Telling a computer how to behave
(via pseudocode -- a workaround for Computing’s Tower of Babel.)

COS 116: 2/13/2007
Adam Finkelstein
Paul Saffo at Silicon Valley's Institute for the Future says that “Google is a religion posing as a company.”

Playing God
If Google is a religion, what is its God?

It would have to be The Algorithm.
Recall: Scribbler

**Inputs**
- Stall sensor
- Light sensors
- Obstacle sensor emitter
- Obstacle sensor detector
- Line sensor (underneath)

**Outputs**
- Speaker
- Motor/wheels
- Light outputs
Recall: Scribbler’s “Language”

- Several types of simple instructions
  - E.g. “Move forward for 1 s”
- Two types of compound instructions

**Conditional (a.k.a. Branching)**

```
If <condition> Then
{
    List of instructions
}
Else
{
    List of instructions
}
```

**Loop**

```
Do 5 times
{
    List of instructions
}
```
Scribbler language illustrates essential features of all computer languages

- Fundamental features of human languages: nouns/verbs/adjectives, subjects/objects, pronouns, etc.
- Computer languages also share fundamental features, e.g. conditional and loop statements, variables, **ability to perform arithmetic**, etc.
For a computer, everything’s a number

Audio waveform

Sequence of Numbers representing frequency, amplitude, etc.

Image

Sequence of Numbers representing red/green/blue color value of each pixel.
A simple problem

- Our robot is getting ready for a big date...

- How would it identify the cheapest bottle? (Say it can scan prices)
Solution

- Pick up first bottle, check price

- Walk down aisle. For each bottle, do this:
  - If price on bottle is less than price in hand, exchange for one in hand.
Similar question in different setting

- Robot has \( n \) prices stored in memory
- Want to find minimum price
Memory: a simplified view

- A scratchpad that can be perfectly erased and re-written any number of times

- A variable: a piece of memory with a name; stores a “value”

\[ i = 22.99 \]
Examples

\[ i \leftarrow 5 \quad \text{Sets } i \text{ to value 5} \]

\[ j \leftarrow i \quad \text{Sets } j \text{ to whatever value is in } i. \]
\[ \quad \text{Leaves } i \text{ unchanged} \]

\[ i \leftarrow j + 1 \quad \text{Sets } i \text{ to } j + 1. \]
\[ \quad \text{Leaves } j \text{ unchanged} \]

\[ i \leftarrow i + 1 \quad \text{Sets } i \text{ to 1 more than it was.} \]
Arrays

- $A$ is an array of $n$ values, $A[i]$ is the $i$'th value


$$A = \begin{bmatrix} 40.99 & 62.99 & 52.99 & \ldots & 22.99 \end{bmatrix}$$
Solution

- Pick up first bottle, check price

- Walk down aisle. For each bottle, do this:
  - If price on bottle is less than price in hand, exchange for one in hand.
Procedure findmin

- $n$ items, stored in array $A$
- Variables are $i$, $best$
- $best \leftarrow 1$
- Do for $i = 2$ to $n$
  
  ```
    { $best \leftarrow i$ }
  ```
Another way to do the same

\[
\begin{align*}
best & \leftarrow 1; \\
i & \leftarrow 1 \\
\text{Do while } (i < n) & \\
& \begin{cases}
\quad i & \leftarrow i + 1; \\
\quad \text{if } ( A[ i ] < A[best] ) \text{ then} \\
\quad & \begin{cases}
\quad best & \leftarrow i \\
\end{cases}
\end{cases}
\end{align*}
\]
```c
#include <stdio.h>

int main(void)
{
    int count;
    for (count = 1; count <= 500; count++)
        printf(“I will not throw paper airplanes in class.”);
    return 0;
}
```
New problem for robot: sorting

Arrange them so prices **increase** from left to right.
Solution

Do for $i=1$ to $n-1$
{ 
  Find cheapest bottle among those numbered $i$ to $n$

  Swap that bottle and the $i$’th bottle.
}

“selection sort”
Swapping

- Suppose $x$ and $y$ are variables. How do you swap their values?

- Need extra variable!

  $$
tmp \leftarrow x
  $$
  $$
x \leftarrow y
  $$
  $$
y \leftarrow tmp
  $$
Algorithm

- A precise unambiguous procedure for accomplishing a task

- Named for Abu Abdullah Muhammad bin Musa al-Khwarizmi

- Examples: recipe, long division, selection sort.
Love, Marriage, and Lying

Standard disclaimer.
Stable Matching Problem

Problem: Given N men and N women, find a "suitable" matching between men and women.
- Participants rate members of opposite sex.
- Everyone lists preferences from best to worst.

### Men’s Preference List

<table>
<thead>
<tr>
<th>Man</th>
<th>1(^{st})</th>
<th>2(^{nd})</th>
<th>3(^{rd})</th>
<th>4(^{th})</th>
<th>5(^{th})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Victor</td>
<td>Bertha</td>
<td>Amy</td>
<td>Diane</td>
<td>Erika</td>
<td>Clare</td>
</tr>
<tr>
<td>Wyatt</td>
<td>Diane</td>
<td>Bertha</td>
<td>Amy</td>
<td>Clare</td>
<td>Erika</td>
</tr>
<tr>
<td>Xavier</td>
<td>Bertha</td>
<td>Erika</td>
<td>Clare</td>
<td>Diane</td>
<td>Amy</td>
</tr>
<tr>
<td>Yancey</td>
<td>Amy</td>
<td>Diane</td>
<td>Clare</td>
<td>Bertha</td>
<td>Erika</td>
</tr>
<tr>
<td>Zeus</td>
<td>Bertha</td>
<td>Diane</td>
<td>Amy</td>
<td>Erika</td>
<td>Clare</td>
</tr>
</tbody>
</table>
Stable Matching Problem

Problem: Given N men and N women, find a "suitable" matching between men and women.

- Participants rate members of opposite sex.
- Everyone lists preferences from best to worst.

Women’s Preference List

<table>
<thead>
<tr>
<th>Woman</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>Zeus</td>
<td>Victor</td>
<td>Wyatt</td>
<td>Yancey</td>
<td>Xavier</td>
</tr>
<tr>
<td>Bertha</td>
<td>Xavier</td>
<td>Wyatt</td>
<td>Yancey</td>
<td>Victor</td>
<td>Zeus</td>
</tr>
<tr>
<td>Clare</td>
<td>Wyatt</td>
<td>Xavier</td>
<td>Yancey</td>
<td>Zeus</td>
<td>Victor</td>
</tr>
<tr>
<td>Diane</td>
<td>Victor</td>
<td>Zeus</td>
<td>Yancey</td>
<td>Xavier</td>
<td>Wyatt</td>
</tr>
<tr>
<td>Erika</td>
<td>Yancey</td>
<td>Wyatt</td>
<td>Zeus</td>
<td>Xavier</td>
<td>Victor</td>
</tr>
</tbody>
</table>

best
worst
Stable Matching Problem

- Problem: Given N men and N women, find a "suitable" matching between men and women.
  - PERFECT matching: everyone matched monogamously.
    - each man gets exactly one woman, and vice-versa

- STABILITY: no incentive for some pair of participants to undermine assignment by joint action.
  - a pair that is not matched with each other is UNSTABLE if they prefer each other to current partners
  - unstable pair: each improve by dumping spouses and eloping

- STABLE MATCHING (Gale and Shapley, 1962) = perfect matching with no unstable pairs.
Example

Men’s Preference List

<table>
<thead>
<tr>
<th>Man</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xavier</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Yancey</td>
<td>B</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>Zeus</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

Women’s Preference List

<table>
<thead>
<tr>
<th>Woman</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>Y</td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>Bertha</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>Clare</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>

Lavender assignment is a perfect matching. Are there any unstable pairs?

Yes. Bertha and Xavier form an unstable pair. They would prefer each other to current partners.
Example

<table>
<thead>
<tr>
<th>Men’s Preference List</th>
<th>Women’s Preference List</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Man</strong></td>
<td><strong>Woman</strong></td>
</tr>
<tr>
<td><strong>1st</strong></td>
<td><strong>1st</strong></td>
</tr>
<tr>
<td>Xavier</td>
<td>Amy</td>
</tr>
<tr>
<td>A</td>
<td>Y</td>
</tr>
<tr>
<td>B</td>
<td>X</td>
</tr>
<tr>
<td>C</td>
<td>Z</td>
</tr>
<tr>
<td>Yancey</td>
<td>Bertha</td>
</tr>
<tr>
<td>B</td>
<td>X</td>
</tr>
<tr>
<td>A</td>
<td>Y</td>
</tr>
<tr>
<td>C</td>
<td>Z</td>
</tr>
<tr>
<td>Zeus</td>
<td>Clare</td>
</tr>
<tr>
<td>A</td>
<td>X</td>
</tr>
<tr>
<td>B</td>
<td>Y</td>
</tr>
<tr>
<td>C</td>
<td>Z</td>
</tr>
</tbody>
</table>

**Green assignment is a stable matching.**
Example

Gray assignment is also a stable matching.
Propose-And-Reject Algorithm

- Guarantees a stable matching.

### Gale-Shapley Algorithm (men propose)

- Initialize each person to be free.
- **while** (some man $m$ is free and hasn't proposed to every woman)
  - $w =$ first woman on $m$'s list to whom he has not yet proposed
    - **if** ($w$ is free)
      - assign $m$ and $w$ to be engaged
    - **else if** ($w$ prefers $m$ to her fiancé $f$)
      - assign $m$ and $w$ to be engaged, and $f$ to be free
    - **else**
      - $w$ rejects $m$
For a given problem instance, there may be several stable matchings.

Do all executions of Gale-Shapley yield the same stable matching? If so, which one?

Yes. Gale-Shapley finds MAN-OPTIMAL stable matching!

Gale-Shapley finds WOMAN-PESSIMAL stable matching.
Extensions

- Unacceptable partners
  - Every woman is not willing to marry every man, and vice versa.
  - Some participants declare others as “unacceptable.”

- Sets of unequal size
  - Unequal numbers of men and women, e.g. 100 men & 90 women

- Limited Polygamy
  - e.g., Bill wants to be matched with 3 women.
Matching Residents to Hospitals

- Hospitals ~ Men (limited polygamy allowed).
- Residents ~ Women (more than hospitals)
- Started just after WWII (before computer usage).
- Ides of March, 13,000+ residents are matched.
- Rural hospital dilemma.
  - Certain hospitals (mainly in rural areas) were unpopular and declared unacceptable by many residents.
  - How to find stable matching that benefits rural hospitals?
Lessons Learned

- Powerful ideas learned in computer science.
- Sometimes deep social ramifications.

- Hospitals and residents…
- Historically, men propose to women.
  Why not vice versa?
- Men: propose early and often.
- Women: ask out the guys.
- Computer scientists get the best partners!!!