Lecture 1: Introduction

COS 126
Princeton University
Spring 2005

Douglas Clark

Overview

What is COS 126?

- Broad, but technical, intro to fundamental ideas of CS.
- No prerequisites, intended for novices.

Goals.

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

Topics

Programming. Java.

Machine architecture. Fictional computer that resembles real one.

Theory of computation. Capabilities and limitations of computers.

Applications.

- Scientific computing.
- Cryptography.
- Software systems.

The Usual Suspects

Lectures: Doug Clark.

- Tuesdays and Thursdays, 10:00 - 10:50, Friend 101.

Precepts: Zafer Barutcuoglu, Benedict Brown, Dan Dantas, Donna Gabai, Phil Shilane, Mona Singh

- Mondays and Fridays - tips on assignments (Fridays esp.), clarification of lecture material.
- Where? Many different locations (see Web or posted lists)

Staffed lab: Undergrad lab assistants.

- Friend 016, 017.
- Schedule to be posted on Web.

Grading

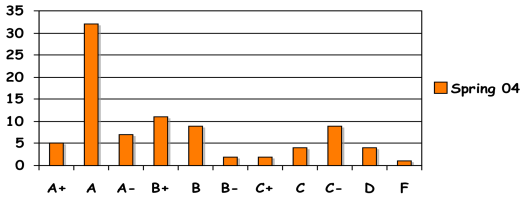
Programming assignments: 35%
- Can drop lowest one.

2 exams: 50%

Final project: 15%

Staff discretion. Adjust borderline cases.

Course grades. No preset curve. Last spring's results:



Course Materials

Course website.
- Programming assignments.
- Lecture notes.
- Exam archive.
- `www.princeton.edu/~cos126`

Required readings.
- Sedgewick and Wayne. *Intro to Computer Science.*
  - draft of some sections: Triangle Copy (150 Nassau)
  - booksite: `www.cs.princeton.edu/IntroCS`

Recommended readings. (U-Store)
- King. *Java Programming From the Beginning.*
- Harel. *What computers can't do.*

Programming Assignments

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

Examples.
- N-body simulation.
- DNA sequence alignment.
- Digital signal processing of MP3 files.
- Traveling salesperson problem.
- Markov model of natural language.

Solve scientific and commercial problems from scratch.

Due: Wednesdays 11:59pm via Web submission.

Computing equipment.
- Your machine. (Linux, OS X, Unix, Windows, Java cell phone, . . .)
- OIT machines. (Friend 016 and 017 labs).

What's Ahead?

ASAP. Read sections 1.0 – 2.2 of Intro to CS.
Thursday. Lecture: Introduction to Java.
Friday. First precept meets.
- Check course web page or posted list to see which precept you're in.
- If not in precept, see Donna Gabai after class or this afternoon before 2 PM in CS 205, 35 Olden Street.

Monday. Precept.
Tuesday. Lecture: Yet More Java
Wednesday. Assignment 0 due.
- Setup Java programming environment.
- For assistance, go to Friend 016 or 017.

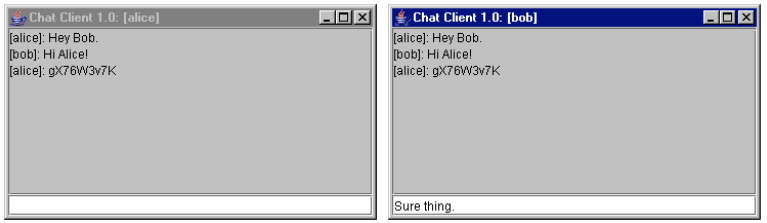
END OF ADMINISTRATIVE STUFF

A Simple Machine

Secure Chat

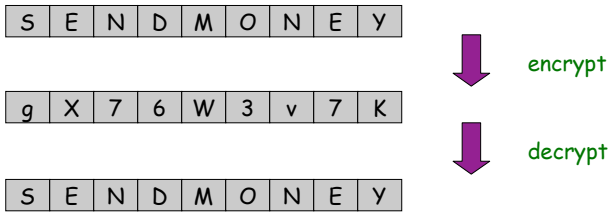
Alice wants to send a secret message to Bob?

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



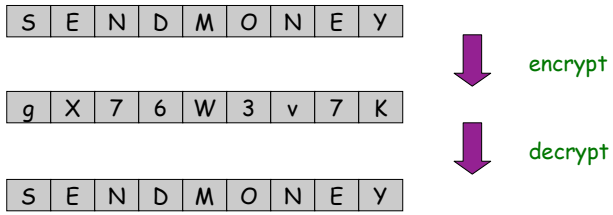
Encryption Machine

Goal: design a machine to encrypt and decrypt data.



Encryption Machine

Goal: design a machine to encrypt and decrypt data.



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.

Digital Data

Computers store all data as a sequence of bits.

- Text.
- Images (JPEG), audio (MP3), video (DivX).
- Programs.

Bit = 0 or 1

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

1. Convert text message to N bits.

Bit = 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	binary

One-Time Pad Encryption

1. Convert text message to N bits.
2. 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	binary
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits

One-Time Pad Encryption

1. Convert text message to N bits.
2. Generate N random bits (one-time pad).
3. Take bitwise XOR of two strings.
   - Sum 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	binary
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits
100000	010111	111011	111010	010110	110111	101111	111011	001010	XOR

One-Time Pad Encryption

1. Convert text message to N bits.
2. Generate N random bits (one-time pad).
3. Take bitwise XOR of two strings.
4. 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	binary
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

1. Convert encrypted message to binary.

g	X	7	6	W	3	v	7	K	encrypted

One-Time Pad Decryption

1. 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	binary

One-Time Pad Decryption

1. Convert encrypted message to binary.
2. 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	binary
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits

One-Time Pad Decryption

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

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	binary
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

1. Convert encrypted message to binary.
2. Use same N random bits (one-time pad).
3. Take bitwise XOR of two strings.
4. 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	binary
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?

Notation	Meaning
a	original message
b	one-time pad
^	XOR operator
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 is always 0.
- ^ is *associative.*
- Therefore: (a ^ b) ^ b = a ^ (b ^ b) = a ^ 0 = a

Goods and Bads of One-Time Pads

Good:

- Very simple encryption/decryption processes.
- Theoretically unbreakable if pad is truly random.

Bad:

- "One time" means *one time only.*
  - Very breakable if pad is re-used
- Pad must be as long as the message.
- Pad must be distributed securely.
- Truly truly random bits are very hard to come by.

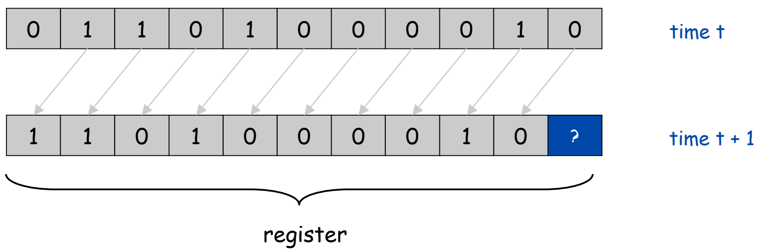
So how about this:

- Let's make an almost-random-bit-generator gadget
- Correspondents each get identical small gadgets rather than identical large one-time pads

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.

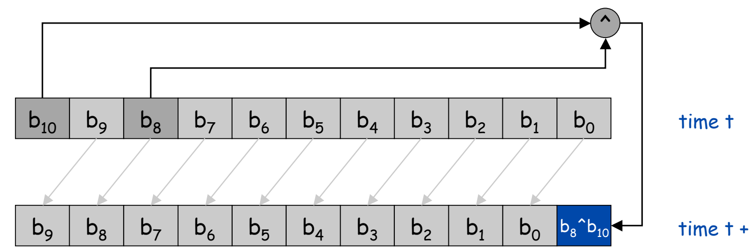


Linear Feedback Shift Register

{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

Are these 2000 numbers random?

```
11001001001111011011100101101011100110001011111101001000010011010010111100110010011111101
1100000101011000100001110101001101000011110010011001110111111101010000010000100010100101 01
```


The Science Behind It

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

How did the computer scientist die in the shower?
The instructions on the shampoo read "lather, rinse, repeat."

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

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

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

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

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

Java program prints exactly same bits as LFBSR.

- You'll understand this program by next week.

```
public class LFBSR {
   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++) {
         boolean bit = b3 ^ b10;                          <= update
         b10 = b9; b9 = b8; b8 = b7; b7 = b6; b6 = b5;
         b5  = b4; b4 = b3; b3 = b2; b2 = b1; b1 = b0;
         b0 = bit;

         if (bit) System.out.print(1);                    <= print
         else     System.out.print(0);
      }
      System.out.println();
   }
}
```

```
% java LFBSR 2000
01001100100000011000100010
11010101111001001111001110
10
00100001101111111100101
...
```