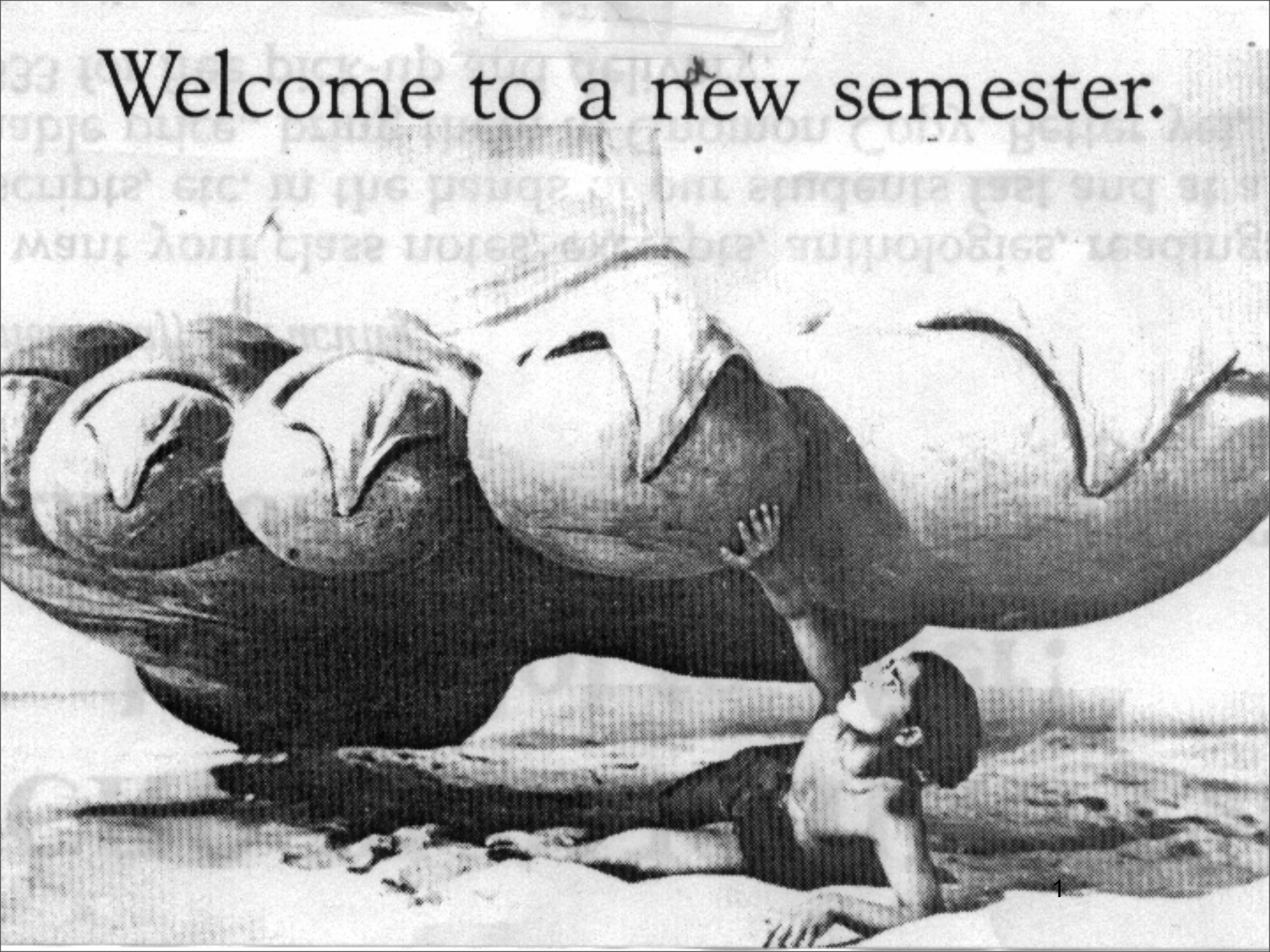
Welcome to a new semester.



CS/ENGRD 2110 (formerly CS 211) Fall 2008

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
- Assignment 1 (of 5) is up, due 9/10. Get started early!
- Need a Java refresher? Check out the CS 1130 web site (link on the 2110 web site)
- Luc Vincent talk today, 4:15, Upson B17: Google Maps

Course Staff

Instructor

Dexter Kozen
kozen@cs.cornell.edu

Administrative Assistant

Kelly Patwell
patwell@cs.cornell.edu

More contact info

See Staff 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 will occasionally make small last minute changes to the notes
- Readings and examples will be posted online together with lecture notes

Sections

CS	ENGRD					
4633	9282	DIS 201	T	12:20-1:10	OLH 165	Morgan, Ashwin
4634	9283	DIS 202	T	1:25-2:15	HLS 110	Atul, Bobby
4635	9284	DIS 203	T	2:30-3:20	OLH 165	Jonathan, Parvati
4636	9285	DIS 204	W	12:20-1:10	HLS 206	Fang, Zoe
4637	9287	DIS 205	W	1:25-2:15	UPS 205	Sonica, Parvati
4638	9290	DIS 206	W	2:30-3:20	UPS 205	Fang, Morgan
4639	9291	DIS 207	T	12:20-1:10	HLS 110	**Canceled -** please register for DIS 201

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

CS2111 (formerly 212)

CS 2111: Programming Practicum

- 1 credit project course
- Substantial project (IthacaQuest)
- No formal lectures
- Required for CS majors; recommended for others
- OK to take 2111 concurrently with 2110 or after (but not before)

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)
- Textbook: Frank M. Carrano, *Data Structures and Abstractions with Java*, 2nd ed., Prentice Hall (1st edition is obsolete!)
- Additional material on the Prentice Hall website

Obtaining Java

- See Resources on website
- Use Java 6 if you can
- Java 5 is ok
- Need Java Development Kit (JDK), not just Java Runtime Environment (JRE)
- Latest production release is 1.6.0_7, but we have been using 1.6.0_10 with good results

Eclipse

- We highly recommend use of an IDE
- Eclipse tutorial in section
- Use version 3.4 (Ganymede) if you can
- Version 3.3 (Europa) is ok
- See Resources on website

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

Coursework

- 5 assignments involving both programming and written answers (45%)
  - We AI check each homework assignment
  - The software is very accurate!
- Two prelims (15% each)
- Final exam (20%)
- Course evaluation (1%)
- Occasional quizzes in class (4%)

Assignments

- Assignments may be done by teams of two students (except for A1)
  - A1 is already posted on CMS
- You may choose to do them by yourself
- Finding a partner: choose your own or contact your TA. Newsgroup may be helpful.
- Monogamy encouraged
- Please read partner info and Code of Academic Integrity on website

Course Objectives

An introduction to computer science and 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, hashtables, graphs
- Organizing large programs

Using Java, but not a course on Java!

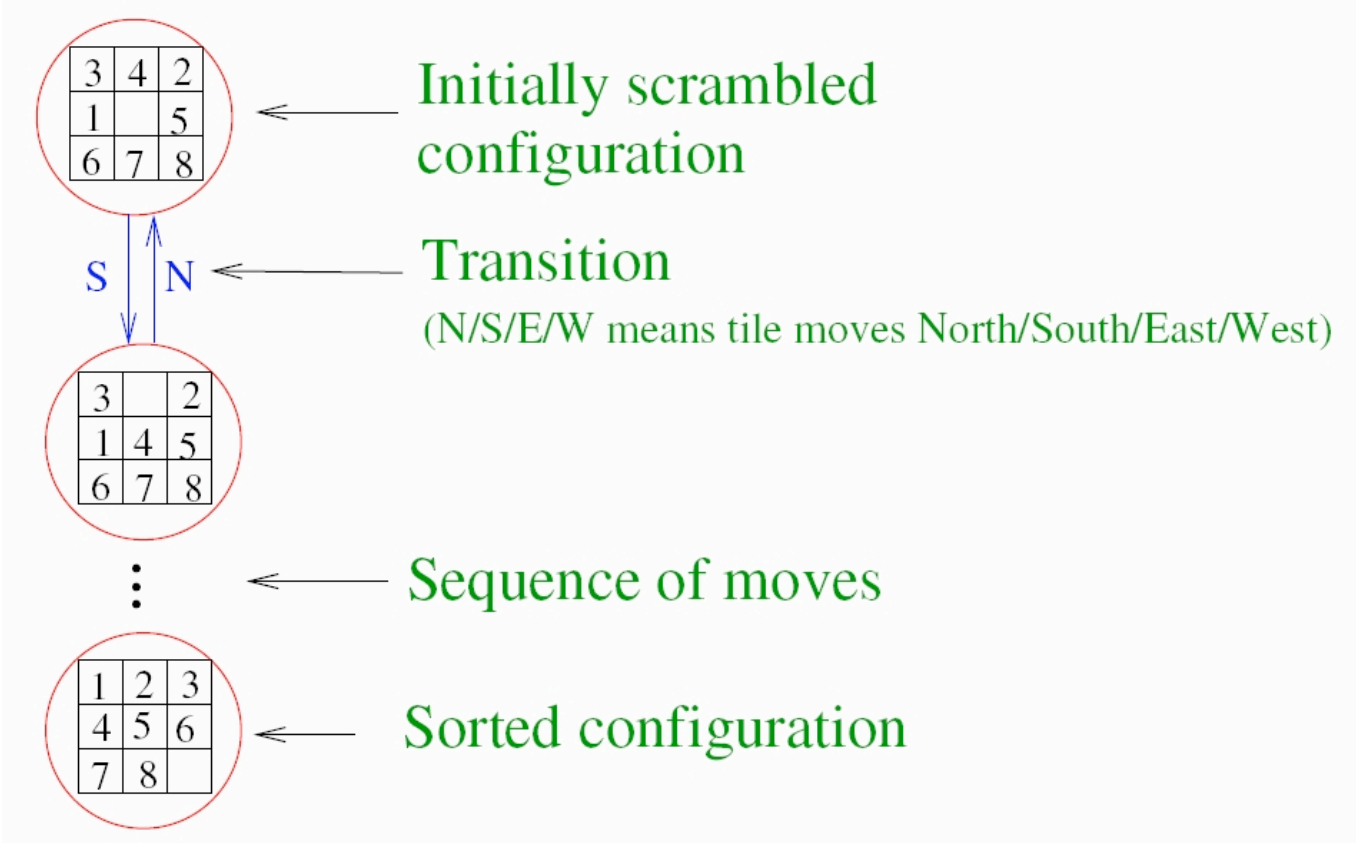
Lecture Topics

- 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

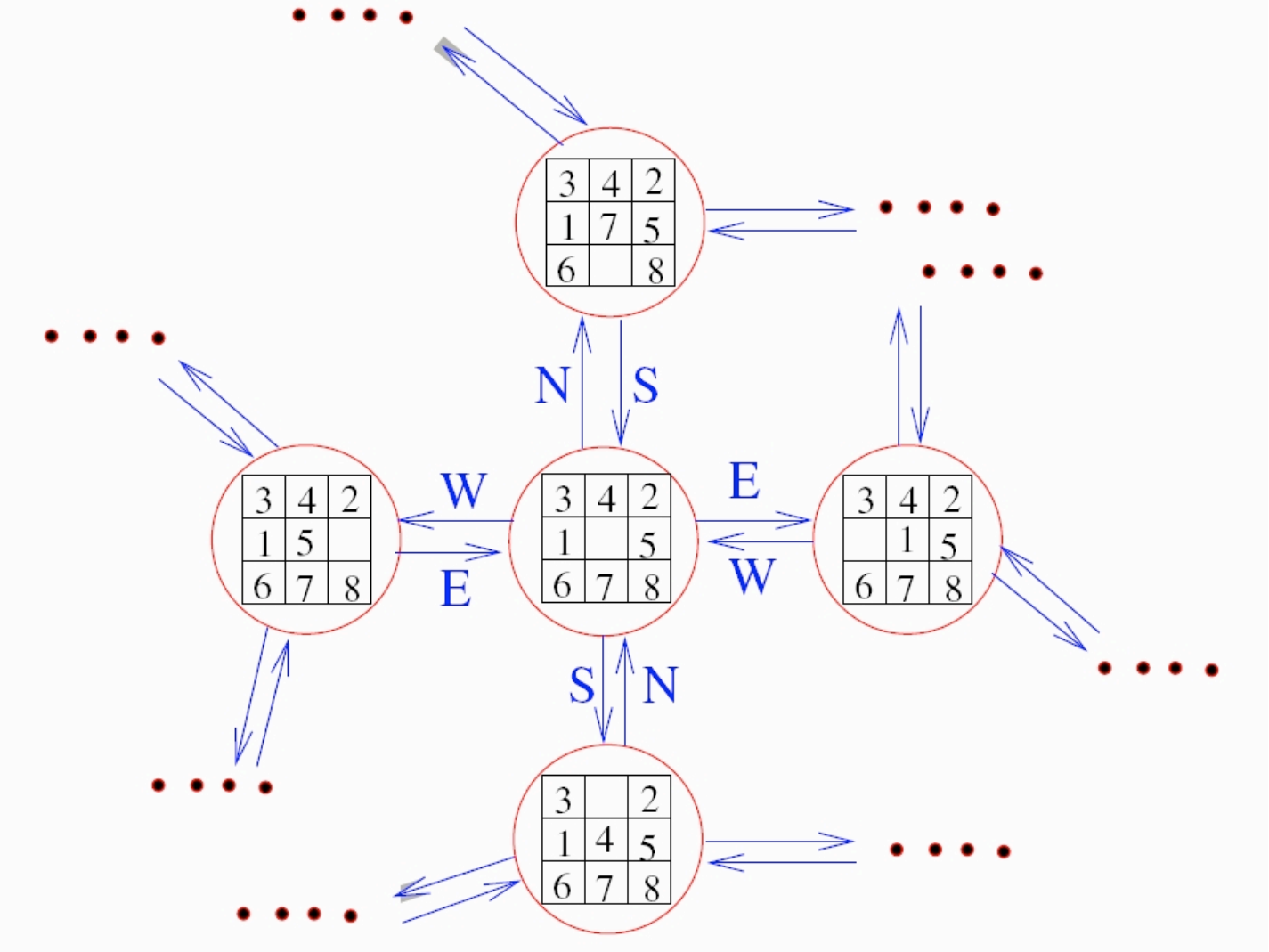
Sam Loyd's 8 Puzzle



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.

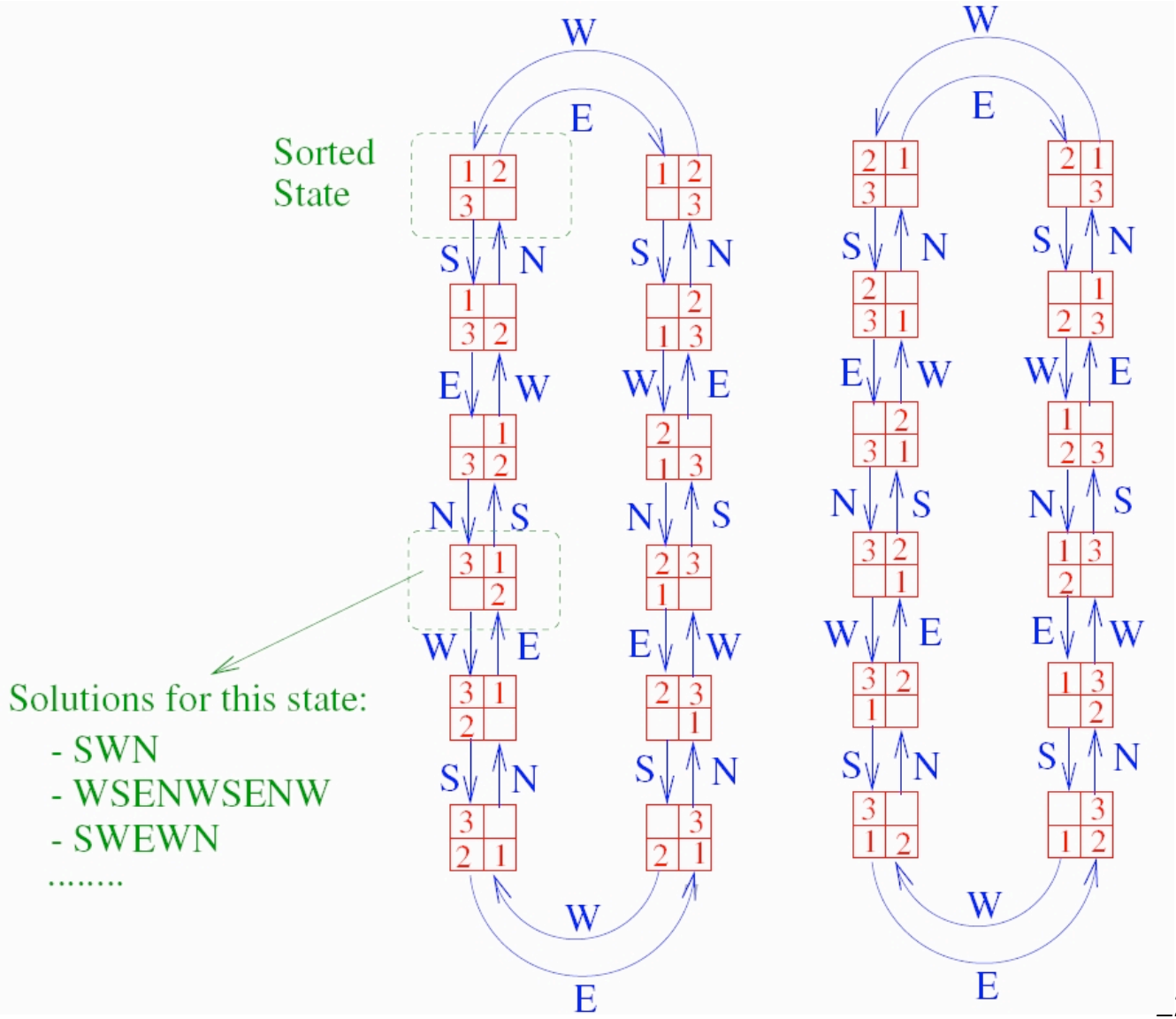
State Transition Diagram of 8-Puzzle



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: airline routes, roadmaps, . . .
  - 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?
  - 8-puzzle (efficiently)
  - MapQuest
- 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

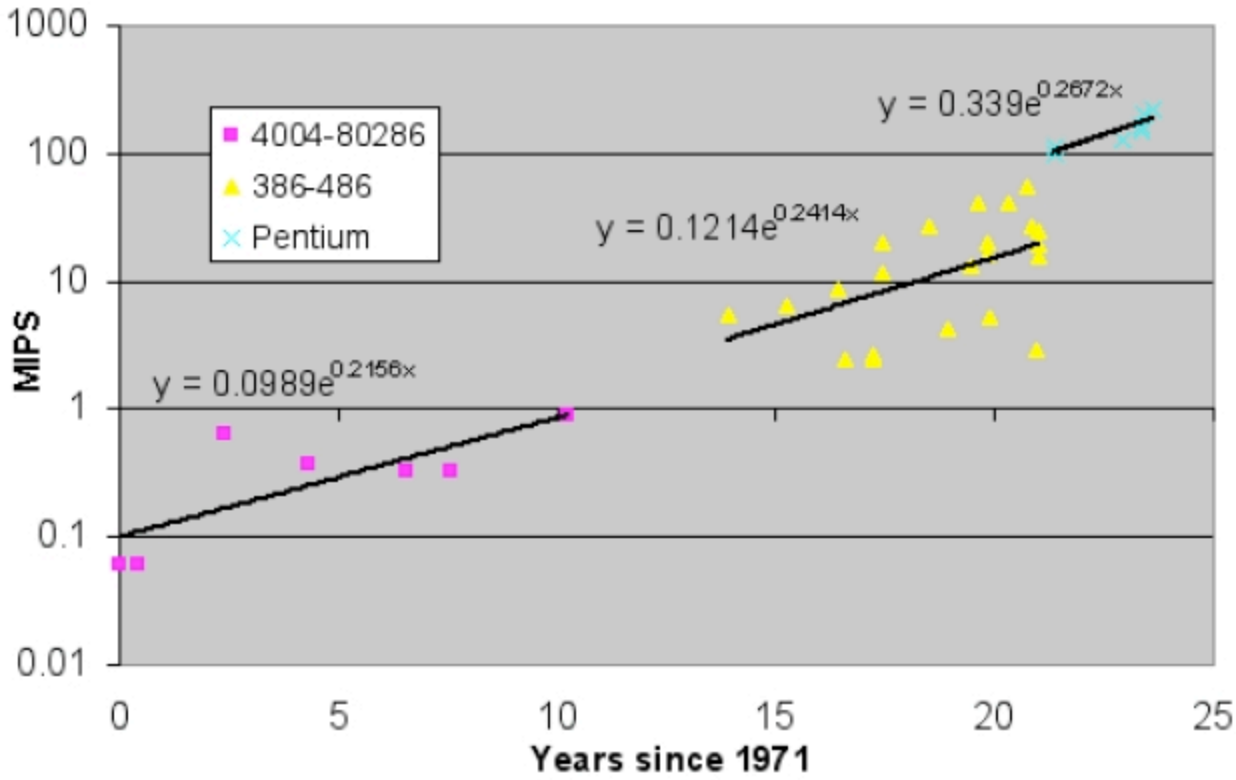


Figure 5: Processor performance in millions of instructions per second (MIPS) for Intel processors, 1971-1995.

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!

Motivation

- Moore's Law will not continue forever...
- 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

Welcome!

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