CS 211 (ENGRD 211) Summer 2007

Lecture 1: Overview

http://www.cs.cornell.edu/courses/cs211

Course objectives

- An introduction to CS and software engineering
- Our focus is not on Java, nor on programming
- But we use Java to implement and test CS ideas
- Almost everything in CS211 is applicable to other languages, and even to other fields

Prerequisites

- CS 100J or something similar
- We assume basic Java knowledge
  - classes, objects, fields, methods, constructors, static and instance variables, control structures, arrays, strings, exposure to inheritance
- Need review?
  - ***Java Refresher/Bootcamp***
    - self-guided tutorial—material (including solutions) on website (Help & Software)
    - Live help in Upson B7, Mon 7/25 and Tue 7/26, 7**:00-9:00pm**
    - **Same material both days**

Lectures

- MTWRF 10:00-11:15am
- Attendance is mandatory
- Lecture notes will be online, along with reading assignments and examples
- ENGRD 211 or CS 211?
  - **Same course**! We call it CS211
  - Non-engineers sign up for CS 211
  - Engineers sign up for ENGRD 211

Course staff

- Instructor:
  - David Crandall
- Teaching Assistants:
  - Nikos Karampatziakis, Raghu Ramanujan
- Consultants
  - Jonathan Hui, Noah Santorello
- See website for office hours, contact info

Course Resources

- Course web site
  http://www.cs.cornell.edu/courses/cs211
  - Announcements, course info, online resources…
- Course newsgroups
  cornell.class.cs211, cornell.class.cs211.talk
  - Great place to ask questions (carefully)
- Required textbook
  Mark Weiss, *Data Structures and Problem Solving using Java (3/E)*, 3rd ed., Addison-Wesley

Grading

- Five assignments (10% each)
- Prelim (15%)
- Final exam (25%)
- Occasional quizzes in class (10%)

7

Assignments

- Written and programming parts
- Some assignments may be done by teams of two students
  - You may choose to do them by yourself
  - Choose your own partner. Newsgroup may be helpful.
  - Monogamy encouraged
- Late policy
  - Up to 24 hours late, with a 20% penalty
- Mandatory reading: partner info and Code of Academic Integrity on website

8

Exams

- Prelim exam (15% of grade)
  - Tentative date: Tuesday July 17
  - Let me know ASAP if you have a conflict
- Final exam (25% of grade)
  - Tuesday August 7, 8:00am

9

Quizzes

- 5-10 unannounced quizzes
- Idea is to reward attendance, give you (and me) feedback on your understanding of material
  - You will earn some points just for attempting the quiz
  - You will receive a 0 if you're not in class
  - No make-up quizzes
  - Lowest quiz score will be dropped

10

Obtaining Java

- We do not require an IDE
  - But most people use one
- See **Help & Software** under **Java Resources** on website
- Use Java 5 (aka 1.5.0_10)
- Do not use Java 1.6!
  - Still in beta

11

Academic integrity

- Read and understand AI code on course website
- We will look for and prosecute AI violations
- Be especially diligent about assignments

"You may discuss homework problems with other students at a high level. That is, you may discuss general approaches to a problem, high-level algorithm design, and the general structure of code. However, you may not share written code with any other students except your own partner, nor may you possess code written by another student who is not your partner, either in whole or in part, regardless of format." -- CS211 website

12

CS212

- ***CS 212: Java Practicum***
- 1 credit project course
- Substantial project
- 1 lecture per week
- Required for CS majors; recommended for others
- Best to take 211 and 212 in the same semester

13

Succeeding in 211

- CS211 moves quickly, especially in the summer

"The course is extremely demanding; only students who are willing to commit themselves to its rigors and who have met the prerequisite should enroll." -- Cornell Summer Session website

- Be proactive: take full advantage of resources
  - Required and supplementary readings (see website)
  - Course staff office hours
  - Newsgroups
  - Online resources (see website)

14

Course topics

- Introduction and review
- Part 1: Recursion (1 week)
  - Recursion and induction
  - Grammars and parsing
  - Exceptions
- Part 2: Object-oriented concepts (1 week)
  - Objects, classes, references
  - Inheritance, subtyping
  - Abstract classes, interfaces
  - Object oriented design principles

15

Course topics

- Part 3: Algorithms and data structures (2 weeks)
  - Lists, trees
  - Algorithm analysis and asymptotic complexity
  - Searching, sorting algorithms
  - Generic programming, ADTs
  - Stacks, queues, heaps, priority queues
  - Binary search trees, hashing
- Part 4: Graphs and graph algorithms (1 week)
  - Graph concepts
  - Minimum spanning trees, Prim's algorithm
  - Shortest path, traveling salesman problems

16

Course topics

- Part 5: Graphical user interfaces (1 week)
  - Event-driven programming
  - Threads and synchronization

17

Computer Science

- CS is much more than just programming!

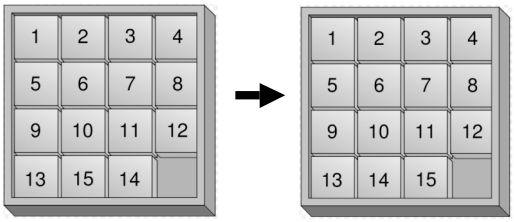
"Computer science is no more about computers than astronomy is about telescopes." --Edsger Dijkstra

- Study very general problems and techniques to solve them efficiently
  - And how to frame a specific problem in terms of one of these general problems
- How to design code that is efficient, maintainable, secure, scalable, …

18

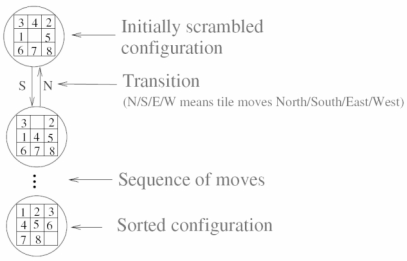
An example: the 15-puzzle

- Invented by Chapman in 1874



19

8-Puzzle

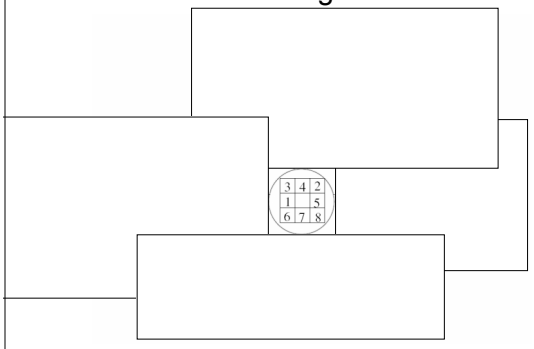


Goal: Given an initial configuration of tiles, find a sequence of moves that will lead to the sorted configuration.

A particular configuration is called a ***state*** of the puzzle.

20

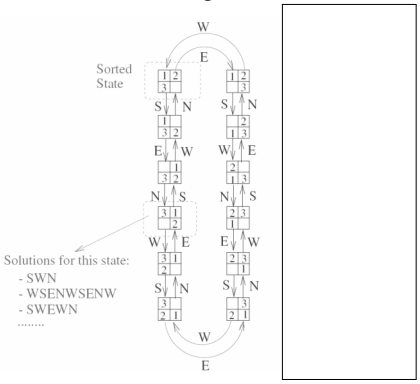
State Transition Diagram of 8-Puzzle



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.

21

State Transition Diagram for a 2x2 Puzzle



22

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, …)

23

Graphs

- Many practical problems can be posed as graph problems, e.g.:
  - Airline scheduling
  - Trip routing
  - Ranking web pages
  - Studying social networks
  - Image enhancement
  - Project planning
  - Many more!

24

Graph Problems

- Is there a path from node A to node B?
  - Solve the 15-puzzle
- What is the shortest path from A to B?
  - Solve the 15-puzzle efficiently
- Traveling salesman problem
- Hamiltonian cycles
- etc…

25

Simulating the 15-puzzle

(implementation issues)

- 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?
- What kind of GUI should we design?
- How to structure the program so it can be understood, maintained, upgraded?

26

Why you need CS 211

- 211 will give you the background and tools you need to write programs that solve real problems
- Fun and intellectually interesting: cool math ideas meet engineering
  - Recursion, induction, logic, discrete structures, …
- Crucial to any engineering or science career
  - Good programmers are >10x more productive
  - Leverages knowledge in other fields, makes new possibilities
  - Where will you be in 10 years?

27

Why you need CS211, cont'd

- Real computer systems are complex.
  - You need CS and software engineering to make them work; can't just hack them together.
- Selected software disasters:
  - Therac-25 (1985-1987)
  - AT&T long distance outage (1990)
  - Denver automated baggage system (1995-2005)
  - FBI Virtual Case File (2000-2005)

28

Why you need CS211, cont'd

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

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 Beta 2	50

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

*source: Wikipedia
†source: CMU CyLab Sustainable Computing Consortium

29

Moore's Law

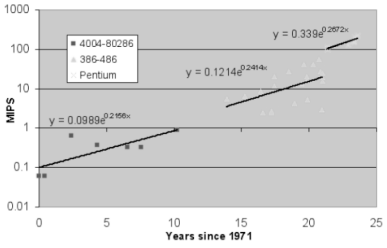


Figure 5: Processor performance in millions of instructions per second (MIPS) for Intel processors, 1971-1995.

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

30

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!

31

Motivation

- Computers double in speed every 18 months
  - Software doubles in size every M Years
  - Data doubles in size every N Years
  - Your brain never doubles in speed
  - But we do get smarter, and can work in teams
- Computer science is increasingly important
  - Better algorithms
  - Better data structures
  - Better programming languages
  - Better understanding of what is (and is not) possible

32