010010	000100	001101	000011	001100	001110	001101	000100	011000	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits
100000	010111	111011	111010	010110	110111	101111	111011	001010	XOR

message

0 ^ 1 = 1

One-Time Pad Encryption

Encryption.

- Convert text message to N bits.
- Generate N random bits (one-time pad).
- Take bitwise XOR of two bitstrings.
- Convert binary back into text.

Base64 Encoding

char	dec	binary
A	0	000000
B	1	000001
...	...	...
X	23	010111
...	...	...

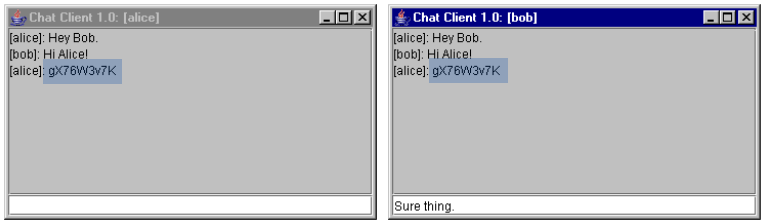
S	E	N	D	M	O	N	E	Y	
010010	000100	001101	000011	001100	001110	001101	000100	011000	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits
100000	010111	111011	111010	010110	110111	101111	111011	001010	XOR
g	X	7	6	W	3	v	7	K	encrypted

message

Secure Chat

Alice wants to send a secret message to Bob?

- Can you read the secret message `gX76W3v7K` ?
- But Bob can. How?



One-Time Pad Decryption

Decryption.

- Convert encrypted message to binary.

g	X	7	6	W	3	v	7	K	encrypted

One-Time Pad Decryption

Decryption.

- Convert encrypted message to binary.

Base64 Encoding

char	dec	binary
A	0	000000
B	1	000001
...	...	...
X	23	010111
...	...	...

g	X	7	6	W	3	v	7	K	encrypted
100000	010111	111011	111010	010110	110111	101111	111011	001010	base64

One-Time Pad Decryption

Decryption.

- Convert encrypted message to binary.
- Use same N random bits (one-time pad).

g	X	7	6	W	3	v	7	K	encrypted
100000	010111	111011	111010	010110	110111	101111	111011	001010	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits

One-Time Pad Decryption

Decryption.

- Convert encrypted message to binary.
- Use same N random bits (one-time pad).
- Take bitwise XOR of two bitstrings.

XOR Truth Table

x	y	x ^ y
0	0	0
0	1	1
1	0	1
1	1	0

g	X	7	6	W	3	v	7	K	
100000	010111	111011	111010	010110	110111	101111	111011	001010	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits
010010	000100	001101	000011	001100	001110	001101	000100	011000	XOR

One-Time Pad Decryption

Decryption.

- Convert encrypted message to binary.
- Use same N random bits (one-time pad).
- Take bitwise XOR of two bitstrings.
- Convert back into text.

Base64 Encoding

char	dec	binary
A	0	000000
B	1	000001
…	…	…
Y	24	011000
…	…	…

g	X	7	6	W	3	v	7	K	
100000	010111	111011	111010	010110	110111	101111	111011	001010	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits
010010	000100	001101	000011	001100	001110	001101	000100	011000	XOR
S	E	N	D	M	O	N	E	Y	message

Why Does It Work?

Crucial property. Decrypted message = original message.

Notation	Meaning
a	original message bit
b	one-time pad bit
^	XOR operator
a ^ b	encrypted message bit
(a ^ b) ^ b	decrypted message bit

Why is crucial property true?

- Use properties of XOR.
- (a ^ b) ^ b = a ^ (b ^ b) = a ^ 0 = a

associativity of ^ · always 0 · identity

XOR Truth Table

x	y	x ^ y
0	0	0
0	1	1
1	0	1
1	1	0

Goods and Bads of One-Time Pads

Good.

- Very simple encryption/decryption processes.
- Provably unbreakable if pad is truly random. [Shannon, 1940s]

eavesdropper Eve sees only random bits

"one time" means one time only

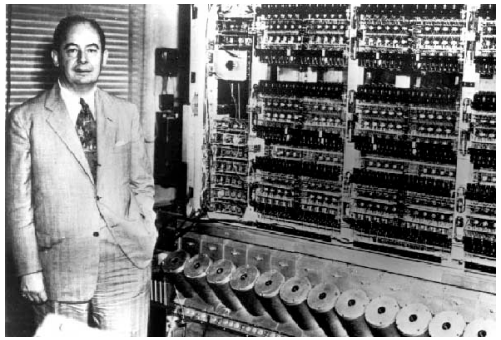
Bad.

- Easily breakable if pad is re-used.
- Pad must be as long as the message.
- Truly random bits are very hard to come by.
- Pad must be distributed securely.

impractical for Web commerce

Random Numbers

> Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.



Jon von Neumann (left), ENIAC (right)

Pseudo-Random Bit Generator

Practical middle-ground.

- Let's make a pseudo-random bit generator gadget.
- Alice and Bob each get identical small gadgets.

instead of identical large one-time pads

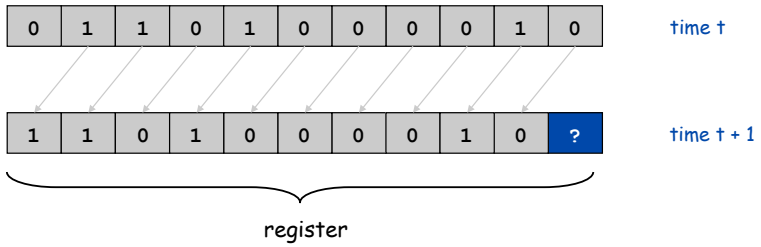
How to make small gadget that produces "random" numbers.

- Linear feedback shift register.
- Linear congruential generator.
- Blum-Blum-Shub generator.
- ...

Shift Register

Shift register terminology.

- Bit: 0 or 1.
- Cell: storage element that holds one bit.
- Register: sequence of cells.
- Seed: initial sequence of bits.
- Shift register: when clock ticks, bits propagate one position to left.

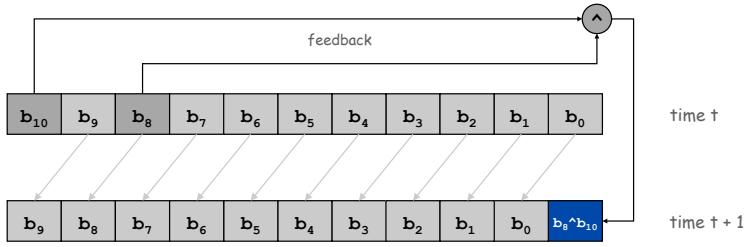


Linear Feedback Shift Register

{8, 10} linear feedback shift register.

- 11 bit shift register.
- New output bit 0 is XOR of previous bits 8 and 10.
- Output bit = bit 0.

LFSR demo



Random Numbers

Q. Are these 2000 numbers random? If not, what is the pattern?

```
11001001001111011011100101101011100110001011111101001000010011010010111100110010011111
11011100000101011000100001110101001101000011110010011001110111111101010000010000100010
10010101000110000010111100010010011010110111100011010011011100111101011110010001001110
10101110100000101001000100011010101011100000001011000001001110001011101101001010110011
00001111111001100000111111000110000110111100111010011110100111001001110111011101010101
01000000000001000000001010000000100010000101010100100000000110100000111001000110111010111
01010001010000101000100100010101101010000110000100111100101110011100101111011100100101
01110110000101011100100001011101001001010011011000111101110110010101011110000001001100
00101111100100100011101101011010110001100011101111011010100101100001100111001111110111
10000101001100100011111101011000010001110010101101110000110101100111000111110110110001
01101110100110101001111000011100110011011111111101000000010010000010110100010011001010
11111100001000011001010011111000111000110110110111011011010101101100000110111000111010
11011010001101100101110111100101010011100000111011000110101110111000101010110100000011
00100001111101001100010011111010111000100010110101010100110000000111110000110001100111101
11111001010000111000100110110101111011000100101110101100101000111100010110011010011111
10011100001111011001100101111111100100000011101000011010010011100110111011111010101000
10000001010100001000000100101000101100010100111010001110100101101001100110011111111110
000000001100000001111000001100110001111111101100000001011100001001011001011001111001111
10011110001111001101100111110111110001010001101000101110010100101110001100101101111100
110100011111001011000111001110110111101011010010001100110101111111100010000011010100011
10000101101100100110111101111010010100100110001101111101110100010101001010000011000100
01111010101100100000111101000110010010111110110010001011110101001001000011011010011101
10011101011111010001000100101010101100000000111000000110110000111011100110101011111000
0010001100010101111010
```

A. No. This is output of an 11 bit LFSR!

The Science Behind A LFSR

Q. Are the bits really random?
A. No! Real machines are deterministic.

Q. Will bit pattern repeat itself?
A. Yes, after $2^{11} - 1 =$ 2047 steps.

Q. What if I need more bits?
A. Scalable: 20 cells for 1 million bits, 30 for 1 billion.

Q. Will the machine work equally well if we XOR bits 4 and 10?
A. No! Need to understand theory of finite groups.

Q. How many cells do I need to guarantee a certain level of security?
A. Subject of active research.

Other LFSR Applications

What else can we do with a LFSR?

- DVD encryption with CSS.
- DVD decryption with DeCSS!
- Subroutine in military cryptosystems.

```
/*     efdtt.c     Author:  Charles M. Hannum <root@ihack.net>          */
/*     Usage is:  cat title-key scrambled.vob | efdtt >clear.vob        */

#define m(i)(x[i]^s[i+84])<<

                   unsigned char x[5]      ,y,s[2048];main(
                   n){for( read(0,x,5     );read(0,s ,n=2048
                            ); write(1   ,s,n)         )if(s
                  [y=s       [13]%8+20]  /16%4  ==1       ){int
                  i=m(       1)17  ^256 +m(0)   8,k       =m(2)
                  0,j=       m(4)   17^ m(3)   9^k*      2-k%8
                  ^8,a      =0,c    =26;for   (s[y]    -=16;
                 --c;j  *=2)a=      a*2^i&    1,i=i /2^j&1
                 <<24;for(j=         127;     ++j<n;c=c>
                                     y)
                                     c

                          +=y=i^i/8^i>>4^i>>12,
                  i=i>>8^y<<17,a^=a>>14,y=a^a*8^a<<6,a=a
                >>8^y<<9,k=s[j],k         ="7Wo~'G_\216"[k
                 &7]+2^"cr3sfw6v;*k+>/n."[k>>4]*2^k*257/
                      8,s[j]=k^(k&k*2&34)*6^c+~y
                                  ;}}
```

http://www.cs.cmu.edu/~dst/DeCSS/Gallery

LFSR and "General Purpose Computer"

Important properties.

- Built from simple components.
- Scales to handle huge problems.
- Requires a deep understanding to use effectively.

Basic Component	LFSR	Computer
control	start, stop, load	same
clock	regular pulse	2.8 GHz pulse
memory	11 bits	1 GB
input	seed	sequence of bits
computation	shift, XOR	logic, arithmetic, …
output	pseudo-random bits	Sequence of bits

Critical difference. General purpose machine can be programmed to simulate ANY abstract machine.

Simulating The Abstract Machine in Java

A Java program.

- Prints same bits as LFSR.
- The code will make sense next week!

```
public class LFSR {
   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);
      boolean b10 = false, b9 = true,  b8 = true,  b7 = false;
      boolean b6  = true,  b5 = false, b4 = false, b3 = false;
      boolean b2  = false, b1 = true,  b0 = false;

      for (int i = 0; i < N; i++) {                    repeat N times
         boolean bit = b8 ^ b10;
         b10 = b9; b9 = b8; b8 = b7; b7 = b6; b6 = b5;
         b5  = b4; b4 = b3; b3 = b2; b2 = b1; b1 = b0;
         b0 = bit;                                          update

         if (bit) System.out.print(1);                print next bit
         else     System.out.print(0);
      }
   }
}
```

```
% java LFSR 2000
11001001001111101101110010110
10111001100010111111010010000
01001101001011110011001001111
...
```