

CS/ENGRD 2110  
(FORMERLY CS 211)  
FALL 2009

Lecture 1: Overview  
<http://courses.cs.cornell.edu/cs2110>

## Announcements

- Please take a look at the course web site
- All lectures will be posted online but we tend to revise them at the last minute.
- Assignment 1 (of 5) is up, due 9/9. Start early!
- Need a Java refresher? Check out the CS 1130 web site (link on the 2110 web site)
- CS colloquium: Andy Wilson (Microsoft) – 4:15pm “Surface Computing” Thursday, 8/27 Upson B17.

## Announcements

- CS colloquium: Surface Computing; Andy Wilson (Microsoft) – 4:15pm Thur. 8/27 Upson B17.
  - Not “aimed” at people just getting started in CS
  - But fascinating anyhow!
  - Each week we have a different famous speaker
- Added reason to attend?
  - You can get 1 credit if you sign up to attend weekly
  - Free food from 3:45-4:15 before the talk in the 4<sup>th</sup> floor Upson Computer Science atrium

## Getting to Details: Course Staff

- Instructor
  - Ken Birman
  - [ken@cs.cornell.edu](mailto:ken@cs.cornell.edu)
- Administrative Assistant
  - Bill Hogan
  - [whh@cs.cornell.edu](mailto:whh@cs.cornell.edu)
- More contact info
  - See [Staff](#) button on website



## Course Staff

- Teaching Assistants
  - Lead sections (“recitations”, “discussions”) starting next week
  - Act as your main contact point
- Consultants
  - In Upson 360, hours online
  - “Front line” for answering questions
  - consulting hours start next week
- More info?
  - See [Staff](#) on website

## Lectures

- TR 10:10-11am, Olin 155
  - Attendance is mandatory
- ENGRD 2110 or CS 2110?
  - **Same course! We call it CS 2110**
  - **Non-engineers sign up for CS 2110**
  - **Engineers sign up for ENGRD 2110**
- We often make last minute changes to the notes
- Readings and examples will be posted online together with lecture notes
  - Unlike the lecture notes, these won't change often



## Sections



- Like lecture, attendance is mandatory
- Usually review, help on homework
- Sometimes new material
- Section numbers are different for CS and ENGRD
- Each section will be led by a member of the teaching staff
- No permission needed to switch sections
- You may attend more than one section if you wish

## Sections

Non-Eng	Eng	Day	Time	Room
5741	DIS 201	T	12:20PM - 01:10PM	HLS 110
5743	DIS 202	T	01:25PM - 02:15PM	HLS 306
5745	DIS 203	T	02:30PM - 03:20PM	HLS 306
5747	DIS 204	W	12:20PM - 01:10PM	HLS 306
5749	DIS 205	W	01:25PM - 02:15PM	HLS 306
5751	DIS 206	W	02:30PM - 03:20PM	BRD 140
5753	DIS 207	T	12:20PM - 01:10PM	PHL 219

## CS2111 (formerly 212)

- Not offered anymore
- CS 2111 used to be a special “extra” project for CS majors, but now everyone does the same project

## Resources

- Course web site
  - ▣ <http://courses.cs.cornell.edu/cs2110>
  - ▣ Watch for announcements
- Course newsgroups
  - ▣ cornell.class.cs2110, cornell.class.cs2110.talk
  - ▣ Good place to ask questions (carefully)

## Resources


- Book: Frank M. Carrano, *Data Structures and Abstractions with Java, 2<sup>nd</sup> ed.*, Prentice Hall
  - ▣ *Note: The 1<sup>st</sup> edition is pretty obsolete by now*
  - ▣ If you somehow end up with it, consult 2<sup>nd</sup> edition!
  - ▣ Copies of 2<sup>nd</sup> Edition on reserve in Engr Library
- Additional material on Prentice Hall website
- Great Java resource: the online materials at the Sun JDK web site. Google has it indexed.

## Obtaining Java

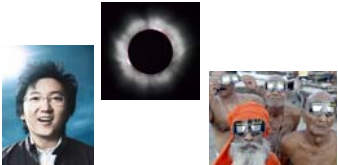


- See Resources on website
- Use Java 6 if you can
  - ▣ Java 5 is ok, but why not upgrade?
- Need Java Development Kit (JDK), not just Java Runtime Environment (JRE)
- Many production releases... latest is usually best

## Eclipse IDE



- IDE: Interactive Development Environment
  - ▣ Helps you write your code
  - ▣ Protects against many common mistakes
  - ▣ At runtime, helps with debugging



## Eclipse IDE



- We **highly** recommend use of an IDE
- Eclipse tutorial will be discussed in section
- Use version 3.4 (Ganymede) if you can
  - ▣ V 3.3 (Europa) is ok, but why not upgrade??
- See [Resources](#) on website



*"In my country of Kazakhstan everyone is use Eclipse!  
Excellent for hack American web site and steal credit  
card. Also very good for to meet girls."*

## Java Help

- CS 2110 assumes basic Java knowledge
  - ▣ classes, objects, fields, methods, constructors, static and instance variables, control structures, arrays, strings, exposure to inheritance
- Need a refresher?
  - ▣ **CS 1130, Transition to Object-Oriented Programming**
  - ▣ formerly 101J
  - ▣ self-guided tutorial, material on website

## Academic Excellence Workshops

- Two-hour labs in which students work together in cooperative setting
- *One credit S/U course based on attendance*
- Time and location TBA
- See the website for more info

**[www.engineering.cornell.edu/student-services/learning/academic-excellence-workshops/](http://www.engineering.cornell.edu/student-services/learning/academic-excellence-workshops/)**

## Coursework

- 5 assignments involving both programming and written answers (45%)
- Two prelims (15% each)
- Final exam (20%)
- Course evaluation (1%)
- Occasional quizzes in class (4%)

## Assignments

- Except for assignment A1, assignments may be done by teams of two students
  - ▣ A1 is already posted on CMS
- We encourage you to do them by yourself
- Finding a partner: choose your own or contact your TA. Newsgroup may be helpful.
- Monogamy is usually the wisest policy
- Please read partner info and Code of Academic Integrity on website

### Academic Integrity... Trust but verify!



- We use artificial intelligence tools to check each homework assignment
  - ▣ The software is very accurate!
  - ▣ It tests your code and also notices similarities between code written by different people
- Sure, you can fool this software
  - ▣ ... but it's easier to just do the assignments
  - ▣ ... and if you try to fool it and screw up, you might fail the assignment or even the whole course.

### Using Java, but not a course on Java!

#### Introduction to computer science, software engineering

- Concepts in modern programming languages
  - ▣ recursive algorithms and data structures
  - ▣ data abstraction, subtyping, generic programming
  - ▣ frameworks and event-driven programming
- Algorithm analysis and designing for efficiency
  - ▣ asymptotic complexity, induction
- Concrete data structures and algorithms
  - ▣ arrays, lists, stacks, queues, trees, hashables, graphs
- Organizing large programs

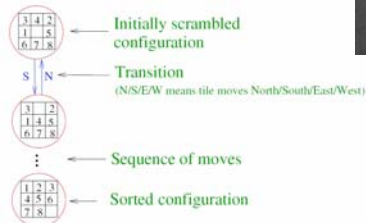
### Lecture Topics

- Introduction and Review
- Software Engineering concepts
- Recursion and induction
- Object-oriented concepts: data abstraction, subtyping
- Data structures: Lists and trees
- Grammars and parsing
- Inheritance and frameworks
- Algorithm analysis, Asymptotic Complexity
- Searching and Sorting

### More Lecture Topics

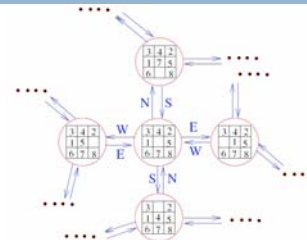
- Generic Programming
- Data Structures
  - ▣ Sequence Structures: stacks, queues, heaps, priority queues
  - ▣ Search Structures: binary search trees, hashing
  - ▣ Graphs and graph algorithms
- Graphical user interface (GUI) frameworks
  - ▣ Event-driven programming
  - ▣ Concurrency and simple synchronization

### Sam Loyd's 8 Puzzle



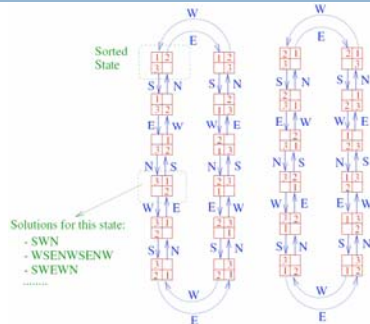
- Goal: Given an initial configuration of tiles, find a sequence of moves that will lead to the sorted configuration.
- A particular configuration is called a **state of the puzzle**.

### State Transition Diagram of 8-Puzzle



**State Transition Diagram:** picture of adjacent states.  
A state Y is **adjacent** to state X if Y can be reached from X in one move

## State Transition Diagram for a 2x2 Puzzle



## Graphs

- State Transition Diagram in previous slide is an example of a **graph**: a mathematical abstraction
  - **vertices** (or **nodes**): (e.g., the puzzle states)
  - **edges** (or **arcs**): connections between pairs of vertices
  - vertices and edges may be labeled with some information (name, direction, weight, cost, ...)
- Other examples of graphs: roadmaps, airline routes, . . .
  - A common vocabulary for problems

## Path Problems in Graphs

- Is there a path from node A to node B?
  - Solve the 8-puzzle
- What is the shortest path from A to B?
  - Find fastest way to solve the 8-puzzle
  - Or the Google Maps / Mapquest problem
- Traveling salesman problem
- Hamiltonian cycles

## « Simulating » the 8-puzzle

- What operations should puzzle objects support?
- How do we represent states?
- How do we specify an initial state?
- What algorithm do we use to solve a given initial configuration?
- How should we present information to the user? (GUI design)
- How to structure the program so it can be understood, maintained, upgraded?

## Why you need CS 2110

- You will be able to design and write moderately large, well-structured programs to simulate such systems.
- Computer systems are complex. Need CS to make them work; can't just hack it
- Selected software disasters:
  - CTAS air traffic control system 1991-present
  - Ariane 5 ex-rocket
  - Denver airport automated baggage handling
  - German parliament
  - and dare I say ... *PeopleSoft*?

## Why you need CS 2110, cont'd

- Fun and intellectually interesting: cool math ideas meet engineering (and make a difference)
  - Recursion, induction, logic, discrete structures, ...
- Crucial to any engineering or science career
  - Good programmers are 10x more productive
  - Leverage knowledge in other fields, create new possibilities
  - Where will you be in 10 years?

## Why you need CS 2110, cont'd

- Real systems are large, complex, buggy, bloated, unmaintainable, incomprehensible.

Year	Operating System	Millions of lines of code*
1993	Windows NT 3.1	6
1994	Windows NT 3.5	10
1996	Windows NT 4.0	16
2000	Windows 2000	29
2001	Windows XP	40
2005	Windows Vista	50

- Commercial software typically has 20 to 30 bugs for every 1,000 lines of code†

\*source: Wikipedia

†source: CMU CyLab Sustainable Computing Consortium

## Moore's Law

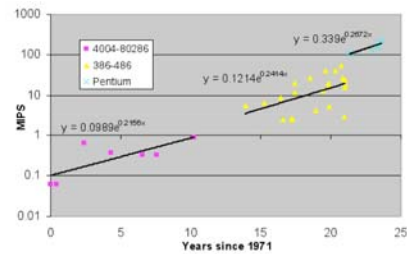


Figure 5: Processor performance in millions of instructions per second (MIPS) for Intel processors, 1971-1995.

- From *Lives and death of Moore's Law*, Ilkka Tuomi, 2002

## Grandmother's Law

- Brain takes about 0.1 second to recognize your grandmother
  - About 1 second to add two integers (e.g. 3+4=7)
  - About 10 seconds to think/write statement of code
- Your brain is not getting any faster!
  - So better make those 10 seconds count!
  - If you write code ineffectively, or can't understand why your own code will work... it won't work!
  - Learning to think "algorithmically" really helps!

## Anyhow... don't you want to get rich?



- Moore's Law won't continue forever... But it probably will for a while.
- Let's invent some cool ways to use all that computing power!
  - Your brain never doubles in speed, so don't count on producing exponentially more code!
  - But we do get smarter, and can work in teams. So we can think of clever uses of computers... go where no man (or woman) has gone before...

## The universal tool!

- Computer science* is becoming a universal tool
  - Algorithms that answer fundamental questions
  - Data structures to represent immense amounts of information
  - Programming languages that let you express your goals more clearly
  - Insights into what is (and is not) possible
- Once, mathematics / physics played this role.
- Today *computational thinking* is a key part of every exciting story



We hope you have fun, and enjoy programming as much as we do