ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI FORLÌ

DIT PhD Introduction to Computational Thinking and Programming

Lesson 1. Computational Thinking

Alberto Barrón-Cedeño
a.barron@unibo.it

29/10/2024

L'idonietà

This activity includes two modules: Programming and Statistics

You will submit your solution to a couple of problems/exercises from each module

Details at due time

Table of Contents

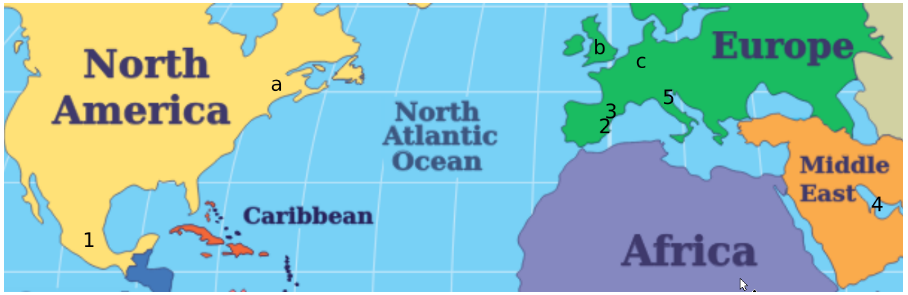
1. You and your instructor
2. Contents
3. Computational Thinking

You and your instructor

Quick Introduction

Who are you?

The instructor



1. BEng in Computing at UNAM, Mexico
   MSc in Computing at UNAM, Mexico
   - Internship at UdeM, Canada
2. MSc in AI at UPV, Spain
   PhD in AI at UPV, Spain
   - Internship at UofS, UK
3. Post-doc at UPC, Spain
   - Internship at BUW, Germany
4. Scientist at QCRI, Qatar
5. Professor at UniBO, Italy

PhDs that I am supervising

4th year (about to graduate)

Arianna Muti

Hidden in Plain Sight: Detecting Misogyny beneath Ambiguities and Implicit Bias in Language

- Internship at Expert.ai (Modena, Italy)
- Internship at U. of Groningen (Groningen, The Netherlands)
- 10+ peer-reviewed full papers published (one upcoming at EMNLP)
- Transitioning towards a PostDoc at Bocconi University

Katerina Korre

A Universal and Cross-language Approach to Internet Hate Speech Detection and Analysis

- Internship at Symanto.ai (Valencia, Spain)
- 8+ peer-reviewed full papers published (two under review in journals)
- Transitioning towards a PostDoc at Athens University

PhDs that I am supervising

2nd year

Paolo Gajo

NLP Technologies for Gastronomy

- Internship at Dalhousie University (Halifax, Canada)
- 4+ peer-reviewed full papers published (two during his masters)

Unfinished

Francesco Fernicola

Return to the Source: Assessing Machine Translation Suitability

- In co-supervision with EURAC Research (Bolzano, Italy)
- 5+ peer-reviewed full papers published (two during his masters)
- Currently Computational Linguist at the European Parliament

Computing at DIT

Recent and ongoing research projects[1]

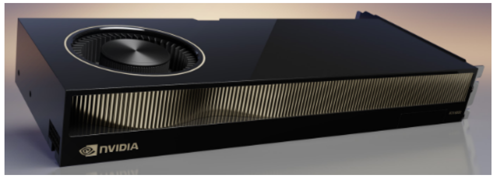
- UpSkills on upgrading the (technological) skills of language students
  https://upskillsproject.eu
- UNITE on exploiting LLMs for language learning
  http://site.unibo.it/unite
- !Translate on augmenting machine translation with explanations
  https://site.unibo.it/no-translate
- Gastrowiki on producing and fixing definitions
  https://site.unibo.it/gastrowiki

[1] Non exhaustive

Computing at DIT

Computing Power[2]

- 4 NVIDIA RTX 6000 Ada
  https://www.nvidia.com/en-us/design-visualization/rtx-6000
- 2 NVIDIA Quadro P4000
  https://www.techpowerup.com/gpu-specs/quadro-p4000.c2930



[2] Dedicated to deep learning (training and out-of-the-box)

Lesson coordinates

Slides and code available at:

https://albarron.github.io/teaching/phd-comp-thinking/

Tools

Python 3 programming language
We will use Google's Colab: https://colab.research.google.com

For (more) serious affairs, you could consider

1. Command line or
2. Integrated development environment; e.g., Pycharm[3], Eclipse[4] or local Jupyter[5]

[3] https://www.jetbrains.com/pycharm/

[4] https://www.eclipse.org/

[5] https://jupyter.org/

Contents

Lesson contents

Introduction to computational thinking

1. Problem definition and solving
2. Decomposition
3. Pattern recognition
4. Abstraction
5. Algorithmic thinking

Programming

6. Introduction to programming
7. Jupyter notebooks
8. Basic operations
9. Dealing with text

Computational Thinking

The tools we use have a profound and devious influence on our thinking habits, and therefore on our thinking abilities.

Edsger W. Dijkstra[6]

[6] https://amturing.acm.org/award_winners/dijkstra_1053701.cfm

Computational Thinking

"[Computational Thinking] represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use"

Jeannette M. Wing, CMU (2006)

Humans and Computers

Computational methods and models give us the *courage* to solve problems and design systems

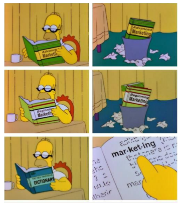
Computational thinking confronts the riddle of machine intelligence:

- What can humans do better than computers?
- What can computers do better than humans?

  Some examples of each?

- What is computable?

A few definitions



Problem

1. A difficulty that has to be resolved or dealt with
2. A question to be answered, schoolwork exercise
   Antonyms: solution

System

1. A group of interacting or interrelated elements that act according to a set of rules to form a unified whole

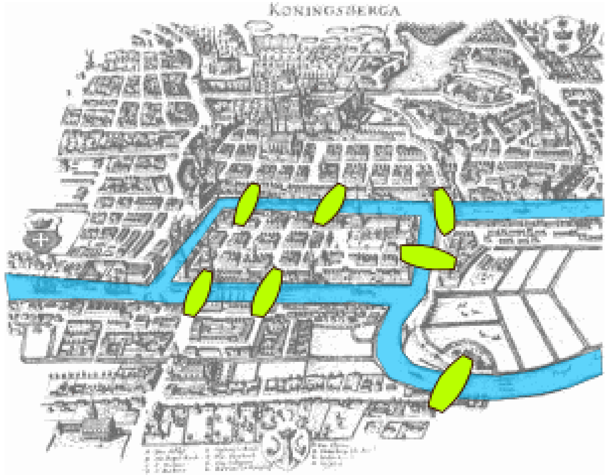
Computability

1. The ability to solve a problem in an effective manner

```
https://en.wiktionary.org/wiki/problem
https://en.wikipedia.org/wiki/System
https://en.wikipedia.org/wiki/Computability
```

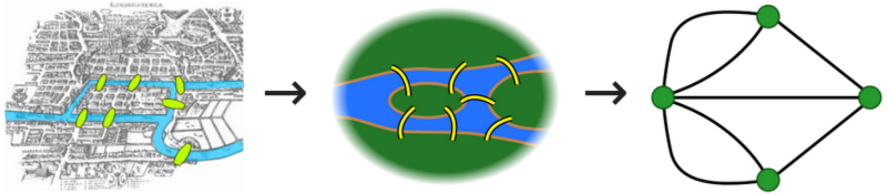
Activity 1: The Seven Bridges of Königsberg

Task: Devise a path through the city of Königsberg that would cross each of the bridges once and only once



Activity 1: The Seven Bridges of Königsberg

Looking for a solution



Can you devise a solution using this abstraction?

Solution: There is no solution

The foundations of graph theory
Leonhard Euler (1736)

What is involved in computational thinking

- Defining problems
- Solving problems
- Designing systems
- Understanding human behavior

All by drawing on the concepts fundamental to computer science

- How difficult is it to solve?
- What's the best [doable|acceptable|affordable] way to solve it?
- An approximate solution is good enough?
- False positives or false negatives are allowed?

		predicted label	
		positive	negative
true	positive	true positive	false positive
label	negative	false negative	true negative

How to deal with a difficult problem?

[By] reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation

[. . .] using abstraction and decomposition when attacking a large complex task or designing a large complex system

Have you solved a problem using any of these techniques?

Copying a drawing?
https://www.wikihow.com/Copy-a-Drawing-or-Picture-by-Hand

The thinking in computational thinking

Thinking in terms of . . .

- Prevention
  Do you backup your mobile phone?
  There are two kinds of people
  1. those who backup
  2. those who have never lost all their data [mobile phone]
- Protection
  Do you use a case to protect your mobile phone?

Getting ready to recover from worst-case scenarios through

redundancy	→	If I keep money in my backpack, I can go home even if I loose my wallet
damage containment	→	If I have an exam, I will take an earlier train than usual
error correction	→	Before handling my report, I will pass a spell checker

Computational thinking is search, search, and more search

How do you

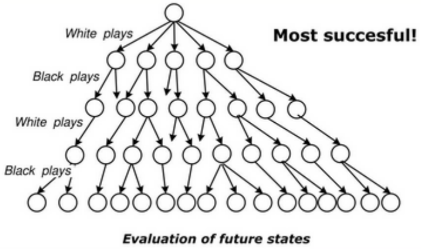
- Buy the best possible item on Amazon?
- Find the best match on Tinder?
- Spot the most entertaining tiktok?

How do you win at UNO?

How do you win at dominoes?

How do you win at chess?

How do *standard computers* win at chess?



What computational thinking is and is not

Characteristics

Conceptualising, not programming

- Computer science is not computer programming
- Beyond (∼beside) being able to program a computer
- Thinking at multiple levels of abstraction

Fundamental skill

- A skill every human being must know to function in modern society

A way that humans, not computers, think

- A way humans solve problems
- Not trying to get humans to think like computers

Computers are dull and boring
Humans are clever and imaginative
Humans make computers exciting

Computational Thinking: The three As

An iterative process based on three stages:

Abstraction (Problem Formulation). One attempts to conceptualize a problem verbally, e.g., by trying to formulate a question such as "How does gravity work?," or through visual thinking, e.g., by drawing a diagram identifying objects and relationships

Automation (Solution Expression). It is expressed in a non-ambiguous way so that the computer can carry it out; e.g., through computer programming (or through *prompting*?)

Analysis (Execution & Evaluation). The solution gets executed (by the computer) in ways that show the direct consequences of one's own thinking. Visualisations could support the evaluation of solutions

(Repenning et al., 2016)

Algorithm

An algorithm is. . .

Definition 1 A finite sequence of well-defined (computer-implementable) instructions, typically to solve a class of problems or to perform a computation

`https://en.wikipedia.org/wiki/Algorithm`

Definition 2 An explicit, precise, unambiguous, mechanically-executable sequence of elementary instructions, usually intended to accomplish a specific purpose.

Erickson (2019, p. 1)

Activity 2: The panino[7]

Problem: Write the algorithm to prepare a *panino*

[7] Since recipes are *just* algorithms

Activity 6: The panino

My algorithm to prepare a *panino*[8]

Ingredients:

bread • *prosciutto crudo* • *pecorino di Pienza* • *carciofini sott'olio*

1. Cut the bread into two halves horizontally
2. Add three slices of *prosciutto* on top of the bottom half
   * get sure not to go beyond the border of the bread
3. Evenly distribute some slices of *pecorino*
4. Add 3 pieces of *carciofini*
   * get sure not to get too much oil
5. Put the top half of bread on top
6. Enjoy

[8] Via Taranto from `https://ilpaninobologna.com`

Activity 6: The panino

Possible *issues* in your/my recipes[9]

- Under-specification?
- Lack of identification of the input?
- Imprecise identification of the problem?

[9] Keep in mind that this is a toy problem

Describing Algorithms

The 4 components of an algorithm

What: A precise specification of the problem that the algorithm solves

How: A precise description of the algorithm itself

Why: A proof that the algorithm solves the problem it is supposed to solve[10]

How fast: An analysis of the running time of the algorithm[11]

- No particular development order
- Write for an audience; this is not intended for yourself
- Write for people who is not as clever as you are[12]

From (Erickson, 2019, p. 11)

[10] Not covered in this lesson

[11] idem

[12] For instance, yourself 6 months ago

Natural vs Programming languages

Natural languages

- An *ordinary* language (e.g., Italian)
- Written or oral
- It has evolved naturally in humans, usually without specific and deliberate planning[13]
- *Problem*: ambiguity
  (e.g., "visiting relatives can be annoying")

Programming languages

- Formal-born languages
- Specific syntactic rules that avoid ambiguous statements
- *Sentences* convey one single meaning
- They can have a significant degree of abstraction

[13] Consider Klingon or Sith

Programming language

A formal language comprising a set of instructions that produce various kinds of output [given an input][14]

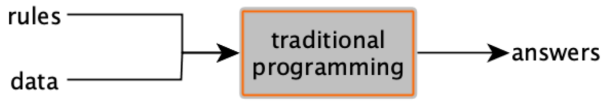


Diagram from L. Moroney's Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning

[14] `https://en.wikipedia.org/wiki/Programming_language`

References I

Erickson, J.
2019. *Algorithms*. Independently published.

Repenning, A., A. Basawapatna, and N. Escherle
2016. Computational thinking tools. In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Pp. 218–222.

Wing, J. M.
2006. Computational thinking. *Communications of the ACM*, 49(3):33—-35.