THE IDEA OF AN ALGORITHM

PIETRO KREITLON CAROLINO

Today we're going to learn about an idea that plays a fundamental role in math, computer science, and even philosophy: the *algorithm*. Intuitively, an algorithm is a fixed set of clear, unambiguous instructions that a person or machine can follow "without thinking". For instance, here is an algorithm for checking if a deck of cards has the five of spades:

Algorithm 1 Finding 5♠

```
foundIt := "no"
while(foundIt = "no" and there are cards left)
    draw a card from the deck
    if(current card is 5♠)
         foundIt := "yes"
    discard current card
output foundIt
```

In elementary school you learned several algorithms, like how to add together numbers with several digits, and how to multiply such numbers. You memorized a few rules that told you which digits to write down where, and once you had the rules down, you could add and multiply numbers without thinking.

In contrast, here is a set of instructions that is *not* an algorithm, because the steps are *not* clear or unambiguous:

Algorithm 2 Writing a novel (not really an algorithm)

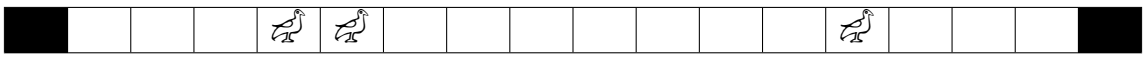
```
come up with an interesting premise
come up with complex characters
come up with a cool story involving the characters and the premise
```

Thinking up interesting premises is not a clear, straightforward task like checking if a card is the five of spades, or adding $3+5$. In the sequel we'll come up with algorithms to do various things, in order to get a feel for what kinds of things can be done mechanically, "without thinking".

Robots.

For the first few problems we will program a robot to do certain things for us. Our robot — let's call her Dora — lives in a long, thin corridor, where the floor is made of stone tiles of the same size. There are doors at both ends of the corridor. On a few of the tiles, birds have made their nests.

Here's a sketch of Dora's world:



The doors are represented above by black squares, and the birds by, well, birds.

Dora understands the following instructions:

(1) move one tile right/left;
(2) check if there is a bird nest at current position;
(3) check if there is a door immediately to the right/left of the current position;
(4) shut down.

In addition, Dora understands `if` and `while` statements like the ones in Algorithm 1, and she can create and update variables. For instance, here is a program that makes Dora move to the door all the way to the right, and shut down:

Algorithm 3 Find right door and shut down

```
while(there is not a door immediately to the right)
    move right
shut down
```

And here's a program that does the same, but Dora outputs how many steps she took before shutting down:

Algorithm 4 Count steps to right door and shut down

```
numberOfSteps := 0
while(there is not a door immediately to the right)
    move right
    numberOfSteps := numberOfSteps + 1
output numberOfSteps
shut down
```

For the all problems, assume Dora starts on the tile next to the left door. Also, to conserve energy, make sure Dora shuts down after her tasks are done.

1. Program Dora so that she outputs "yes" if there are any nests in the corridor, and "no" otherwise.

2. Program Dora so that she outputs the number of bird nests in the corridor.

3*. Write a program that makes Dora output "yes" if there are two nests right next to each other somewhere, and "no" otherwise.

4. Suppose we upgrade Dora so that she is now able not only to tell if there is a nest somewhere, but how many eggs are in it. For instance, here's a program that has Dora look for a nest with a single egg. She shuts down if she finds one, or if she hits the right door:

Algorithm 5 Look for single egg

```
numEggs := 0
while(numEggs ≠ 1)
    if(there is a nest here)
         numEggs := number of eggs in this nest
    if(numEggs = 1)
         shut down
    if(there is a door immediately to the right)
         shut down
    move right
```

Program Dora so that she outputs "yes" if there is an *empty* nest somewhere, and "no" otherwise.

5. Write a program so that Dora outputs the largest number of eggs in any single nest. For instance, if there are nests with 2, 5, and 3 eggs, and no other nests, then Dora should output 5.

6*. Write a program so that Dora outputs "yes" if the number of eggs in the nests is increasing from left to right, and "no" otherwise.

Calculating Robots.
Now we're going to forget about the bird corridor for a bit, and focus on Dora's mental powers. Our first goal is to show that, even if Dora comes from the factory with very limited abilities, we can program her to do much more.

For example, imagine that factory Dora only knows how to add one to numbers, but not two, three, etc. That is, if she is currently storing a variable `x`, she understands the command `y := x+1`, but *not* `y := x + 2`. Nevertheless, we can program Dora to add two by just adding one twice:

Algorithm 6 PlusTwo(`x`)

```
temp := x+1
temp := temp+1
return temp.
```

Then, if we ever needed to do `y := x + 2`, we would use `y := PlusTwo(x)`.

Now suppose factory Dora only knows how to add and subtract numbers, but not how to multiply them. That is, if she is currently storing two variables, `x` and `y`, then she understands statements involving `x+y` and `x−y`, such as `if(x + y > 0)`; but she does *not* understand those involving `x × y`.

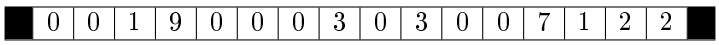
7*. Can you nevertheless program Dora to multiply numbers? In other words, can you write an algorithm `Multiply(x, y)` that takes `x` and `y` as input (like the one above takes `x` as input) and `returns` the product `x × y` (like the one above returns `x + 2`)? You may assume `x` and `y` are both non-negative. (Hint: what does 4×5 actually mean?)

8*. You learned in school what the *remainder* of a division is: the remainder of 13 divided by 4 is 1, the remainder of 25 divided by 7 is 4, etc. By now Dora knows addition, subtraction, and multiplication; can you teach her remainders? That is, write an algorithm `Remainder(x, y)` that takes non-negative numbers `x` and `y` as input and `returns` the remainder of `x` divided by `y`. (Hint: you only need subtraction.)

9. Write an algorithm `DivisibleBy(x, y)` which `returns` "yes" if `x` is exactly divisible by `y`, and "no" otherwise. For instance, `DivisibleBy(35, 7)` and `DivisibleBy(48, 2)` should return "yes", while `DivisibleBy(36, 5)` and `DivisibleBy(13, 4)` should return "no". You may use all the capabilities you have taught Dora so far.

10.** Can you write an algorithm `IsPrime(x)` which takes as input a positive number `x` and `returns` "yes" if `x` is prime, and "no" otherwise? You may use all the capabilities you have taught Dora so far.

11.** Let's go back to the bird corridor. Suppose every nest has between zero and nine eggs (tiles without nests count as zero eggs). Here is a possible configuration:



Program Dora so that she interprets the egg configuration as a number in base 10 (the configuration above corresponds to the number 19000303007122) and outputs "yes" if this number is prime, "no" otherwise. You may use all the capabilities you have taught Dora so far.