Today’s topics

- **Computability**
  - *Great Ideas* Ch. 14

- **Artificial Intelligence**
  - *Great Ideas* Ch. 15

- **Reading up to this point**
  - Course Pack
  - *Great Ideas* Ch. 1-4, 7, 11-12
What can be computed

• What class of problems can be solved?
  ➤ G4, 500Mhz Pentium III, Cray, pencil?
  ➤ Alan Turing proved some things, hypothesized others
    • Halting problem, Church-Turing thesis

• What class of problems can be solved efficiently?
  ➤ Problems with no practical solution
    • what does practical mean?
  ➤ Problems for which we can’t find a practical solution
    • solving one solves them all
Schedule students, minimal conflicts

- **Given student requests, available teachers**
  - write a program that schedules classes
  - Minimize conflicts

- **Add a GUI too**
  - Web interface
  - ...
  - ...

I can’t write this program because I’m too dumb
One better scenario

I can’t write this program because it’s provably impossible
Another possible scenario

I can’t write this program but neither can all these famous people
Not impossible, but impractical

- **Towers of Hanoi**
  - How long to move n disks?

- **What combination of switches turns the light on?**
  - Try all combinations, how many are there?
  - Is there a better way?
Travelling Salesperson

- Visit every city exactly once
- Minimize cost of travel or distance
- Is there a tour for under $2,000? less than 6,000 miles?
- Is close good enough?

Try all paths, from every starting point -- how long does this take?

a, b, c, d, e, f, g
b, a, c, d, e, f, g ...
Complexity Classifications

- This route hits all cities for less than $2,000 --- verify properties of route efficiently.
- Hard to *find* optimal solution

Pack trucks with barrels, use minimal # trucks

Ideas?

Problems are the “same hardness”: solve one efficiently, solve them all
Are hard problems easy?

- **P** = easy problems, **NP** = “hard” problems
  - **P** means solvable in polynomial time
    - Difference between **N**, **N**², **N**¹⁰ ?
  - **NP** means non-deterministic, polynomial time
    - guess a solution and verify it efficiently

- **Question:** **P** = **NP** ?
  - if yes, a whole class of difficult problems can be solved efficiently---one problem is *reducible* to another
  - if no, none of the hard problems can be solved efficiently
  - showing the first problem was in **NP** was an exercise in intellectual bootstrapping (1971)
Theory and Practice

- **Number theory: pure mathematics**
  - How many prime numbers are there?
  - How do we factor?
  - How do we determine primeness?

- **Computer Science**
  - Primality is “easy”
  - Factoring is “hard”
  - Encryption is possible

*top secret*

public-key cryptography
randomized primality testing
A Puzzle ("Post’s Correspondence Problem")

Given a set of cards:
- N card types (can use as many of each type as possible).
- Each card has a top string and bottom string.

Example 1:

<table>
<thead>
<tr>
<th></th>
<th>BAB</th>
<th>A</th>
<th>AB</th>
<th>BA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>ABA</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

N = 4

Puzzle:
- Is it possible to arrange cards so that top and bottom strings are the same?
A Puzzle ("Post’s Correspondence Problem")

Given a set of cards:

- N card types (can use as many of each type as possible).
- Each card has a top string and bottom string.

Example 1:

<table>
<thead>
<tr>
<th></th>
<th>BAB</th>
<th>A</th>
<th>AB</th>
<th>BA</th>
<th>N = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>ABA</td>
<td>B</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>ABA</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Puzzle:

- Is it possible to arrange cards so that top and bottom strings are the same?

Solution 1.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>BA</th>
<th>BAB</th>
<th>AB</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ABA</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>ABA</td>
</tr>
</tbody>
</table>

|   | 1   | 3   | 0   | 2   | 1   |
A Puzzle ("Post’s Correspondence Problem")

Given a set of cards:
- N card types (can use as many of each type as possible).
- Each card has a top string and bottom string.

Example 2:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ABA</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>BAB</td>
<td>B</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

N = 4

Puzzle:
- Is it possible to arrange cards so that top and bottom strings are the same?
A Puzzle ("Post’s Correspondence Problem")

Given a set of cards:
- N card types (can use as many of each type as possible).
- Each card has a top string and bottom string.

Example 2:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>N = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ABA</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>BAB</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0  1  2  3

Puzzle:
- Is it possible to arrange cards so that top and bottom strings are the same?

Solution 2.
Post’s Correspondence Problem.

Halting Problem.

- Write a program that reads in another program and its inputs, and decides whether or not it goes into an infinite loop.
  - infinite loop often signifies a bug

Program 1.
- 8 6 4 2 4 2 4 2 4 2 4 2 4 2 4
- 9 7 5 3 1

```java
odd.java
...
while (x > 1) {
    if (x > 2) {
        x = x - 2;
    } else {
        x = x + 2;
    }
}
```
Undecidable Problems

Post’s Correspondence Problem.

Halting Problem.
- Write a program that reads in another program and its inputs, and decides whether or not it goes into an infinite loop.
  - infinite loop often signifies a bug

- Program 2.
  - 8 4 2 1
  - 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

hailstone.java

```java

while (x > 1) {
    if (x % 2 == 0)
        x = x / 2;
    else
        x = 3*x + 1;
}

```
Undecidable Problems

Post’s Correspondence Problem.
Halting Problem.
Program Equivalence.
Optimal Data Compression.
Virus Identification.

Impossible to write Java program to solve any of these problem!
Some facts…

- **100 Billion neurons**
- **On average, connected to 1 K others**
- **Neurons are slow. Firing rates < 100 Hz.**
- **Can be classified into**
  - Sensory
  - Motor
  - Central
Sensory Systems

- **Vision** (nearly 30-50%)
- **Audition** (nearly 10%)
- **Somatic**
- **Chemical**
  - Taste
  - Olfaction
Motor Systems

- Locomotion
- Manipulation
- Speech
Central Systems

- Reasoning and problem solving
- Language
- ........
Moravec’s argument

- Hans Moravec: ROBOT
- Trends in biological and machine brain evolution

- 1 neuron = 1000 instructions/sec
- 1 synapse = 1 byte of information
- Human brain then processes $10^{14}$ IPS and has $10^{14}$ bytes of storage
- In 2000, we have $10^9$ IPS and $10^9$ bytes on a desktop machine
- In 25 years, assuming Moore’s law we obtain human level computing power
Evolution of Computer Power/Cost

MIPS per $1000 (1997 Dollars)

Brain Power Equivalent per $1000 of Computer

1965 Trend
1985 Trend
1997 Trend
1983 Trend

Human
Monkey
Mouse
Lizard
Spider
Nematode
Worm
Bacterium
Manual Calculation

Year
Computer power available to AI and Robot programs

MIPS
Million

1000

1

1000

1

Million


Brain Power Equivalent
Human

IBM Deep Blue

Monkey

Cray Blitz

C90/16, $20,000,000,000

Mouse

CMU Deep Thought

Lizard

CMU Hitech

Spider

DEC PDP 1

$700,000

Newell, Simon, Shaw
Logic Theorist

JOHNNIAC

$500,000

J. McCarthy

Samuel Checker Program
match with champion
Robert Neale

IBM 7090

$15,000,000

CDC 6400,
$5,000,000

Ken Thompson

Cray 1,
$15,000,000,000

E. Fredkin

DEC PDP 6

$1,000,000

Guzman Computer Vision

DEC PDP 10

$2,000,000

Ernst Robot Arm

DEC VAX 780

$400,000

Greenblatt Chess Program MacHack

SUN 2

$30,000

Stanford Cart crosses
30 meter obstacle course in five hours

SUN 3

$20,000

Personal Computers become preferred
for Expert Systems

SUN Sparc

$20,000

CMU RALPH road
following system drives
minivan 3,000 miles
from Washington DC to
San Diego, autonomous
98.2% of the distance

486 PC

$5,000

Pentium Pro

$5,000

Pentium

$5,000

Nematode

Worm

Bacterium

Manual Calculation
Chess Machine Performance versus Processing Power

Chess Rating

- Kasparov
- Grand Master
- Senior Master
- Master
- Expert
- Class A
- Class B
- Class C
- Class D
- Bernstein, IBM 704, 1957

Search depth, ply
Chess positions/second
Equivalent MIPS
Agents and Environments
Reflex agent

Agent

Sensors

What the world is like now

Condition-action rules

What action I should do now

Effectors

Environment
Reflex agent with state

- State
- How the world evolves
- What my actions do
- Condition–action rules
- What the world is like now
- What action I should do now

Agent

Sensors

Effectors

Environment
Goal-oriented agent
Utility-based agent