Lecture 8: Functions and Mappings

Dr John Levine

CS103 Machines, Languages and Computation
October 19th 2015

The Story So Far

- Lecture 1: computers are programmable function machines, all computers are equivalent (undecidable problems are undecidable on all computers).
- Lecture 2: types of mathematical proof, direct proof, proof by contradiction, proof in a formal system.
- Lecture 3: the MIU-system, deriving theorems by hand, deriving theorems automatically, but is there a decision procedure for theoremhood?
- Lecture 4: introduction to sets, finite sets, infinite sets, countable sets, doing proofs on infinite sets.

The Story So Far

- Lecture 5: recap of proof by contradiction, proof by induction (the domino proof).
- Lecture 6: functions and algorithms, what is a problem, how to write an algorithm, why bother with algorithms, how specify algorithms using unambiguous language, flowcharts and pseudocode.

Assignment 3

Using unambiguous English, flowcharts or pseudocode, create *efficient* algorithms for the following functions.

a) Given an unordered 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.

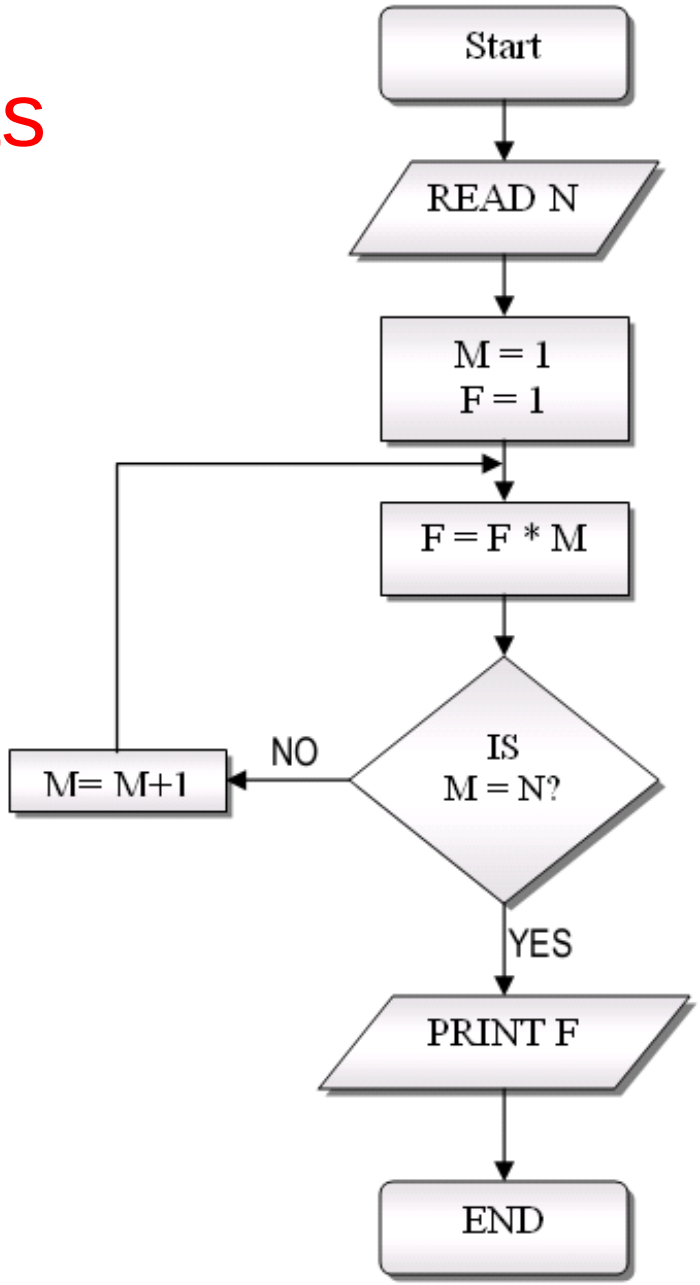
b) Given an arbitrary integer, return TRUE if that integer is a prime number and FALSE if it is not.

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

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.

Flowcharts



Pseudocode

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

Two Views of Functions

- So far we've regarded functions as being input-output boxes (i.e. sausage machines).
- The mathematical view of functions is rather different: a function is a set of pairs $\{(x_0, y_0), (x_1, y_1), (x_2, y_2) \ldots \}$
- The x variables are chosen from a set called the *domain* and the y variables are chosen from a set called the *range*.
- The association can have some "sense", or it can be an arbitrary association between members of one set and members of another.

Two Views of Functions

- For each domain value, there must be one and only one value in the range (otherwise the relation is not a function).
- Example function: $\{(1, 1), (2, 4), (3, 9), (4, 16)\}$
- The domain $\{1, 2, 3, 4\}$, the range is $\{1, 4, 9, 16\}$.
- Another example: $\{(1, 0), (2, 1), (3, 0), (4, 1)\}$
- The domain $\{1, 2, 3, 4\}$, the range is $\{0, 1\}$.
- Not a function: $\{(1, 0), (1, 1), (2, 0), (2, 1)\}$

Mappings

- If we can find an algorithm for automatically computing the range value given the domain value, it is called a *mapping*.
- Example: for $\{(1, 1), (2, 4), (3, 9), (4, 16)\}$ the mapping function could be $f(x) = x^2$.
- Another example: for $\{(1, 0), (2, 1), (3, 0), (4, 1)\}$ the mapping function could be the “even” function given in on the board (with true replaced by 1 and false by 0)
- Other mapping functions are possible, including (for finite sets) complete enumeration of all cases.

Mappings to Show the Size of Sets

- Consider two sets shown below:

$$A = \{1, 2, 3, \ldots, 1000\}.$$
$$B = \{1, 3, 5, \ldots, 2001\}.$$

- Which set is bigger?
- We could count the elements of each set, but this would be tedious.
- We can make a mapping from elements of the first set to elements of the second set, and then see if that mapping accounts for all elements of both sets.

Dogs and Collars

- Consider an infinite set of dogs.
- Each dog has a name, which is a finite string of letters (all in uppercase), such as “FIDO” or “GROMIT”.
- Consider an infinite set of collars, where every collar is labelled with an integer $n \in \{0, 1, 2, \ldots\}$.
- Are there enough collars for all the dogs?
- In other words, can I find a computable mapping from the dog’s name to an integer, so that every dog gets a unique collar?
- And does every collar have a unique dog?

The Dogs and Collars Problem

- Try to find a way to map the names of the dogs onto a unique collar (i.e. an integer).
- Is there a collar for every dog?
- Try to specify how the mapping would work in reverse, i.e. how to turn the collar number into the name of the dog that would own that collar.
- Try writing a program to perform the reverse mapping.
- Is there a dog for every collar?