

# COMP 9 / EN47 - Exploring Computer Science

## Lecture 11: The Limits and Future of Computing

April 5, 2011

### 1 Introduction

How many of you believe me that computer science isn't really about computers, but about *computation*?

Refresher: what is an algorithm?

- A specific, step-by-step plan for a procedure that begins with an input value and yields an output value in a finite number of steps
- What are some algorithms we have explored?
  - Pythagoras's theorem
  - sequence alignment/BLAST
  - others?

### 2 Search

How do you look for a name in a phone book?

- Linear search (page 1, page 2, page 3)
  - for  $n$  items, takes on average  $n$  ( $n/2$ ) operations.
- *Binary* search (page  $n/2$ , page  $n/4$ , page  $n/8$ )
  - for  $n$  items, takes  $\log n$  operations.

We don't use phone books any more, but the algorithm for any *indexed search* is the same.

Clearly, choice of *algorithm* matters

### 3 Cryptography

Sometimes the *fastest known* algorithm is still slow. This can be useful!

Why is your mail/bank/facebook login safe?

- Multiplying two large prime numbers to produce a *composite* is pretty fast
- Finding the two prime factors of a composite is *really slow*.
- Takes  $2^n$  operations for an n-bit number!

### 4 Unsolvable problems

Some problems *cannot* be solved!

Problem 'A':

```
while x != 1 do
  x = x - 2
```

Clearly, this only terminates for even values of x

Problem 'B': while x != 1 do if x.even? x = x / 2 else x = 3 \* x + 1 end

This terminates for *some* values of x, but nobody has been able to prove it terminates for all!

An example sequence of values of x:

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

#### 4.1 The Halting Problem

Could we write a program that takes another program and decides whether or not it will terminate?

It turns out we cannot! The proof is provided as a poem, but the intuitive approach is that if only our program would wait *just a little longer* it *might* finish. How can we bound it?

## 5 Quantum Computing

- Modern computers use transistors to represent *bits*, values of 1 or 0.
- Quantum computers use *quantum bits* (qubits), which have values of  $|0\rangle$  or  $|1\rangle$ 
  - This means “0 with some probability” and “1 with some probability”
  - this probability function *collapses* upon measurement to the most likely value (usually)
- Entangled groups of these qubits can interact

### 5.1 Shor’s algorithm

- In 1994, Peter Shor developed an algorithm for quantum prime factorization
  - far too complex to explain here
- If we could actually build a quantum computer, we could quickly break known crypto systems
- Nobody has yet managed to build anything more than a ‘toy’ quantum computer, and it may not be feasible

### 5.2 Undecidable is still undecidable

Quantum computers still cannot solve *undecidable* problems such as the Halting Problem.

## 6 Biocomputing

- How much does Google spend on electricity?
  - Estimated at \$12 million per *month*
- Len Adleman, 1994, came up with a way to represent and solve a known computationally hard problem, “Hamiltonian Path”, related to the “Traveling Salesman problem”, with DNA molecules.
- So-called “biocomputing” may make computationally hard problems of *limited* size actually tractable
- They’re also incredibly energy efficient: possibly a *billion* times more energy efficient than digital electronic computers

- I predict we are 15–20 years from having a compiler and programming language for biocomputing, much like for electronic computers today.

## 7 The future

So, what changes in computer science if we develop quantum and bio-computers?

Not much.

Formal models of computation *predate* modern digital electronic computers, and will *outlive* them too.

This is why computer science is about *computation*, not *computers*.