A quick glance at Computer Science

Alberto Tonda, Ph.D., Senior permanent researcher (DR)

UMR 518 Mathématiques et Informatique Appliquées - PS, INRAE, U. Paris-Saclay
UAR 3611 Institut des Systèmes Complexes Paris Ile-de-France, CNRS

alberto.tonda@inrae.fr

Outline

- This is some text
  - And some smaller text

Computer science

- What is computer science?
  - **Computer science is the study of computation, information, and automation.** It explores theoretical principles underlying algorithms, logic, and complexity; and practical design and implementation of hardware and software systems. CS seeks to formalize problem-solving, optimize efficiency, and develop intelligent systems that interact with the world. (**ChatGPT**)
- Some people claim it's not *science*, it's **engineering**
  - "Can you do computer science without a computer?"
  - (Italy) Curriculum differences CS/CSE are minimal
  - I share this point of view, but it's not entirely correct

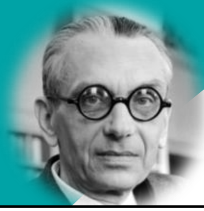
Pre-computer science

M. Al-Khwarizmi

G. W. von Leibniz

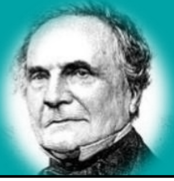
G. Boole

E. Post



K. F. Gödel

A. Lovelace

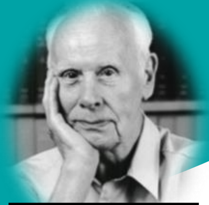


C. Babbage

J. P. Eckert

J. Mauchly

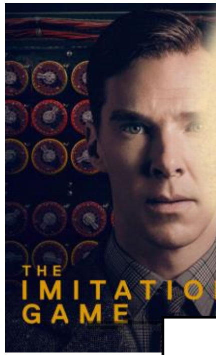
T. Flowers

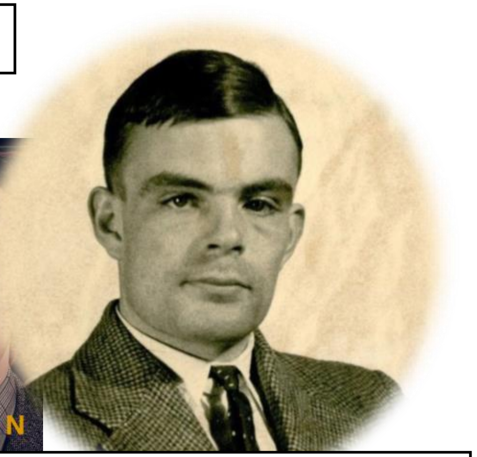


G. Stibiz

K. Zuse

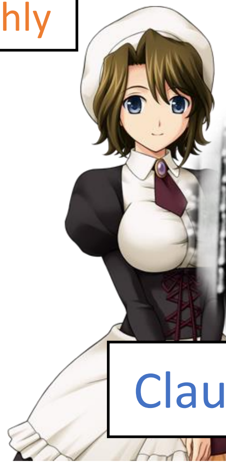
Computer science





Alan Turing

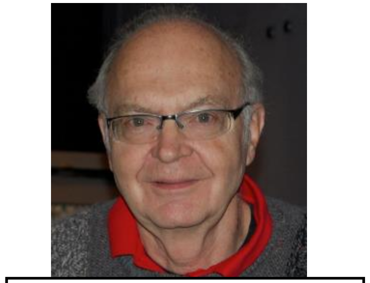
John Von Neumann



Claude Shannon

Alonzo Church

Grace Hopper



Donald Knuth

What are the 3 fundamental parts of a computer?

What are the 3 fundamental parts of a computer?

- Disk storage (hard drive)
- Computation (Central Processing Unit, CPU)
- Temporary memory (Random Access Memory, RAM)

- When programs are executed
  - Loaded from hard drive to RAM
  - CPU reads and writes information from/to RAM*
  - Eventually, results might be written back to hard drive

*There are also intermediate memories, like **cache**, but that's not too important

Why should we care?

- RAM data is volatile, hard drive data persists
- Why does RAM exist, then?

Why should we care?

- RAM data is volatile, hard drive data persists
- Why does RAM exist, then?
    - Access to RAM is *F A S T E R* (by orders of magnitude!)
    - Reading and writing to hard drive is *slow*

Practical advice

- Avoid reading and writing to/from disk more than necessary
  - Ideally, load everything at the beginning, write at the end
  - However, we might want to save intermediate results

- You can improve speed by using more RAM
  - For example, store results of computations (no recomputing)
  - However, your amount of RAM is limited (**out of memory errors**)
  - OOM errors are the most common errors I got
  - **Memory/CPU trade-off**

Memory management

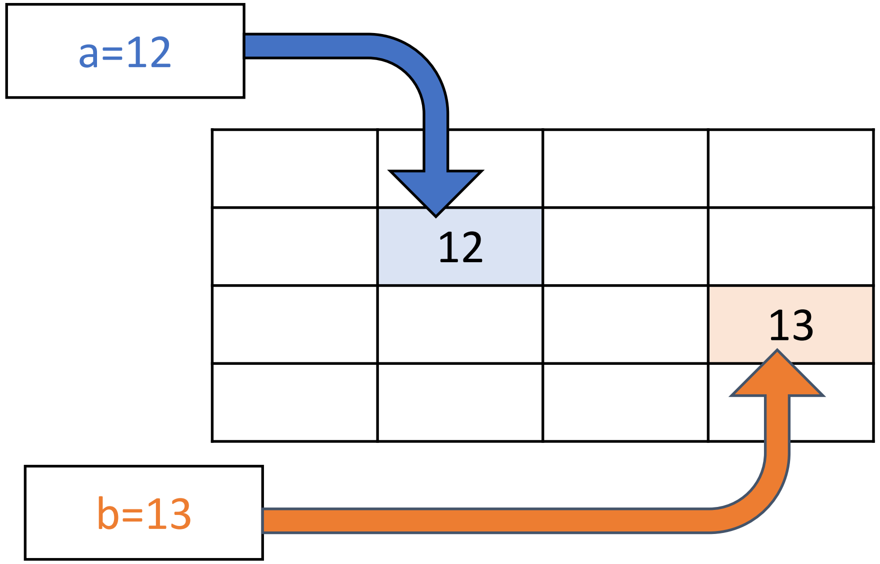
- Everything in a computer is stored as…?

Memory management

- Everything in a computer is stored as **bits**! 0/1
  - One single element is a **bit**, 8 bits make a **Byte (octet)**
  - If your RAM memory is 16 GB (GigaBytes), it means it can store up to 16 * 10^9 Bytes (and even more, using tricks)
  - Memory is now so large that minimal amounts are kB (10^3 Bytes)
- Estimate or measure how much memory your program takes
  - E.g. in Python a **float** takes 8 Bytes (8*8=64 bits)
  - So you could store ~2 * 10^9 Python floats in RAM memory
  - But your operative system is also using memory; and so are all the other running programs, so in practice much less

Memory management

- Computer memory is divided into “cells” (memory blocks)
  - Each block has its own address, and can store bits
  - **Values** inside variables are stored in blocks
  - Sometimes, **references** to blocks



Memory management

- Memory management
    - Get rid of the stuff in memory you no longer use
    - Load new stuff in memory, properly
    - How do you do that in Python?

Memory management

- Memory management
  - Get rid of the stuff in memory you no longer use
  - Load new stuff in memory, properly
  - How do you do that in Python?

- YOU DON’T*
  - Python does that for you (**garbage collector**)
  - *however, the GC is not perfect, and you can invoke it manually
  - See the documentation of the **gc** Python package

Practical advice

- Other languages allow for more precise control of memory
  - C, C++ (they are also called more low-level)
  - They are also much faster, but way more annoying
  - Screw up memory allocation -> program crashes (**segmentation fault**, aka “there is a problem *somewhere*”)
  - *Memory leaks* are also possible (do not release used memory)

Images

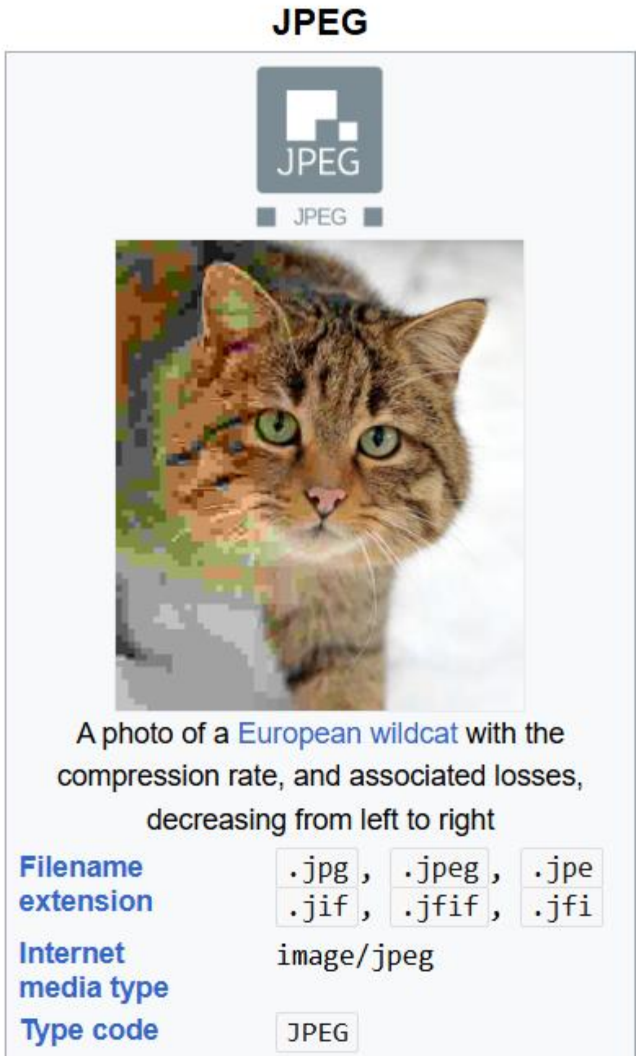
- How much memory for an image?
  - 745 x 1,134 x RGB, 1 Byte/channel

Images

- How much memory for an image?
  - 745 x 1,134 x RGB, 1 Byte/channel
  - You would expect something like 745 x 1,134 x 3 = 2,534,490 Bytes ~= 2.5 MB
  - But this image **only takes ~105 KB**!
  - How is that possible?

Images

- Naïve encoding of images (BITMAP) is rare
- JPG, PNG, WEBP are algorithms
- You *decompress* a *compressed* image
- Trade-off **computation** (de/compress) for **memory**



JPEG

A photo of a European wildcat with the compression rate, and associated losses, decreasing from left to right

Filename extension	.jpg, .jpeg, .jpe .jif, .jfif, .jfi
Internet media type	image/jpeg
Type code	JPEG

Why are some languages faster than others?

- Fast languages
  - Assembly, C, C++, Fortran, …
- Slow languages
  - Java, Python, R, …

Why are some languages faster than others?

- Compiled languages (Assembly, C, C++, …)
  - Computers represent EVERYTHING as bits
  - Including **instructions**! But “000101010” is not very readable
  - Compilers transform **human-readable programs** into **binary code**
  - Binary code is optimized for your PC! So it’s very fast
- Disadvantages
  - Compiling a program can take some time
  - Code works with restrictions (only Windows, only Linux, only Intel, only AMD, …)

Why are some languages faster than others?

- Interpreted languages (Java*, Python, R, …)
  - Rely on software layer (interpreter) which reads and executes code
  - Same code can work on very different architectures!
  - Take care of memory management automatically; no compiling
- Disadvantages
  - Interpreter adds extra steps for execution, it's *slower*
  - Orders of magnitude slower (easily 10-100x)
  - Also, extremely difficult to optimize memory use and speed

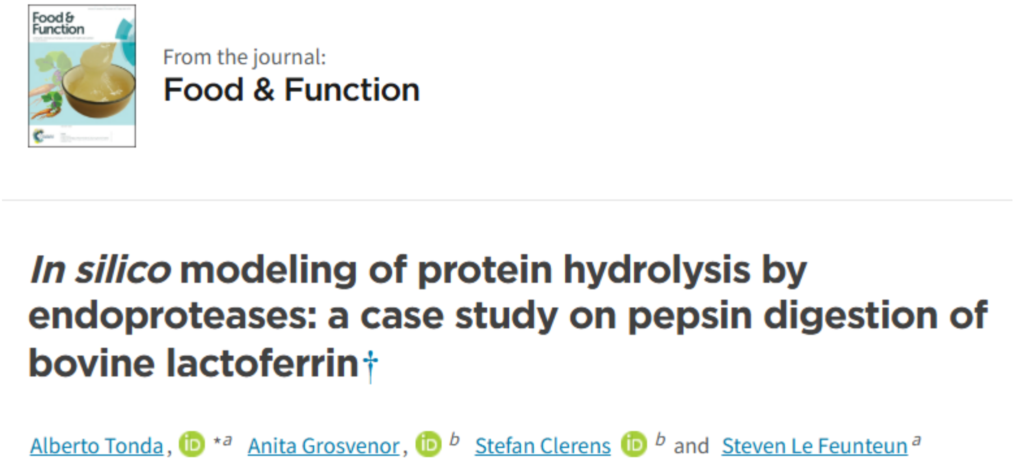
**Java actually makes a hybrid compiled/interpreted mess*

Python is interpreted, *but*…

- Python is popular for its tricks to improve speed
  - Lots of Python libraries use a technique called *binding*
  - Compiled code is <u>called and executed</u> from Python code
  - So, the most time-consuming parts run **faster**!
  - Libraries like **numpy**, **pytorch**
- In principle, <u>you</u> can also create bindings
  - It’s a bit hard, because it requires knowing well both Python AND the compiled language (usually C, C++, or Fortran)
  - A colleague (US LANL) did it for old Fortran code

Why should we care?

- Sometimes code needs to be fast and Python is not enough
- Example from the paper below
  - Simple code, running stochastic simulations
  - It needed to run *thousands* of times
  - C++ improved execution time **10-100x**
  - Months to hours



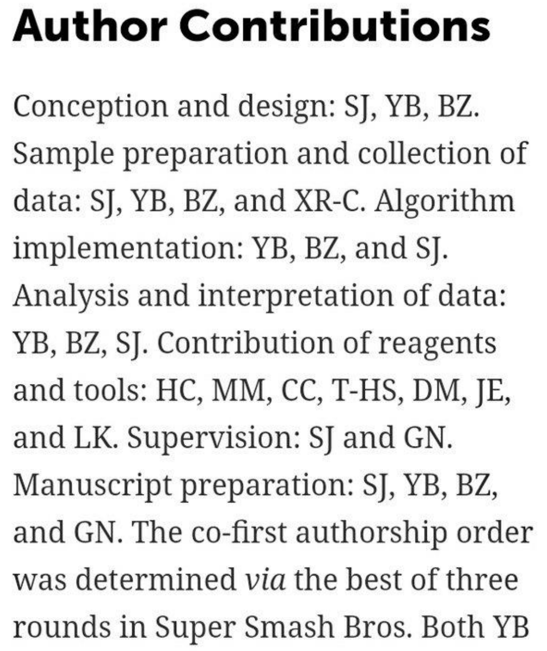
From the journal:
Food & Function

In silico **modeling of protein hydrolysis by endoproteases: a case study on pepsin digestion of bovine lactoferrin†**

Alberto Tonda, *a Anita Grosvenor, b Stefan Clerens b and Steven Le Feunteun a

In conclusion

- Computer Science is a relatively young field
  - Two souls: one more theoretical, the other more engineering
  - Tradition of anarchy and issues with authority (open source)
  - Maybe less “dogmatic” than other disciplines? IDK
  - We can easily rerun other people’s experiments!
  - Fast to adapt, CS changes really quickly over time
  - Computer Scientists think they are funny and witty



Author Contributions

Conception and design: SJ, YB, BZ. Sample preparation and collection of data: SJ, YB, BZ, and XR-C. Algorithm implementation: YB, BZ, and SJ. Analysis and interpretation of data: YB, BZ, SJ. Contribution of reagents and tools: HC, MM, CC, T-HS, DM, JE, and LK. Supervision: SJ and GN. Manuscript preparation: SJ, YB, BZ, and GN. The co-first authorship order was determined *via* the best of three rounds in Super Smash Bros. Both YB

INRAE

MIA PARIS-SACLAY EKINOCS

Institut des Systèmes Complexes Paris Île de France

université PARIS-SACLAY

Questions?

alberto.tonda@inrae.fr