Announcements

- Due Dates
  - Monday, March 17, 11pm
    - Project 3B
    - Required Lab 14
    - Extra-Credit Lab 13

Announcements

- Last week of class!
- No final exam!

Announcements

- Labs this week
  - Thursday-Friday
    - Quiz on Chapter 17
    - TA evaluations
    - Project 3B work time
    - Pick up Reflection paper 3

Announcements

- Lecture this week
  - Friday
    - Reflection paper 4
    - Wrap-up
    - Course evaluations for lecture/instructor

Calculating Your Grade

- Excel spreadsheet

Computers Can Do Almost {Everything, Nothing}

Limits to Computation

Can Computers Think?

- The Turing Test:

Alan Turing in 1950

  - Two identical rooms labeled A and B are connected electronically to a judge who can type questions directed to the occupant of either room. A human being occupies one room, and the other contains a computer. The judge's goal is to decide, based on the questions asked and the answers received, which room contains the computer. If after a reasonable period of time the judge cannot decide for certain, the computer can be said to be intelligent.
  - The computer is intelligent if it acts enough like a human to deceive the judge.

Can Computers Think? (cont'd)

- Passing the Test
  - Test sidesteps definition of thinking or intelligence
  - Does not focus on any specific ability
  - Advances in the last half century:
    - Parsing grammatical structure of natural language
    - Machine translation of natural language
    - Recognizing semantically meaningful information

Acting Intelligently?

- Eliza, the Doctor Program, was programmed to ask questions in dialog like a psychotherapist and a patient
- Took word cues, noticed use of negatives
- Dialog was essentially pre-planned to appear intelligent, but was not
- *Artificial Intelligence (AI):* To exhibit intelligence, computer has to "understand" a complex situation and reason well enough to act on its understanding (no scripting)

Playing Chess

- Clean task: Clear rules and definition of success (beat a grand master)
- The Board Configuration

Figure 23.1. A chessboard configuration.

Playing Chess (cont'd)

- The Game Tree
  - To decide on move, computer explores moves to determine whether the move will make it better or worse off
  - *Evaluation Function*: Assigns a numerical value to each piece and computes a score for the move. Positive score is better; negative score is worse
    - Computer checks Evaluation Function on every legal move

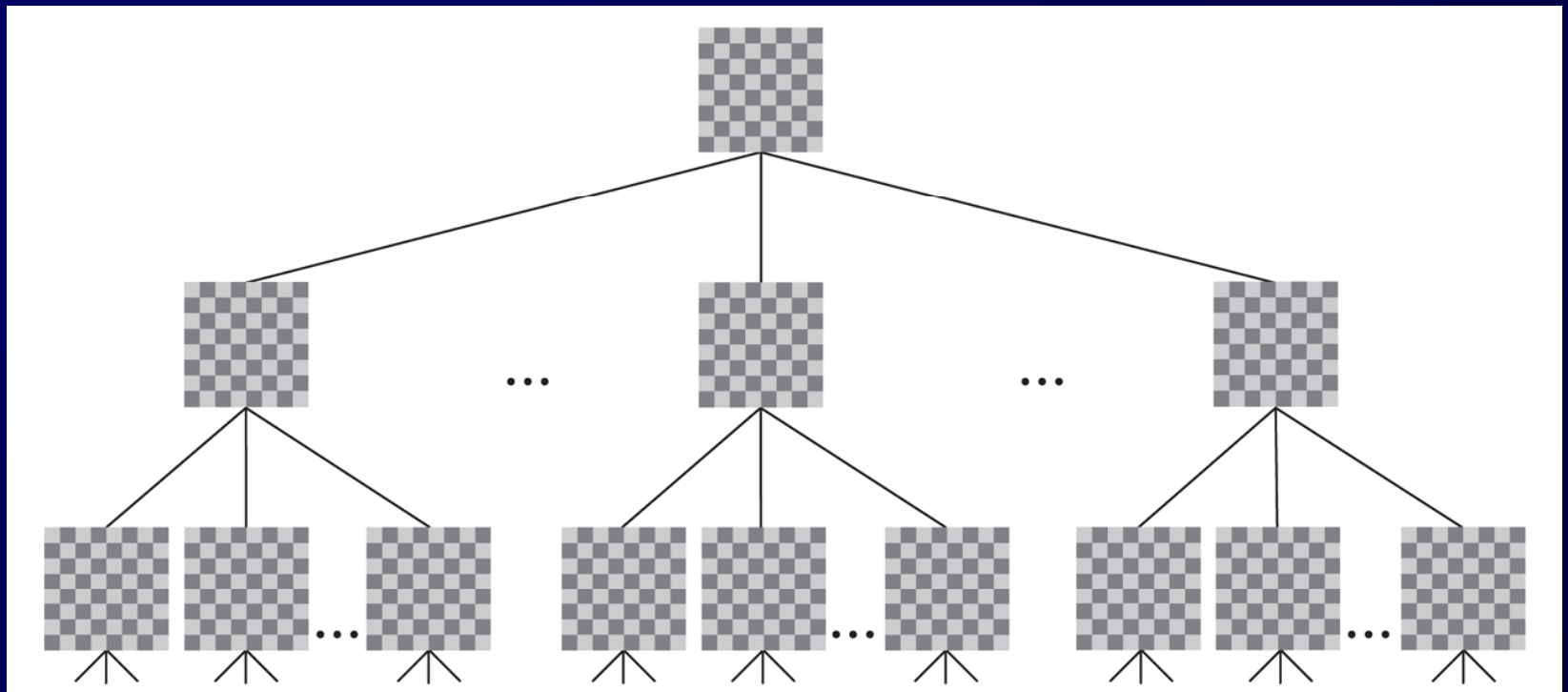

Figure 23.2. *A schematic diagram of a game tree for chess. The current board position is at the top (root). The boards produced in a single move are on the layer below, those reachable in two moves are on the layer below that, and so forth.*

Using the Game Tree Tactically

- For each board configuration under the Evaluation Function, computer considers every possible next move from opponent and analyzes them
  - Can't go all the way to end of game in planning because the number of possible moves increases geometrically
- Using a Database of Knowledge
  - Computer also has "knowledge"—a database of openings and endgames.

Using Parallel Computation

- Chess programs improved under this basic logic
  - Combination of faster computers, more complete databases, and better evaluation functions
  - *Parallel computing*—application of several computers to one task—and custom hardware brought possibility of beating a grand master in a tournament

The Deep Blue Matches

- 1996, Garry Kasparov vs. IBM's Deep Blue
  - Deep Blue: parallel computer composed of 32 special purpose computers and 256 custom chess processors
  - Kasparov won
- 1997, Kasparov vs. Deep Blue
  - Kasparov lost

Interpreting the Outcome of the Matches

- The problem was basically solved by speed
  - Deep Blue could look "deeper" into the game tree
  - But it did so intelligently
  - Intelligence may be the ability to consider many alternatives in an informed and directed way
  - Did it demonstrate that the computer was intelligent, or that IBM's experts and programmers were intelligent?

Acting Creatively

- Can a computer be creative?
  - Can it create art?
  - Creativity is by definition a process of breaking rules
  - Java applets that create designs in the style of Mondrian are only producing variations on application of rules, using random numbers
    - But the computer or program did not invent the rules

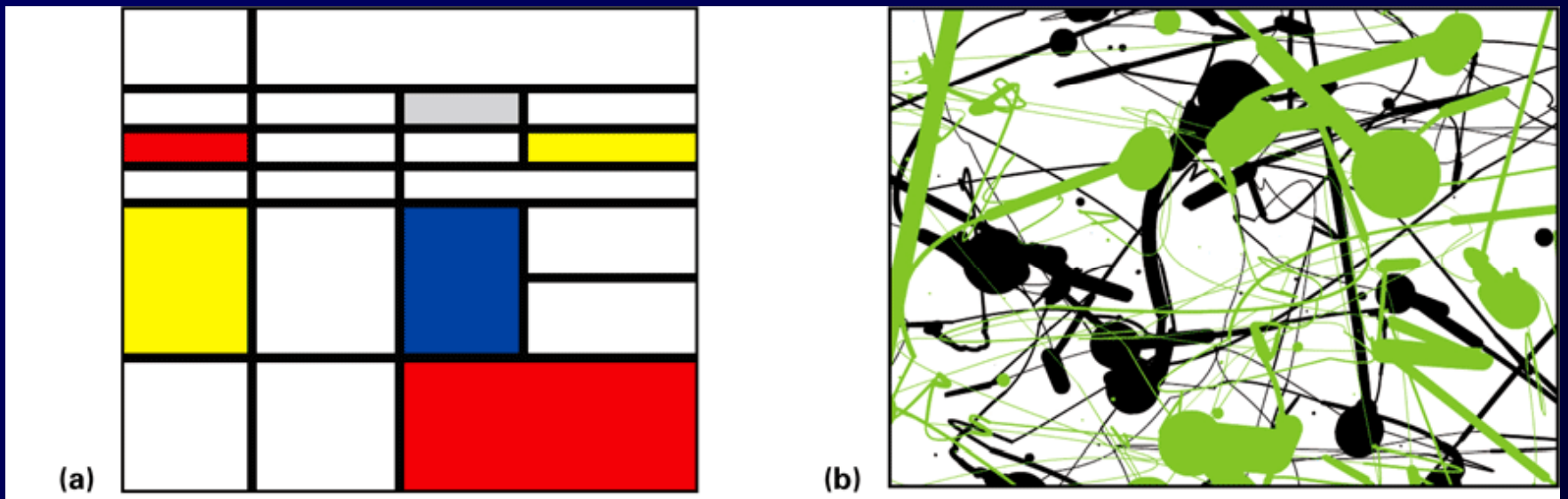

Figure 23.3 Examples of artist-free art. (a) Graphic design suggestive of the style of Piet Mondrian (find programs on the Web by searching *mondrian AND java*.) (b) Graphic design suggestive of the style of Jackson Pollock (go to `www.jacksonpollock.org`, and move and click the mouse).

Acting Creatively (cont'd)

- Creativity as a spectrum
  - Inspiration ranges from "flash out of the blue" to hard work "incremental revision"
  - The Hard Work Forms
    - Bruce Jacob wrote a program to compose musical canons (variations on a theme)
      - Randomly generates new themes, assesses as good or bad, discards bad ones
      - Forcing random variation to fit rules sometimes produces new techniques
      - Incremental revision appears to be algorithmic

What Part of Creativity is Algorithmic?

- The more deeply we understand creativity, the more we find ways in which it is algorithmic
- To the extent that creativity is algorithmic, a computer can be creative
- Progress in understanding creativity benefits people too

The Universality Principle

- What makes one computer more powerful than another?
- Any computer using simple instructions could simulate any other computer
  - *Universality Principle*: All computers have the same power
    - All computers compute the same set of computations

Universal Information Processor

- Computer is sufficient for any task if we can buy or write the software needed
- General purpose, not specialized
- People play a big role in customizing computers for specific task—one reason IT fluency is important

Practical Consequences of the Universality Principle

- Computers perform same computations, but at different rates
- Obvious difficulties:
  - Why doesn't Macintosh software run on PC?
  - How do old machines become outmoded if they're all the same?
  - Is the computer in my laptop the same as the one in my microwave?

Macintosh vs. PC

- Processors have different combinations of instructions
  - Encoded differently, operate differently, and contain separate instructions
  - It is possible to write a program for each machine to process instructions of the other machine
  - The real difference is that the operating systems work differently

Outmoded Computers

- Speed is the main difference
- New software runs slowly on old machines
- Hardware and software products may be incompatible with older machines
  - Business decision, not technology impediment

The Laptop and the Microwave

- Computers are embedded in consumer products because it is cheaper than implementing same system with custom electronics
- Embedded computers could run other computer applications, but
  - Their programs are fixed
  - They have limited input/output devices

More Work, Slower Speed

- Why do some tasks take longer to compute?
- Comparing *Alphabetize CDs* with *Face Forward*
  - Recall Alphabetize CDs from Chapter 10
  - Consider task of making sure all CDs face forward
    - The amount of work is proportional to the amount of data
      - Work-proportional-to-n algorithm

Work Proportional to n^2

- When we observe that one computation is taking more time than another despite requiring the same amount of data, it is generally because the algorithm does more work to solve the problem
- Work-proportional-to-*n* algorithm:
  - Doing the task on 1000 items takes 1000 times as long as on 1 item
- Work-proportional-to-n^2 algorithm:
  - Doing the task on 1000 items takes 1 million times as long as on 1 item

How Hard Can a Problem Be?

- NP-Complete Problems
  - No known practical algorithmic solutions
  - Intractable: Only solution is to try all possible solutions and pick the best (like finding the cheapest set of airline tickets for touring n cities)
    - Large data sets cannot be solved with a realistic amount of computer time

How Hard Can a Problem Be? (cont'd)

- Unsolvable Problems
  - Some problems have no algorithms at all
  - Computer cannot determine if a program has a bug (like an infinite loop) in it
- The Nonexistent Loop Checker
  - Trying to solve the *Halting Problem*