CS 211

Computers and Programming

http://www.cs.cornell.edu/courses/cs211/2005su

Lecture 1: Introduction
Summer 2005

Announcements

- Assignment 1 is up
- Read/Review Chapters 1-3 of Weiss
- Java Boot Camp is *tomorrow* 7-10pm in Upson B7

Course Staff

Instructor

- Mohan Rajagopalan

Teaching Assistants

- Jeff Hartline
- Peter Sirokman

They're available in office hours

Consultants

- Alec Bernston
- Mehmet Saglam

There are consulting hours every day, where students can come in for help. See the web page for times, and the location of the consulting room. (Upson 328, I think)

Lectures and Readings

- M-F 10:00-11:15am, Upson111
- Attendance is **mandatory!**
- Lecture notes will be online, either immediately before or after the lecture
- The Textbook:
  Data Structures & Problem Solving Using Java, Mark Allen Weiss, Addison-Wesley, third edition.
- Read the book! Readings assignments will posted online
  - Today's is Ch 1-3
  - Tuesday and Wednesday are Chapter 7

Structure

- Monday-Thursday will be lecture days
- Fridays will be “section” style meetings. There will be a quiz, and the plenty of time afterwards for general questions and answers (usually about the current assignment)
- Assignments are weekly and will usually be due Tuesdays in class
- There are two in-class prelims on Tuesday July 12th and Wednesday July 27th
- The Final Exam is August 9th, 8-10am

Assignments

- Assignments will have two parts: a written portion and a programming portion.
- You make work on and submit the programming portion in pairs.
- You must submit the written portion individually, although you may discuss it with your programming partner for the current assignment.
- Submit the written portion in class, and the programming portion through CMS.

Java Bootcamp

- This is not a Java programming course.
- We will not be teaching general Java programming, and we expect you to have basic Java competence.
- If you have little or no Java knowledge, or think you need a refresher, you should attend the Java Bootcamp
- Bootcamp will be run by the consultants, Alec and Mehmet
- Time and place: Tuesday, June 28, Upson B7 7-10pm

Java

- We use Java 5.0 for this class. Make sure your personal machines are running it. We won't be using the latest features of Java 5 immediately, but soon.

- We recommend DrJava, an open source program as your IDE (Interactive Development Environment). It will make programming much easier.

- You may use another IDE if you like

- DrJava has some weirdness with Java 5. It should work fine in the labs, but you may have trouble installing it on your personal machines. If that happens, you can use Java 4 instead (temporarily)

Grades

- 6 assignments involving both programming and written answers: 44% of grade (1-5: 7% each, 6: 9%)
- Two prelims: 15% of grade each
- Final exam: 25% of grade
- Course evaluation: 1% of grade
- These weights may change!

Objectives of CS 211

- Concepts in modern programming languages
  - recursion, induction
  - classes, objects
  - inheritance, interfaces
- Efficiency of programs
- Data structures: arrays, lists, stacks, queues, trees, hash-
tables, graphs
- Software engineering: How to organize large programs

This is not a course on Java programming!

Lecture Sequence

Part 1

- Introduction
- Induction and Recursion
- Grammars and Parsing
- Object-Oriented Programming (OOP)
- Inheritance and Interfaces

Lecture Sequence

Part 2

- Lists
- Trees
- Searching and Sorting
- Complexity (Efficiency/Running Time)
- Abstract Data Types (ADTs)

Lecture Sequence

Part 3

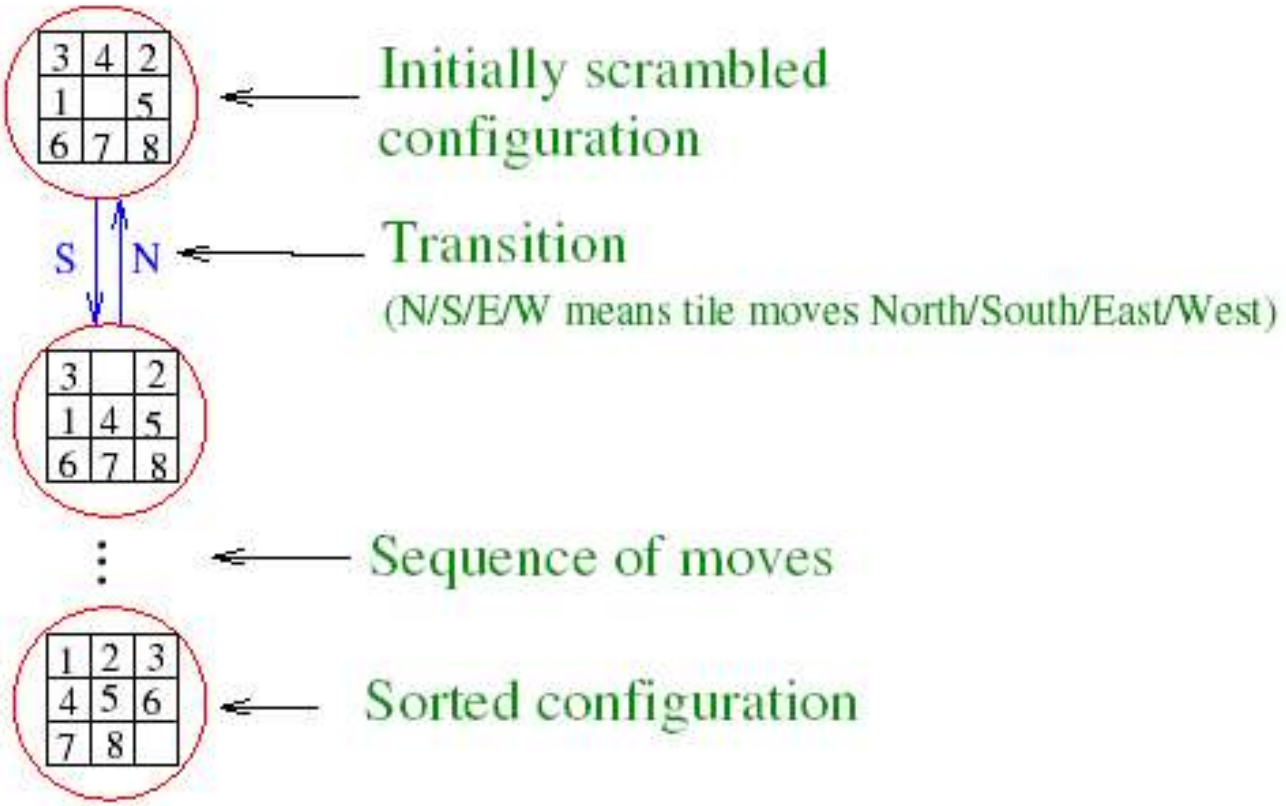
- Graphs
- Generic Programming
- Search Trees
- Other Algorithms

An Example: The 15-puzzle

1	2	3	4
5	6	7	8
9	10	11	12
13	15	14	

- http://www.javaonthebrain.com/java/puzz15/

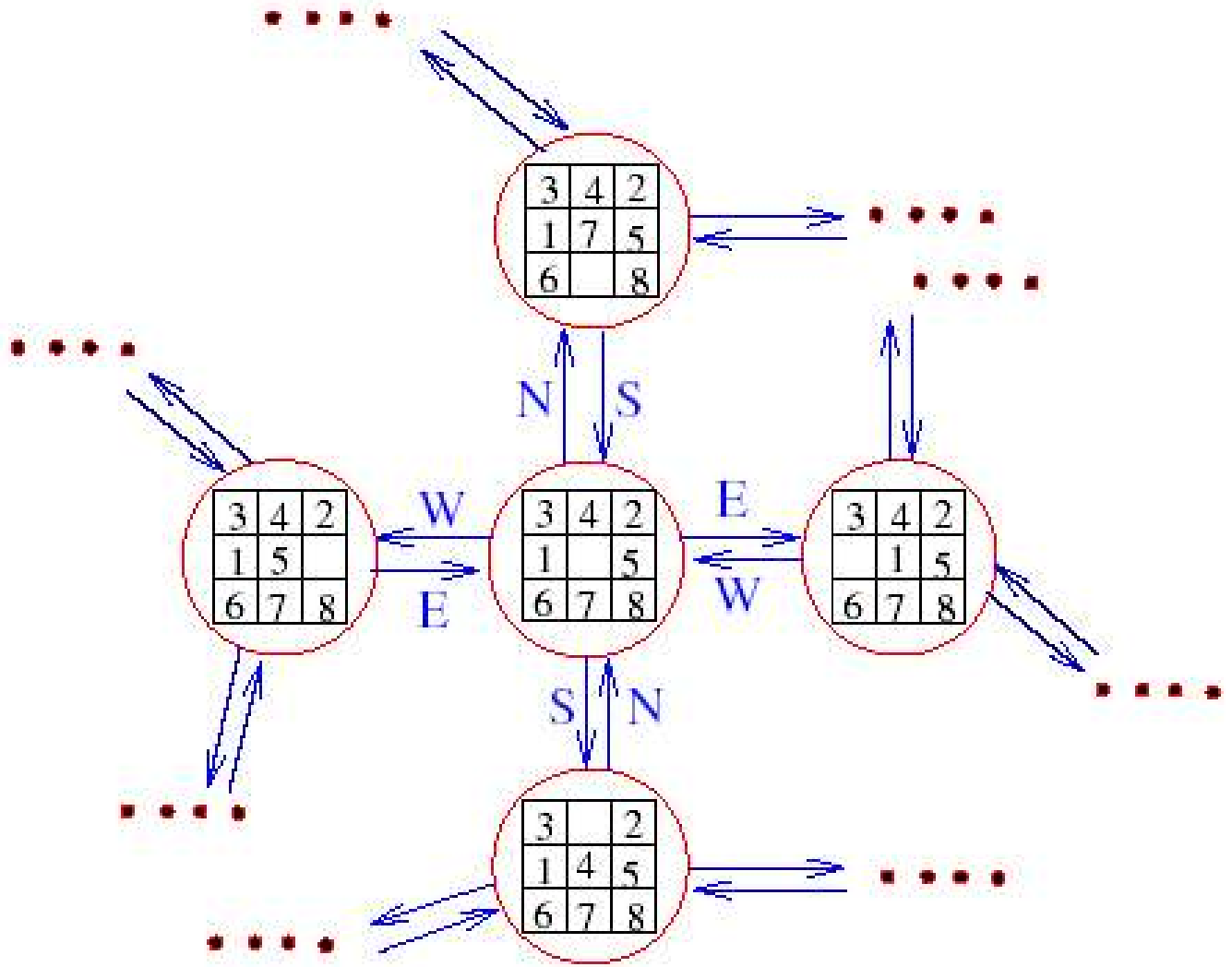
Sam Loyd's 8-puzzle



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.

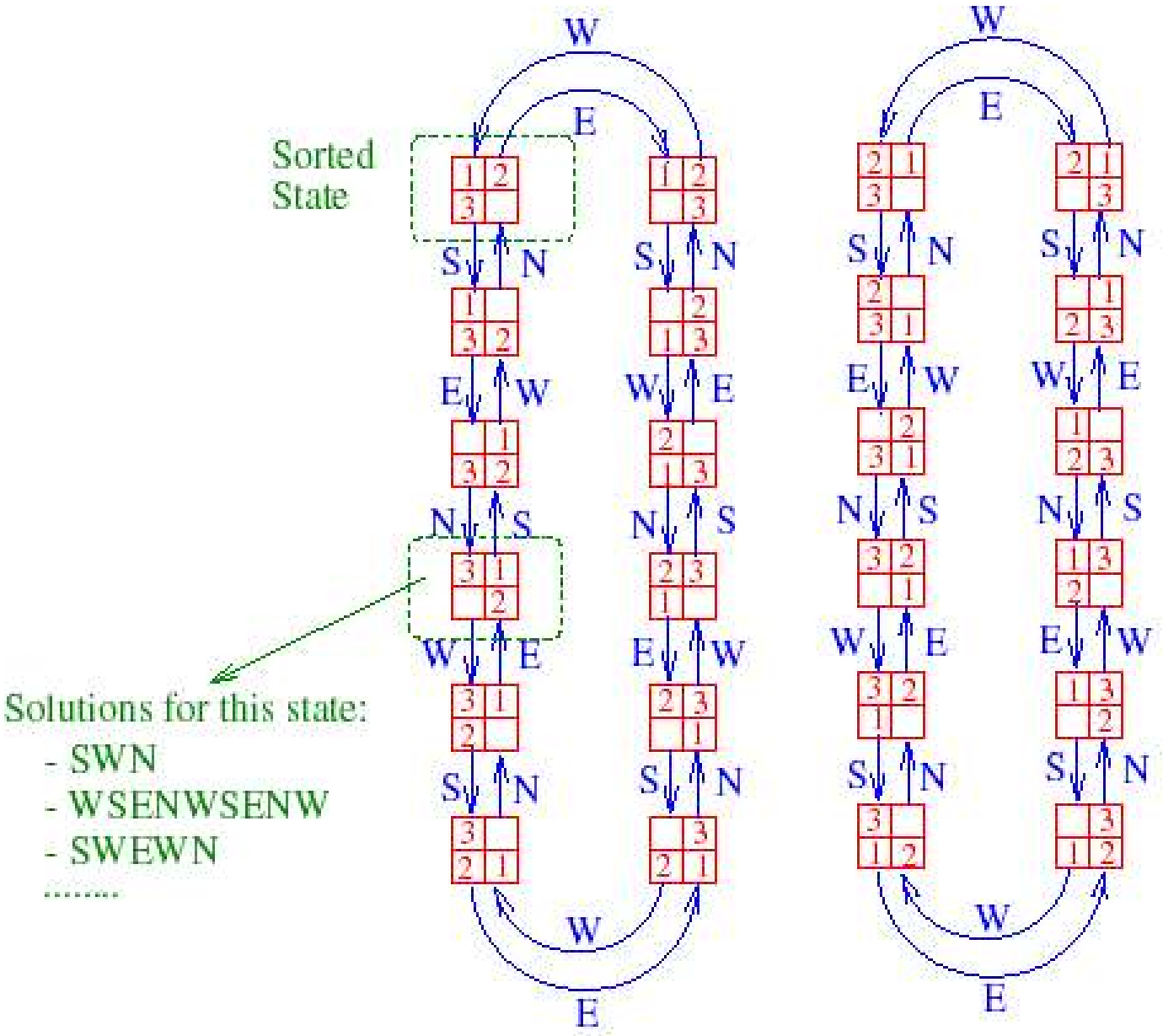
State Transition Diagram of 8-puzzle



A state Y is ADJACENT to state X if Y can be reached from X in one move

State Transition Diagram: picture of adjacent states

State Transition Diagram for a 2x2 Puzzle



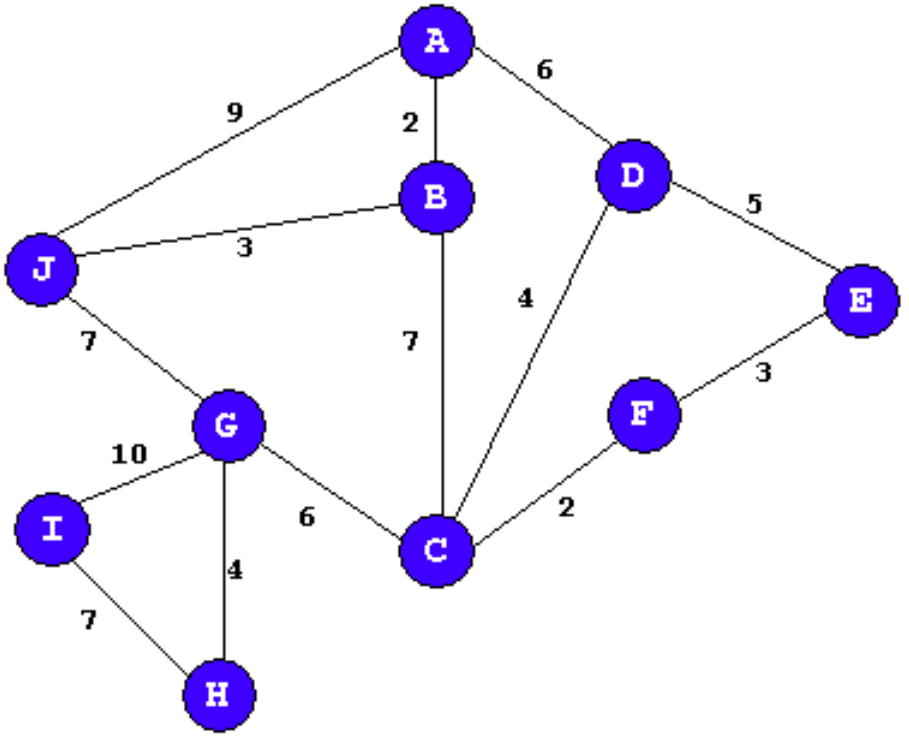
Graphs

- State Transition Diagram in previous slide is an example of a graph

- Graph has
  - vertices (or nodes): in our example, these are the puzzle states
  - edges (or arcs): connections between pairs of vertices
  - vertices and edges may be annotated with some information (like edge "weights")

- Other examples of graphs: airline routes, roadmaps, links on the internet. . .

Another Graph



Graph Problems:

- Is there a path from node A to node B?
- What is the shortest path from A to B?
- Traveling salesman problem
- Hamiltonian cycles
- . . . will see later in semester

Data Structures and Algorithms

Java – Brief Review

- What's so special about Java? Lots of things, but a few things stand out (for this class):
  - It's object-oriented
  - It's platform independent
  - Passing values is easy (no "pointers" to worry about)
  - It has a nice library

Classes and Objects

A program usually consists of several classes. Each class has some fields and some methods. A typical class declaration looks like

```
class ClassName {

   field declarations;

   method(parameters) {
     local variable declarations;
     body;
   }

   method(parameters) {
     local variable declarations;
     body;
   }
}
```

main() and fields

The main class has a main method, which is always declared as

```
public static void main(String[] args) {
   ... body ...
}
```

- static fields
  - belong to a class, shared by all objects of the class.
  - declared using keyword "static“
- instance fields
  - belong to objects of a class
  - a different one for each object

An Example

```
class Widget {

    static int nextSerialNumber = 10000;
    int serialNumber;

    Widget() {
        serialNumber = nextSerialNumber++;
    }

    Widget(int sn) {
        serialNumber = sn;
    }

    public static void main(String[] args) {
        Widget a = new Widget();
        Widget b = new Widget();
        Widget c = new Widget();
        Widget d = new Widget(42);
        System.out.println(a.serialNumber);
        System.out.println(b.serialNumber);
        System.out.println(c.serialNumber);
        System.out.println(d.serialNumber);
    }
}
```

Methods

- Methods can be static or instance
  - static methods may not refer to instance fields or instance methods
  - static methods can be called even if no objects of the class have been created

- Parameters, local variables of a method
  - exist only while the method is running
  - use parameters to pass input data to a method
  - use local variables for temporary data used in a method
  - use fields for persistent data or data shared by several methods

Referencing

- Reference fields, methods in own class by name
  - serialNumber, nextSerialNumber
- Reference static fields in another class by qualified name
  - Widget.nextSerialNumber
- Reference fields of objects with object nam
  - a.serialNumber

Example:

System.out.println(a.serialNumber)

  - out is a static field in class System
  - value of out is an instance of a class that has a method println(int)

Other Class and Object stuff

Overloading: look at String.valueOf(...) in Java API
- there are 9 of them, one for each of 9 different argument types
- think of argument types as part of the name of the method

Class hierarchy
- arranged in a tree -- at the root (top) is Object
- e.g. String and StringBuilder are subclasses of Object
- methods and fields of superclass are available in subclass

Reference vs Primitive types

Primitive types
- int, long, float, byte, char, boolean, ...
- take a single word or 2 words of storage
- not objects in the Java sense
- variable of that type contains the actual data

Reference types
- arrays and objects (e.g. String, StringBuffer, Vector)
- usually take more storage
- variable of that type contains a POINTER to the actual data
- null is a reference type

== vs. equals()

use == for primitive types
use == for reference types (e.g. Strings) ONLY if you mean actual identity of the two objects (note that this is ALMOST NEVER what you want!)
E.g. can have two different strings, both with value "hello"
use equals() instead, e.g. x.equals("hello") instead of x == "hello".

Arrays

- Arrays are reference types

- Elements of arrays can be reference or primitive
  - e.g. int[] or String[]
  - if a is an array, a.length is its length
  - elements are a[0], a[1], ..., a[a.length - 1]

- Multidimensional arrays are really arrays of arrays
  - e.g. int[][] is an array of integer arrays
  - can be "ragged"; e.g. all the arrays in the 2nd dimension need not be the same length

Array Example

```
class MultiArray {

   static int[][][] a = new int[2][3][];

   public static void main(String[] args) {
      for (int i = 0; i < a.length; i++) {
         for (int j = 0; j < a[i].length; j++) {
            a[i][j] = new int[i+j];
         }
      }
      for (int i = 0; i < a.length; i++) {
         for (int j = 0; j < a[i].length; j++) {
            System.out.println(a[i][j].length);
         }
      }
   }
}

> java MultiArray
0
1
2
1
2
3
```

Arrays, Vectors, Hashtables

- Array
  - storage is allocated when they are created, cannot change

- Vector (in java.util)
  - "extensible" arrays -- can grow as needed
  - can append or insert elements, access i-th element, reset to 0 length
  - can get an enumeration of the elements

- Hashtable (in java.util)
  - save data indexed by keys
  - can lookup data by its key
  - can get an enumeration of the elements

Hashtable example

- Create a hashtable of numbers, using the names of the numbers as keys:

```
Hashtable numbers = new Hashtable();
numbers.put("one", new Integer(1));
numbers.put("two", new Integer(2));
numbers.put("three", new Integer(3));
```

To retrieve a number:

```
Integer n = (Integer)numbers.get("two");
if (n != null) {
    System.out.println("two = " + n);
}
```

Caveat: returns null if does not contain the key
- need to check in order to avoid null pointer exception

Command Line Interface

- Arguments are contained in the String array parameter main()

```
class CmdLineArgs {

   public static void main(String[] args) {
      System.out.println(args.length);
      for (int i = 0; i < args.length; i++) {
         System.out.println(args[i]);
      }
   }
}
```

```
> java Foo
0
> java Foo asdf zxcv ss
3
asdf
zxcv
ss
> java Foo hello world
2
hello
world
```

- Try your programs in a command window, not just in DrJava, because that's how we'll be testing them. Behavior may be a little different.