Lecture 7: Problems and Algorithms

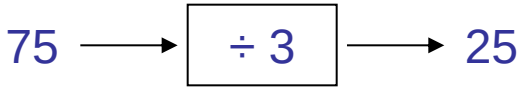
Dr John Levine

CS103 Machines, Languages and Computation
October 16th 2015

University of Strathclyde Science

Function Machines

- Function machines are simple mathematical devices for transforming inputs into outputs:



- Computers are programmable function machines.
- The inputs and outputs can be numbers, text, pictures, sounds, or anything you like really...
- … but it all gets turned into binary integers anyway.

University of Strathclyde Science

Solving Problems

- We use these programmable function boxes to solve *problems* for us.
- Example problem: find the best route between two points on a map.
- The input data are the map (represented in a way that the computer understands it) and the two points.
- The output is a string of human-understandable text which gives details of the route to be taken.

University of Strathclyde Science

What is a Problem?

- Usual meaning of ‘problem’ is something specific to be solved: e.g. ‘My ring binder has come open and now my notes are all muddled up.’
- In Computer Science a *problem* is a name used for *all* specific problems of a certain type.
- The specific problems to be solved by the computer are called *instances* of the problem.

University of Strathclyde Science

Example Problems

- Given an unsorted list of numbers, find the largest number in the list.
- Given an unsorted list of numbers, sort them into ascending order.
- Given a set of cities with roads between them, find the shortest route which visits each city exactly once.
- Given a computer program *P* and an input to that program *I*, determine if *P* will halt when run on *I*.

University of Strathclyde Science

Thinking about a Problem

- Given an unsorted list of numbers, find the largest number in the list.
- To solve this problem, I will need to look at *every* number in the list; if I don't, I might miss the largest.
- However, I will only need to look at each number *once*, so long as I can remember the largest number I've seen so far.
- So, if I can look at one number every second, I can process the whole list in length(*L*) seconds.

University of Strathclyde Science

What is an Algorithm?

- An algorithm is a finite set of instructions for solving a problem, which, given a well-defined initial state, will result in a corresponding well-defined end-state.
- Think of it as “Dobby’s instructions” for solving a problem.
- A well-specified algorithm will solve *all* specific instances of the problem.
- An algorithm will *always* terminate in a finite number of steps.

University of Strathclyde Science

Algorithms vs. Programs

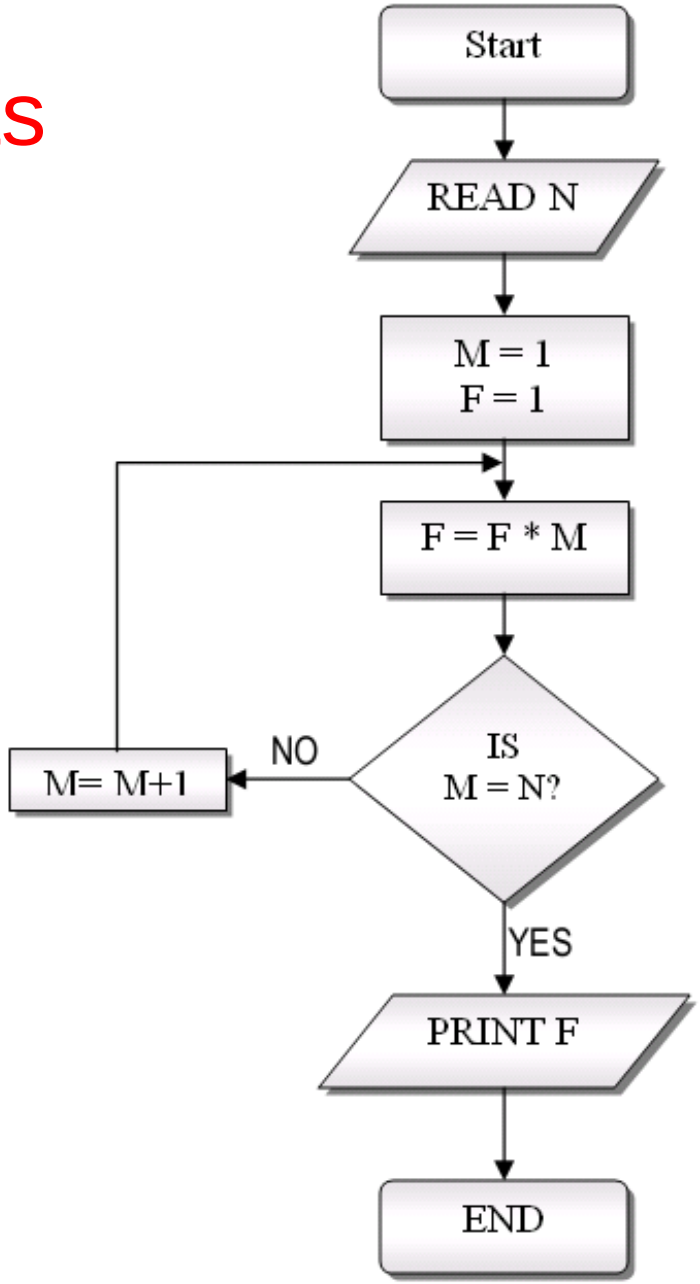
- Algorithms are more abstract and fundamental than computer programs: you can implement an algorithm in any language you want to.
- Algorithms can be specified using normal language, flowcharts or *pseudocode*.
- The specification of an algorithm must be completely unambiguous. This is very important!

Unambiguous English

1. Read in an integer, N.
2. Set M = 1 and set F = 1.
3. For each value of M from 1 up to N, set F = F * M.
4. Return F and halt.

University of Strathclyde Science

Flowcharts



Pseudocode

```
input N
M = 1
F = 1
loop forever
   F = F * M
   if M = N
      return F
   else
      M = M + 1
```

Example Algorithm

- Given an unsorted list of numbers, L, find the largest number in the list.

1. Set Max to be the first item of L and delete it from L.
2. If L is empty, return Max (and halt).
3. If the first item of L is larger than Max, set Max to be the first item of L.
4. Delete the first item of L and go to Step 2.

University of Strathclyde Science

Why Bother with Algorithms?

- Often, programmers will not worry too much about such details: just slam in some code, fight with the compiler, then go for a beer.
- Bugs in syntax or even semantics are easy to find: but bugs in algorithms are the most difficult to find.
- Bad design at the algorithm stage always leads to bad programs being written.

University of Strathclyde Science

Assignment 3

Using flow charts, pseudocode or unambiguous English, create algorithms for the following functions:

(a) Given an unsorted list of integers and a target integer as inputs, return TRUE if the integer is a member of the list and FALSE if it is not.

(b) Given an arbitrary integer, return TRUE if that integer is a prime number and FALSE if it is not.

(c) Given an arbitrary string, return TRUE if the string is a theorem of the MIU system and FALSE if it is not.

University of Strathclyde Science

Testing your algorithms

Your algorithms need to pass “the Dobby test”:

- Find a willing helper (a friend or family member who doesn’t do this class)
- Write down your algorithm on a piece of paper and give it to your helper
- Give your helper an input to the algorithm
- See if your helper can follow the algorithm to produce the correct output