

# Introduction to Computer Science

## Introduction

Ryan Stansifer

Department of Computer Sciences  
Florida Institute of Technology  
Melbourne, Florida USA 32901

<http://www.cs.fit.edu/~ryan/>

2 November 2017

# Overview of Course

- ▶ Introduction and Context. What is CS?
- ▶ Java review. Data, control constructs, static methods
- ▶ Classes. Incorporation, instantiation, inheritance
- ▶ Generics. Code reuse
- ▶ Program analysis. Steps the program takes
- ▶ Data structures. Lists, stacks, queue

# Course Goals

- ▶ Programming
  - ▶ exciting to translate ideas into reality
  - ▶ basics are simple, yet programming well is difficult; do not underestimate the challenge
  - ▶ delivery high-quality programs on time; be able to express control flow and design data in Java
  - ▶ problem solving is hard and difficult to teach
- ▶ Computer Science
  - ▶ Computer Science is not just programming
  - ▶ It is easy to lose sight of the big picture, so we have a general introduction
  - ▶ Other (non-programming) topics from time to time: architecture, Monte Carlo methods,  $O(N)$ , invariants, and so on

## Outline of Introduction

There are couple of topics that put programming in context and that are helpful if pointed out in advance and getting mired in the details.

- ▶ What is Computer Science? Areas of study: AI, OS, ...
- ▶ What is a computer? Architecture, CPU, memory hierarchy
- ▶ Interface layers: hardware, operating system, application
- ▶ The Java platform
  - ▶ JVM and a million other pieces
  - ▶ Java history, pragmatics
- ▶ Programming languages — not just Java
- ▶ Program development; debuggers and so on
- ▶ Program style. A program is a text file
- ▶ I/O, streams

The single most important skill in programming, computer science, and science in general is abstraction. Yet I think that belaboring the idea may be too philosophical at this time. If one is observant, one will see abstraction at work in all the topics above.

# What is Computer Science?

# What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

# What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

- ▶ automaton:

# What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

- ▶ automaton: “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)



# What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

- ▶ automaton: “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
- ▶ information:

# What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

- ▶ automaton: “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
- ▶ information: knowledge represented in a form suitable for transmission, manipulation, etc.

# What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

- ▶ automaton: “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
- ▶ information: knowledge represented in a form suitable for transmission, manipulation, etc.
- ▶ protocol:

# What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

- ▶ automaton: “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
- ▶ information: knowledge represented in a form suitable for transmission, manipulation, etc.
- ▶ protocol: rules for exchanging information without problems

# What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

- ▶ automaton: “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
- ▶ information: knowledge represented in a form suitable for transmission, manipulation, etc.
- ▶ protocol: rules for exchanging information without problems
- ▶ algorithm:

# What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

- ▶ automaton: “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
- ▶ information: knowledge represented in a form suitable for transmission, manipulation, etc.
- ▶ protocol: rules for exchanging information without problems
- ▶ algorithm: an unambiguous, finite description in simple steps or actions

# What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

- ▶ automaton: “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
- ▶ information: knowledge represented in a form suitable for transmission, manipulation, etc.
- ▶ protocol: rules for exchanging information without problems
- ▶ algorithm: an unambiguous, finite description in simple steps or actions

Computer Science is not the study of computers, nor is it the practice of their use.





*How does this class (studying Java) fit into the study of Computer Science?*

*How does this class (studying Java) fit into the study of Computer Science?*

Learn some algorithms, some real and idealized machines, learn something about information. Mostly learn some mechanisms which can *express computation*.

Mathematics, science, or engineering?

**Mathematics.** *The science of numbers, interrelations, and abstractions.*

**Science.** *Systematic knowledge or practice. Acquiring knowledge through the scientific method of natural phenomena (natural sciences) or human or social behavior (social sciences).*

**Engineering.** *The applied science of acquiring and applying knowledge to design, or construct works for practical purposes.*

# What is CS?

- ▶ Engineering? Application of science?
- ▶ Natural science? Observable phenomena?
- ▶ Mathematics? Invisible abstractions?
- ▶ Social science? Functioning of human society?

# What is CS?

- ▶ Engineering? Application of science?
- ▶ Natural science? Observable phenomena?
- ▶ Mathematics? Invisible abstractions?
- ▶ Social science? Functioning of human society?

CS is exciting and difficult as it is all these things.

# What is CS?

- ▶ To do anything requires programming
- ▶ To do something useful requires domain knowledge

Do you become skillful at programming, or an expert in some small area?

## Existential Angst



*The Scream* by the Norwegian artist Edvard Munch, painted in 1893.

We are at the dawn of new era. The, as yet unfinished, language of computation is the language of science and engineering and is overtaking mathematics as the Queen of Science.

*Philosophy is written in this grand book, the universe which stands continually open to our gaze. But the book cannot be understood unless one first learns to comprehend the language and read the letters in which it is composed. It is written in the language of mathematics, and its characters are triangles, circles and other geometric figures without which it is humanly impossible to understand a single word of it; without these, one wanders about in a dark labyrinth.*



Galileo Galilei in *The Assayer*



# What Does A Computer Scientist Do?

Similar to mathematics, most everyone in modern society uses computing. So getting a computer science degree prepares you for everything and nothing.

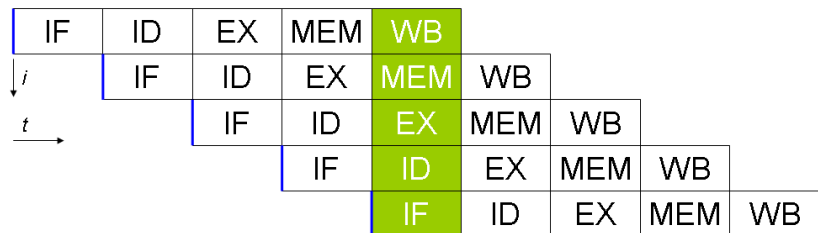
The most visible activity is commanding computers to do our bidding, i.e., programming.

*What do you want to do?*

# Fields

- ▶ Computer architecture
- ▶ Operating systems
- ▶ Programming languages and compilers
- ▶ Algorithms, data structures, complexity
- ▶ Computability theory
- ▶ Numerical analysis
- ▶ Networking and distributed computing
- ▶ Parallel computing
- ▶ Information Management/Database systems
- ▶ Software development (aka Software Engineering)
- ▶ Human-computer communication/interaction
- ▶ Graphics and Visual Computing
- ▶ Intelligent Systems (aka Artificial Intelligence)

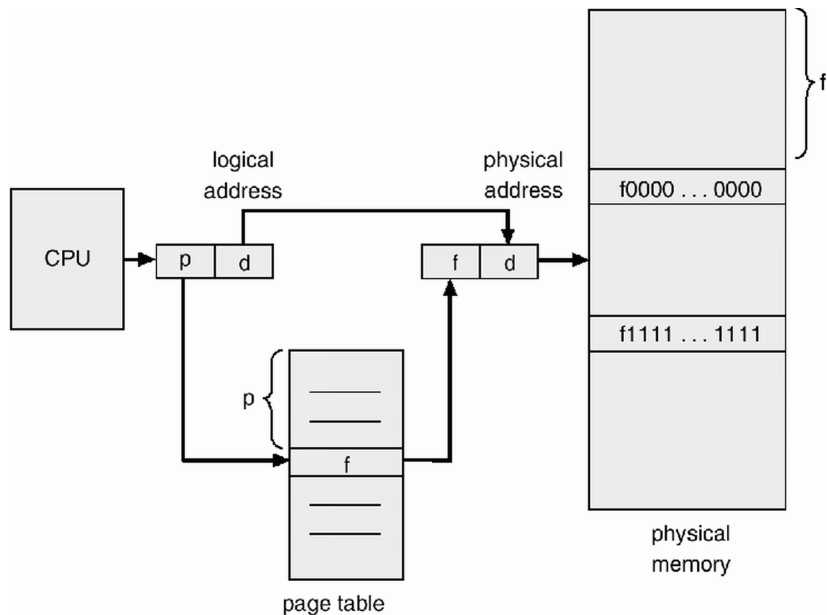
# Architecture



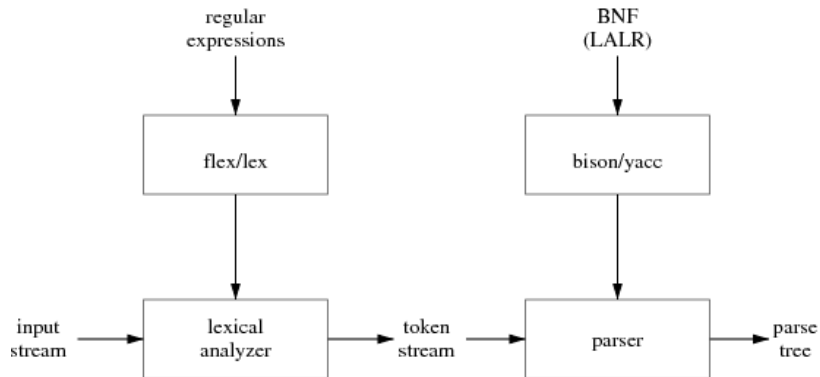
Basic five-stage pipeline in a RISC machine: instruction fetch, instruction decode, execute, memory access, register write back.

The IBM PowerPC G5 has 21 pipeline stages; the Intel Pentium 4E has 31 stages.

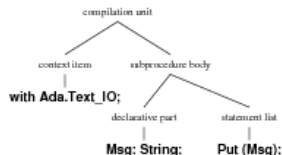
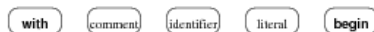
## Operating Systems — paging



# Programming Languages and Compilers



```
with begin 2 * x ;
```



# Algorithms and Data Structures — Sorting

## Theory of Computation — halting problem

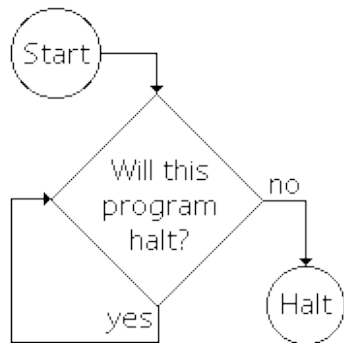
*The argument that the power of mechanical computations is limited is not surprising. Intuitively we know that many vague and speculative questions require special insight and reasoning well beyond the capacity of any computer that we can now construct or even foresee. What is more interesting to computer scientists is that there are questions than can be clearly and simple stated, with an apparent possibility of an algorithmic solution, but which are know to be unsolvable by any computer.*

Linz 6th, Section 12.1, page 310

An example of such a problem is if a grammar is ambiguous or not. (This can formalized and is an interesting issue in constructing compilers.)

The proof that there are specific problems that cannot be solved, if not remarkably simple.

## Theory of Computation — halting problem





# Numerical Analysis

*A report from the United States General Accounting Office begins “On February 25, 1991, a Patriot missile defense system operating at Dhahran, Saudi Arabia, during Operation Desert Storm failed to track and intercept an incoming Scud. This Scud subsequently hit an Army barracks, killing 28 Americans.” The report finds the failure to track the Scud missile was caused by a precision problem in the software.*

Nicholas J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, 1996, ISBN13 9780898713558. Page 505.

<http://www.ima.umn.edu/~arnold/disasters/disaster.html>

# Networking

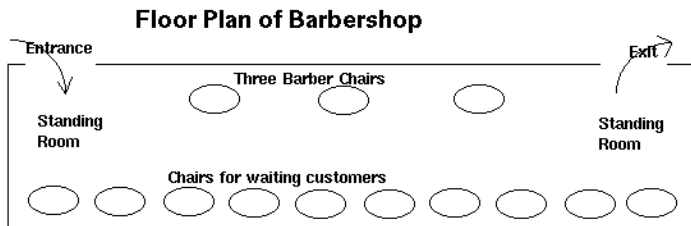
In the sliding window protocol the window size is the amount of data a sender is allowed to have sent into the network without having yet received an acknowledgment for it.

In Internet routers, active queue management (AQM) is a technique that consists in dropping packets before a router's queue is full.

Historically, queues use a *drop-tail* discipline: a packet is put onto the queue if the queue is shorter than its maximum size. Drop-tails queue have a tendency to penalize bursty flows.

Active queue disciplines drop packets before the queue is full based on probabilities. Active queue disciplines are able to maintain a shorter queue length than the drop-tail queue which reduces network latency.

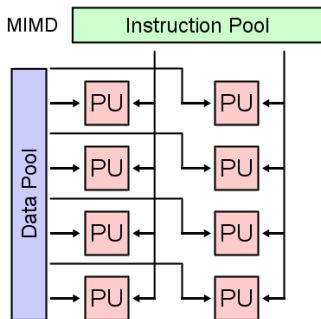
# Distributed Computing — barber shop problem



# Parallel Computing

	sing instr	mult instr
single data	SISD	MISD
multiple data	SIMD	MIMD

Flynn's taxonomy



# Information Management/Database Systems

Name	
A	B
Dot	cat
Sue	dog
Zan	cat
Bip	bird

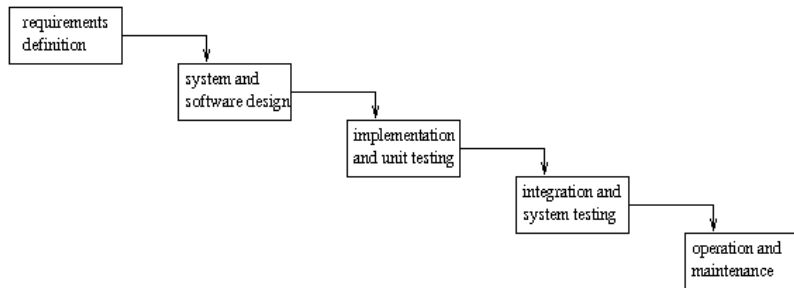
Have	
B	C
cat	shots
dog	shots
dog	leash

Result of Join		
A	B	C
Dot	cat	shots
Sue	dog	shots
Sue	dog	leash
Zan	cat	shots

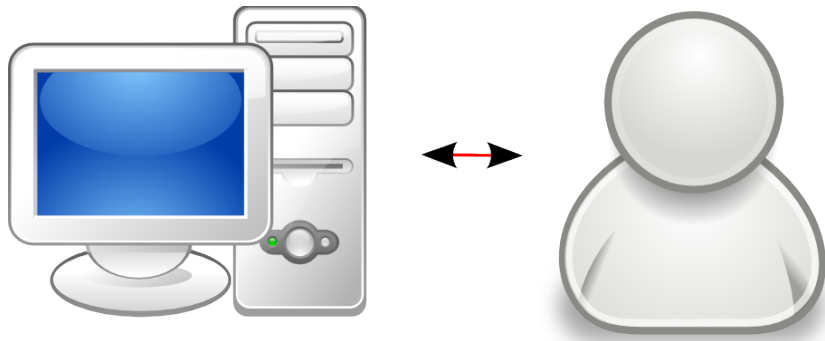
The join of two relational tables



# Software Engineering — waterfall model



# Human-Computer Communication/Interaction

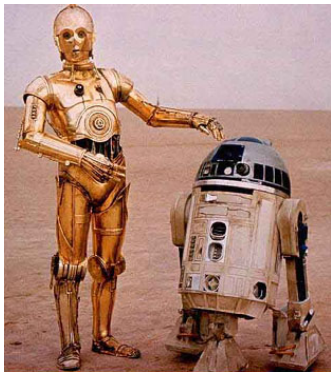


The Miracle Worker scene from Star Trek 4: The Voyage Home on YouTube.





# Intelligent Systems



C3PO and R2D2 are fantasy robots from the movie Star Wars, while Kiva's industrial robots can efficiently and intelligently move shelves in a warehouse.



End of the overview of different fields of study in computer science

## Layers, Scale, Interfaces

The solution to vast complexity is layers. Good layers enable high-quality specialization, Bad layers just increase the complexity.

Software development—programming—is often about creating layers, even in small programming projects.

Computing is complex. There are many layers of interesting stuff between the person and the automaton.



# Scale of the Universe 2

by Cary and Michael Huang

# The vastness and minuteness of time and space is a challenge to comprehend.

From Wikipedia, the free encyclopedia

Section	Range (m)		Unit	Example Items
	≥	<		
Subatomic	0	$10^{-15}$	am	electron, quark, string
Atomic to Cellular	$10^{-15}$	$10^{-12}$	fm	proton, neutron
	$10^{-12}$	$10^{-9}$	pm	wavelength of gamma rays and X-rays, hydrogen atom
	$10^{-9}$	$10^{-6}$	nm	DNA helix, virus, wavelength of optical spectrum
Human Scale	$10^{-6}$	$10^{-3}$	μm	bacterium, fog water droplet, human hair <sup>[1]</sup>
	$10^{-3}$	$10^0$	mm	mosquito, golf ball, soccer ball,
	$10^0$	$10^3$	m	human being, American football field, Eiffel Tower
	$10^3$	$10^6$	km	Mount Everest, length of Panama Canal, asteroid
Astronomical	$10^6$	$10^9$	Mm	the Moon, Earth, one light-second
	$10^9$	$10^{12}$	Gm	Sun, one light-minute, Earth's orbit
	$10^{12}$	$10^{15}$	Tm	orbits of outer planets, Solar System,
	$10^{15}$	$10^{18}$	Pm	one light-year, distance to Proxima Centauri
	$10^{18}$	$10^{21}$	Em	galactic arm
	$10^{21}$	$10^{24}$	Zm	Milky Way, distance to Andromeda Galaxy
	$10^{24}$	∞	Ym	visible universe



## SI Prefixes

A study of people in Nicaragua who were born deaf and never learned Spanish or a formal sign language has concluded that humans need language in order to understand large numbers. “Up to three, they’re fine,” says Elizabet Spaepen, a researcher at the University of Chicago and an author of the study. “But past three, they start to fall apart.”

<http://www.npr.org/2011/02/09/>

# SI Prefixes

peta	P	quadrillion	$10^{15}$	1 000 000 000 000 000
tera	T	trillion	$10^{12}$	1 000 000 000 000
giga	G	billion	$10^9$	1 000 000 000
mega	M	million	$10^6$	1 000 000
kilo	k	thousand	$10^3$	1 000
hecto	h	hundred	$10^2$	100
deca	da	ten	$10^1$	10
(none)		one	$10^0$	1
deci	d	tenth	$10^{-1}$	0.1
centi	c	hundredth	$10^{-2}$	0.01
milli	m	thousandth	$10^{-3}$	0.001
micro	$\mu$	millionth	$10^{-6}$	0.000 001
nano	n	billionth	$10^{-9}$	0.000 000 001
pico	p	trillionth	$10^{-12}$	0.000 000 000 001
femto	f	quadrillionth	$10^{-15}$	0.000 000 000 000 001

XKCD PRESENTS:  
SOME NEW  
SCIENCE MNEMONICS



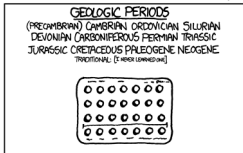
PLEASE EMAIL MY DAD A SHARK  
OR PEOPLE EXPECT MORE DRUGS AND SEX



BIG  
KARL MARX GAVE THE PROLETARIAT ELEVEN ZEPPELINS, Y.O.  
SMALL  
MICROSOFT MADE NO PROFIT FROM ANYONE'S ZINES, Y.O.



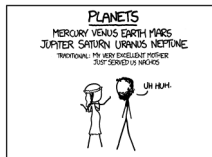
KATY PERRY CLAIMS ORGASMS  
FEEL GOOD SOMETIMES  
OR: KERNEL PEAS ORSH OUR FAMILY GAME SYSTEM



POLYCYSTIC OVARIAN SYNDROME DOES CAUSE PROBLEMS  
THAT JUDDICIOUS CONTRACEPTIVES PARTIALLY NEGATE.



"BIG BROTHER REPTILIAN OVERLORDS," YELLED  
GLENN, "BRAINWASHING VIA GROUND WRITER!!"  
OR: BE BOLD, RESPECT OTHERS, YOU'LL GRADUALLY  
BECOME VERSATILE, GREAT WIMPEDGING!



MARY'S "VIRGIN" EXPLANATION MADE  
JOSEPH SUSPECT UPSTAIRS NEIGHBOR

# A Memory Aid for the SI Prefixes

[deca, hecto], kilo, **mega**, **giga**, **tera**, **peta**, **exa**, **zetta**

**Karl Marx g**ave **the proletariat eleven zeppelins**.

[deci, centi], **milli**, **micro**, **nano**, **pico**, **femto**, **atto**, **zepto**

**MicroSoft made no profit from anyone's zunes**.

Planets:

My very excellent mother just served us nachos.

Mary's "Virgin" explanation made Joseph suspect upstairs neighbor.

Man very early made jars serve useful purposes.

## Powers of Two

Because computers represent information in binary form, it is important to know how many pieces of information can be represented in  $n$  (binary) bits.  $2^n$  pieces of information can be stored in  $n$  bits, and so is it necessary to be familiar with powers of two.

It is obvious that  $\lceil \log_2 n \rceil$  bits are required to represent  $n$  things. Some bit patterns might be unused.

$2^0$	1
$2^1$	2
$2^2$	4
$2^3$	8
$2^4$	16
$2^5$	32

(What does it mean that one thing can be represented with zero bits? There is no need to represent one thing as there is nothing else which can be confused with it.)

In Java, the expression

```
32 - (Math.numberOfLeadingZeros(n-1))
```

will compute the number of bits necessary to represent  $n$  things.

The number of bits necessary to represent a natural number  $n$  is a closely related, but different notion.  $\lfloor \log_2 n \rfloor + 1$  is *not* the same thing as  $\lceil \log_2 n \rceil$ .

$n$	$\log_2 n$	$\lceil \log_2 n \rceil$	$\lfloor \log_2 n \rfloor + 1$
1	0.0	0.0	1.0
2	1.0	1.0	2.0
3	1.6	2.0	2.0
4	2.0	2.0	3.0
5	2.3	3.0	3.0

See [jgloss](#).



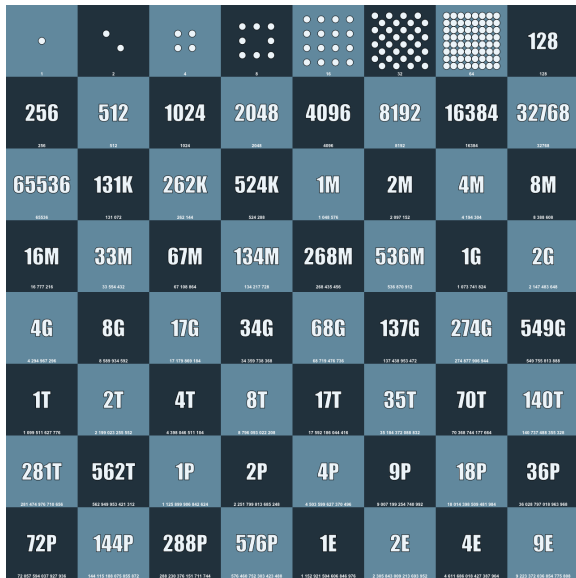
## Legend of the Ambalappuzha Paal Payasam

According to the legend, Lord Krishna once appeared in the form of a sage in the court of the king who ruled the region and challenged him for a game of chess (or chaturanga).

The sage told the king that he would play for grains of rice—one grain of rice in the first square, two grains in the second square, four in the third square, eight in the fourth square, and so on. Every square would have double the number of grains of its predecessor. The king lost the game and soon realized that even if he provided all the rice in his kingdom, he would never be able to fulfill the promised reward. The sage appeared to the king in his true form, that of Lord Krishna. and said he could serve paal-payasam (sweet pudding made of milk and rice) in the temple freely to the pilgrims every day until the debt was paid off.



How many squares in a checkerboard? How big is  $2^{64}$ ?



$10^3$  kilo k  
 $10^6$  mega M  
 $10^9$  giga G  
 $10^{12}$  tera T  
 $10^{15}$  peta P  
 $10^{18}$  exa E

Maybe we can comprehend it better as weight or time. How much does a gain of rice weigh? One grain of rice weighs about 0.029 grams. And, approximately 470 million tons of milled rice is harvested annually in the world today.



0	1	0.029g
1	2	0.058g
2	4	0.116g
3	8	0.232g
4	16	0.464g
5	32	0.928g
6	64	1.856g
7	128	3.712g
8	256	7.424g
9	512	14.848g

10	1 024	29.696g	about an ounce
11	2 048	59.392g	
12	4 096	118.784g	
13	8 192	237.568g	
14	16 384	475.136g	about a pound
⋮			
25	33 554 432	973.079kg	about a ton
⋮			
40	1 099 511 627 776		35 thousand tons
50	1 125 899 906 842 624		36 million tons
60	1 152 921 504 606 846 976		36 billion tons
64	18,446,744,073,709,551,616		589 billion tons

$2^{64}$  grains of rice is the total rice harvest for over 1200 years.

## Powers of Two (Time in Seconds)

Suppose we double 1 second and double the amount of time again.  
And, we do this again and again.

0	1	1 second
1	2	2 seconds
2	4	4 seconds
3	8	8 seconds
4	16	16 seconds
5	32	32 seconds
6	64	about a minute
7	128	about 2 minutes
8	256	about 4 minutes
9	512	about 8 minutes

# Powers of Two

10	1 024	17 minutes
⋮		
19	524 288	one week
20	1 048 576	two weeks
⋮		
25	33 554 432	a year
⋮		
30	1 073 741 824	34 years
40	1 099 511 627 776	37 millennia
50	125 899 906 842 624	
60	1 152 921 504 606 846 976	age of universe

## Powers of Two

Notice that  $2^{10} \approx 10^3$ , so these powers have significance:

10	1 024	$\approx 10^3$	kilo
20	1 048 576	$\approx 10^6$	mega
30	1 073 741 824	$\approx 10^9$	giga
40	1 099 511 627 776	$\approx 10^{12}$	tera
50	1 125 899 906 842 624	$\approx 10^{15}$	peta
60	1 152 921 504 606 846 976	$\approx 10^{18}$	exa
70	1 180 591 620 717 411 303 424	$\approx 10^{21}$	zetta

SI prefixes are not supposed to be used for powers of 2 (just powers of 10).

Sadly, abuse of SI prefixes in computer technology has led to confusion. Whereas 1GHz usually means  $10^9$  instructions per second, 1GB usually means  $2^9$  bytes.

# Powers of Two

Some other powers have special significance in computing.

<i>exponent</i>	<i>bit patterns</i>	
7	128	size of ASCII char set
8	256	size of Latin-1 char set
16	65 536	size of Java <b>short</b>
31	2 147 483 648	no. of neg <b>int</b>
32	4 294 967 296	size of Java <b>int</b>
63	9 233 372 036 854 775 808	no. of neg <b>long</b>
64	18 446 744 073 709 551 616	size of Java <b>long</b>



One challenge in computer science is the vast scale of computing devices.

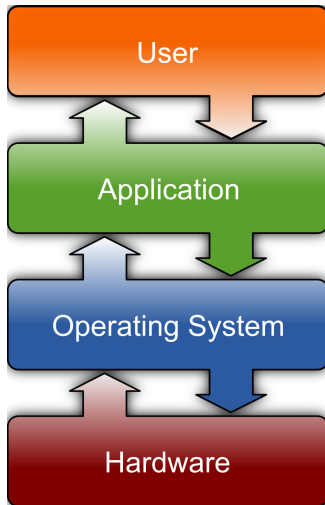
A computer may have a terabyte ( $10^{12}$  bytes) worth of storage. A computer may execute ten instructions every nanosecond ( $10^{-9}$  seconds).

How does one deal with such complexity?

# Layers



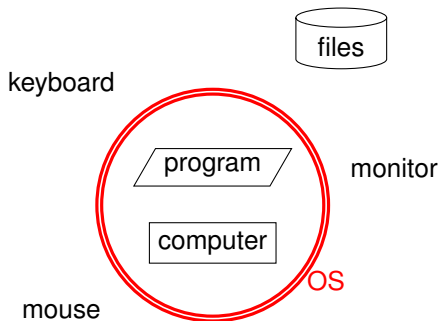
# Interface Layers



# Definitions

- ▶ interface — An *interface* defines the communication boundary between two entities, such as a piece of software, a hardware device, or a user. It generally refers to an abstraction that an entity provides of itself to the outside.
- ▶ API — An *application programming interface (API)* is a set of procedures that an operating system, library, or service provides to support requests made by computer programs.  
For example, the Java API . . . .
- ▶ IDE — In computing, an *Integrated development environment (IDE)* is a software application that provides facilities to computer programmers of a source code editor, a compiler and/or interpreter, build automation tools, and usually a debugger.  
[The definition of IDE does not really belong here.]

# Simple View of Programming



The program controls the computer, yet it needs critical assistance (from the operating system) to communicate with the outside environment and even to run effectively.

Can a Java program print or display a ß (es-zet) character?

Can a Java program draw a red line on the display?

Can a Java program determined which button on the display was clicked or touched?

Can a Java program print or display a ß (es-zet) character?

Can a Java program draw a red line on the display?

Can a Java program determined which button on the display was clicked or touched?

No, not really; yet we write Java programs to these things all the time. In reality the capabilities of the system are determined by the hardware and managed by the operating system.

Nothing can happen without the support of the operating system.

Java APIs abstract external communication. How easy these API's are to use is determined by the skill of the software designers.



For a deeper appreciation of programming a computer, we should examine briefly the many layers upon which the user depends. An important lesson in organizing these complex systems is that the boundaries should be well chosen. Rapidly changing technology, competing business interests, and new insights make it impossible to settle these boundaries once and for all. Whole college classes like computer architecture, operating systems, compiler construction, and programming languages go into the subjects more deeply.

# Hardware and Operating System Platform

Application

System calls: `open()`, `read()`, `mkdir()`, `kill()`



File system

Memory management

**OS:**

Process management

Networking

CPU

Memory

**Hardware:**

Network interface

Monitor

Disk

Keyboard

## Example Platforms

- ▶ Hardware: IBM PowerPC, Intel x86, Sun Ultra-SPARC II
- ▶ OS: Microsoft Windows XP, Mac OS X v10.5 “Leopard”, Linux, Solaris 10

Try:

```
cs> uname -io  
RackMac3, 1 Darwin
```

```
olin> uname -io  
X86_64 GNU/Linux
```

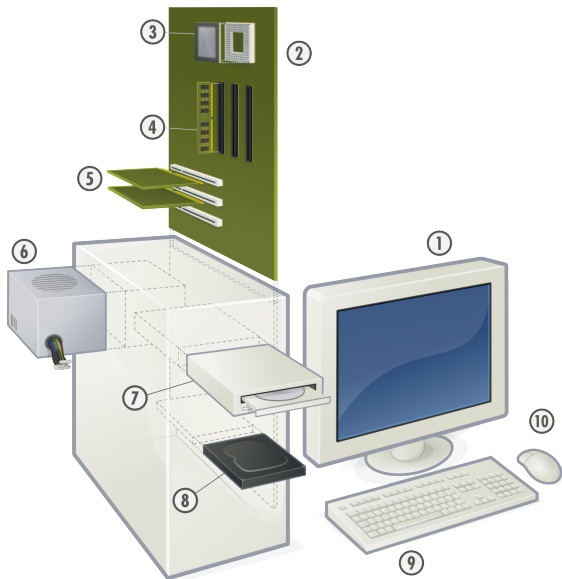
The Java programming language (and other high-level languages) try to form a high-level platform.

Good interfaces mean you don't have to understand the lower layers. For example, you don't have to understand electronic flip-flops in order to program.

The point is:

- ▶ Many interfaces are software constructions, and software interfaces are an important design problem for programmers
- ▶ Many existing interfaces are in flux requiring an understanding of the lower layers.

# Computer Hardware

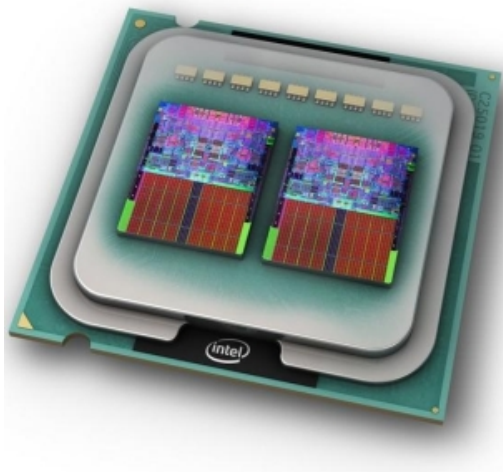


# Computer Hardware



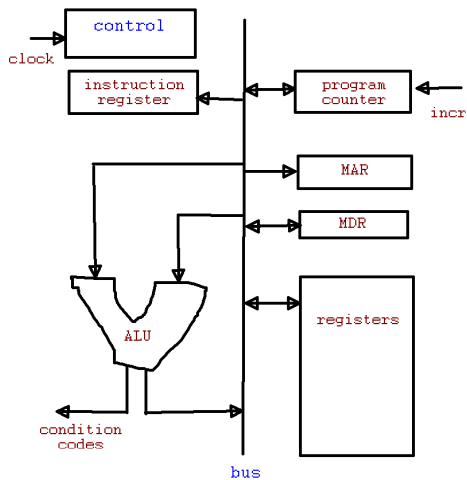
AMD 64X2 dual core

# Computer Hardware



Intel quad

# Computer Architecture—CPU





# Computer Architecture—CPU

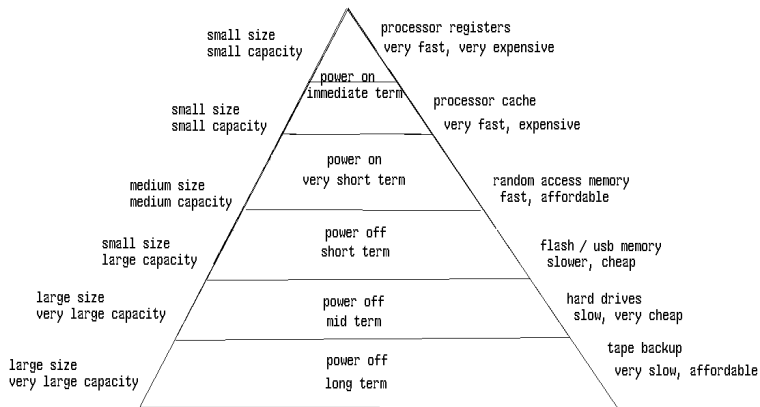
*control unit* is the part of the cpu that controls all the internal actions of the cpu, especially the fetch/execute cycle.

*arithmetic/logic unit (ALU)* is the part of the cpu that does operations: addition, multiplication, etc.

*memory data register (MDR)* is the register of the cpu that contains the data to be stored in the computer's main storage, or the data after a fetch from the storage. It acts like a buffer keeping the contents of storage ready for immediate use by the cpu.

## Computer Memory Hierarchy

by Dan Lash (.com)



# Computer Registers

*Unfortunately, registers are always comparatively few in number, since they are among the most expensive resources in most implementations, both because of the read and interconnection complexity they require and because the number of registers affects the structure of instructions and the space available in instructions for specifying opcodes, offsets, conditions, and so on.*

Muchnick, page 110

# Memory Hierarchy

type	access	size	cost
registers	5ns	1e2	
caches (SRAM)	10ns	1e6	100.00 \$/MB
main memory (DRAM)	100ns	1e9	1.00 \$/MB
hard disk	5000ns	1e11	.05 \$/MB

As the technology improves and the costs go down over time, the typical size of each layer goes up. The ratio in access time between two layers influences the design of the computer hardware. When the ratio changes significantly a different design may achieve better performance.

# Cloud Computing

A final note about computers. The computing platform today is less concerned about the individual computer and more concerned about the network of interconnected computers on the Internet.

*The network is the computer*  
Slogan of Sun Microsystems

Cue *The Network is the Computer*

# Java as a Teaching Language

The subject of this course is programming and teaching programming without any particular programming language does not seem possible. There are good reasons to learn any programming language. There is no good reason to be proficient in one programming language over the rest.

There is no best language for learning the others. So a comprise of different pedagogic, societal, practical, and scientific factors govern the chose of Java.

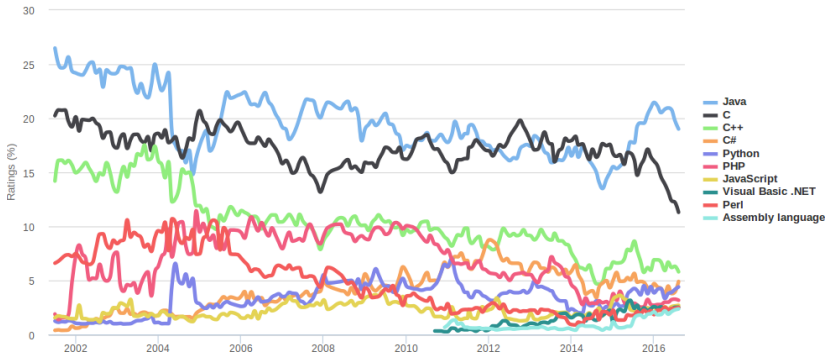
# “Buzz” about Java

Buzz about Java might mean more jobs, more student engagement.

- ▶ TIOBE based on search results
- ▶ RedMonk based on github/stack overflow

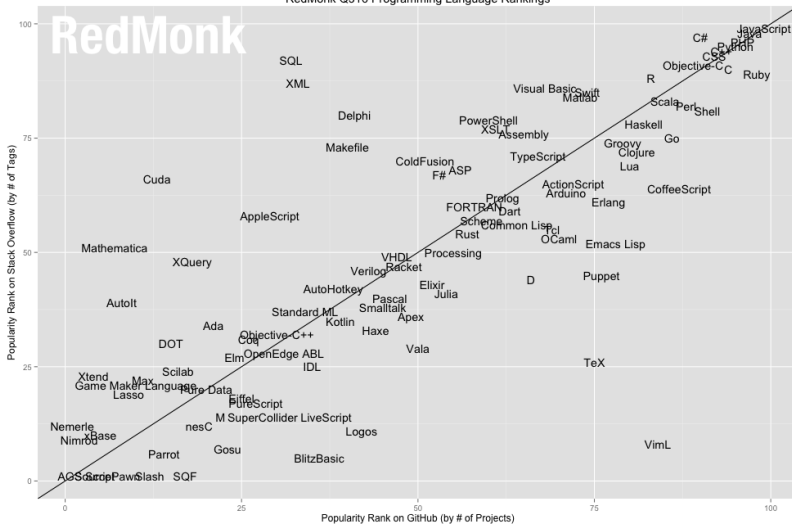
## TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)





RedMonk Q316 Programming Language Rankings



# History of Java

- ▶ 1991: a small group led by James Gosling at Sun Microsystems rejected C/C++ as the basis for digital consumer devices.
- ▶ 1993: Failed in win the contract from Time-Warner for the interactive cable television trial in Orlando, Florida.
- ▶ 1995: WWW, browsers, Java, applets
- ▶ 2009: Oracle buys Sun
- ▶ 2010: Oracle sues Google over Java use in Android Java. “The lawsuit is one battle in a whole war of the mobile industry. Virtually every major player is locked in courtroom battles with another – many fighting on multiple fronts. Software patents, which tend to be broad and subject to multiple interpretations, make for useful tools to bludgeon competitors.”
- ▶ 2014: Java 8 (lambdas added)

enterprise computing

networking

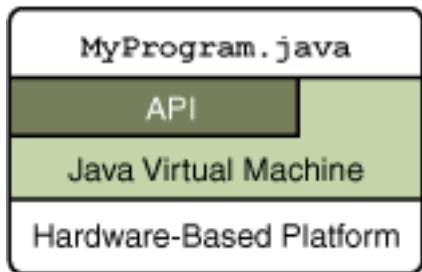
telecommunications

programs

WWW applications

databases

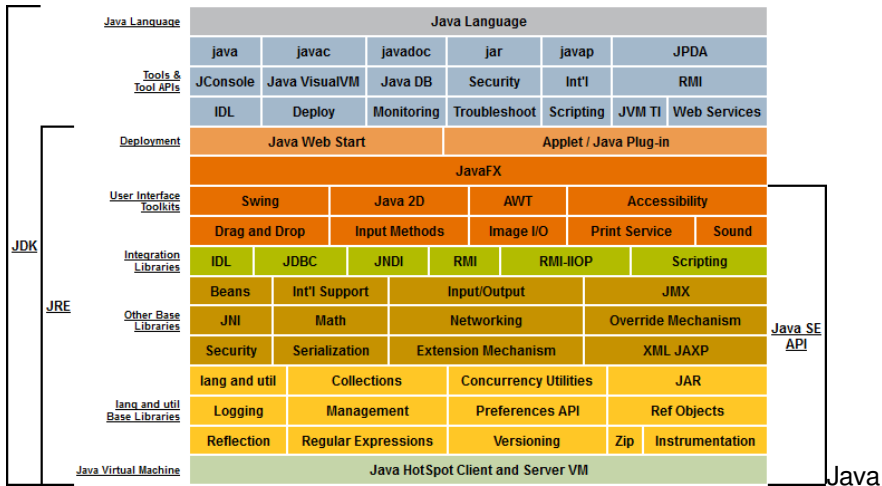




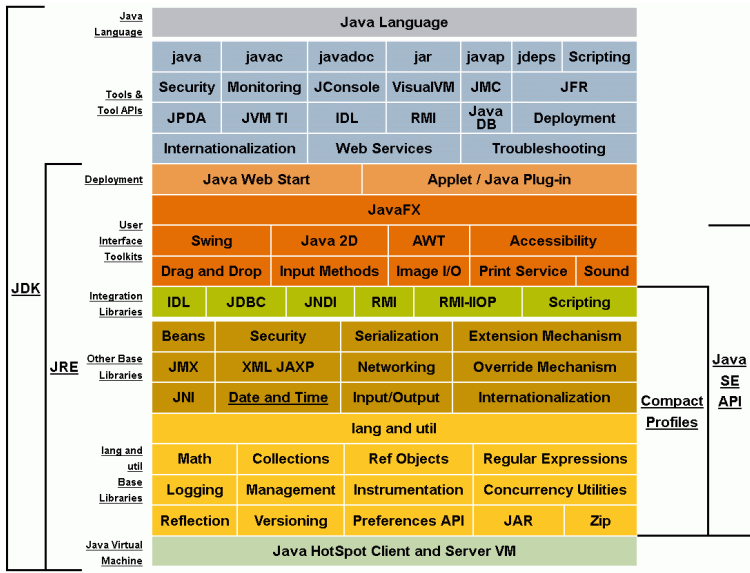
} Java platform

Java platform

overview



## SE platform overview



## Java SE 8 platform overview

Will programming evolve into specialized (and different) skills for telecommunications, enterprise computing, etc, each supported by sophisticated libraries?

Will programming evolve into specialized (and different) skills for telecommunications, enterprise computing, etc, each supported by sophisticated libraries?

I don't think so. Clearly, domain specific knowledge and powerful libraries are useful, but the I think programming will always be the glue that is necessary, independent of the application.



# Java Platform

Some of the major components surrounding Java:

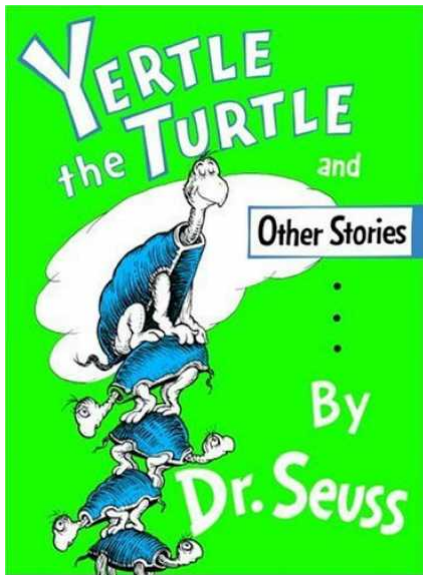
- ▶ Java virtual machine (JVM) specification
- ▶ Virtual machine implementation (for Solaris, Window, and Linux), translation tools (`java` and `javac`), and development tools
- ▶ Java programming language specification
- ▶ A core library (the package `java.lang`), extensive libraries (APIs) for networking, graphics, etc., and additional APIs for special purposes (e.g., telephony)

# Java Platform

Additional components surrounding Java:

- ▶ API documentation  
(No reference material will be given to you. We expect you to go out on the Internet, find it, and know some parts of it in detail.)
- ▶ An IDE for developing Java programs and GUIs: Netbeans  
(We expect you to be able to develop Java programs; we don't explicitly teach using an IDE.)

(In lab and lecture we have other priorities and we expect a lot from you. However, don't be reluctant to ask your classmates, instructors, the help desk, etc., if you have questions. Asking knowledgeable people is still the fastest way to learn.)



Precarious pyramid

- ▶ Software development cycle
- ▶ Files
- ▶ Language translation
- ▶ Streams

See notes elsewhere.

# International Olympiad in Informatics (IOI)

World-wide competitions for pre-college students exist in many subjects. In addition to computer science, competitions are held in math, physics, chemistry, biology linguistics, geography, *inter alia*.

The national organization of the US, USA Computing Olympiad (USACO), trains and selects the team representing the USA.

Unlike the other subjects, it is easy to participate with USACO. They offer on-line practicing, and exciting world-wide contests every month.

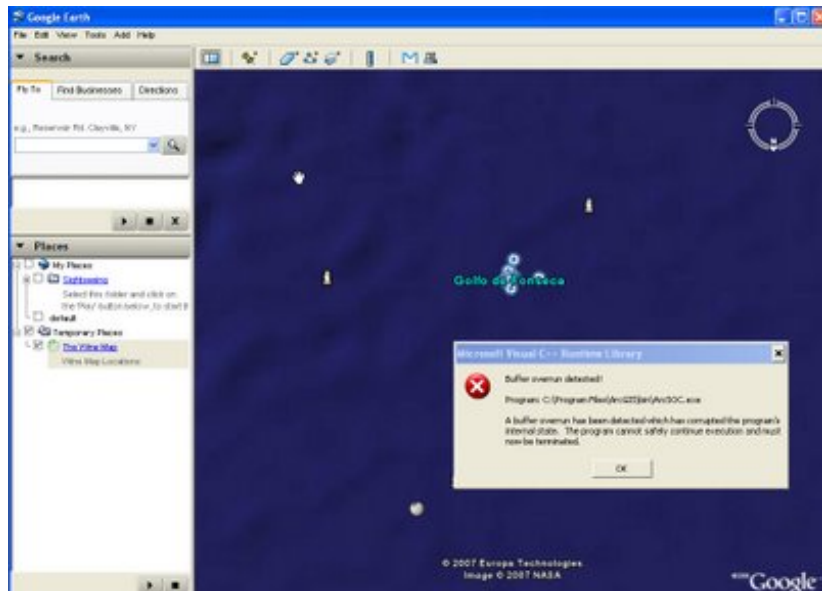
[usaco.org](http://usaco.org)

# Computers Are Misunderstood

- ▶ Buffer overruns in old languages like C/C++ cause risks
- ▶ Intellectual property, copyrights
- ▶ Computer voting, computer money
- ▶ Free software works better
- ▶ Correct software is hard

sunmider sotod  
misstood under  
sdo underomist  
undermisstood  
msiudnesrtodo

# Buffer Overrun



Society has not found the right incentive for the creation of useful and correct software.



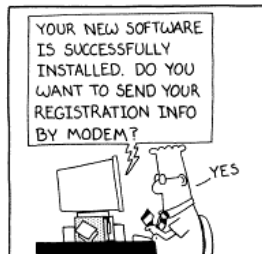
S. Adams E-mail: SCOTTADAMS@AOL.COM



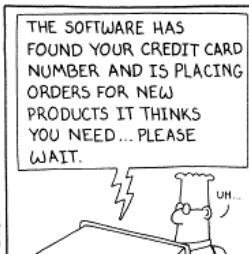
© 1995 United Feature Syndicate, Inc. (NYC)



Copyright © 1995 United Feature Syndicate, Inc.  
Redistribution in whole or in part prohibited



S. Adams E-mail: SCOTTADAMS@AOL.COM



© 1995 United Feature Syndicate, Inc.



Copyright © 1995 United Feature Syndicate, Inc.  
Redistribution in whole or in part prohibited



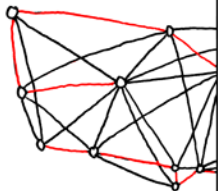
Software is a tool the enables us do things more efficiently and quicker than ever thought possible.



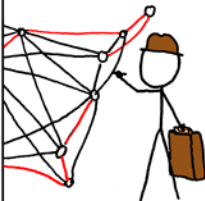
Everyone wants to be the person causing the earth to move. But the lever is really important.

# Traveling Salesman Problem

BRUTE-FORCE  
SOLUTION:  
 $O(n!)$



DYNAMIC  
PROGRAMMING  
ALGORITHMS:  
 $O(n^2 2^n)$



SELLING ON EBAY:  
 $O(1)$

STILL WORKING  
ON YOUR ROUTE?

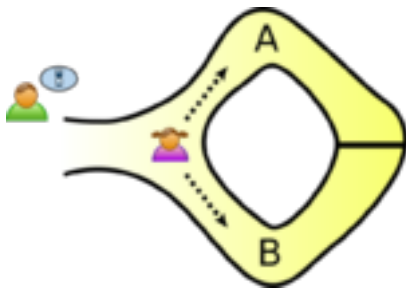
SHUT THE  
HELL UP.



## Zero Knowledge Proof

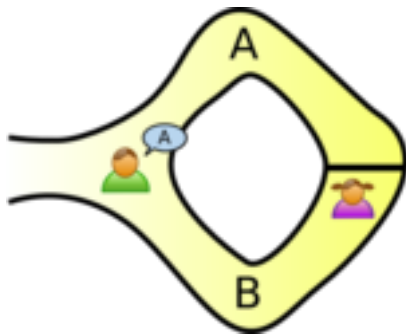
Peggy has a secret word used to open a door in a cave shaped like a circle. Victor says he'll pay her for the secret, but not until he's sure she really knows it. Peggy says she'll tell him the secret, but not until she receives the money. They devise a scheme by which Peggy can prove that she knows the word without telling to Victor.

First, Victor waits outside while Peggy goes out of sight and stands at one side or the other of the door. Victor does not see which way she goes, nor can he hear or watch Peggy if she opens the door.



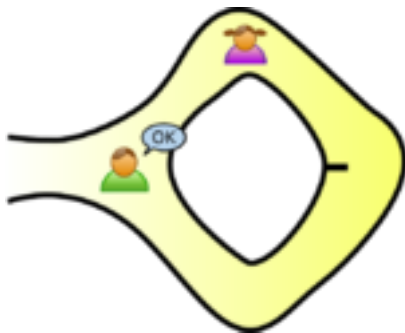
# Zero Knowledge Proof

Second, Victor randomly calls out a path for Peggy to take back to the mouth of the cave.



# Cryptography

Third, Peggy takes the indicated path proving she may have opened the door.



With a 50% probability, Peggy has returned by using the password. Victor can repeat the test as long as it takes to convince himself that Peggy indeed knows the password. Each test decreases the chance the Peggy got lucky by standing on the side of the door which Victor will pick.