CSC104 fall 2012
Why and how of computing
week 6

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~heap/104/F12/
416-978-5899

Text: Picturing Programs
Outline

algorithms questions

Notes
could algorithms run the world?

Spectacular algorithm success leads to questions:

▶ Is there, potentially, an algorithm to solve every problem?  
   \textbf{NO}

▶ If there are two or more algorithms solving the same problem, how do you choose?  
   compare efficiency

▶ How do you discover new algorithms?  
   heuristics + tips no guarantee

▶ How do you maintain and improve massive, possibly buggy, algorithms?
problems without an algorithm

before electronic, programmable computers
Alonzo Church and Alan Turing showed there were many unsolvable algorithms

- also showed class of solvable algorithms

Classic example: Halting Problem
another example

If there an algorithm for each problem, how about one to decide whether declarative English sentences are true? How about:

This statement is false.

What should the algorithm that verifies (or not) sentences do?
algorithms that take too long

\[
\begin{align*}
\text{fib}(0) &= 0 \\
\text{fib}(1) &= 1 \\
\text{fib}(2) &= \text{fib}(0) + \text{fib}(1) = 1 \\
\text{fib}(3) &= \text{fib}(1) + \text{fib}(2) \\
\text{fib}(4) &= \text{fib}(2) + \text{fib}(3)
\end{align*}
\]

An algorithm may exist, but take too long to be feasible:

\[
(\text{define (fib n)}

\begin{cases}
\text{if (< n 2)} \\
\quad \text{n} \\
\quad (+ \text{(fib (- n 1)) (fib (- n 2)))})
\end{cases}
\]

\[\text{rewrite several efficient ways.}\]

Of interest from rabbit-breeding to biology to computer science (see Vi Hart), calculating Fibonacci sequence this way gets slow for numbers over 40.
an everyday (once) algorithm

Before Canada-411, we used to look up phone numbers in white pages. There are (at least) two different, correct ways to find the leaf (2-sided sheet) with the business you’re looking for (or conclude it’s not there).

- linear search ~ order of a thousand steps
- binary search ~ order of 8 steps
how to solve it
it being a new problem

Clearly there’s no fool-proof method, but there’s some techniques that often make progress. It helps to write down the whole process:

- Understand the problem
  - draw pictures use symbols
- Devise (one or more) plan(s)
  - breadth-first search versus depth-first search.
- Try the plan
- Look back
paper folding?
try it out

- centre the patterns — so that first fold is visible
- given natural number
- need pattern
  - Ds and Us
  - “up” and “down”

- Understand the problem (what’s given, what’s required)?
- Devise a plan
  - choose # of folds
  - create pattern
  - small examples → looking for a pattern
  - working backward.

- Try at least one plan (be ready to abandon it too)

- Look back