T1 – handed back after class, grades already posted online

average 78% ☺

"hard questions" Q1, Q6 ≈ 60%

Rubric on part II tomorrow.

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?

  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:

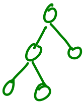
This statement is false.

What should the algorithm that verifies (or not) sentences do?

algorithms that take too long

fib(0) — 0
fib(1) — 1

fib(2) = fib(0) + fib(1) = 1
fib(3) = fib(1) + fib(2)
fib(0) + Fib(1).



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

```
(define (fib n)
  (if (< n 2)
      n
      (+ (fib (- n 1)) (fib (- n 2)))))
```

fib(4) = fib(2) + fib(3)
fib(0) + fib(1)   fib(1) + fib(2)

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 $\sim$ order of a thousand steps
- binary search $\sim$ order 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:

what's required
what's given
; solver-name : input → output

- 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" "down"

- Understand the problem (what's given, what's required)?
- Devise a plan chart # folds #crease patter
  - small examples → looking for a pattern.
  - working backward.
- Try at least one plan (be ready to abandon it too)
- Look back

Notes

Notes