Computers Can Do (a) Anything, (b) Almost Nothing (Pick One)

FIT 100

The speed of computers and the breadth of their application is awesome, leaving us with the impression that they can do anything. They cannot. Consider some of the limitations to computation.

Universality

- One property of computers is that anything one computer can do, any other computer can do it, too.
  - Usually one computer can do the task faster than the other, but they are both able to do it
- This is a fundamental property, called *universality*

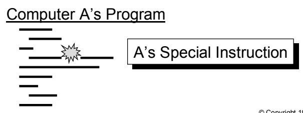
Universality: Any problem solvable by some computer can be solvable by any computer

- Thus, no one can claim to create a more powerful computer in the sense that it is capable of more sophisticated computations than another computer

Compete on speed, memory capacity, reliability, price, etc. but not the ability to solve more complex problems

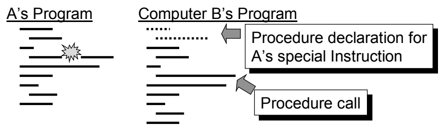
Why Is Universality True?

- Recall (Lecture 4) that computers interpret instructions using the fetch/execute cycle in hardware
- Everything that a computer does, except the basic instructions, is formulated as software, i.e. a series of instructions accomplishing the task
- Suppose computer A has a special instruction in its hardware that computer B doesn't have in its hardware … and A's software uses that instruction



Make Program Work For B

- It looks like B cannot execute the program …
- However,
  - By writing a procedure that performs the special A operation by simulating it with B's instructions, and
  - Replacing each occurrence of that instruction with a call to that procedure,
- B can run software equivalent to the software A runs
- Therefore, the computers are equivalently powerful



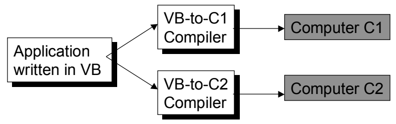
Universality Means ...

- Universality sets computers apart from machines that perform physical operations
  - Consider a chainsaw, a bread knife and a cheese slicer
- Buy one computer and perform any information processing task imaginable … you may not have the right input/output devices to enjoy the result, but the computer can do all the work if given the software

Universality trumps the argument that "using a computer is like driving a car": People drive successfully without knowing how a car works … they can use computers without knowing how they work … but one needs to know more to apply a versatile tool

Practical Universality

- There seem to be many ways in which computers are different …
  - Carburetor computer, laptop computer, cell-phone computer
  - Software for the Mac doesn't work on a PC and vice versa
    - Though computers mostly have the same instructions, their encoding in binary numbers is different …
- A compiler is a translator from a programming language to the machine language of a computer



FIT 100 Computers Use Resources

- Each operation of a computer (each step in the Fetch/Execute Cycle) takes a small amount of time
  - As a rough approximation, assume 1 instruction per clock tick
  - A 500MHz computer does 500,000,000 instructions in a second
- Generally, the number of instructions needed to solve a problem determines how much time it will take …
  - A 10 billion instruction computation takes 20 seconds
- Networks have bandwidth limits: 100 Mb/sec
- Every part of a computation takes memory, too.

Every letter or digit takes 1-2 bytes
Every instruction takes 4 bytes
Every integer takes 2 or 4 bytes
Every decimal number takes 4 or 8 bytes

… And everything is limited by the speed of light

FIT 100 The Resources Limit Computation

- A computation's execution time (how long it runs) is specified in terms of the amount of data it is given …
  - A problem that runs for a minute on a given amount of data, and runs for two minutes on twice as much data, etc. is said to be a "linear computation"
  - Computing the weekly pay and deductions for employees is an example because twice as many employees would take twice as long to compute pay and deductions
- Other ways of specifying execution time are possible, such as guesses per interval size as in Day Finder
  - Required 5 guesses to find a day among 32 things
  - How many guesses would it take if an interval had 64 items?
  - How many for 128 items?
  - Double the interval use only one more guess … logarithmic

FIT 100 Typically More Data, More Time

- There are much more complicated computations
  - A quadratic computation takes 4 times the time when there is twice the data, 9x time for triple the data, etc.
  - Checking to see if an employee has the same address as another employee might be an example
- There are even more complex computations ...
  - "Exponential problems" require double the time when adding just one more data value
  - "Try all combinations" is a typical example

The "knapsack problem" concerns packing different sized items to fit in a give space … need to fill maximum items

Fast solutions for the knapsack problem would exist if computers could correctly guess

FIT 100 Difficult Problems

- There are a variety of kinds of problems that computers "cannot" perform
  - Some problems could be solved in principle, but it would take so long and take so many resources that it is impractical … simulating the positions of the stars in the Milky Way galaxy over a million years
  - Some problems cannot be solved because the inputs and process determining the outcome cannot be known … predicting today's closing price for Amazon.Com
  - Unsolvable problems cannot be computed … solving them is logically inconsistent … a philosophically interesting topic

The "halting problem" is a problem not solvable by computer: No program can tell if another program will halt eventually or loop forever

FIT 100 Halting Problem

- Imagine someone claims to have a program that tells whether another program halts or not

testHalt(programCode As String, data As String, answer As String)

- You write a program as follows

```
Private Sub refuteClaim(programCode As String)
Dim ans As String
  Call testHalt(programCode, programCode, ans)
  If ans = "yes" Then
     Do While 0=0
     Loop
  End If
End Sub
```

Text of your test program

The claimed program cannot exist since a task can be set up for it for which all outputs are logically inconsistent

```
testHalt("Private Sub refuteClaim(…", "Private Sub refuteClaim(…", reply)
```

There can be no reply … "yes" implies "no", "no" implies "yes"!

FIT 100 Summary

- Computers are universal … no computer is more powerful than another; faster perhaps, but not more powerful
- Computers use resources, and the amount of a resource they use to solve a problem on a given amount of data is how to measure the complexity of that computation
- Computation introduces some deep philosophical questions: the inability to guess correctly and unsolvability