



General Computer Science
Princeton University
Spring 2007

Kevin Wayne

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.

2

The Usual Suspects

Lectures. [Kevin Wayne]

- Tuesdays and Thursdays, Frist 302.
- Same lecture at 10 and 11am.

lead preceptor

Precepts. [Siddhartha Brahma · Maia Ginsburg · Chang Kim · Woo Chang Kim · Ganesh Lakshminarayana · Prof. Andrea LaPaugh · Ananya Misra · Sonya Nikolova · Tim O'Connor · Yi Wang · Forrest Zhang · Yaping Zhu]

- Tue+Thu or Wed+Fri.
- 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.

Grades

Course grades. No preset curve or quota.

8 programming assignments. 40%.

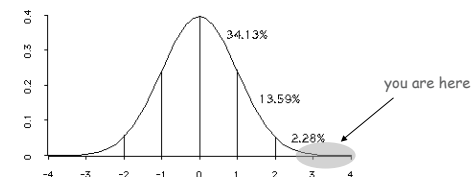
2 exams. 50%.

can drop lowest one

Final programming project. 10%.

Staff discretion. Adjust borderline cases.

participation helps, frequent absences hurts



3

4

Course Materials

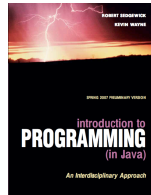
Course website. [www.princeton.edu/~cos126]

- Submit assignments, check grades.
- Programming assignments.
- Lecture notes.

← print slides before lecture;
annotate during lecture

Required readings.

- Sedgewick and Wayne. *Intro to Programming in Java: An Interdisciplinary Approach*. [U-store]
- Packet. [stay tuned]



Recommended readings.

- Harel. *What computers can't do*. [U-Store]

What's Ahead?

Lecture 2. Intro to Java.

Precept 1. Meets today/tomorrow. Everyone should go.

Precept 2. Meets Thu/Fri. Intended for novices.

Not registered? Go to any precept today;
must officially register to submit assignments.

Assignment 0. Due Monday 2/12, 11:55pm.

- Read Sections 1.1 and 1.2 in textbook.
- Install Java programming environment + a few exercises.
- Lots of help available, don't be bashful.

END OF ADMINISTRATIVE STUFF

Programming Assignments

Desiderata.

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

Examples.

- N-body simulation.
- Digital signal processing.
- DNA sequence alignment.
- Estimate Avogadro's number.
- Markov model of natural language.

} you solve scientific
problems from scratch!

Due. Mondays 11:55pm via Web submission.

Computing equipment.

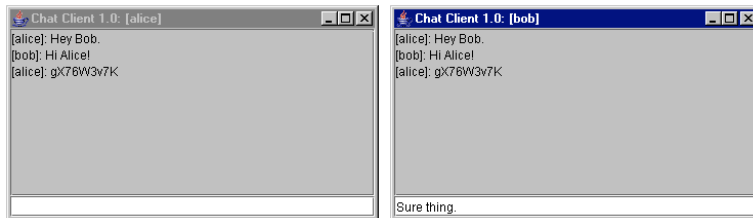
- Your laptop. [OS X, Windows, Linux, iPhone, ...]
- OIT desktop. [Friend 016 and 017 labs]

Prologue: A Simple Machine

Secure Chat

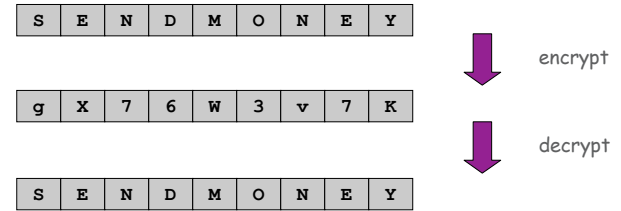
Alice wants to send a secret message to Bob?

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



Encryption Machine

Goal. Design a machine to encrypt and decrypt data.



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.



9

11

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



computer with a steamer

12

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

13

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	message
010010	000100	001101	000011	001100	001110	001101	000100	011000	base64

One-Time Pad Encryption

Encryption.

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

XOR Truth Table

x	y	$x \wedge y$
0	0	0
0	1	1
1	0	1
1	1	0

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

S	E	N	D	M	O	N	E	Y	message
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

↑
 $0 \wedge 1 = 1$

15

17

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	message
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

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	encrypted
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

18

24

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.
 - $(a \oplus b) \oplus b = a \oplus (b \oplus b) = a \oplus 0 = a$
- ↑ associativity of ^
↑ always 0
↑ identity

XOR Truth Table

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

25

Goods and Bads of One-Time Pads

Good.

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

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.

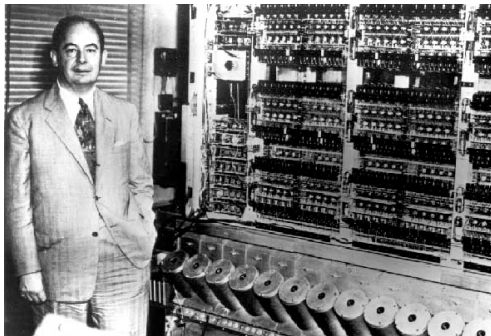
"one time" means **one time only**

↑ impractical for Web commerce

32

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)

34

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.

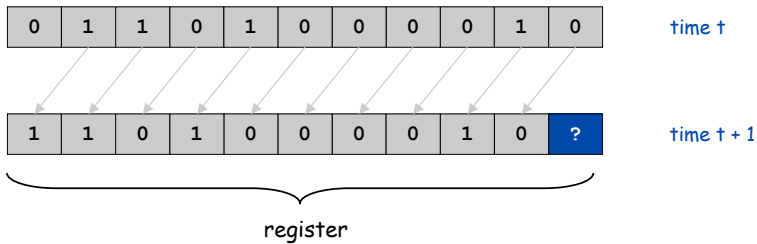
- Linear feedback shift register.**
- Linear congruential generator.
- Blum-Blum-Shub generator.
- ...

35

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.



36

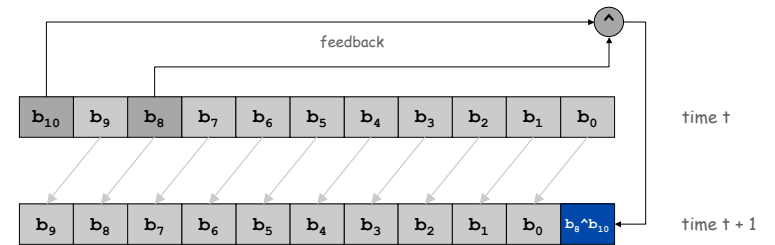
Linear Feedback Shift Register

{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



37

Random Numbers

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

```

1100100100111101101110010110101110011000101111101001000010011010010111100110010011111
101110000010101100010000111010100110100001110010011001110111111010100000100001001010
01010100011000001011100010010011010110111000110100110111001110101110010001001110101
011101000001010010001000110101011100000001011000001001110001011101010101011001100010000
11111100110000011111000110000110111001110100111101001110110110101010100001010
0000000100000000101000000100010000101010100100000001101000001100100011011101011010100
01010000101000100010001010110101000011000010011110010111001110010111011100100101011011
0000101011100100001011101001001010011011000111011101100101010111000000100110000010111
10010010001110110101101011000110001101111011010100101100001100111110111100001010
011001000111110101100010001110010101101100001101011001110011110110110001011011010
011010100111100001110011001101111111010000000100100000101101000100110010101111100001
0000110010100111100011000110110110110110101010100000110110001110101101101010100011
01100101110111001010100111000000110110001101011101100010101010100000011001000011110
10011000100111101011100010001011010101001100000011110000110001100111101111001010000
1110001001101010111011000100101101011001010001110001011001101001111100111000011110
110011001011111100100000111010000110100011010011100110111011110101010001000000101010000
100000100101000101100010100011101001011010011001100111111110000000001100000000
111100000110011000111111101100000010111000010010110010110011110011110011110001110011
0110011111011110001010001101000101100101001011000110010110111001101000111110010110
00111001110101111010110100100011001101011111000100000110101000111000010110110010110
11110111101001010010011000101111011010001010100101000001100010001110101011001000001
1110100011001001011110110010001011101010010010000011010100111011001110100111101001000100
0100101010101100000000111000000110110001101110011010101111000001000110001010111010
    
```

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

38

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.

39

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 ==2)
0,j= m(4) 17^ m(3) 9^k* 2-k&8
^8,a =0,c =26;for (s[y] --m16;
--c;j *=2)a= a*2^i& 1,i=i /2^j&1
<<24;for(j= 127; ++j<n;c>
y)
c
+=y^i/8^i>>4^i>>12,
i=i>>8^y<<17,a^=a>>14,y=a^8^a<<6,a=a
>>8^y<<3,i=s[j],k =7Wo-'G_1216'[k
&7]+2^i*c&3&f&6w;+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>

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

```

% java LFSR 2000
110010010011110110111001011101
01110011000101111110100100001
00110100101111001100100111...
    
```