Algorithmic Thinking

FIT 100

To be effective computer users it is necessary to have a general idea of how to make a computer solve a problem. Thinking algorithmically is a necessary first step toward solving a problem by computer.

FIT 100 Unambiguous Instructions

- An *algorithm* is an unambiguous sequence of step-by-step instructions for producing a specified result
- Two activities – "telling what to do" and "doing it"
  - Specifying the algorithm (telling what to do) – let's call the person who does this the programmer
  - Executing the algorithm (following its instructions or doing it) without the intervention of the programmer – let's call the agent (person or computer) who does this the computer

FIT 100 Recipes

- Recipes are an example of algorithms written by chefs (programmers) and followed by cooks (computers) to produce a specified food

> S'mores: Place a toasted marshmallow on a Graham cracker and then place a square of chocolate on top

FIT 100 The 5 Properties of Algorithms

- All algorithms must have certain properties if a computer is to execute them successfully without intervention by the programmer
  - Input Specification
  - Output Specification
  - Definiteness
  - Effectiveness
  - Finiteness

FIT 100 Input Specification

- The “input” is the data that will be transformed by the algorithm to create the output
- The input must exist in a format the “computer” can access and manipulate
- In giving an algorithm, state:
  - The types of data expected -- whole numbers, letter strings
  - The number of data items expected (or how the computer will know it has reached the end of the data)
  - The structure, if any, of the data expected -- a list, table, etc.

FIT 100 Output Specification

- The “output” is the result of the computation -- its description often forms the name of the algorithm
- The “output” must be specified in a format that the “computer” can express (e.g., on a screen, audio)
- The features specified are the same as for input:
  - The types of data forming the result
  - The number of data items forming the result (or how the computer will know it has reported all of the data)
  - The structure of the result

FIT 100 Definiteness

- An algorithm must be explicit about how to realize the computation
- Definiteness is achieved by giving commands that state unambiguously what to do, in sequence
- The commands may be …
  - Conditional, i.e. require a decision to be made, and so must be explicit about how to respond to all different outcomes
  - Repeated, and so must be explicit about when to stop the repetition

The definiteness property assures that the executing agent will always know what command to perform next

FIT 100 Effectiveness

- Effectiveness assures that the computer can perform the algorithm's operations mechanically without intervention
  - No additional inputs, special talent, clairvoyance, creativity or help from Superman
- Effectiveness is achieved by reducing the task to the primitive operations known to the computer
- Definiteness assures the computer knows what command to perform next; effectiveness assures the computer can accomplish the command

FIT 100 Finiteness

- An algorithm must eventually terminate with either
  - The "right" output
  - An indication that no solution is possible
- A non-terminating algorithm is useless since it is impossible to distinguish between continued progress and being "stuck"
- Finiteness is relevant to computer algorithms since they typically repeat instructions

```
   3.3
3)10.0000000000…
  9
  10
   9
   1
```


FIT 100 Trying Your Hand at Writing an Algorithm

- Write an algorithm to sort 5 numbers from largest to smallest.

- Take out a piece of paper. Tear it into 5 small pieces. Write the following numbers, one on each piece of paper: 2, 17, 33, 4, 6.
- Shuffle them around.
- Take out a second piece of paper. Write your name on it.
- Write down the steps (an algorithm) to sort these 5 numbers. Note: You can only view and compare 2 numbers at any single point in time.
- (I'll collect these at the end of class today. These won't be graded.)

FIT 100 Alphabetize CDs

- *Input*: Unordered CDs filling a slotted rack
- *Output*: CDs in slotted rack, alphabetized

FIT 100 Alphabetizing Algorithms

1 "*Artist_Of*" means the name of the group

2 Pick one end of the rack to be the beginning of the alphabetic sequence. Call that end's slot the "*Alpha*" slot

3 Call the slot adjacent to the *Alpha* slot the "*Bet*" slot

4 If the *Artist_Of* the CD in the *Alpha* slot is later in the alphabet than the *Artist_Of* the CD in the *Bet* slot, then interchange the CDs

5 If there is a slot following the *Bet* slot, begin calling it the "*Bet*" slot and go to step 4; otherwise, continue on

6 If there are two or more slots following the *Alpha* slot, then begin calling the slot following the *Alpha* slot, "*Alpha*" and the slot following it the "*Bet*" slot, and go to step 4; otherwise, stop

FIT 100 Some Ideas for Sorting Algorithms

- Insertion Sort
  - Make the first number a list by itself – it is already sorted
  - "Insert" each number, one at a time, into the correct place in the list; shift the other numbers if you need to
  - The list slowly "grows" in sorted order
- Bubble Sort
  - Compare each pair of numbers, one pair at a time; if the pairs are out of order, swap them.
  - Keep doing this step until you go through the complete list without having to swap a single pair.
- Exchange Sort
  - Go through the list, at each step swapping the smallest number into the first slot in the list.
  - Repeat this step with each successive position in the list.

FIT 100 Algorithm vs. Program

- A program is simply an algorithm specialized to a particular situation …
- Alphabetize CDs is an instance of Exchange Sort
- Exchange Sort can be specialized to other cases
  - Sort CDs by other criteria, e.g. title
  - Sort books by title or other criteria
  - Sort canceled checks, students' homework assignments, vehicles, etc.

The algorithm, being a process with only a limited number of specifics given, is more abstract than is the program

All programs are algorithms. However, not all algorithms are programs.