Algorithms II

Can't stop, won't stop

Math Circle

3/11/2018

Intuitively you might understand what an algorithm is. It is a set of instructions that provide an exact description of how to do something. The important thing about an algorithm, is that it is a series of steps, where each step is exact, and doesn't require any intuition.

1. Which of the following are algorithms, and which aren't?

   (a) A recipe from a cook book, that specifies how to make lasagna (...stir in tomatoes, tomato paste, water, sugar, 3 tablespoons parsley, basil, fennel, 1/2 teaspoon salt and pepper; bring to a boil. Reduce heat; simmer, uncovered, 30 minutes, stirring occasionally...).

   (b) A set of directions from Google maps to go from UCLA to the Sana Monica Pier. (Head west on Gayley Ave toward Veteran Ave, Continue onto Montana Ave, Turn left onto S Sepulveda Blvd...)

(c) Vague life advice from a Kung fu instructor (You must find inner peace, only then can you hope to perfect the golden flying pan fried monkey technique.)

(d) Euclid's algorithm that we used to find the greatest common divisor of two numbers.

(e) An algorithm for writing your first novel (Come up with an interesting premise and characters. Find a compelling conflict, ...)

(f) What are other examples of algorithms that are not covered above? What about things that aren't algorithms?

In order to first talk about writing our own algorithms, it helps to take a specific example.

Suppose that we have a robot named Dora. Dora lives in a hallway which has two doors (denoted by the letter 'D') which has a tiled floor. On some of the tiles, some birds have made their nests (the birds are funny looking, they look like ♣). Here is an example of what Dora's hallway might looks like.

D			♣			♣	♣					♣		D

Dora can only understand the following simple instructions:

(a) Move one tile to the right or left.

```
move_left, move_right
```

(b) Check if there is bird nest in the current tile that Dora is in.

```
bird_in_tile
```

(c) Check if Dora has a door on her right/left.

```
door_on_left, door_on_right
```

(d) turn off

```
turn_off
```

Dora also knows how to execute a series of command if a statement is true, and how to repeat something. For example, if I wanted to write an algorithm to make Dora move all the way to the left door, then I could write:

```
while (door_on_left == false)
    move_left
turn off
```

Dora can also count, and can store variables, and can do some very basic arithmetic. If I wanted to have Dora move to the end of the hallway, and count how many steps it took and output that number, I could modify the above algorithm to do the following:

```
number_of_steps = 0
while (door_on_left == false)
    move_left
    number_of_steps = number_of_steps + 1
if (number_of_steps == 1)
    output "1 step"
else
    output number_of_steps
    output " steps"
turn off
```

Notice that Dora's output changes slightly if she stepped once, or more than once. We can use an 'if statement' like above in order to cause something to happen if a certain condition is met, and not otherwise.

2. Now it is your turn to write some algorithms. For the following, assume that Dora starts next to the right door. Also, make sure that Dora turns off after she finished implementing the algorithms!

   (a) Have Dora output 'yes' of she finds any nests in the hallway, and 'no' otherwise.

(b) Have Dora output the number of nests in the hallway.

(c) Have Dora walk down the hallway, and output 'Double Nests' if there are two nests adjacent to each other anywhere in the hallway, and 'No Double Nests' otherwise.

We upgraded Dora, and now she has the additional command

```
number_of_eggs
```

which will tell her how many eggs are in the current nest that she is standing over.

(a) Write an algorithm that will output the largest number of eggs in any single nest. For example, if there are four nests, and they have 1,4, 5 and 3 eggs, then Dora should output 5.

(b) **Write an algorithm that will output 'yes' if the number of eggs in each nest increases from left to right, and 'no' otherwise. Assume that Dora knows how to interpret the statement $A < B$ for integers A, B.**

Now that you have an idea of what an algorithm is, and some practice writing your own, let's work on writing some algorithms to do math.

Suppose that you have a function

```
add_one(x)
```

which takes in one input, x, and returns the number x + 1. You have no idea how it works, you just know that it works. The following algorithm would output 7

```
a = 6
y = add_one(a)
output y
```

If you think about if, if you have this function, then you can easily define another function `'add_two(x)'` as follows:

```
function add_two(x)
y = add_one(x)
z = add_one(y)
return z
```

The way to read this algorithm is as follows. The first line in the above function days "This is a function with the name `add_two` which takes one input argument, referred to by the name x." The second line assigns one plus the input number to a variable named y. The next line assigns one plus y to a variable z, and the last line returns z. When I say that the function returns z, I mean that if the function is called (for example by `Q=add_two(3)`) then the value of z will be assigned to Q.

3. Suppose now, that we have a machine that can add or subtract numbers using `add(x,y)`and `subtract(x,y)`, and can correctly interpret statements like $A < B$ for integers A and B. Do the following on the extra paper provided. For all of these problems, I encourage you to use functions that you have written before when you are writing new functions!

   (a) Write a function called `multiply(x,y)` which takes in two arguments, x and y, and returns x times y.

   (b) Write a function called `exponentiate(x,y)` which takes in two arguments, x and y, and returns x^y.

   (c) Write a function called `mod(x,y)` which takes in two arguments, x and y, and returns x modulo y,. For example, $mod(4, 7)$ should return 4, $mod(13, 4)$ should return 1, etc.

   (d) Write a function called `divide(x,y)` which takes in two arguments, x and y, and returns x/y. You can assume that

   (e) **Write a function called `gcd(x,y)` which takes in two arguments, x and y, and returns the greatest common divisor of x and y.**

   (f) **Write a function called `isPrime(x)` which takes in one argument, x and returns true if x is prime, and false otherwise.**

4. Here are some challenge problems that relate to things that we'll cover in the future. Do all of these on the provided paper as well.

   (a) **Write a function that will output every single integer.**

   (b) **Write a function that will output every possible pair of integers.**

(c) **Write a function that will run forever if the Goldbach conjecture is true, and will output the smallest counterexample otherwise.**

(d) **Write a function that will run forever if the collatz conjecture is try, and will output the smallest counterexample otherwise.**