Algorithms And Data Structures I Course Notes

Felipe Balbi

October 9, 2019

Week 1

1.1.1 What is a Problem? What is an algorithm?

In computing we deal with problems that are addressable by computers. In other words, we deal with problems that are **computable**. The underlying language used to communicate with a computer needs to be mathematical, regardless of which *Programming Language* we use.

Computers require each and every idea to be converted into a mathematical concept (a number or a truth value).

Toy example:

```
x days of holiday total
y days of holiday used
x > y ? True  or False
x - y = Amount of days left
```

Note that in the case of `x-y` a positive result implies *True* while a negative or zero result implies *False*.

Problems can usually be solved in more than one way. The study of Algorithms and Data Structures gives us tools to decide which method is *better* than the other.

Definition 0.1 (Algorithm) *A general and simple set of step-by-step instructions which, if followed, solve a particular problem.*

Keep in mind that it's highly desirable to have a general purpose algorithm that solves many instances of similar problems. For example, instead of having an algorithm to solve $x^2 = 2$, it would be better to produce an algorithm to solve $x^2 = y$ given x and y are in $\mathbb{Z}$.

1.2.1 Al-Khwarizmi and Euclid

Algorithms predate the digital computer by hundreds of years. The word *algorithm* comes from the latinized name of Persian polymath **Al-Khwarizmi** (written as *algorithmi*).

One of the first known algorithms is Euclid's algorithm for calculating the *Greatest Common Divisor* between two numbers. It was described around 300 B.C.

An algorithm is a **mathematical concept** that can be instantiated as a computer program.

1.2.4 From mathematics to digital computers

Before we think about how to concretely describe an algorithm, we need to consider how to write the input data into a computer. It is **not** always possible to input arbitrary data into a computer. For example the number π is an irrational number (actually, it's transcendental see [1], which means it has an infinite decimal expansion; therefore we can't input π into a computer, as computer memory is a finite resource. We can only approximate it.

When approximating irrational and transcendental numbers with rational numbers, we will be left with an error in our calculations. This error is referred to as the *precision* of our calculation. The smaller the error, the more precisely correct our computer handles the input to our problem.

The need for approximations came to be before digital computers. The Egyptian-Greek mathematician, Heron of Alexandria, produced an algorithm for calculating and approximation of the square root of a number. That algorithm is called Heron's Method.

Heron's Method

Say we want to calculate $x^2 = 2$, give x to 1 d.p.

We **know** x must be $1 < x < 2$, we take the mean to get a candidate:

$\frac{1+2}{2} = 1.5$

$x_g = 1.5 \rightarrow x_g^2 = \frac{9}{4} > 2$

The answer **must** be within the interval $1 < x < 1.5$.

Thus:

$$\frac{2}{x} = x < x_g \rightarrow \frac{2}{x_g} < x$$
$$\rightarrow 1.\dot{3} = \frac{4}{3} < x < \frac{3}{2} = 1.5$$

Take mean to get new candidate:

$x\prime_g = \frac{17}{12} = 1.41\dot{6}$

correct to 1 d.p.

We can repeat this process as many times as we want in order to increase accuracy.

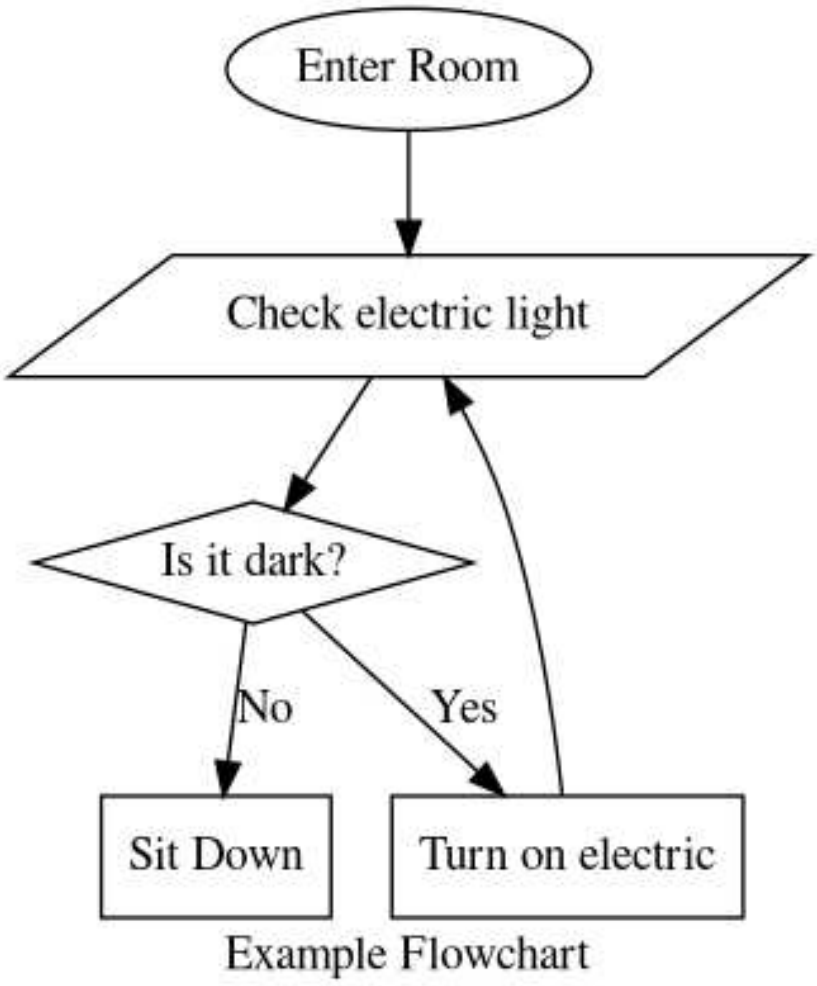
[1] `https://www.youtube.com/watch?v=WyoH_vgiqXM`

Week 2

1.3.1 Introduction to flowcharts

Using flowcharts to describe algorithms. Flowcharts are abstract representations of processes such as workflow and project management. They are composed of differently shaped boxes and arrows connecting them. Boxes typically represent actions, referred to as *activities*, *states of affairs* or *decisions*. Arrows represent workflow or outcomesthat result from the decisions.

Let's look at an example below:



Example Flowchart

Flowcharts use different shapes for different meaning:

- Oval

  Ovals are Terminal nodes. They are used either as *Start* or *End* of an algorithm.

- Parallelogram

These represent I/O actions, like gathering or displaying data.

- Arrows

  Represent control flow of the algorithm by connecting one node to another.

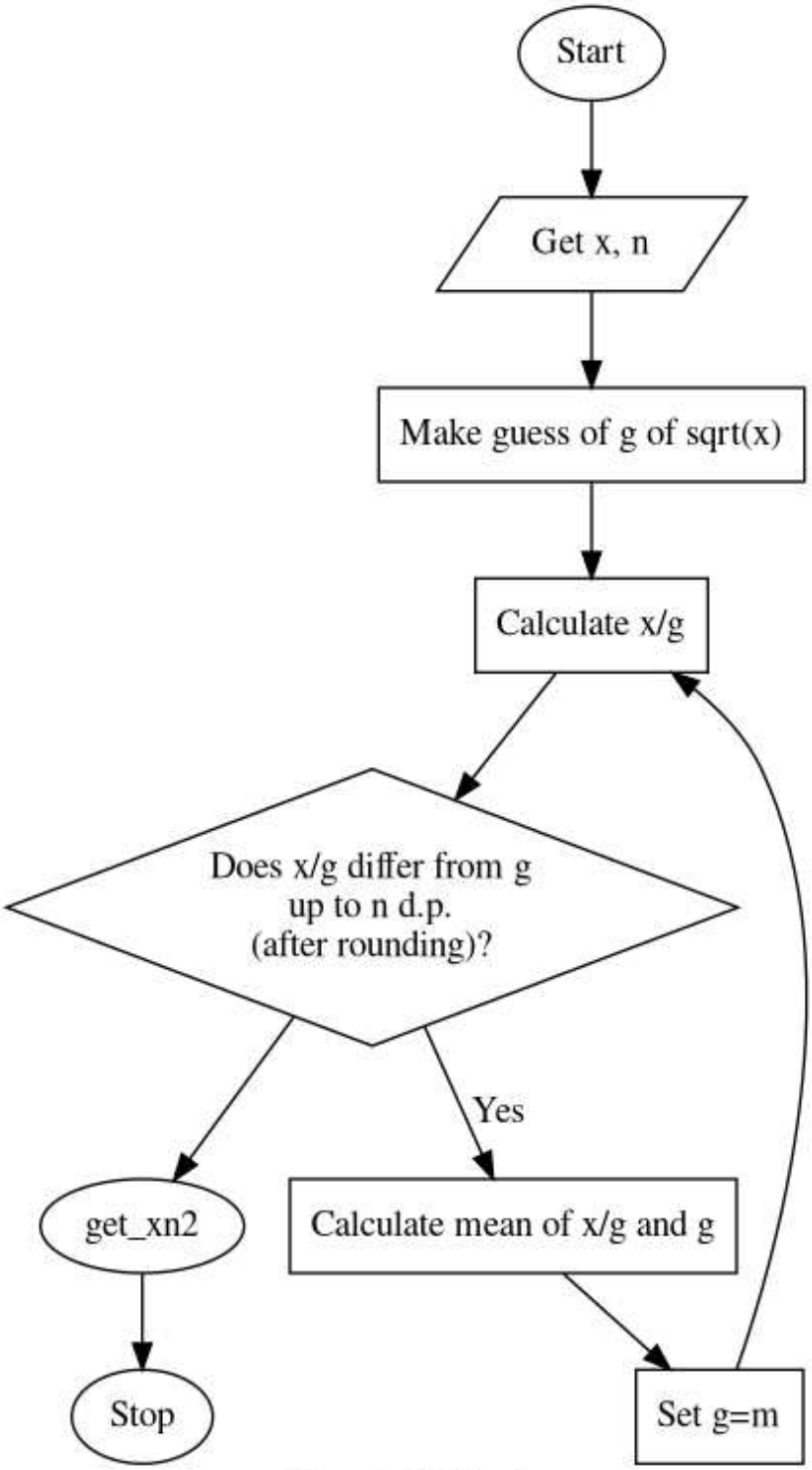
- Diamond

  Diamonds represent decisions blocks. Typically they have two outcomes: Yes/No, True/False, etc.

- Rectangle

  Basic actions, turning on the light, are carried out by rectangle boxes.

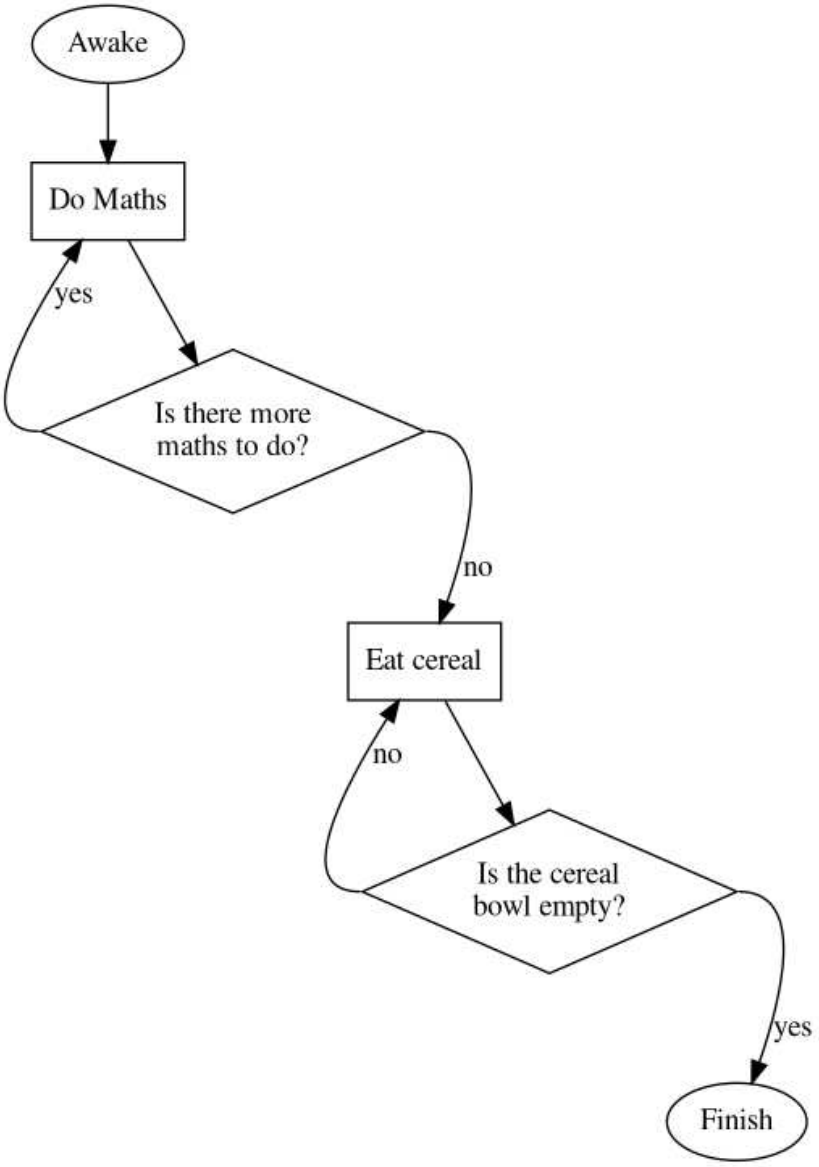
Heron's Method Flowchart



Heron's Method

1.3.4 My example of my morning routing as a flowchart

Professor Matty showed a flowchart of his morning routine. It went something like below:



Morning Routine

1.4.1 Conclusion

We learned concepts of problems, solutions, and algorithms. We learned a bit of algorithmic history and how to describe algorithms in flowcharts.