INTRODUCTION TO COMPUTER SCIENCE

Programming is not Coding

How to build safe and secure software

Stefan Parvu

December 7, 2023

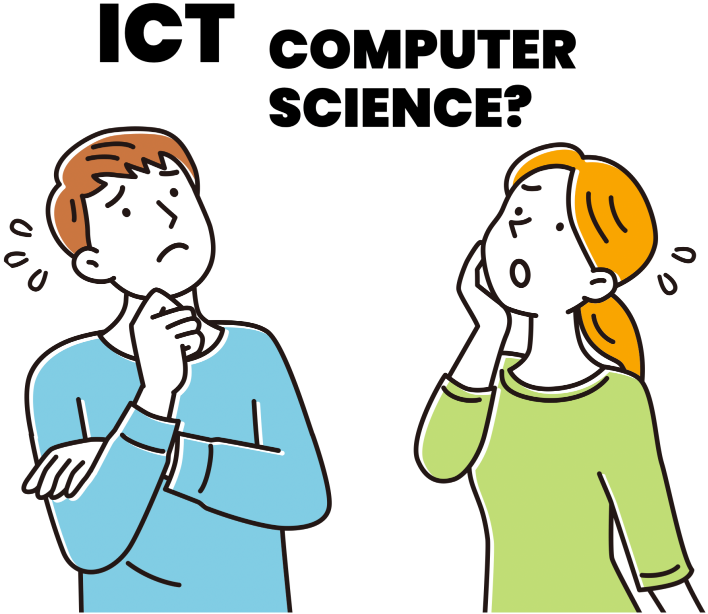
Table of Contents

Lesson 1

Computation. Algorithms. Computer Programs

In this lesson we will talk about the following concepts: computation, algorithms and programs.

Lesson 1



Computer Science (CS)

But what's the difference between IT and CS?

Computer Science

Computer Science is the study of the principles of computing and how computer systems solve problems

Computer scientists design and build tools and software applications

Programming computers using mathematical algorithms, abstract concepts and models like, example the computational complexity theory

Information Technology

IT is the study of current tools and computing techniques that can be used for technological needs of a particular organization

IT professionals apply and use these tools or software applications

IT involves more practical aspects how to use and maintain software applications, to develop, maintain and improve business processes

Computation

What is a Computation?

A **computation** is any type of arithmetic or non-arithmetic calculation that is well-defined.[1][2] Common examples of computations are mathematical equations and computer algorithms.

Mechanical or electronic devices (or, historically, people) that perform computations are known as *computers*. The study of computation is the field of computability, itself a sub-field of computer science.

A **computation** is what a computing device does

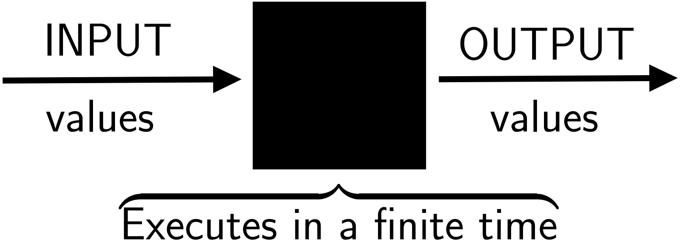
We sometimes call the computation, a behaviour

Computing device

For example, a computer system, a tablet, a mobile phone or a basic calculator. But it can be as well, a non physical device, something more abstract, like an algorithm.

We describe a computing device by describing all its possible computations or behaviours.

What is an algorithm?



Sequence of computational steps that transform the input into the output

Example of computations

- well-defined mathematical statements
- solvable statements
- simple instructions

Note: We have mathematically concepts difficult to solve, like: the halting problem, or busy beaver game.

Computation as a sequence of states

INSTRUCTION 1

INSTRUCTION 2

INSTRUCTION 3

⋮

INSTRUCTION N

$x = 12$
$y = 18$
State 1

y — x

$x = 12$
$y = 6$
State 2

x — y

$x = 6$
$y = 6$
State 3

Initial State

State 1

State 2

⋮

State N

What is a computing state?

A state is an assignment of values to variables.

$$\begin{array}{lcl} x & = & 12 \\ y & = & 18 \end{array}$$

State 1

$$\begin{array}{lcl} y & - & x \\ x & = & 12 \\ y & = & 6 \end{array}$$

State 2

$$\begin{array}{lcl} x & - & y \\ x & = & 6 \\ y & = & 6 \end{array}$$

State 3

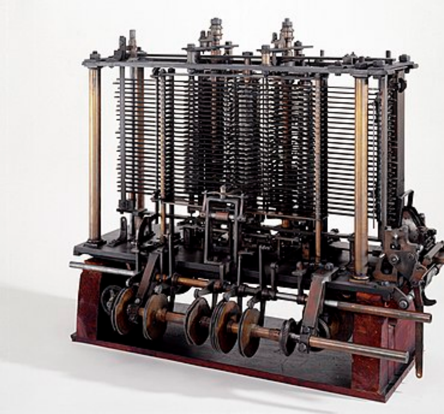
Computing devices are supposed to compute something. Like calculate and predict the weather, render to produce a movie, calculate first 1000 digits of π

Some well-defined computations:

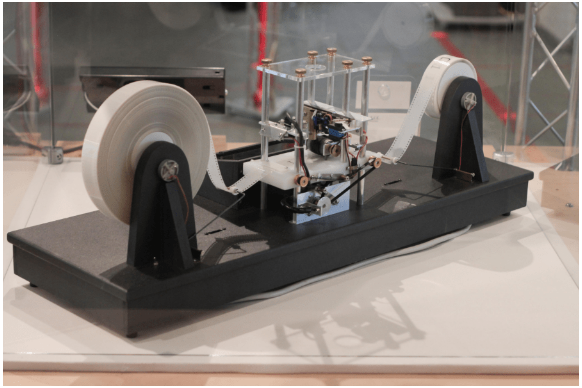
- calculations carried by an electronic computer or calculator
- calculations performed on a analytical engine, Turing machine
- majority of mathematical statements and calculations

The Analytical Engine

- The very first mechanical general purpose computer system
- Designed in 1837 by the English mathematician and computer pioneer Charles Babbage
- Built as a programmable device to solve different things



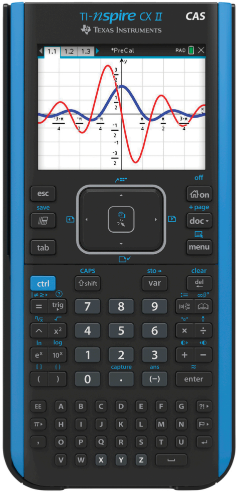
Turing Machine



- Infinite tape, divided into cells with symbols
- A head can read/write symbols on the tape
- A register that stores the state of the machine

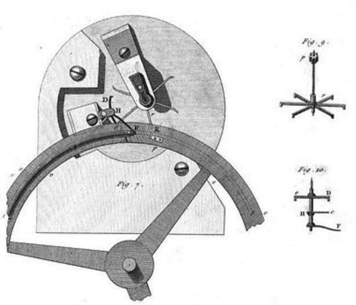
Modern calculator

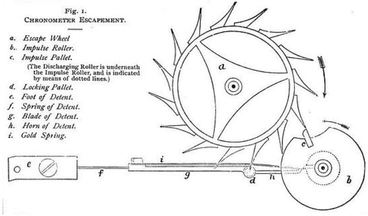
- Execute a number of precise operations
- Designed to contain a set of such operations and instructions
- Includes even more complex operations, graphing charting

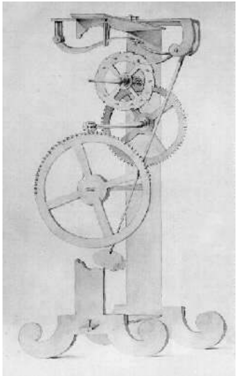


World's first computing device?

The escapement clock, a man-made device, to keep track of time.







Computation can be seen as a **sequence of states** a computing device does.

- Contains one or many sequence of states
- Can be an infinite number of sequence of states
- Or it can terminate with a final state

We can describe a computing device by describing all their possible computations.

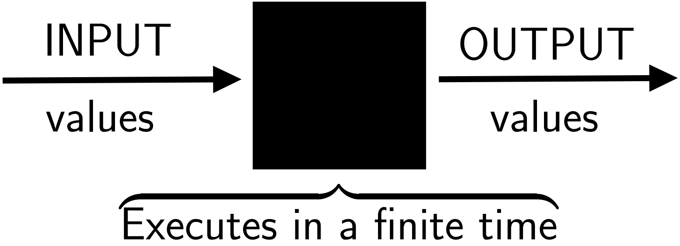
Algorithms

What is an algorithm?

In mathematics and computer science, an **algorithm** (/ˈælɡərɪðəm/ ⓘ) is a finite sequence of rigorous instructions, typically used to solve a class of specific problems or to perform a computation.[1] Algorithms are used as specifications for performing calculations and data processing.

- But can we call this a computer program?
- Or is it just a recipe, something higher than a sequence of code
- A higher level of abstraction of how to implement something we plan to develop. Example: how to find a name in a phone book

What is an algorithm?



Sequence of computational steps that transform the input into the output

An algorithm is

- a method
- or a computational procedure
- or a sequence of computational steps
- or a recipe

to transform the input into the output

An algorithm is characterised

- a method to solve something **speed**: executes in a finite amount of time
- **correctness**: the output should be correct

Algorithms are everywhere

From your kitchen, in your microwave oven, your washing machine, to your phone or computer. When you browse Internet web sites your web browser is using different algorithms to decide how to display data to you.

Our society relies on algorithms to suggest sentences for convicted criminals. We even use algorithms to keep people alive: the control systems from any modern car, or different medical devices.

Do we really need to study algorithms

Even if computing resources were infinite available, we would still need to study and understand algorithms. Why?

Because we would like to be certain that our program terminates in time and does so with the correct answer(s).

Time is more valuable than money

Computers systems might be fast, but they are not infinitely fast. Computing time is a precious resource. The memory of a computer system might be inexpensive but it is neither infinite nor free!

You can get back money after you spend it, but once time is spent, you can never get it back!

Select algorithms that use resources of time and space efficiently!

How to store and organize data

Part of the algorithm design is how we plan to store and organize data in order to compute something. We call this **data structures**

There is no single data structure which works well for all purposes

Hard problems or NP-complete

Algorithms are used to solve different problems in a precise amount of time, the speed of the algorithm. We must be able to answer how long does it take for an algorithm to produce its output.

But there are problems for which there are no algorithms that run in a reasonable amount of time. We call these **NP-complete** problems. and these are important because many real applications can contain such probmes.

But how can we describe or design algorithms?

As an abstraction of something you plan to build or use, including all basic operations to achieve that. For example, think you plan to search in a phone book, a person phone number by the name:

- you can start page by page searching for that name
- or you can jump directly to certain letters and start following from there
- or you can apply a different strategy, by 'cutting' the book into half, checking the letter in which half belongs, and applying all over again the same principle until the name is found

How can we write one algorithm?

You can write it in plain English or in a more precise way using mathematics. Some others are using a form of pseudocode.

In computer science, **pseudocode** is a description of the steps in an algorithm using a mix of conventions of programming languages (like assignment operator, conditional operator, loop) with informal, usually self-explanatory, notation of actions and conditions.[1][2] Although pseudocode shares features with regular programming languages, it is intended for human reading rather than machine control. Pseudocode typically omits details that are essential for machine understanding of the algorithm. The programming language is augmented with natural language description details, where convenient, or with compact mathematical notation.

Example phonebook

```
1: procedure PhoneBook(N)                          ▷ Returns person name: N
2:     N ← 1
3:     Open page number N
4:     Look at the page N
5:     if Person is on page N then
6:         Call person N                            ▷ The person name is N
7:     else
8:         Find next page. N ←N + 1 Go back to 3
9:     end if
10: end procedure
```

But why not using a programming language?

We must think what we are planning to do, and how we plan to do it. And for that, we must not rely on a programming language: we will be restricted by the limits of the specific programming language not being able to design and think freely about it.

A better approach is to think to write it as pseudocode or simple mathematics. This way we can have all flexibility and the power of precise mathematics.

Euclid's Algorithm or GCD

In mathematics, the **Euclidean algorithm**,[note 1] or **Euclid's algorithm**, is an efficient method for computing the greatest common divisor (GCD) of two integers (numbers), the largest number that divides them both without a remainder. It is named after the ancient Greek mathematician Euclid, who first described it in his *Elements* (c. 300 BC). It is an example of an *algorithm*, a step-by-step procedure for performing a calculation according to well-defined rules, and is one of the oldest algorithms in common use. It can be used to reduce fractions to their simplest form, and is a part of many other number-theoretic and cryptographic calculations.

Euclid's Algorithm

Greatest Common Divisor (GCD) of two numbers A and B is the largest number that divides both A and B. (Number here defined as an integer)

An integer is the number zero (0), a positive natural number (1, 2, 3, etc) or a negative integer with a minus sign (-1, -2, -3, etc) In mathematics we call this $\mathbb{Z}$ set of numbers.

Euclid's Algorithm

The very first version:

If $A = 0$ then GCD(A,B)=B, since the GCD(0,B)=B, and STOP
If $B = 0$ then GCD(A,B)=A, since the GCD(A,0)=A, and STOP
Write A in quotient remainder form ($A = B * Q + R$)
Compute then GCD(B,R) since GCD(A,B) = GCD(B,R)

Euclid's Algorithm, pseudocode

```
procedure GCD(A, B)                          ▷ The g.c.d. of A and B
    R ← A mod B
    while R ≠ 0 do                           ▷ We have the answer if R is 0
        A ← B
        B ← r
        R ← A mod B
    end while
    return B                                 ▷ The gcd is B
end procedure
```

Euclid's Algorithm, improved

```
procedure Euclid(A, B)                    ▷ The g.c.d. of A and B
    if B == 0 then
        return A                          ▷ The gcd is A
    else
        return Euclid(B, A mod B)
    end if
end procedure
```

Euclid's Algorithm, using mathematics

MODULE *Euclid*

EXTENDS Integers

VARIABLES x, y

CONSTANTS a, b

$$Init \triangleq (x = \mathrm{a}) \land (y = b)$$

$$\begin{aligned}
\text{Next} \triangleq\ & (\ x > y \\
& \land x' = x - y \\
& \land y' = y\ ) \\
\lor\ & (\ y > x \\
& \land y' = y - x \\
& \land x' = x\ )
\end{aligned}$$

Classes of algorithms

- Divide and Conquer
- Sorting
- Searching
- Dynamic programming
- Greedy algorithms
- Graph algorithms
- Shortest path
- Maximum flow
- Parallel algorithms
- Matrix operations
- Online algorithms
- Machine learning
- Linear programming
- String matching

Computer Programs

What is a computer program?

A **computer program** is a sequence or set of instructions in a programming language for a computer to execute. It is one component of software, which also includes documentation and other intangible components.[1]

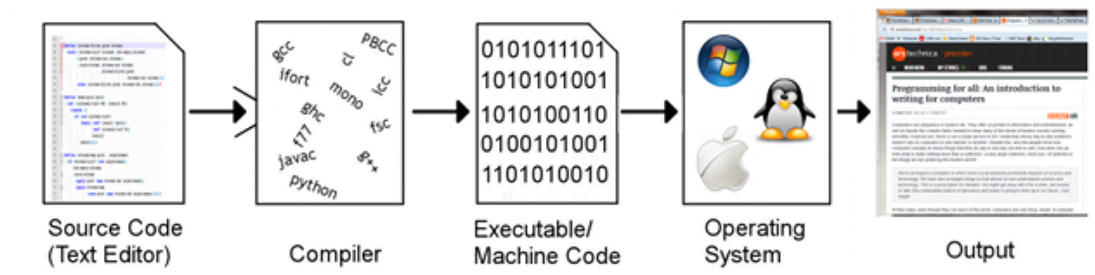
Program. Computation. States

- A program execution is a computation (behaviour)
- A computation is a sequence of states
- A state is an assignment of values to variables

A program is modelled as a set of computations, representing all possible program executions

Computers, digital systems are executing programs

From source code to executable



Program structure

A **computer program** is a sequence or set of instructions in a programming language for a computer to execute. It is one component of software, which also includes documentation and other intangible components.[1]

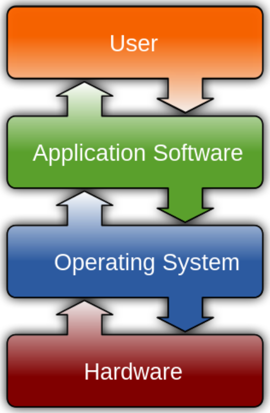
- Some programs run forever, some dont
- A program execution is defined by at least one computation
- A computation is a sequence of states
- And a state is an assignment of values to variables

Program types

- A program is modelled by a set of computations, representing all possible executions
- Remember an algorithm is just an abstract program
- Different programs: software applications and system software
- System software: operating systems

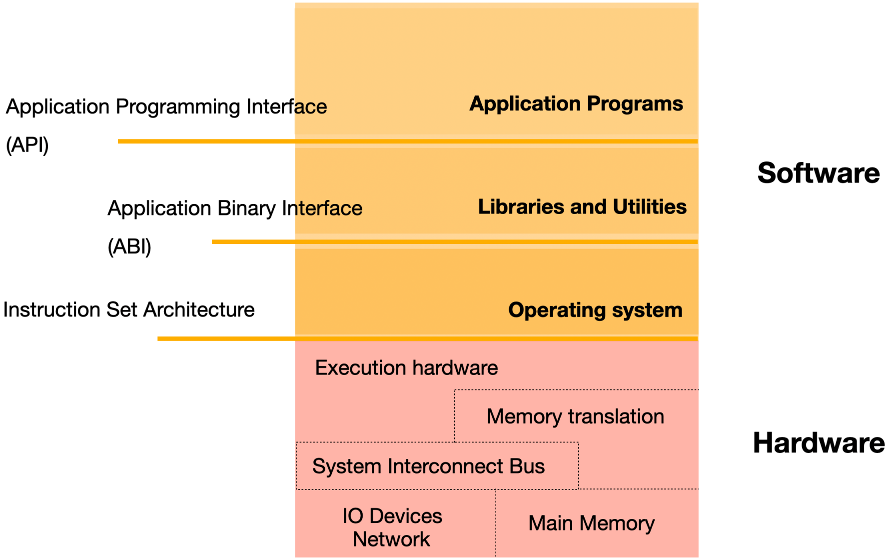
Application and System Programs

- **Software applications**: enterprise resource planning, customer relationship management, supply chain management software, web, middleware, databases
- **Operating systems**: macOS, RedHat, FreeBSD, Windows



Lesson 1

Computer Programs



Lesson 2
Software specifications. Formal methods

Lesson 3
Programming languages. Coding

Lesson 4
Data structures. Algorithms

Lesson 5
Computer system performance

Lesson 6
Build safe and secure software