Principles of Computer Science II

Computational Thinking

Ioannis Chatzigiannakis

Sapienza University of Rome

Lecture 1

Computational Thinking

Wing, J. M. 2006 Computational thinking. CACM 49, 33–35

Computational thinking is taking an approach to solving problems, designing systems and understanding human behaviour that draws on concepts fundamental to computing.

Wing, J. M. 2006 Computational thinking. CACM 49, 33–35

Computational thinking represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Wing, J. M. 2006 Computational thinking. CACM 49, 33–35

Thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction.

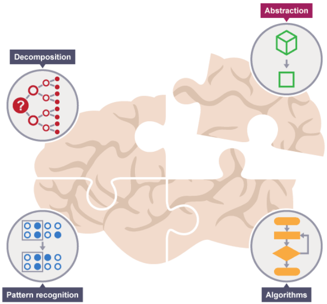
The riddle of machine intelligence

Computational thinking confronts the riddle of machine intelligence:

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

Computational Thinking

- Computers are here to help us.
- What do we need from computers?
- What is our problem?
- Computational Thinking allows us to understand what needs to be solved.
- Four key techniques (cornerstones) to computational thinking:
  1. Decomposition – breaking down a complex problem or system into smaller, more manageable parts
  2. Pattern Recognition – looking for similarities among and within problems
  3. Abstraction – focusing on the important information only, ignoring irrelevant detail
  4. Algorithms – developing a step-by-step solution to the problem, or the rules to follow to solve the problem



Computational Thinking vs Programming

Thinking computationally is not programming.

- . . . not even thinking as a computer.
- Programming tells computer what to do / how to do it.
- Computational thinking enables us to understand what we need to tell to computers.
- . . . what to program.

Examples:

- Explain to a friend how to drive to your house
- Organize a party at the park
- Prepare your luggage
- Teach a kid addition/subtraction
- . . .

Decomposition

Turn a complex problem into one we can easily understand.

- . . . probably you already do every day.
- The smaller parts are easier to solve.
- . . . we already know/have the solutions.

Examples:

- Brushing our teeth
  Which brush? How long? How hard? What toothpaste?
- Solving a crime
  What crime? When? Where? Evidence? Witnesses? Recent similar crimes?
- . . .

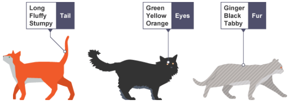
Pattern Recognition

We often find patterns among the smaller problems we examine.

- The patterns are similarities or characteristics that some of the problems share.

Example: Cats

- All cats share common characteristics.
  they all have eyes, tails and fur.
- Once we know how to describe one cat we can describe others, simply by following this pattern.



Abstraction

Hiding irrelevant details to focus on the essential features needed to understand and use a thing

- A compression process – multiple different pieces of constituent data to a single piece of abstract data.
  e.g., "cat"
- Ambiguity – multiple different references.
  e.g., "happiness", "architecture"
- Simplification – no loss of generality
  e.g., "red" - many different things can be red

Thought process wherein ideas are distanced from objects

Abstraction Example: Car vs Car Breaks

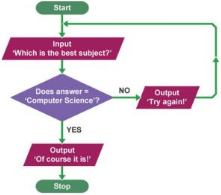
- Do we know how car breaks work?
- Do we know how to use them?

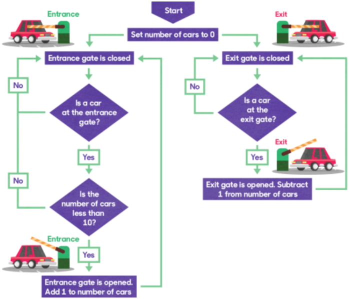
Filter out (ignore) the characteristics that we don't need in order to concentrate on those that we do.

Algorithms

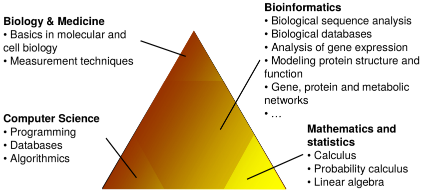
A plan, a set of step-by-step instructions to solve a problem.

- In an algorithm, each instruction is identified and the order in which they should be carried out is planned.





Bioinformatician's skill set



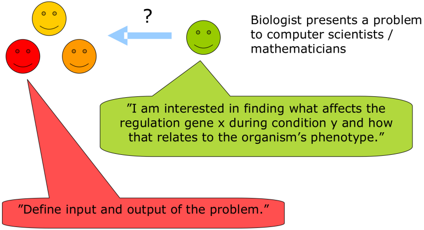
Where would you be in this triangle?

Prof. Juho Rousu, 2006

Bioinformatician's skill set

- Statistics, data analysis methods
  - Lots of data
  - High noise levels, missing values
  - #attributes $\gg$ #data points
- Programming languages
  - Scripting languages: Python, Perl, Ruby, . . .
  - Extensive use of text file formats: need parsers
  - Integration of both data and tools
- Data structures, databases
  - New measurement techniques produce huge quantities of biological data.
- Scientific computation packages
  - R, Matlab/Octave, . . .

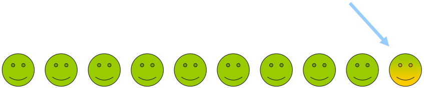
Bioinformatician's Competences



Prof. Esa Pitkänen, 2008

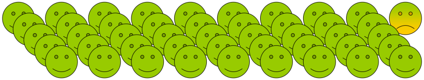
Bioinformatician's Competences

Bioinformatician is a part of a group that consists mostly of biologists.



Prof. Esa Pitkänen, 2008

Bioinformatician's Competences

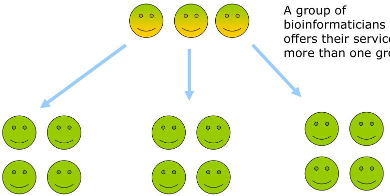


...biologist/bioinformatician ratio is important!

Prof. Esa Pitkänen, 2008

Bioinformatician's Competences

A group of bioinformaticians offers their services to more than one group



Prof. Esa Pitkänen, 2008

Axis 1: Python

1. Integrated Development Environment
2. Data Structures
3. Data Sets
4. Data Formats
5. Data Storage
6. Visualization

Axis 2: Algorithms

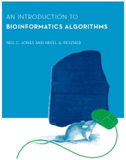
1. Complexity Analysis
2. Sorting
3. Exhaustive Search
4. Branch-and-Bound Algorithms
5. Greedy Algorithms
6. Divide-and-Conquer Algorithms
7. Data Mining Algorithms

Axis 3: Cloud Computing

1. Cloud Storage
2. Databases
3. Elastic Compute
4. Handling Large Data Sets

Literature

Jones, Pevzner: An Introduction to Bioinformatics Algorithms. MIT Press, 2004



1^{st} Assignment

https://www.hackerrank.com/

- Complete a total of 50 Python challenges from the following subdomains:
  - Python: Basic Data Types (any 5), Strings (any 4), Sets (any 4), Math (any 4), IterTools (any 4), Collections (any 4)
  - Algorithms: Warmup (10), Sorting (any 10), Strings (any 5)
- You can cooperate, You can search on the Internet, . . .
- You need to write **your own code**
- Email `ichatz@diag.uniroma1.it`
  Subject: [PCS2] Homework 1
  Your GitHub repository with your solutions, for all challenges.
  Also send your hackerrank user account link:
  https://www.hackerrank.com/{username}