Algorithms

Introduction

Emanuele Rodolà
rodola@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

2nd semester a.y. 2020/2021 · March 03, 2021

Logistics

- **Lecturer:** Prof. Emanuele Rodolà
- **When:** Wednesdays 14:00–**15:30** and Thursdays 10:00–**12:15**
- **Where:** Classroom Psicologia II
- **Office Hours:** Drop me an email
  My office is in via Salaria 113 (Department of Computer Science)
- **Official website:** https://erodola.github.io/Alg-s2-2021/
  Check frequently for news and material (code, papers, ...)

Disclaimer

This course is still young! Only at its second edition.

Lectures will follow a mixed format, consisting of:

- Frontal lectures covering the more theoretical aspects (slides and/or **board**)

Disclaimer

This course is still young! Only at its second edition.

Lectures will follow a mixed format, consisting of:

- Frontal lectures covering the more theoretical aspects (slides and/or **board**)
- Exercises (pseudo-code, actual programming, other formats)

Disclaimer

This course is still young! Only at its second edition.

Lectures will follow a mixed format, consisting of:

- Frontal lectures covering the more theoretical aspects (slides and/or **board**)
- Exercises (pseudo-code, actual programming, other formats)
- Seminars and invited talks (if possible)

Disclaimer

This course is still young! Only at its second edition.

Lectures will follow a mixed format, consisting of:

- Frontal lectures covering the more theoretical aspects (slides and/or **board**)
- Exercises (pseudo-code, actual programming, other formats)
- Seminars and invited talks (if possible)
- Simulations, flipped classrooms

Disclaimer

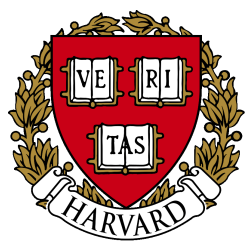
This course is still young! Only at its second edition.

Lectures will follow a mixed format, consisting of:

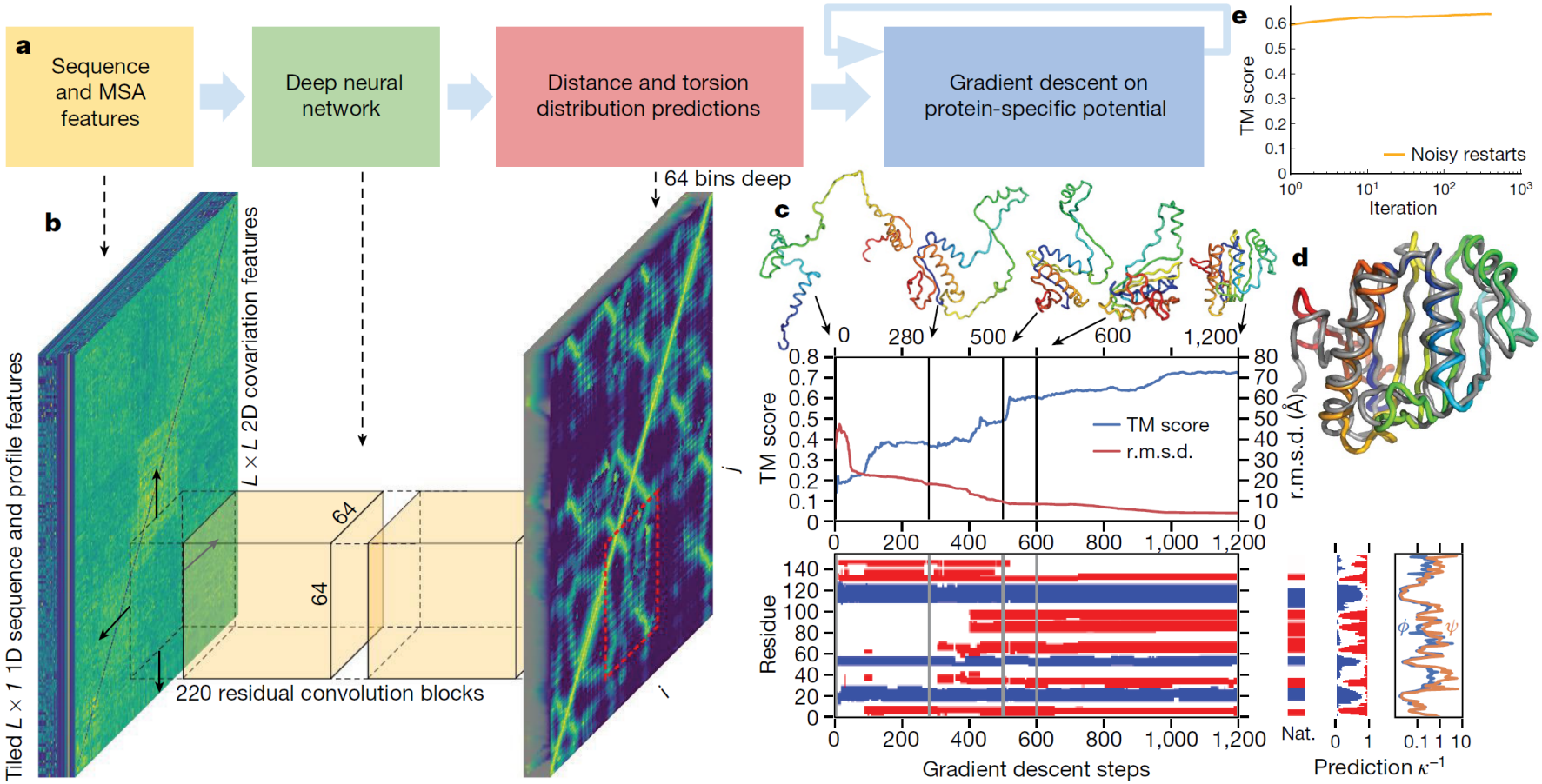
- Frontal lectures covering the more theoretical aspects (slides and/or **board**)
- Exercises (pseudo-code, actual programming, other formats)
- Seminars and invited talks (if possible)
- Simulations, flipped classrooms
- Paper reading

Who am I?

- Had research positions at U Tokyo, TU Munich, U Lugano and visiting positions at Harvard, Stanford, Ecole polytechnique, Technion among others
- Research: digital geometry processing, geometric deep learning
- Team: ∼20 members from physics, engineering, computer science
- Approach me for projects and theses!




Geometric deep learning?



https://deepmind.com/research/case-studies/alphafold

Pre-requisites and reading material

We use multiple sources. Specific references will be given throughout the course in the form of book chapters and scientific articles.

The main suggested textbook is:

"Introduction to Algorithms – 2nd Ed.", Cormen et al.

Pre-requisites and reading material

We use multiple sources. Specific references will be given throughout the course in the form of book chapters and scientific articles.

The main suggested textbook is:

"Introduction to Algorithms – 2nd Ed.", Cormen et al.

Pre-requisites (not mandatory, but welcome):

- **Programming fundamentals**. Any language is alright, although the course will mostly use Python/Matlab for examples
- **Mathematics**. Basic algebra and calculus.

Pre-requisites and reading material

We use multiple sources. Specific references will be given throughout the course in the form of book chapters and scientific articles.

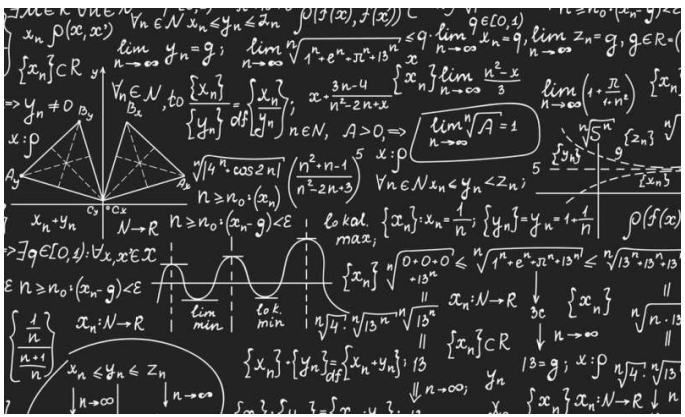
The main suggested textbook is:

"Introduction to Algorithms – 2nd Ed.", Cormen et al.

Pre-requisites (not mandatory, but welcome):

- **Programming fundamentals**. Any language is alright, although the course will mostly use Python/Matlab for examples
- **Mathematics**. Basic algebra and calculus.

Later today: self-evaluation (not graded!).



Grading

Modality: Mid-term written exam + final written exam

Grading

Modality: Mid-term written exam + final written exam

Optional: Oral examination grants ± 3 points

Grading

Modality: Mid-term written exam + final written exam

Optional: Oral examination grants ± 3 points

Everything including in-class exercises are part of the exam!

Grading

Modality: Mid-term written exam + final written exam

Optional: Oral examination grants ± 3 points

Everything including in-class exercises are part of the exam!

Tips:

- Download/print the slides beforehand

Grading

Modality: Mid-term written exam + final written exam

Optional: Oral examination grants ± 3 points

Everything including in-class exercises are part of the exam!

Tips:

- Download/print the slides beforehand
- **Take notes**

Grading

Modality: Mid-term written exam + final written exam

Optional: Oral examination grants ± 3 points

Everything including in-class exercises are part of the exam!

Tips:

- Download/print the slides beforehand
- **Take notes**
- Read the suggested material

Grading

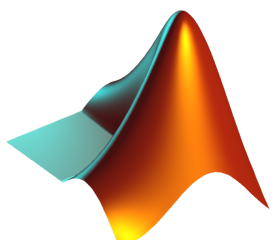
Modality: Mid-term written exam + final written exam

Optional: Oral examination grants ± 3 points

Everything including in-class exercises are part of the exam!

Tips:

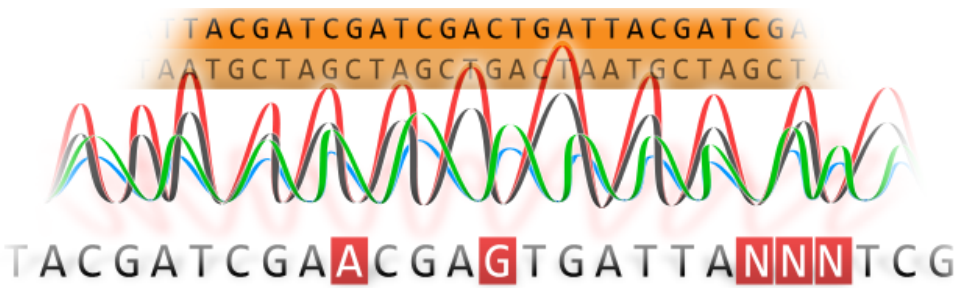
- Download/print the slides beforehand
- **Take notes**
- Read the suggested material
- Bring your laptop: we'll do live coding sessions



Overall objective

What will you get out of this course?

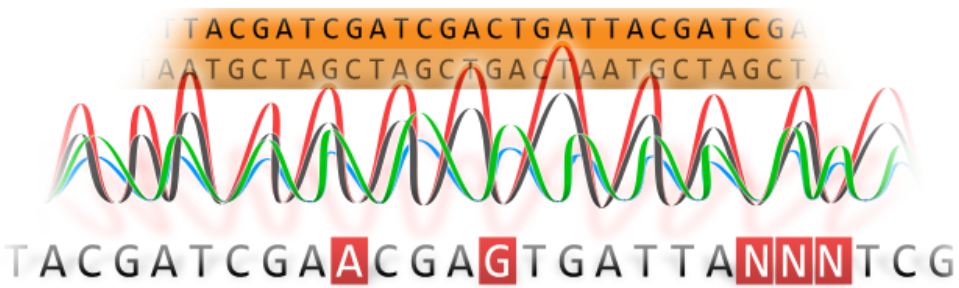
- You will acquire the principles behind the analysis and design of algorithms



Overall objective

What will you get out of this course?

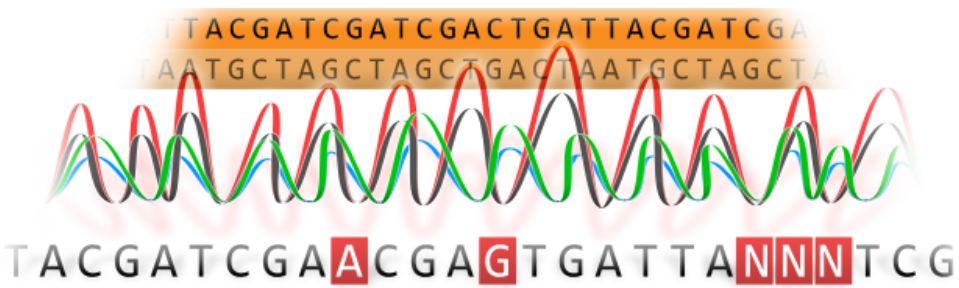
- You will acquire the principles behind the analysis and design of algorithms
- You will apply these principles to address practical problems



Overall objective

What will you get out of this course?

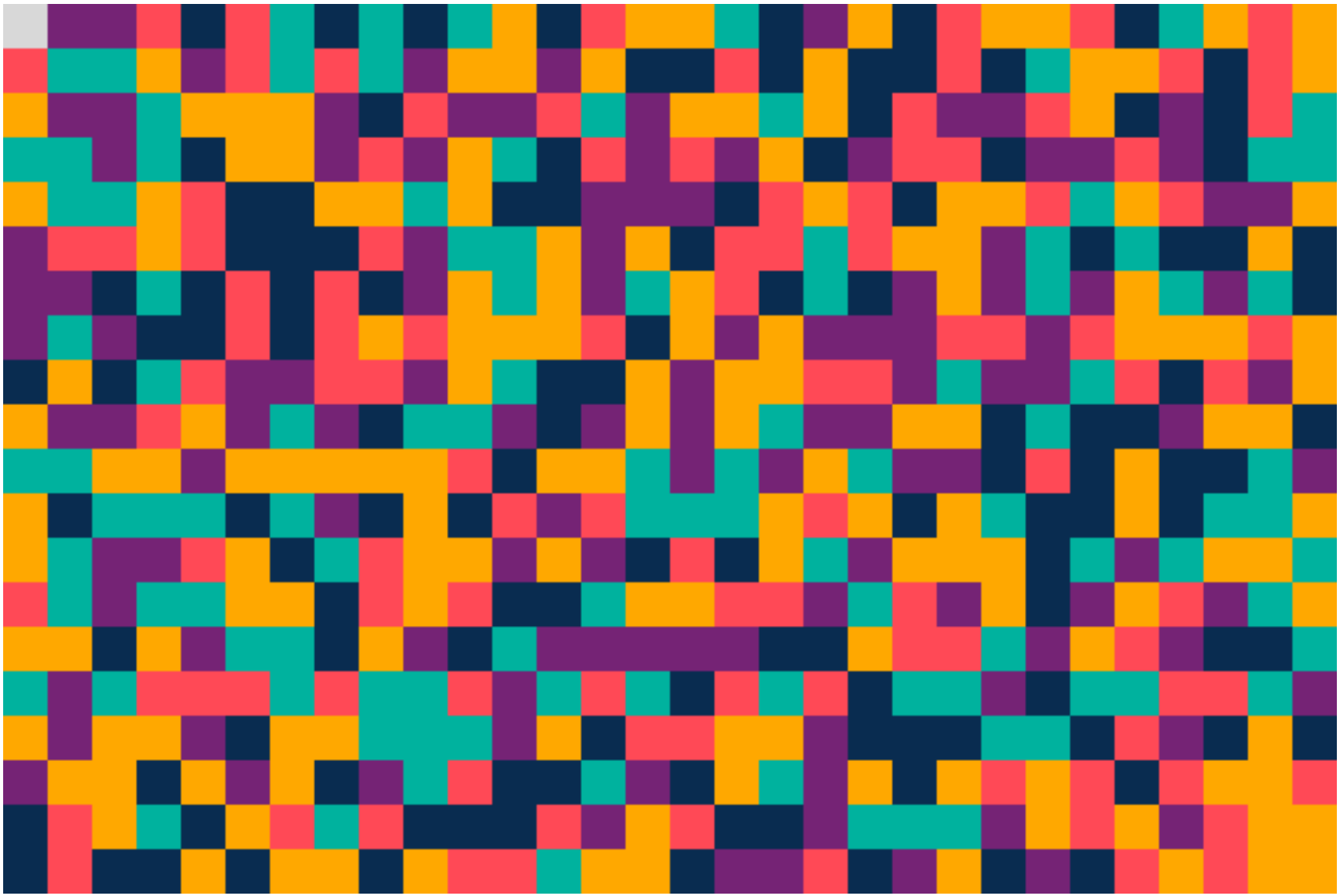
- You will acquire the principles behind the analysis and design of algorithms
- You will apply these principles to address practical problems
- You will get solid fundamental skills to understand current research



What is an algorithm?

Example

Find the biggest group of **reds**



Definition

An algorithm is a well-defined sequence of steps that produces an output given some input

Definition

An algorithm is a well-defined sequence of steps that produces an output given some input

- We use algorithms to solve computational problems

Definition

An algorithm is a well-defined sequence of steps that produces an output given some input

- We use algorithms to solve computational problems
- An algorithm might contain some degree of randomness (i.e. algorithm $\neq$ deterministic)

Definition

An algorithm is a well-defined sequence of steps that produces an output given some input

- We use algorithms to solve computational problems
- An algorithm might contain some degree of randomness (i.e. algorithm $\neq$ deterministic)
- An algorithm might fail to terminate (i.e. algorithm $\neq$ finite time, although this is debatable)

A classical example: Sorting

Input: A sequence of n numbers $(a_1, a_2, \ldots, a_n)$

Output: A reordered sequence $(a'_1, a'_2, \ldots, a'_n)$ such that:

$$a'_1 \leq a'_2 \leq \cdots \leq a'_n$$

A classical example: Sorting

Input: A sequence of n numbers $(a_1, a_2, \ldots, a_n)$

Output: A reordered sequence $(a'_1, a'_2, \ldots, a'_n)$ such that:

$$a'_1 \leq a'_2 \leq \cdots \leq a'_n$$

For example:

$$(1, 7, 3, 15, 2, 3) \Rightarrow$$

A classical example: Sorting

Input: A sequence of n numbers $(a_1, a_2, \ldots, a_n)$

Output: A reordered sequence $(a'_1, a'_2, \ldots, a'_n)$ such that:

$$a'_1 \leq a'_2 \leq \cdots \leq a'_n$$

For example:

$$(1, 7, 3, 15, 2, 3) \Rightarrow (1, 2, 3, 3, 7, 15)$$

The input sequence is an instance of the sorting problem

The sorting algorithm is the sequence of steps that produces the desired output for the given instance.

Another classical example: Shortest paths

Another classical example: Shortest paths

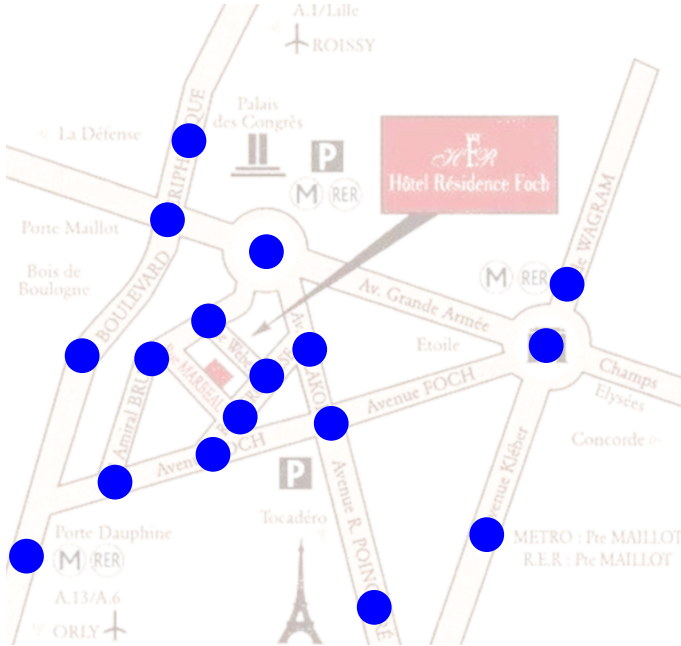
Input: A road map with city landmarks

Output: The shortest path from the airport to the Eiffel Tower

Another classical example: Shortest paths

Input: A road map with city landmarks

Output: The shortest path from the airport to the Eiffel Tower



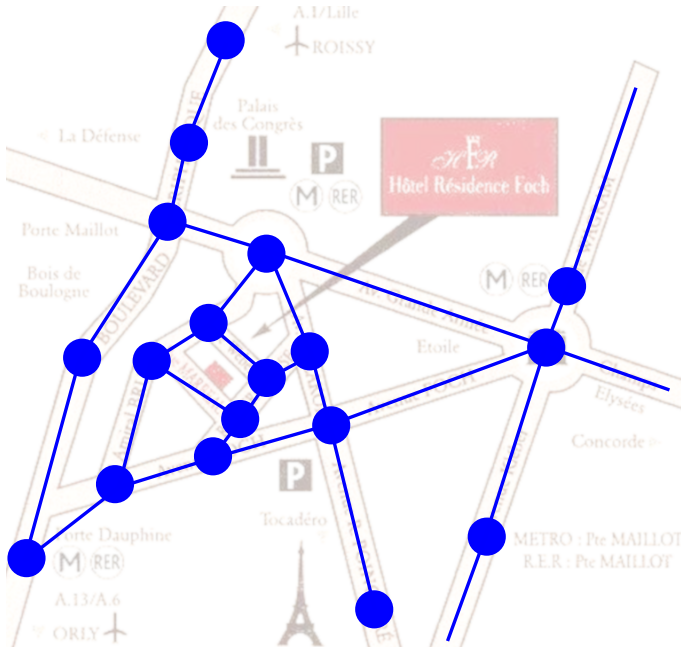
We need a way to formalize the problem.

Absraction: Remove all the unnecessary information.

Another classical example: Shortest paths

Input: A road map with city landmarks

Output: The shortest path from the airport to the Eiffel Tower



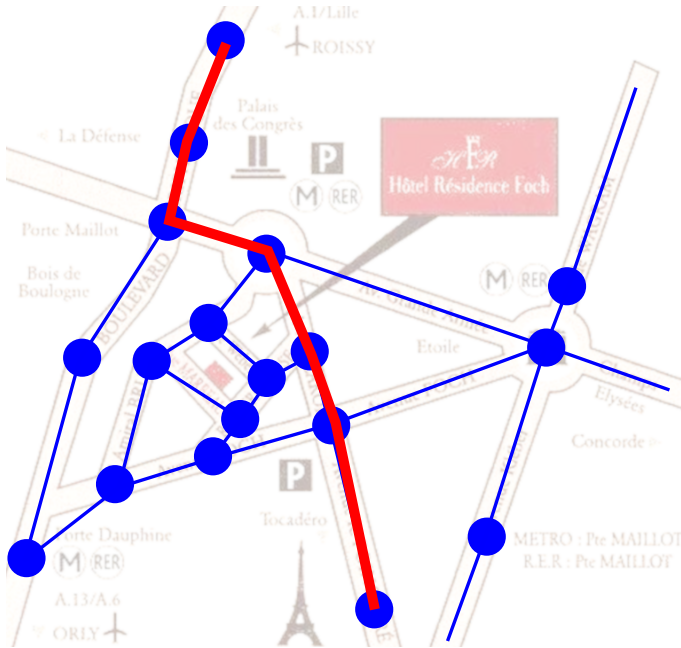
We need a way to formalize the problem.

Absraction: Remove all the unnecessary information.

Another classical example: Shortest paths

Input: A road map with city landmarks

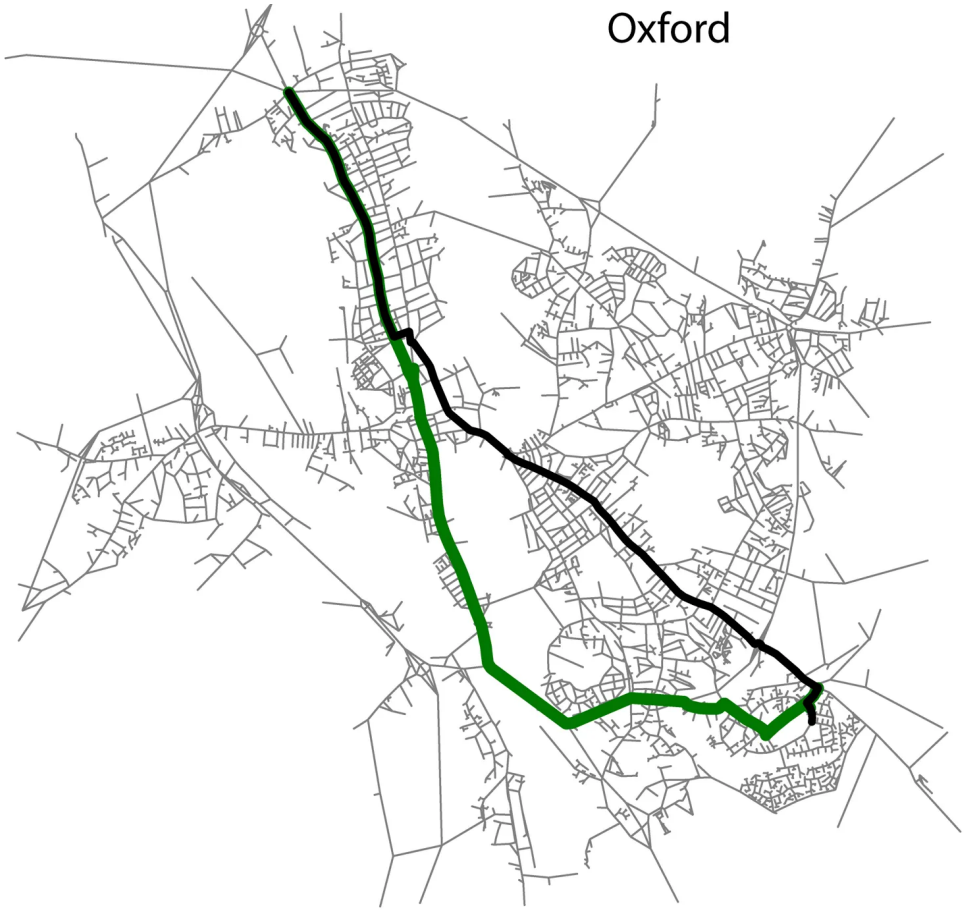
Output: The shortest path from the airport to the Eiffel Tower



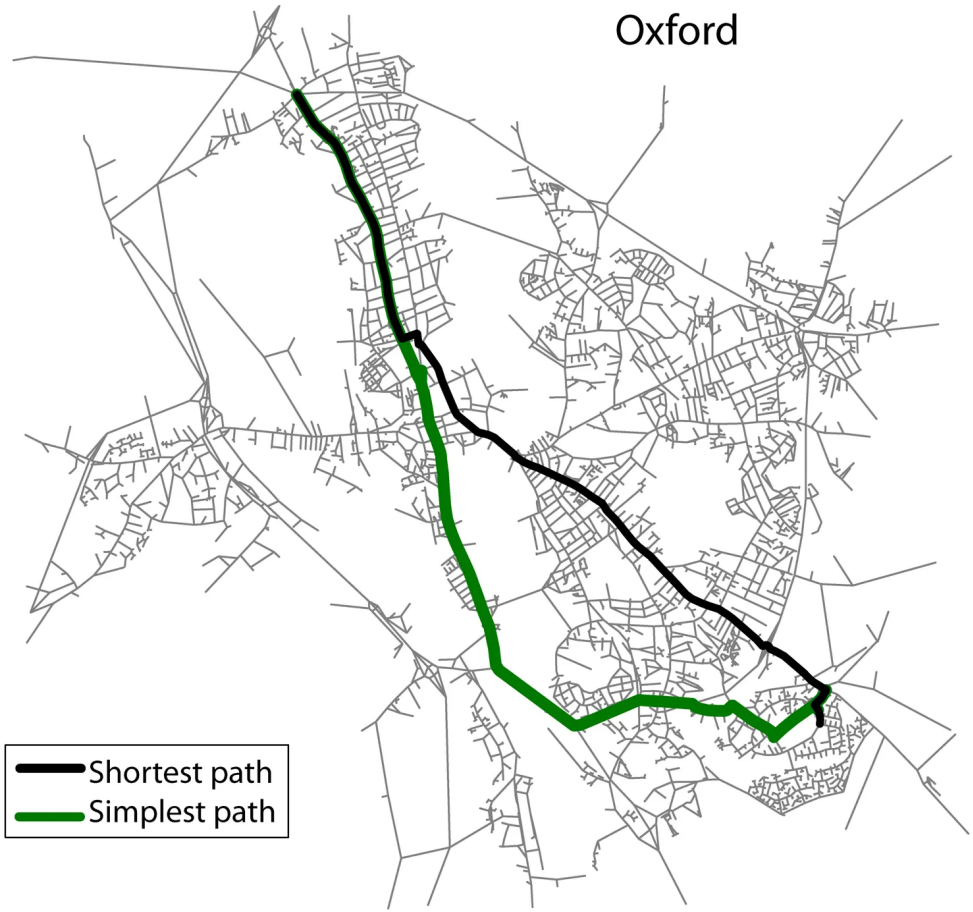
We need a way to formalize the problem.

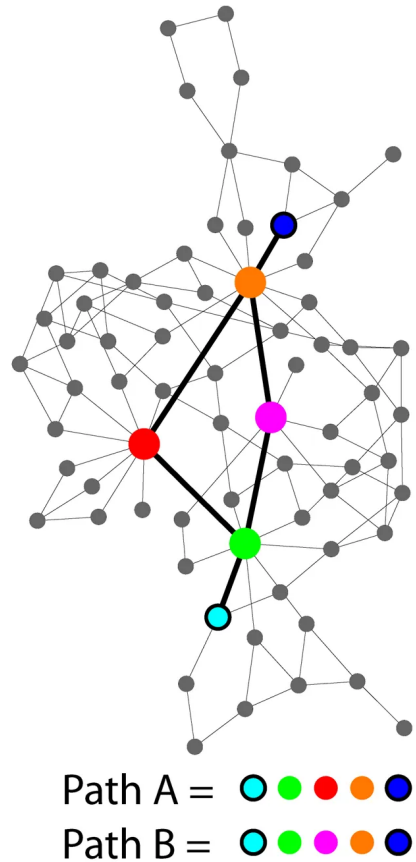
Absraction: Remove all the unnecessary information.

Another classical example: Shortest paths



Another classical example: Shortest paths





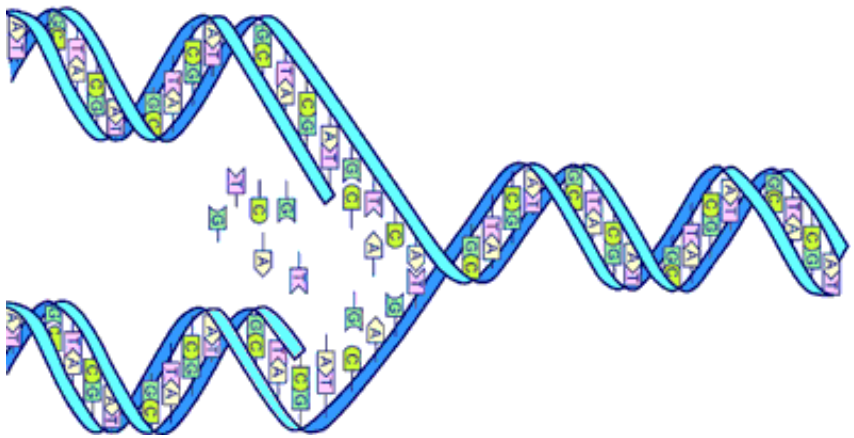
"Shortest" requires the definition of a distance.

A different distance criterion yields a different problem, and in general, a different algorithm to solve it.

The Human Genome Project

Problem: Determine the DNA sequence of the entire human genome.

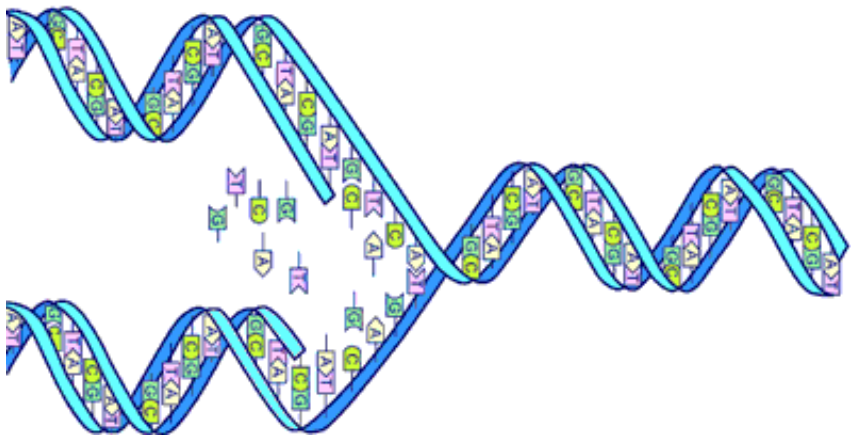
Requires identifying $100,000$ genes, determining the sequences of $3,000,000,000$ base pairs, database storing, data analysis.



The Human Genome Project

Problem: Determine the DNA sequence of the entire human genome.

Requires identifying $100,000$ genes, determining the sequences of $3,000,000,000$ base pairs, database storing, data analysis.

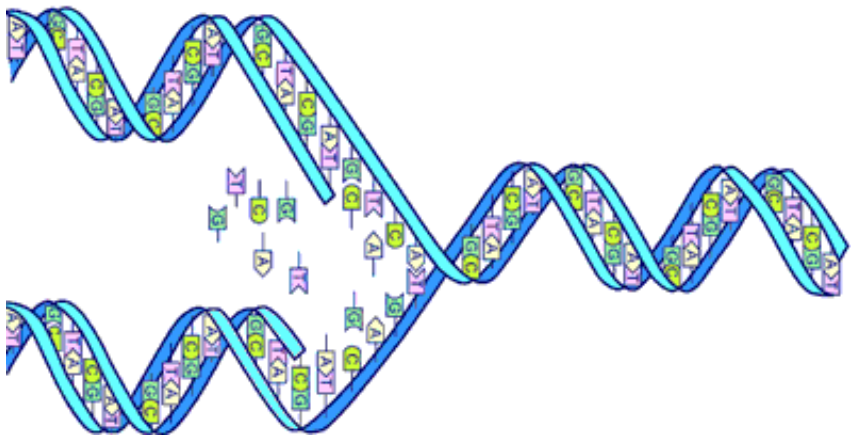


- International effort spanning > 10 years (1990 – 2003)

The Human Genome Project

Problem: Determine the DNA sequence of the entire human genome.

Requires identifying $100,000$ genes, determining the sequences of $3,000,000,000$ base pairs, database storing, data analysis.

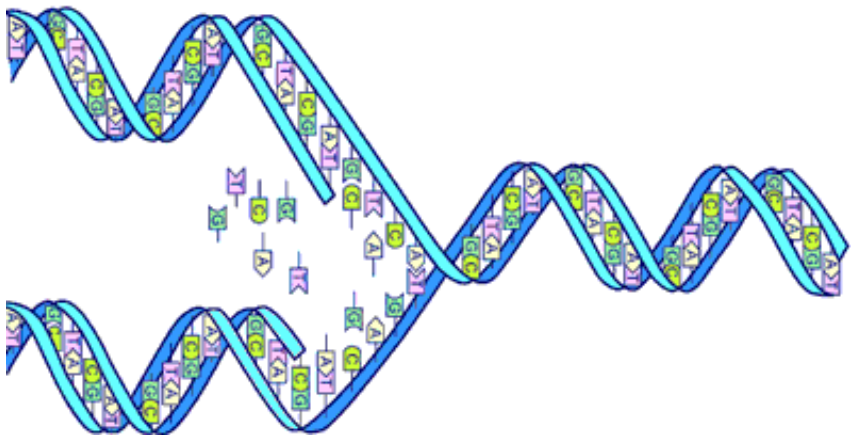


- International effort spanning > 10 years (1990 – 2003)
- Several sub-problems $\rightarrow$ a huge collection of sophisticated algorithms

The Human Genome Project

Problem: Determine the DNA sequence of the entire human genome.

Requires identifying $100,000$ genes, determining the sequences of $3,000,000,000$ base pairs, database storing, data analysis.



- International effort spanning > 10 years (1990 – 2003)
- Several sub-problems $\rightarrow$ a huge collection of sophisticated algorithms
- Efficient algorithms lead to savings in time and money

  But not all the problems admit an efficient solution!

Data structures

These examples suggest that dealing with algorithms requires the study of efficient data structures to store, organize, and facilitate data access.

Data structures

These examples suggest that dealing with algorithms requires the study of efficient data structures to store, organize, and facilitate data access.

Example: In a stack, the last element we insert is also the first we take off (last-in, first-out policy)

Data structures

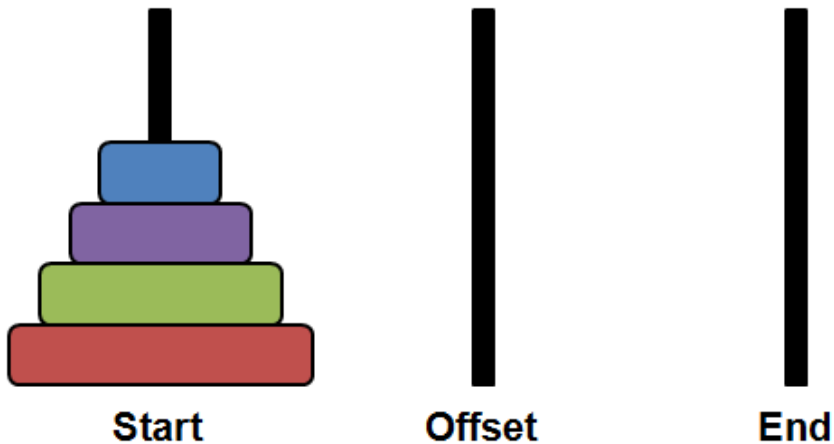
These examples suggest that dealing with algorithms requires the study of efficient data structures to store, organize, and facilitate data access.

Example: In a stack, the last element we insert is also the first we take off (last-in, first-out policy)

Data structures

These examples suggest that dealing with algorithms requires the study of efficient data structures to store, organize, and facilitate data access.

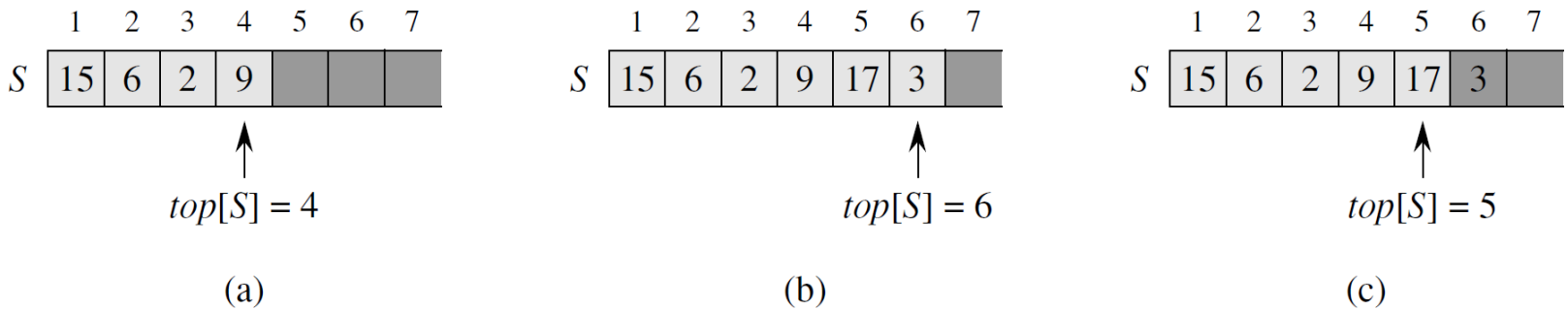
Example: In a stack, the last element we insert is also the first we take off (last-in, first-out policy)



Data structures

These examples suggest that dealing with algorithms requires the study of efficient data structures to store, organize, and facilitate data access.

Example: In a stack, the last element we insert is also the first we take off (last-in, first-out policy)



Insert and remove operations are defined to be efficient at the top.

Exercises

Let us find an answer to some simple questions:

- Find a real-world example where sorting appears

Exercises

Let us find an answer to some simple questions:

- Find a real-world example where sorting appears
- What is a limitation of the stack as a data structure?

Exercises

Let us find an answer to some simple questions:

- Find a real-world example where sorting appears
- What is a limitation of the stack as a data structure?
- Can a problem (e.g. shortest paths) have multiple solutions?

Exercises

Let us find an answer to some simple questions:

- Find a real-world example where sorting appears
- What is a limitation of the stack as a data structure?
- Can a problem (e.g. shortest paths) have multiple solutions?
- Assume that a problem, whose input has size n, can be solved by two different algorithms.

  Algorithm 1 takes $3n^2$ steps to solve the problem.

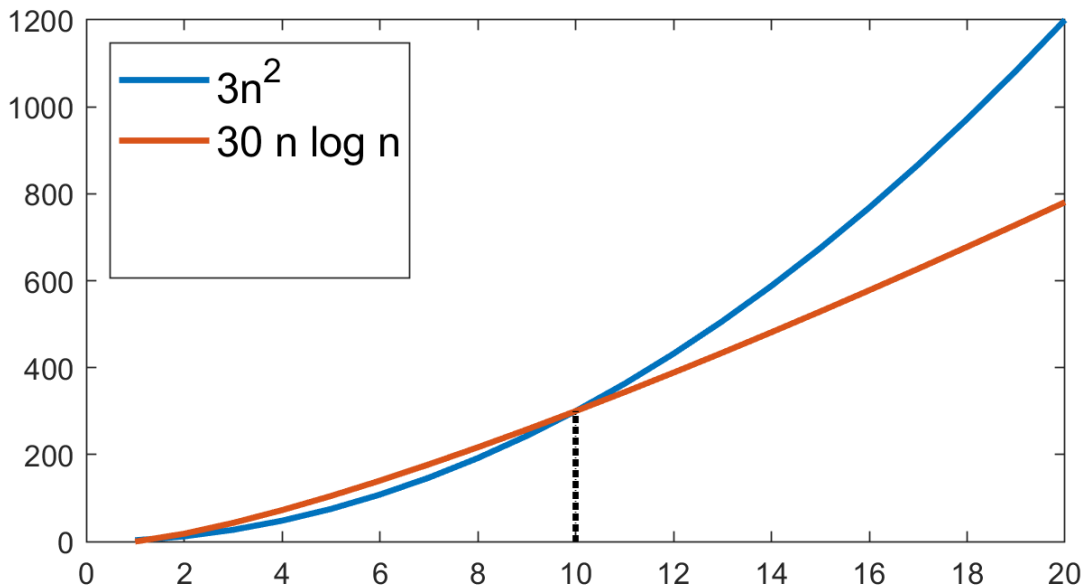
  Algorithm 2 takes $30n \log n$ steps.

  Which one would you choose?

Exercises

Let us find an answer to some simple questions:

- Find a real-world example where sorting appears
- What is a limitation of the stack as a data structure?
- Can a problem (e.g. shortest paths) have multiple solutions?
- Assume that a problem, whose input has size n, can be solved by two different algorithms.



Exercises

In pseudocode or in your favorite language, write an algorithm to solve each of the following problems.

1. Reverse any given sequence of length n:

$$(3, 7, 9, 14) \to (14, 9, 7, 3)$$

2. Given a number x and a sequence $(a_i)_{i=1}^n$, find the closest number to x in the sequence.

$$12.1\,,\ (3, 31, 7, 11, 52) \to 11$$

3. Given a sequence $(a_i)_{i=1}^n$ and a smaller sequence $(b_i)_{i=1}^m, m < n$, find the latter inside the former, and return the index of the first occurrence as output.

$$(C, G, A, T, T, G, C, \dots)\,,\ (T, T, G \dots) \to 4$$