COS 126

General Computer Science
Princeton University
Spring 2009

Kevin Wayne

Overview

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

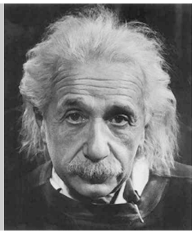
Goals.

- 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 Basics

Lectures. [Kevin Wayne]

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

Precepts. [Donna Gabai (lead) · Aditya Bhaskara · Will Clarkson · Rob Dockins · Michael Golightly · Thomas Mason · Chris Park · JP Singh]

- 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.

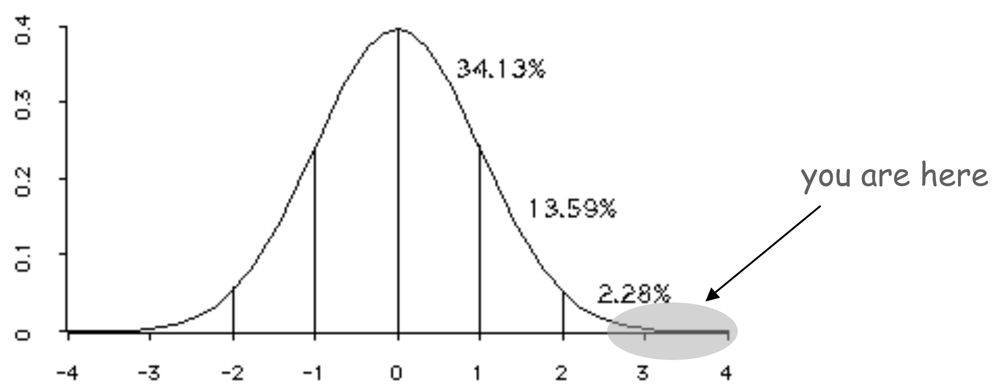
For full details: See **`www.princeton.edu/~cos126`**

Grades

Course grades. No preset curve or quota.

9 programming assignments. 40%.
2 exams. 50%.
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`]

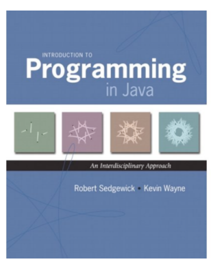
- Submit assignments, check grades.
- Programming assignments.
- Lecture slides.

print before lecture;
annotate during lecture

skim before lecture;
read thoroughly afterwards

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

Princeton royalties
donated to ACM-W



Recommended readings. Harel. *What computers 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. Mondays 11pm via Web submission.

Computing equipment.

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

What's Ahead?

Lecture 2. Intro to Java.

Precept 1. Meets today/tomorrow.
Precept 2. Meets Thu/Fri.

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

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

Assignment 0. Due Monday, 11pm.

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

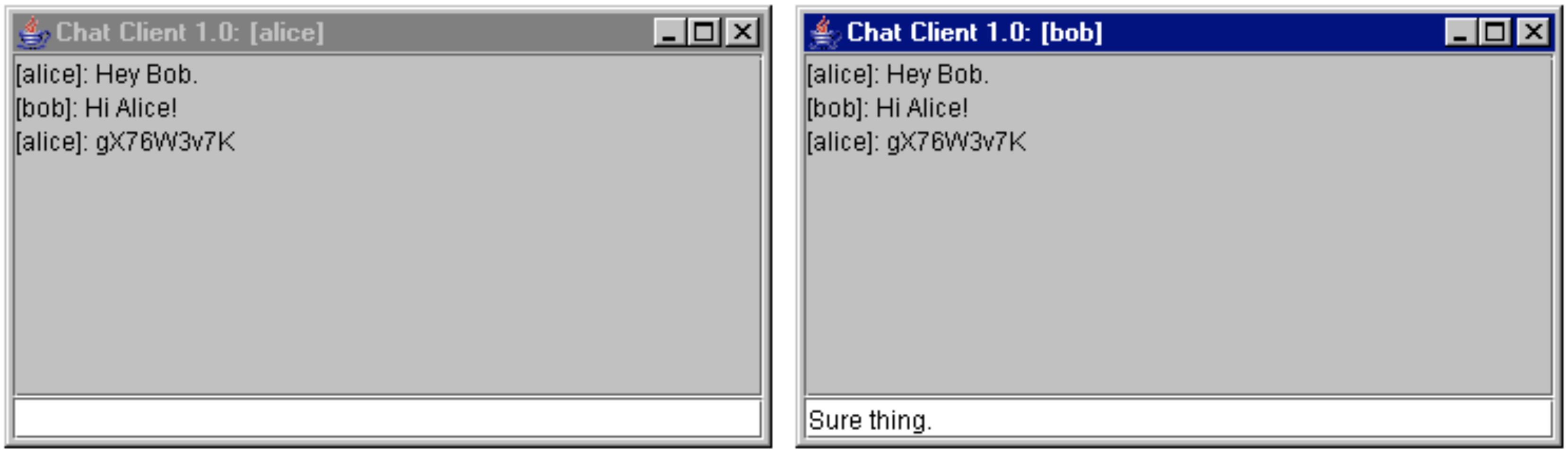
END OF ADMINISTRATIVE STUFF

0. 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.

S	E	N	D	M	O	N	E	Y

encrypt

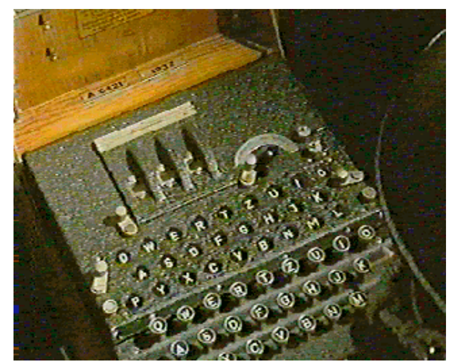
g	X	7	6	W	3	v	7	K

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.

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

Base64 Encoding

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

Encryption.

- Convert text message to N bits.

0 or 1

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).

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

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	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 ^ 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

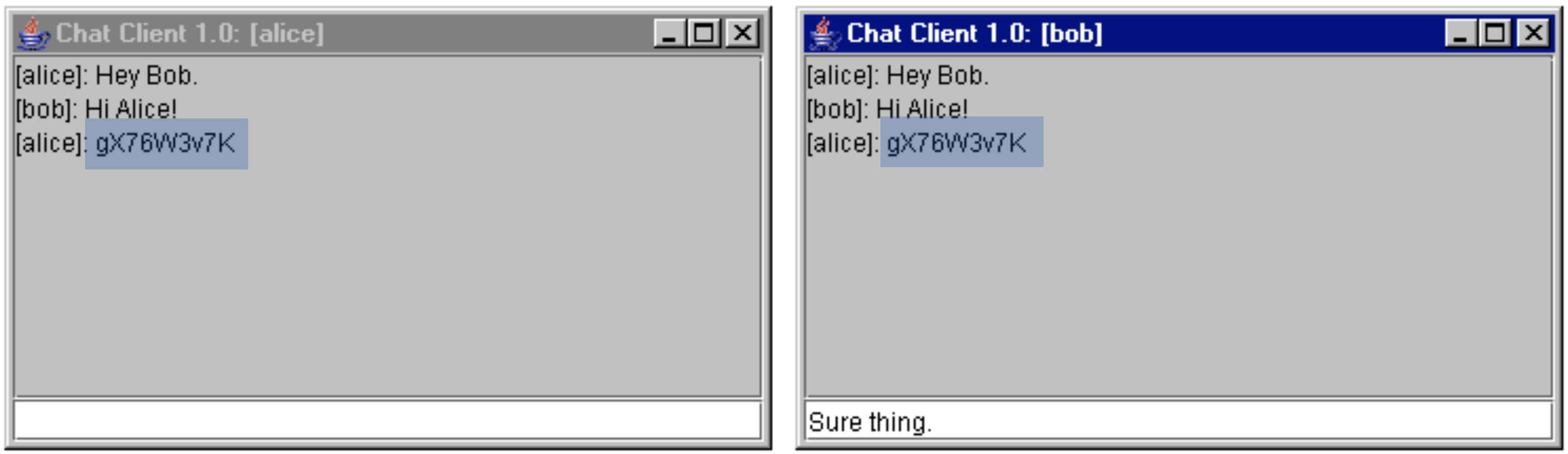
char	dec	binary
A	0	000000
B	1	000001
…	…	…
X	23	010111
…	…	…

S	E	N	D	M	O	N	E	Y	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

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

Base64 Encoding

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

Decryption.

- Convert encrypted message to binary.

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

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	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

1 ^ 1 = 0

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

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

XOR Truth Table

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

Why is crucial property true?

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

associativity of ^ always 0 identity

One-Time Pad Decryption (with the wrong pad)

Decryption.

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

g	X	7	6	W	3	v	7	K	encrypted
100000	010111	111011	111010	010110	110111	101111	111011	001010	base64
101000	011100	110101	101111	010010	111001	100101	101010	001010	wrong bits
001000	001011	001110	010101	000100	001110	001010	010001	000000	XOR
I	L	O	V	E	O	K	R	A	wrong message

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

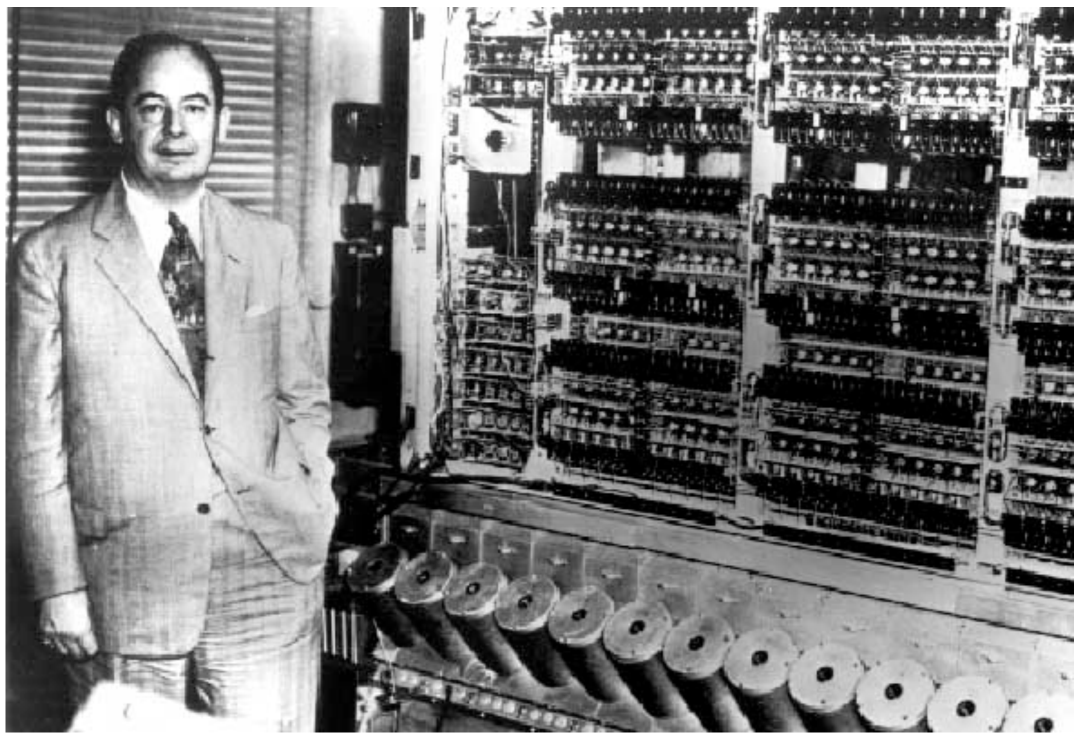
Bad.

- 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

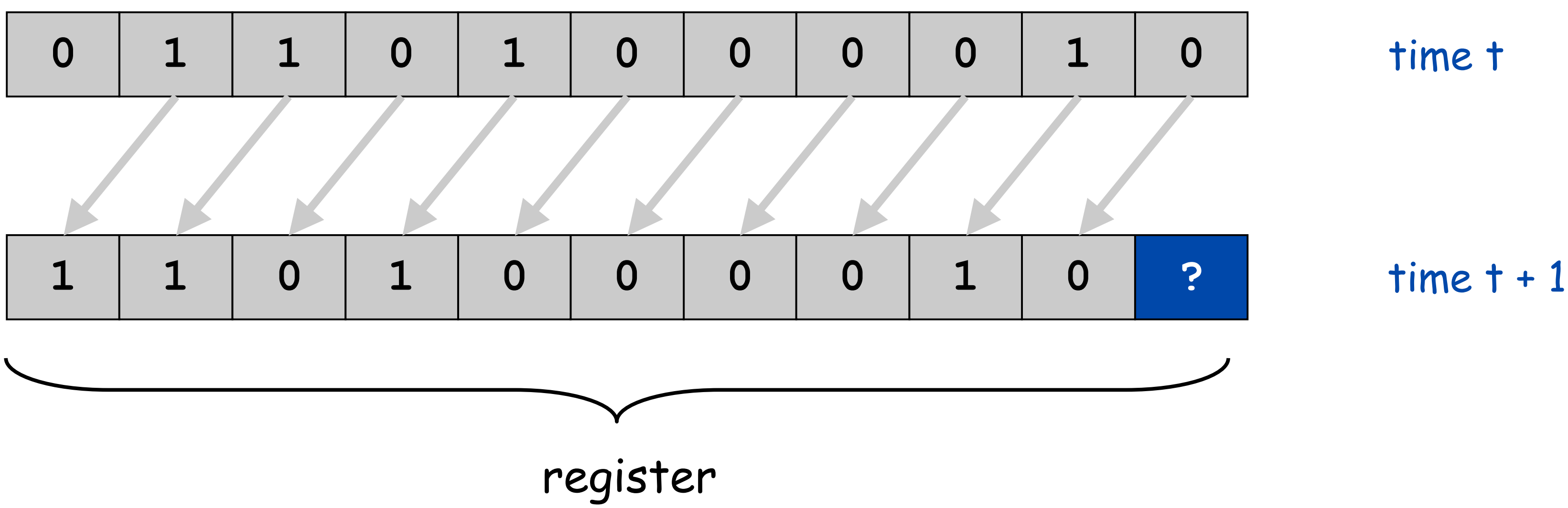
How to make small gadget that produces "random" numbers.

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

Shift Register

Shift register terminology.

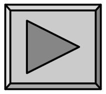
- Bit: 0 or 1.
- Cell: storage element that holds one bit.
- Register: sequence of cells.
- Seed: initial sequence of bits.
- Shift register: when clock ticks, bits propagate one position to left.



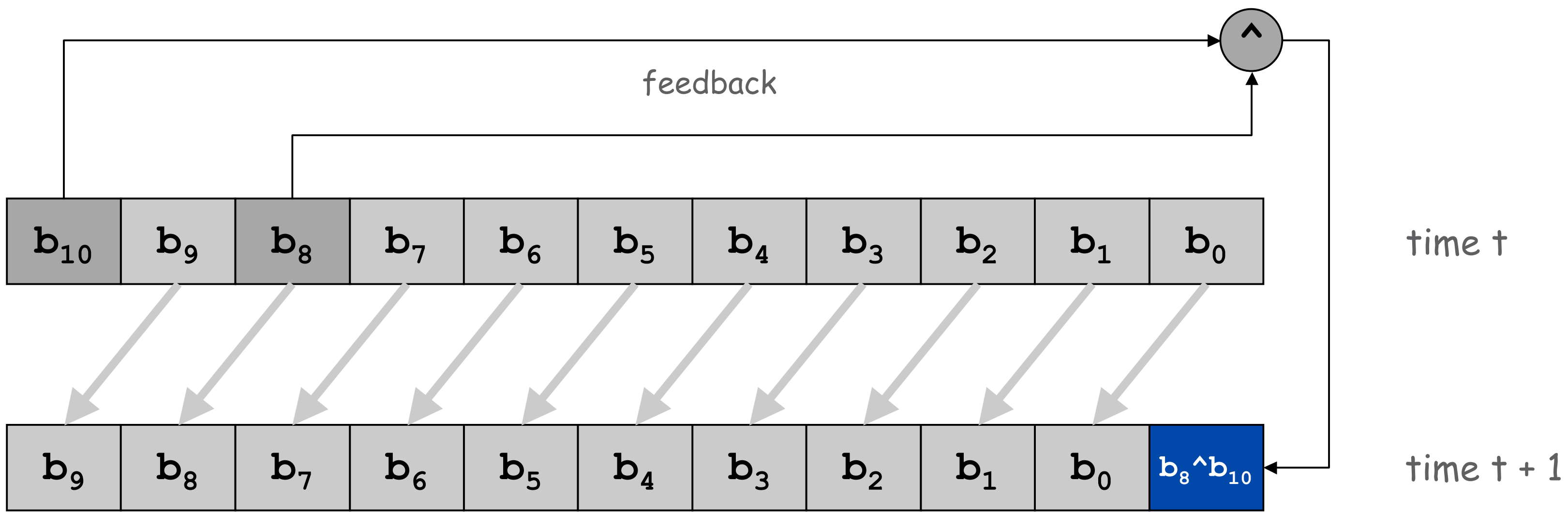
Linear Feedback Shift Register

{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?

```
1100100100111101101110010110101110011000101111110100100001001101001011110011001001111111
1011100000101011000100001110101001101000011110010011001110111111101010000010000100010101
0101010001100000101111000100100110101101111000110100110111001111010111100100010011101010
0111010000010100100010001101010101110000000101100000100111000101110110100101011001100000
1111111001100000111111000110000110111100111010011110100111001001110111011101010101010000
0000000100000000010100000001000100001010101001000000001101000000111001000110111010111010100
0101000010100010010001010110101000011000010011110010111001110010111101110010010101110110
0000101011100100001011101001001010011011000111101110110010101011110000000100110000101111
1001001000111011010110101100011000111011110110101001011000011001110011111101111000010100
0110010001111110101100001000111001010110111000011010110011100011111011011000101101110100
0110101001111000011100110011011111111110100000001001000000101101000100110010101111110000
0000110010100111110001110001101101101110110110101011011000001101110001110101101101000110
0110010111011110010101001110000011101100011010111011100010101011010000000110010000111110
1001100010011111010111000100010110101010011000000111110000110001100111101111111001010000
1110001001101101011110110001001011101011001010001111000101100110100111111001110000111100
1100110010111111110010000000111010000110100100111001101110111110101010001000000010101000
1000001001010001011000101001110100011101001011010011001100111111111110000000000110000000
1111000001100110001111111101100000010111000010010110010110011110011111001111000111100110
0110011111011111000101000110100010111001010010111000110010110111110011010001111100101100
0011100111011011110101101001000110011010111111100010000011010100011100001011011001001100
1111011110100101001001100011011111011101000101010010100000110001000111101010110010000010
1110100011001001011111011001000101111010100100100001101101001110110011101011111010001000
01001010101011000000000111000000110110000111011100110101011111000001000110001010111101
```

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

LFSR Challenge 1

Goal. Decrypt/encrypt 1,000 characters. Can we use an 11-bit LFSR?

A. Yes, no problem.

B. No, the bits it produces are not truly random.
- True, but that's beside the point.

C. No, need a longer LFSR.
- Only 2^{11} bit patterns for register.
- "Random" bits cycle after $2^{11} - 1 = 2047$ steps.

Lesson. LFSR are scalable: 20 cells for 1 million bits; 30 cells for 1 billion.
(but need theory of finite groups to know where to put taps)

LFSR Challenge 2

Goal. Decrypt/encrypt 1 gigabyte movie. How big an LFSR?

A. 30 bits should be enough.

- Probably not; Eve can try all 2^{30} possibilities and see which one results in a movie.

B. 100 bits is safe.

- Maybe; Eve would need 10^{12} centuries to try all 2^{100} possibilities.

C. 1000 bits makes it sufficiently secure.

- Experts have cracked LFSR.
- More complicated machines needed.

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.

A Profound Idea

Programming. Can write a Java program to simulate the operations of any abstract machine.

- Basis for theoretical understanding of computation. [stay tuned]
- Basis for bootstrapping real machines into existence. [stay tuned]

Stay tuned. See Assignment 5.

```
public class LFSR {
   private int fill[];
   private int tap;
   private int N;

   public LFSR(String fill, int tap) { … }

   public int step() { … }

   public static void main(String[] args) {
      LFSR lfsr = new LFSR("01101000010", 8);
      for (int i = 0; i < 2000; i++)
         StdOut.println(lfsr.step());
   }
}
```

```
% java LFSR
11001001001111011011100101101
01110011000101111110100100001
00110100101111001100100111...
```

A Profound Question

Q. What is a random number?

LFSR does not produce random numbers.

- It is a very simple deterministic machine.
- But it is hard to distinguish the bits it produces from random ones.

Q. Are random processes found in nature?

- Motion of cosmic rays or subatomic particles?
- Mutations in DNA?

Q. Is the natural world a (not-so-simple) deterministic machine?

" God does not play dice. " – *Albert Einstein*