

Principles of Computer Science II

Computational Thinking

Marco Zecchini

Sapienza University of Rome

Lecture 1

Do we really need this course?










SAPIENZA
UNIVERSITÀ DI ROMA

Catalogo dei Corsi di studio

Principles of Mathematics	1049371	Second semester	6	UK
Organic and inorganic chemistry	1049372	Second semester	6	UK
Introduction to biomedical statistics	1049376	Second semester	12	UK
Biology of the cell	1049373	Second semester	6	UK

Second year (Year of enrolment 2023/2024)

COURSE	CODE	SEMESTER	CFU	SSD	LANGUAGE
Genetics and computational genomics	1052115	First semester	6	BIO/18	
Principles of Computer Science II	1049261	First semester	6	ING-INF/05	
Microbiology	1049256	First semester	6	BIO/19	
Molecular biology 	1049375	First semester	6		
Biochemistry 	1049377	First semester	6		

Do we really need this course?










SAPIENZA
UNIVERSITÀ DI ROMA

Catalogo dei Corsi di studio

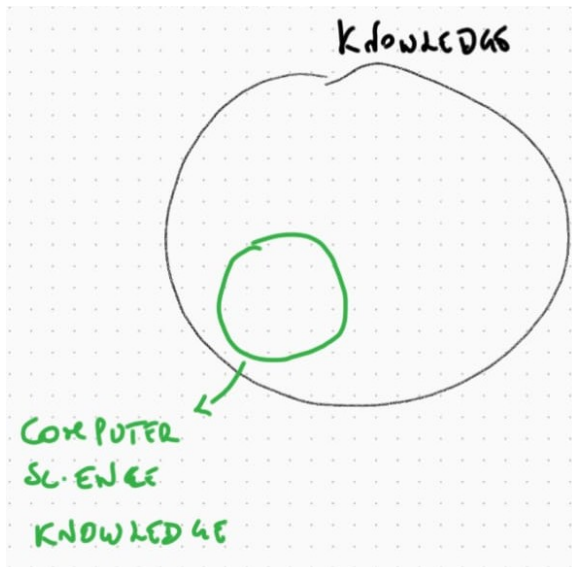
Principles of Mathematics	1049371	Second semester	6	UK
Organic and inorganic chemistry	1049372	Second semester	6	UK
Introduction to biomedical statistics	1049376	Second semester	12	UK
Biology of the cell	1049373	Second semester	6	UK

Second year (Year of enrolment 2023/2024)

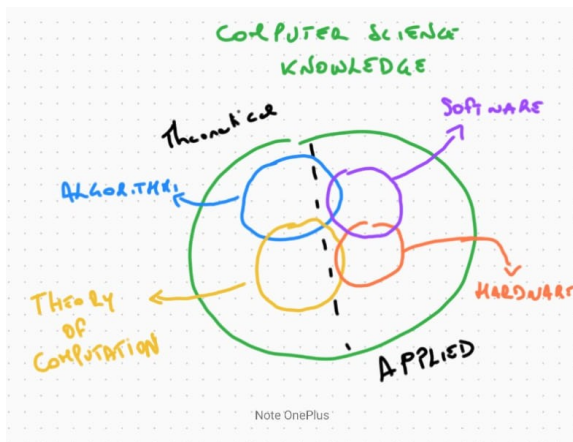
COURSE	CODE	SEMESTER	CFU	SSD	LANGUAGE
Genetics and computational genomics	1052115	First semester	6	BIO/18	
Principles of Computer Science II	1049261	First semester	6	ING-INF/05	
Microbiology	1049256	First semester	6	BIO/19	
Molecular biology 	1049375	First semester	6		
Biochemistry 	1049377	First semester	6		

Yes!

Do we really need this course?



Do we really need this course?



- Introduction to Programming and Computer Architecture;
- Introduction to Python. Installation and usage;
- Get and give information. Input/output from/to the user;
- Standard Python data structures: sequences, iterations, slices, lists, strings, dictionaries;
- Conditions. if, else, elif;
- Organizing the code. Functions, objects, classes, files;
- Python and multimedia. Data structures for images.

In PCSI

```
# Function to filter list items based on their length
def filter_items_by_length(items, min_length):
    # Filter items with length greater than or equal
    # to min_length
    return [item for item in items if len(item) >=
            min_length]

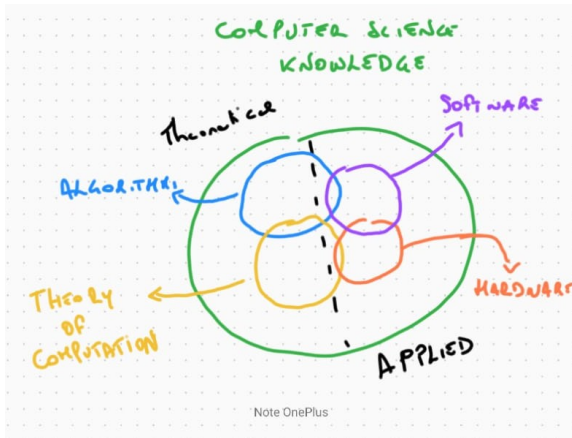
# Create a list of items
lst_listings = ['Item_1', 'Longer_Item_2', 'Item_3', '
                Very_Long_Item_4', 'Item_5']

min_length = 10

filtered_items = filter_items_by_length(lst_listings,
                                         min_length)

for item in filtered_items:
    print(item)
```

In PCSI



What about algorithms?

- **Problem:** We want to sort a vector of elements
- **Instance of the problem:** We want to sort this specific

vector:

8	3	5	4	6
---	---	---	---	---

What about algorithms?

- **Problem:** We want to sort a vector of elements
- **Instance of the problem:** We want to sort this specific vector:

8	3	5	4	6
---	---	---	---	---
- **Application in Bioinformatics:** Sequence Database Search: Sorted DNA sequences can be used for faster searching in large sequence databases, such as BLAST, which relies on efficient sequence comparison algorithms

First algorithm: Selection Sort

Sorting a vector: [8, 3, 5, 4, 6]

Alg. Key intuition: Always Look for the smallest next minimum.

- **Initial Vector:**

8	3	5	4	6
---	---	---	---	---

- **Step 1: Find the minimum and swap with the first element [3]**

3	8	5	4	6
---	---	---	---	---

- **Step 2: Find the next minimum [4] and swap with second element**

3	4	5	8	6
---	---	---	---	---

- **Step 3: Find the next minimum [5] (already in place)**

3	4	5	8	6
---	---	---	---	---

- **Step 4: Find the next minimum [6] and swap with 4th element**

3	4	5	6	8
---	---	---	---	---

First Example: Insertion Sort

Sorting a vector: [8, 3, 5, 4, 6]

Alg. Key intuition: Divide the problem into smaller problem.

- **Initial Vector:**

8	3	5	4	6
---	---	---	---	---

- **Step 1: Insert 3 into the sorted portion [8]**

3	8	5	4	6
---	---	---	---	---

- **Step 2: Insert 5 into the sorted portion [3, 8]**

3	5	8	4	6
---	---	---	---	---

- **Step 3: Insert 4 into the sorted portion [3, 5, 8]**

3	4	5	8	6
---	---	---	---	---

- **Step 4: Insert 6 into the sorted portion [3, 4, 5, 8]**

3	4	5	6	8
---	---	---	---	---

Comparison of the two approaches

- **Insertion Sort:**

- Builds the sorted list one element at a time by inserting each new element into its correct position.
- Performs fewer comparisons when the list is nearly sorted.
- Easier to implement and understand for small lists.

- **Selection Sort:**

- Repeatedly selects the smallest (or largest) element from the unsorted part and swaps it with the first unsorted element.
- Always performs the same number of comparisons, regardless of how sorted the list is.
- Simpler in terms of swapping, but may involve more overall data movement.

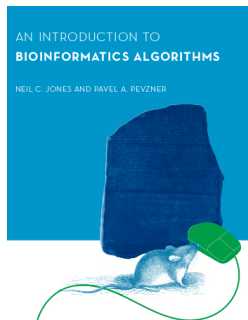
- **Key Differences:**

- **Insertion Sort** tends to be faster when the list is almost sorted.
- **Selection Sort** always performs the same number of comparisons, but it swaps elements more frequently.

Other Example

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

Section 2.5, the US changing problem.

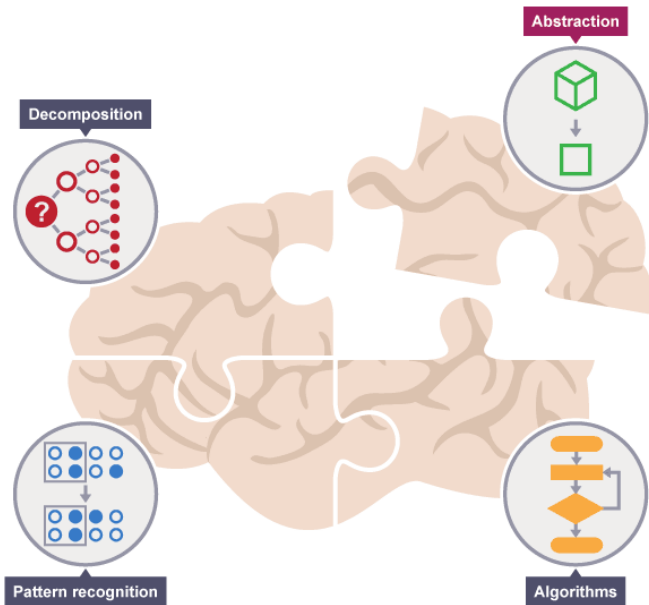


So, why do need this course?

We need this course to learn how to recognize different computer science/bioinformatics problems, **solving** them with an **efficient** algorithms and, finally, realize these algorithms in practice (implementing them).

Computational Thinking

- Computational Thinking allows us to understand what needs to be solved.
- Four key techniques (cornerstones) to computational thinking:
 - ➊ Decomposition – breaking down a complex problem or system into smaller, more manageable parts
 - ➋ Pattern Recognition – looking for similarities among and within problems
 - ➌ Abstraction – focusing on the important information only, ignoring irrelevant detail
 - ➍ 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?
- ...

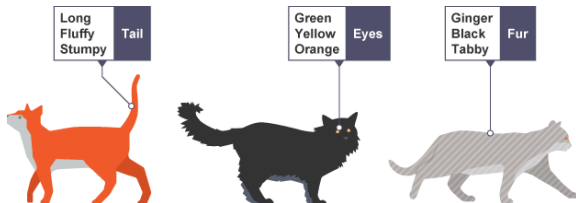
Pattern Recognition

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



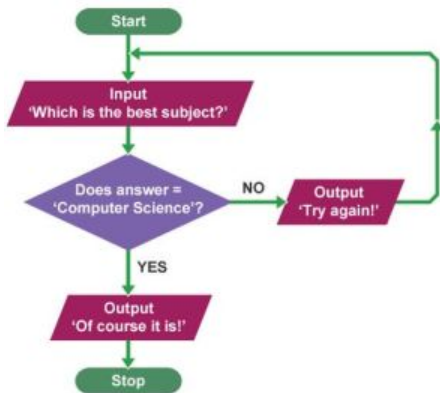
- Do we know how car breaks work?
- Do we know how to use them?

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

Algorithms

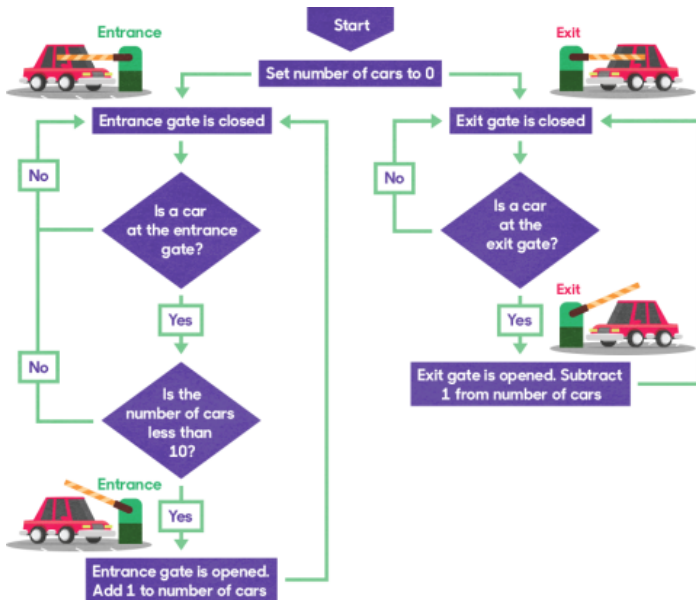
A plan, a set of step-by-step instructions to solve a problem.

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



Introduction





Bioinformatician's skill set

Biology & Medicine

- Basics in molecular and cell biology
- Measurement techniques

Computer Science

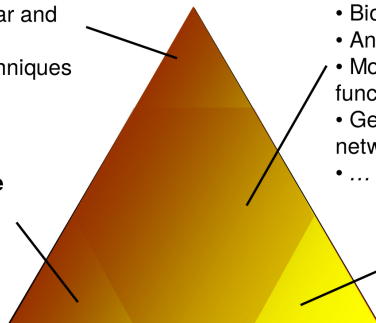
- Programming
- Databases
- Algorithmics

Bioinformatics

- Biological sequence analysis
- Biological databases
- Analysis of gene expression
- Modeling protein structure and function
- Gene, protein and metabolic networks
- ...

Mathematics and statistics

- Calculus
- Probability calculus
- Linear algebra



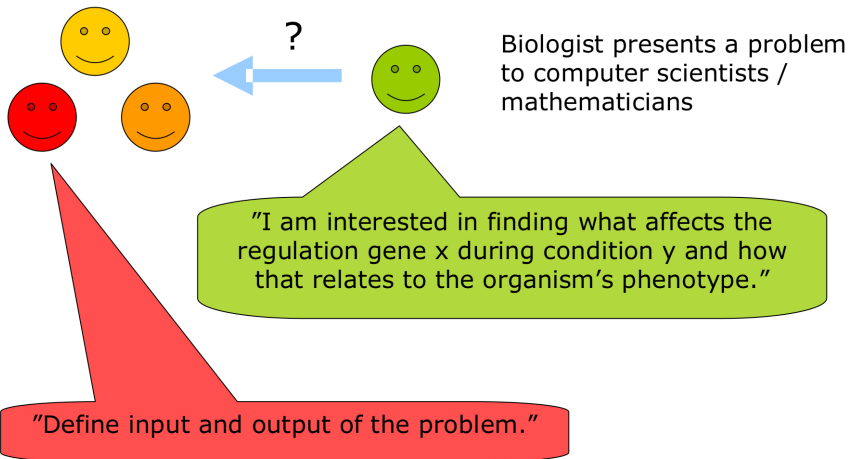
Where would you be in this triangle?

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

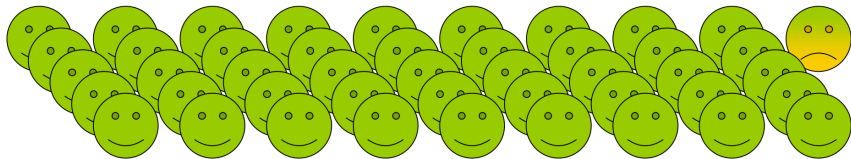
Bioinformatician's Competences

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



Prof. Esa Pitkänen, 2008

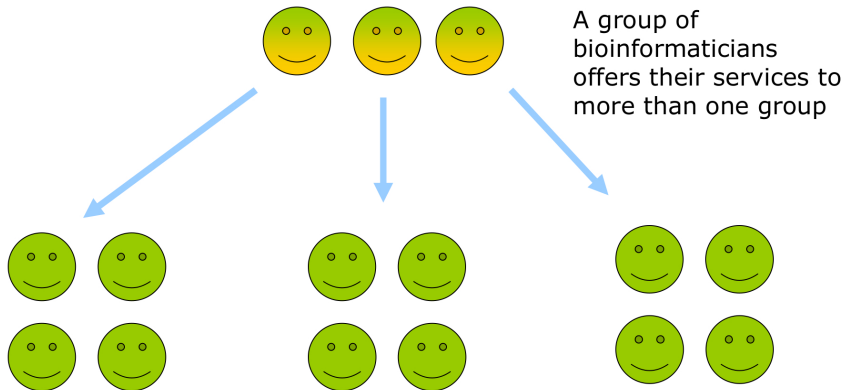
Bioinformatician's Competences



...biologist/bioinformatician ratio is important!

Prof. Esa Pitkänen, 2008

Bioinformatician's Competences



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

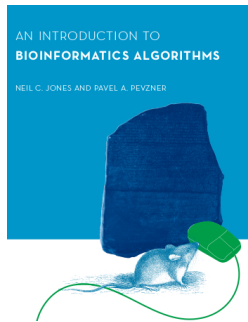
- 1 Complexity Analysis
- 2 Sorting
- 3 Exhaustive Search
- 4 Branch-and-Bound Algorithms
- 5 Greedy Algorithms
- 6 Divide-and-Conquer Algorithms
- 7 Data Mining Algorithms

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



JOHN M. ZELLE: Python Programming: An Introduction to
Computer Science (Third Edition)

Jeff Chang, Brad Chapman, Iddo Friedberg, Thomas Hamelryck,
Michiel de Hoon, Peter Cock, Tiago Antao, Eric Talevich, Bartek
Wilczyński: Biopython Tutorial and Cookbook

Other information

- Website
- Google Classroom