15-110 Principles of Computing – S21

Lecture 1:
Introduction and Logistics

Teacher:
Gianni A. Di Caro

جامعة كارنيجي ميلون في قطر
Carnegie Mellon University Qatar

The 110 team

Teacher:

Prof. Gianni A. Di Caro
(robotics, AI, ML, MAS)

Course Assistants:

- Samar Rahmouni
- Yusuf Saquib
- Jawaria Abbas
- Demetre Devidze
- Abdalla Mohamed

Road map

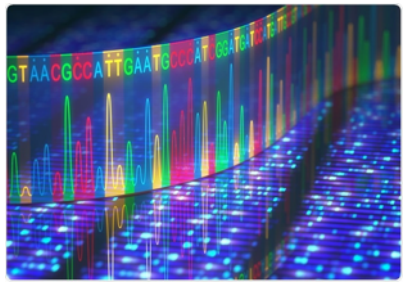
- General overview: what this course is about
- Utility of the course
- Logistics: classes, labs, homework, exams, grading, website, books, piazza, rules
- Software for python programming
- More about the need to use efficient algorithms + computers

Road map

- General overview: what this course is about
- Utility of the course
- Logistics: classes, labs, homework, exams, grading, website, books, piazza, rules
- Software for python programming
- More about the need to use efficient algorithms + computers

Introduction to the *science (art)* of computational problem solving

Daily life, professional activities, leisure, business, … present a variety of ***problems*** that ask for finding a **solution**



- Different constraints (e.g., time, budget)
- Different representations / models / contexts
- Different objectives (e.g., best nutritional balance, minimum traveling time, win!)
- Different requirements for the solution (e.g., provably the best, just one)
- Different available knowledge (e.g., chess vs. financial investment vs. path finding)
- Different degree of (un)certainty (e.g., poker vs. industrial manipulator)
- Different dimensionality / number of variables (e.g., DNA sequencing, root finding, flight control)
- …

Introduction to the *science (art)* of computational problem solving

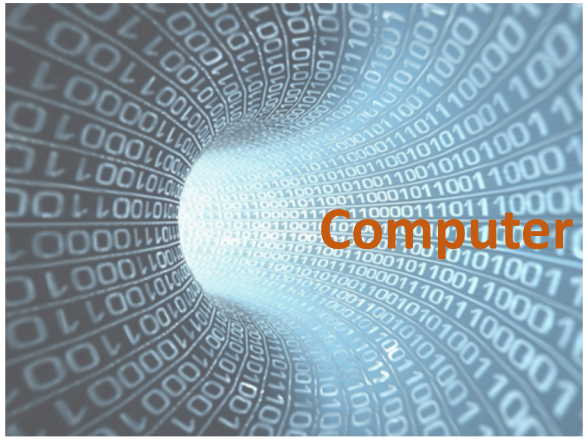
Often (usually) these problems are too complex to be **effectively** and **satisfactorily** solved by humans (alone)…

… ***computers*** can effectively help (us) solving the difficult problems!

- Perform **calculations**, at impressively high rates
  - $+ \; - \; * \; / \; > \; < \; = \; \neq$
  - Fastest computer in the world (IBM AC922): ≈ 10^{15} operations per sec on real numbers: 1,000,000,000,000,000 ops/sec
  - Mega = 10^6, Giga = 10^9, Tera = 10^{12}, Peta = 10^{15}, Exa = 10^{18}
- **Store** and **retrieve** data and results of calculations
  - 160TB internal memory, HP: 160,000,000,000,000 bytes
  - Single storage units > 100 TB: 100,000,000,000,000 bytes

… how do we connect *problem + humans + computers* for effective problem-solving?

Introduction to the *science (art)* of computational problem solving

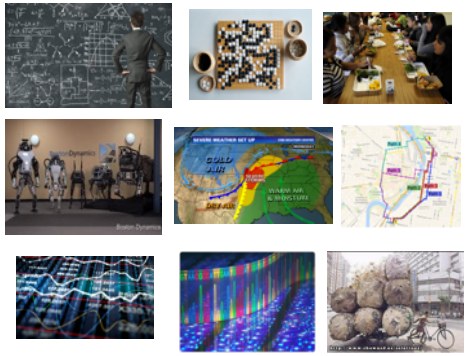


Computer Hardware

Computer programming

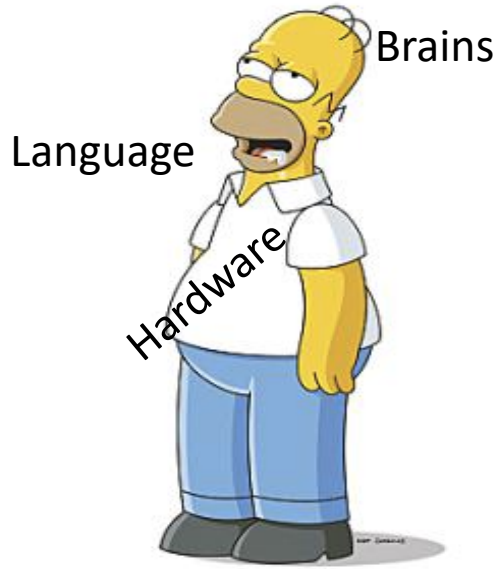
Algorithm

Abstraction
Model formulation

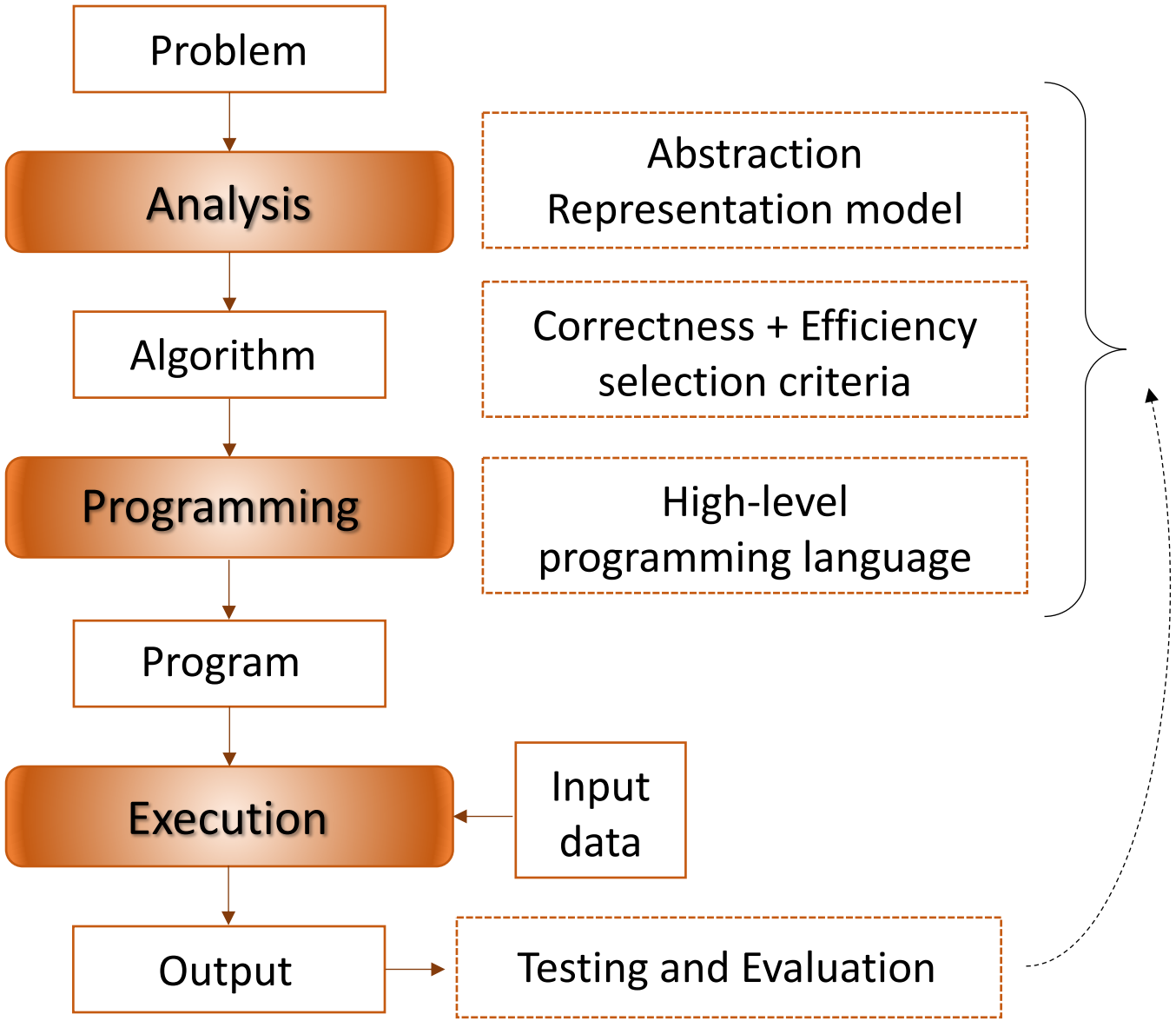


Correctness
Efficiency

- ✓ **Algorithm:** What to do (*step by step*) to (efficiently) solve the problem based on the defined problem abstraction / model → Problem-solving recipe
- ✓ **Programming (coding):** Use a high-level (computer) *language* to precisely describe (code) the algorithm → The code is used to tell the computer to do the things prescribed by the algorithm [what to do and how]
- ✓ **Computer hardware**: The high-level program is translated into a *machine-level* language which is then executed by the specific computer hardware



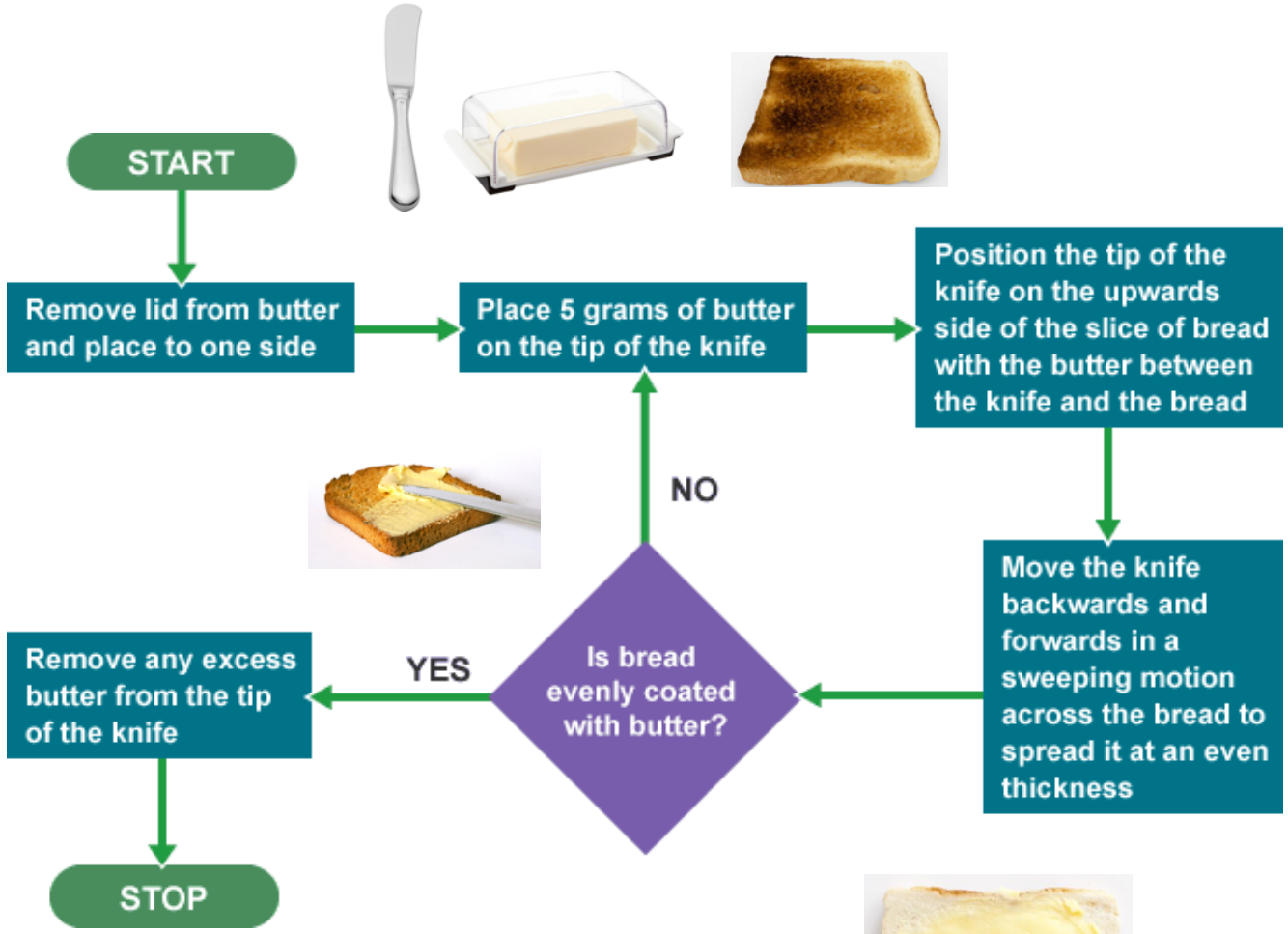
Introduction to the *science (art)* of computational problem solving



- **Algorithm:**
  - ✓ A finite list of instructions that describe a **computation**
  - ✓ when the instructions are executed on a provided set of inputs, the computation proceeds *step by step* through a set of well-defined states (configurations)
  - ✓ eventually, it ends, with some outputs being produced
- **Program:**
  - ✓ Algorithm encoding using a language that the computer understands
  - ✓ > *700 programming languages!*
  - ✓ Primitive constructs, syntax, static semantics, semantics

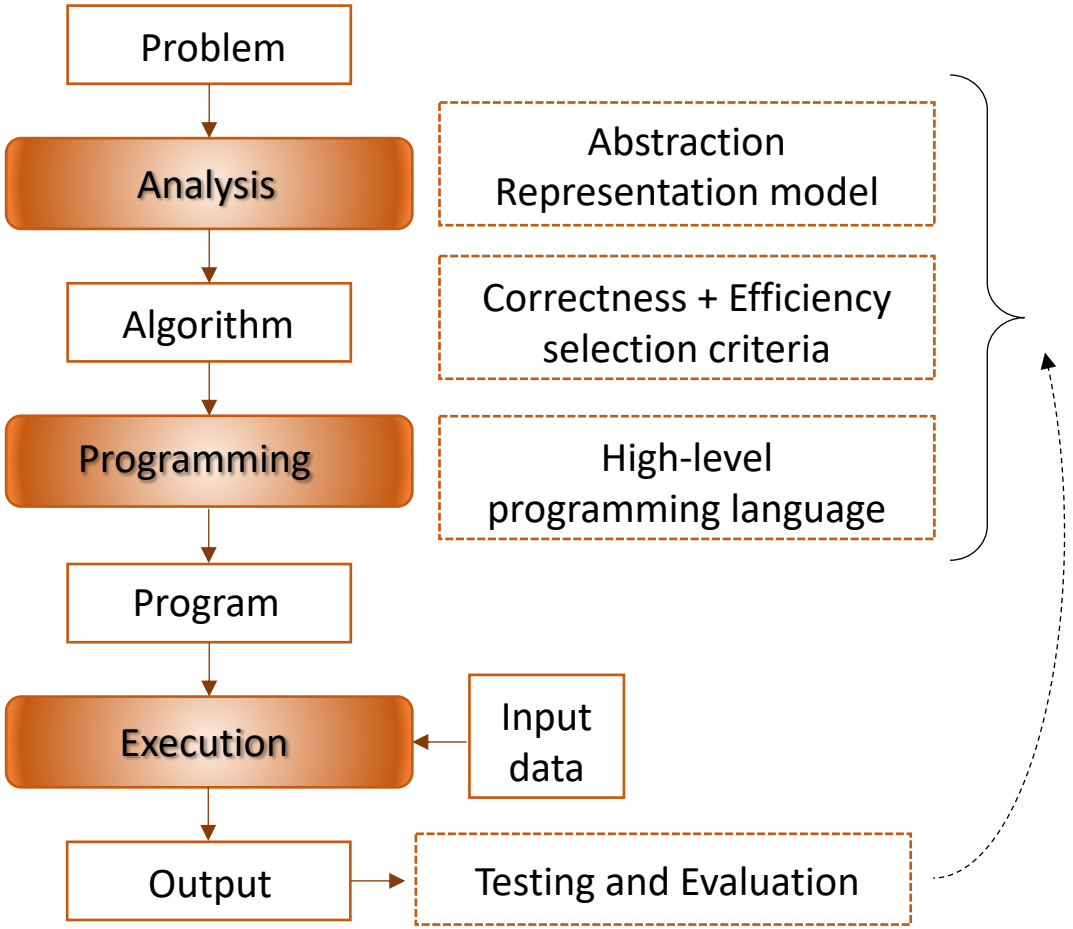
Algorithms

- ✓ A finite list of instructions that describe a **computation**
- ✓ when the instructions are executed on a provided set of inputs, the computation proceeds step by step through a set of well-defined states (configurations)
- ✓ eventually, it ends, with some outputs being produced



- Many ways to **describe** the same algorithm (including natural language)
- To be implemented, it needs to be described in an **unambiguous** form
- Can be applied to a **class of problems**

The 15-110 journey …



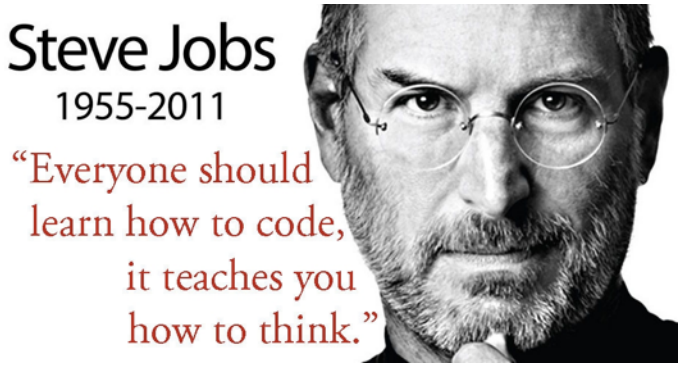
- ✓ **Problem analysis**: abstraction and representation
- ✓ Concepts central to design and understand **computation** (i.e., *algorithms*)
- ✓ Fundamental concepts of **programming**:
  - *Knowledge representation* (data types & structures)
  - *Flow control* (conditionals, iteration, recursion)
  - *Data organization* (lists, matrices, dictionaries)
  - *Decomposition, reusability, information hiding* (modules, functions, objects)
- ✓ ***Python* programming language**
- ✓ Computational **problem-solving techniques**
- ✓ **Correctness** and **efficiency** of algorithms (testing, error finding, complexity, optimization)
- ✓ **Data visualization** and **data publishing**
- ✓ **Put all together into a data science project!**

Road map

- General overview: what this course is about
- Utility of the course
- Logistics: classes, labs, homework, exams, grading, website, books, piazza, rules
- Software for python programming
- More about the need to use efficient algorithms + computers

Motivations for following 15-110 …

- Understanding computation and learning computer programming helps shaping your mind, develops your critical thinking: it's like *math* but more interactive and more fun!



- ✓ Rational thinking for (general) problem analysis and solving
- ✓ Develop abstraction and formalization capabilities
- ✓ Devising an efficient solution to a difficult problem is always rewarding
- ✓ Finding errors in the code is as thrilling as as finding the assassin! ☺

- Why understanding computation and learning computer programming can be rewarding?

- Financial analysts: **$84,300**
- Market research analysts: **$63,230**
- Sales managers: **$121,060**
- Human resource specialists: **$60,350**
- Accountants and auditors: **$69,350**
- Management analysts: **$82,450**
- Personal financial advisors: **$90,640**

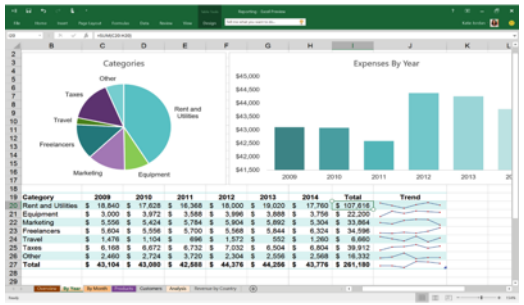
- **Software Engineer:** Average Salary $76,194
- **Senior Software Engineer:** Average Salary $89,599
- **Software Developer:** Average Salary $62,803
- **Senior Software Developer or Programmer:** Average Salary $100,303
- **Web Developer:** Average Salary $62,500
- **Programmer Analyst:** Average Salary $55,250

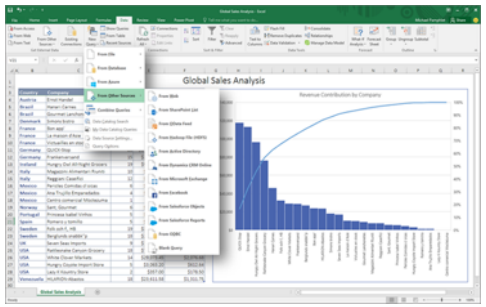
Salary range for a **Biologist**

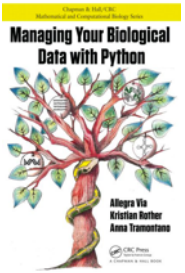
$44,946 to $56,017

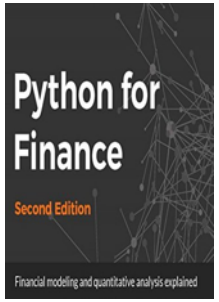
Motivations for following 15-110 …

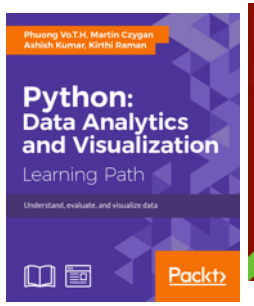
- Why understanding computation and learning computer programming can result in a professional advantage also for non-CS professionals?

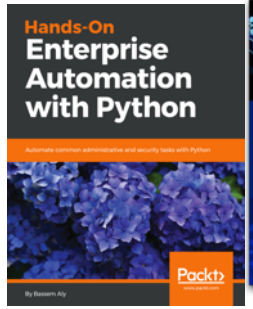


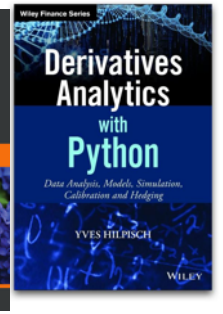


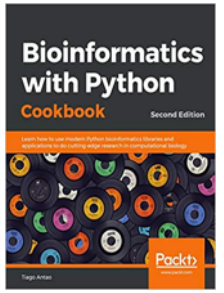


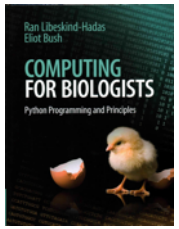


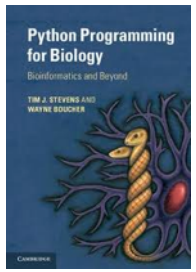


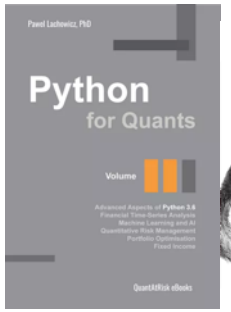


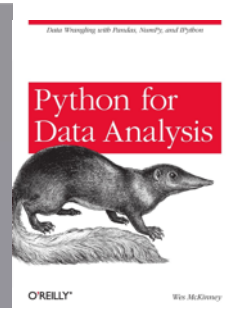


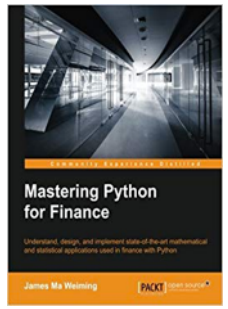


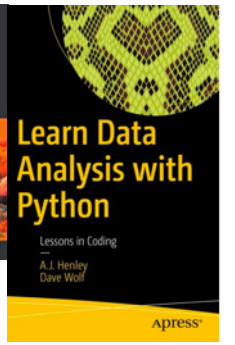












Road map

- General overview: what this course is about
- Utility of the course
- Logistics: classes, labs, homework, exams, grading, website, books, piazza, rules
- Software for python programming
- More about the need to use efficient algorithms + computers

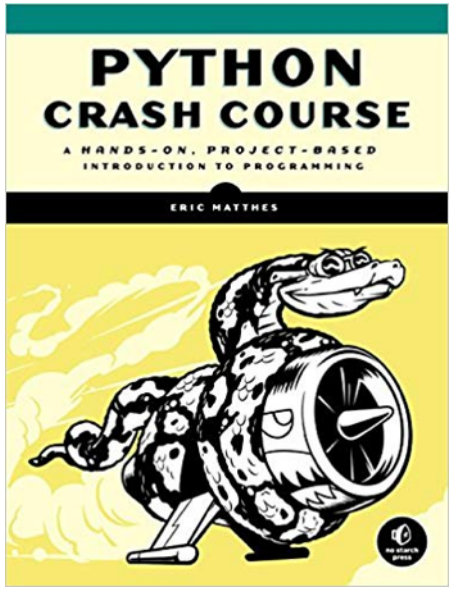
Logistics: Classes and Grading

- **Theory Classes**: Sundays, Tuesdays (1:30pm – 2:50pm), ONLINE
- **Labs, hands-on, Quizzes**: Thursdays (1:30pm – 2:50pm), room 1064, In-presence
- **Homework:** weekly, programming questions (autograded) + theory quizzes
- **Quizzes**: weekly, during lab hours
- **Midterms:** three tests during the course
- **Final exam:** one general test at the end of course
- **Project:** programming project issued at week 10 to be completed by the end of the course (deliver: code/notebook + oral presentation to teacher)

- **Homeworks: 15%**
  - Weekly, except on exam weeks
  - To be solved individually (check the course policy
  - Check the schedule for dates and deadlines
- **Quizzes: 25%**
  - Weekly
  - Administered at the beginning of the lecture or lab
  - Individual and closed book
- **Three midterms: 30%** (10% each)
  - Administered during normal class hours
  - Individual and closed book
- **Project: 10%**
- **Final exam: 20%**

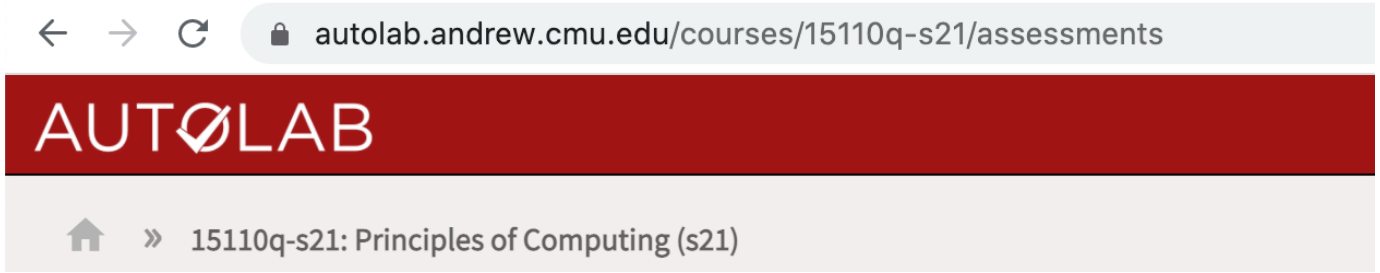
Logistics: Resources

- **Main sources for material:**
  - Lectures + lecture slides + technical notes
  - Python reference book on Canvas
  - Course website: https://cs-cmuq.github.io/110-www/



- **Piazza** for: Questions, announcements, discussions

- **Autolab** for submitting homework and project, and getting grades



Logistics: In-class rules

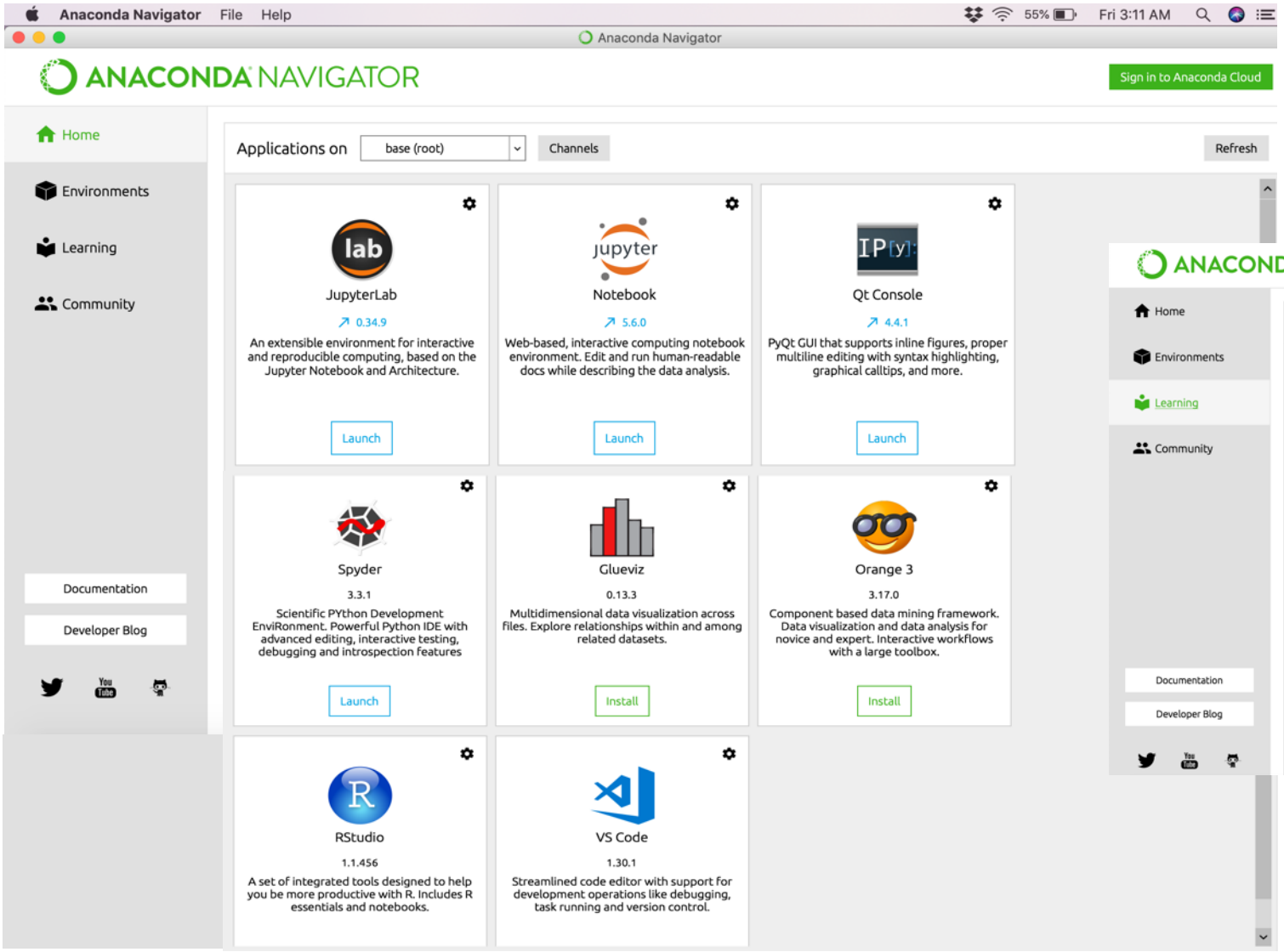
Be *alive* and **participate**!

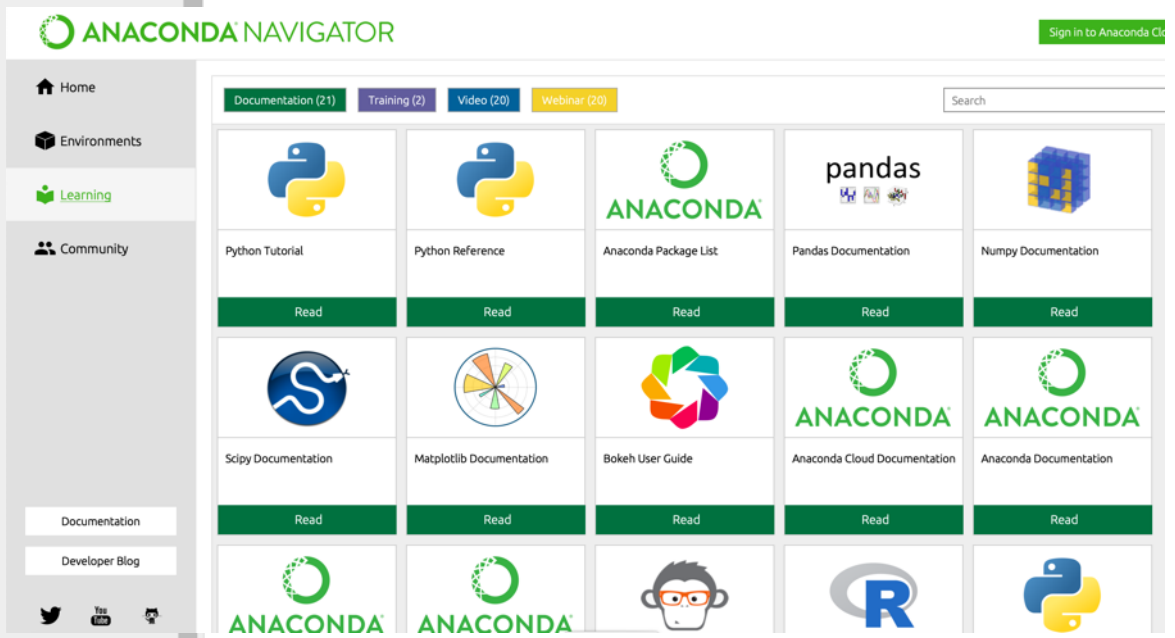
Ask **questions!**

Road map

- General overview: what this course is about
- Utility of the course
- Logistics: classes, labs, homework, exams, grading, website, books, piazza, rules
- Software for python programming
- More about the need to use efficient algorithms + computers

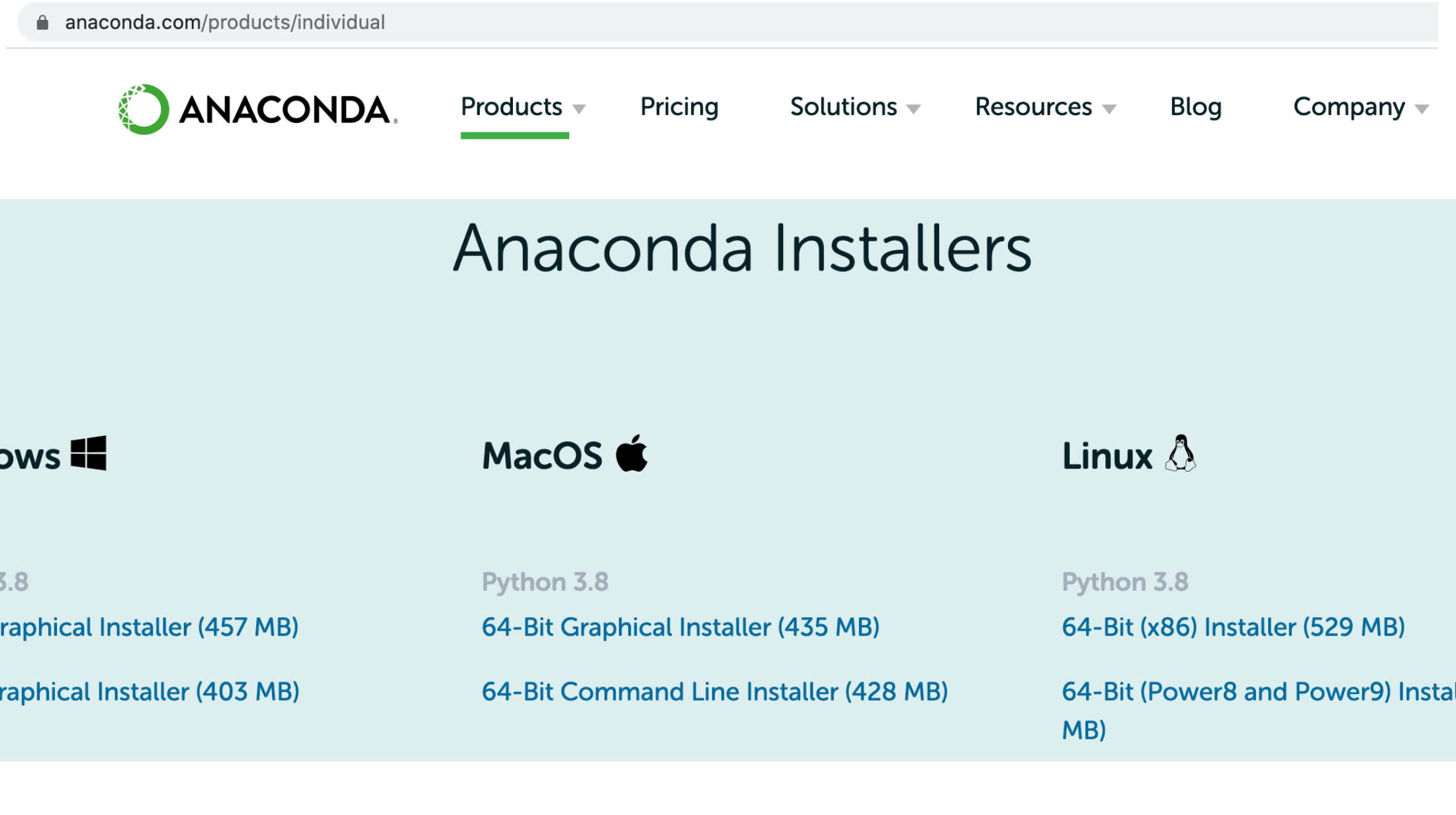
ANACONDA Navigator



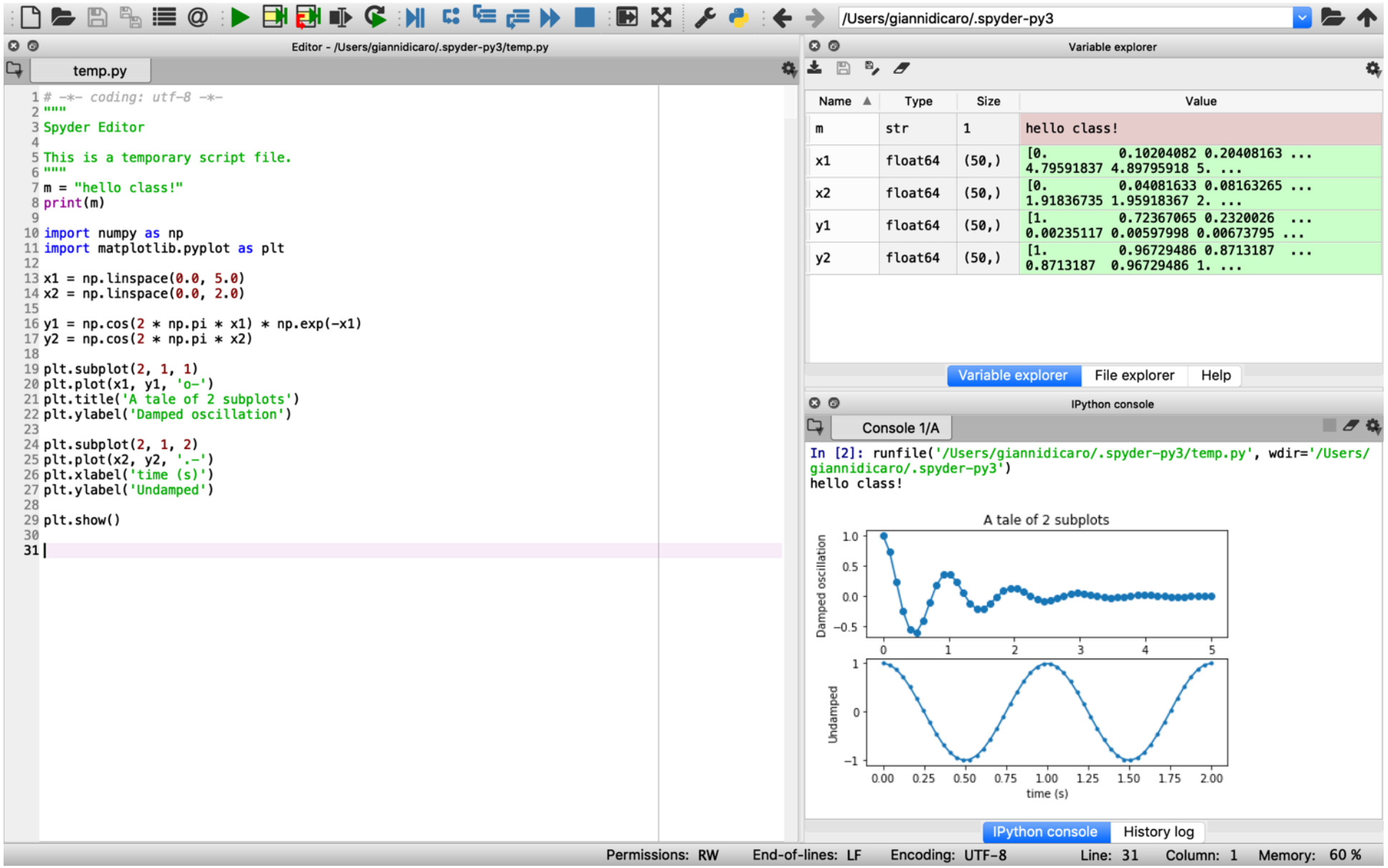


https://www.anaconda.com/

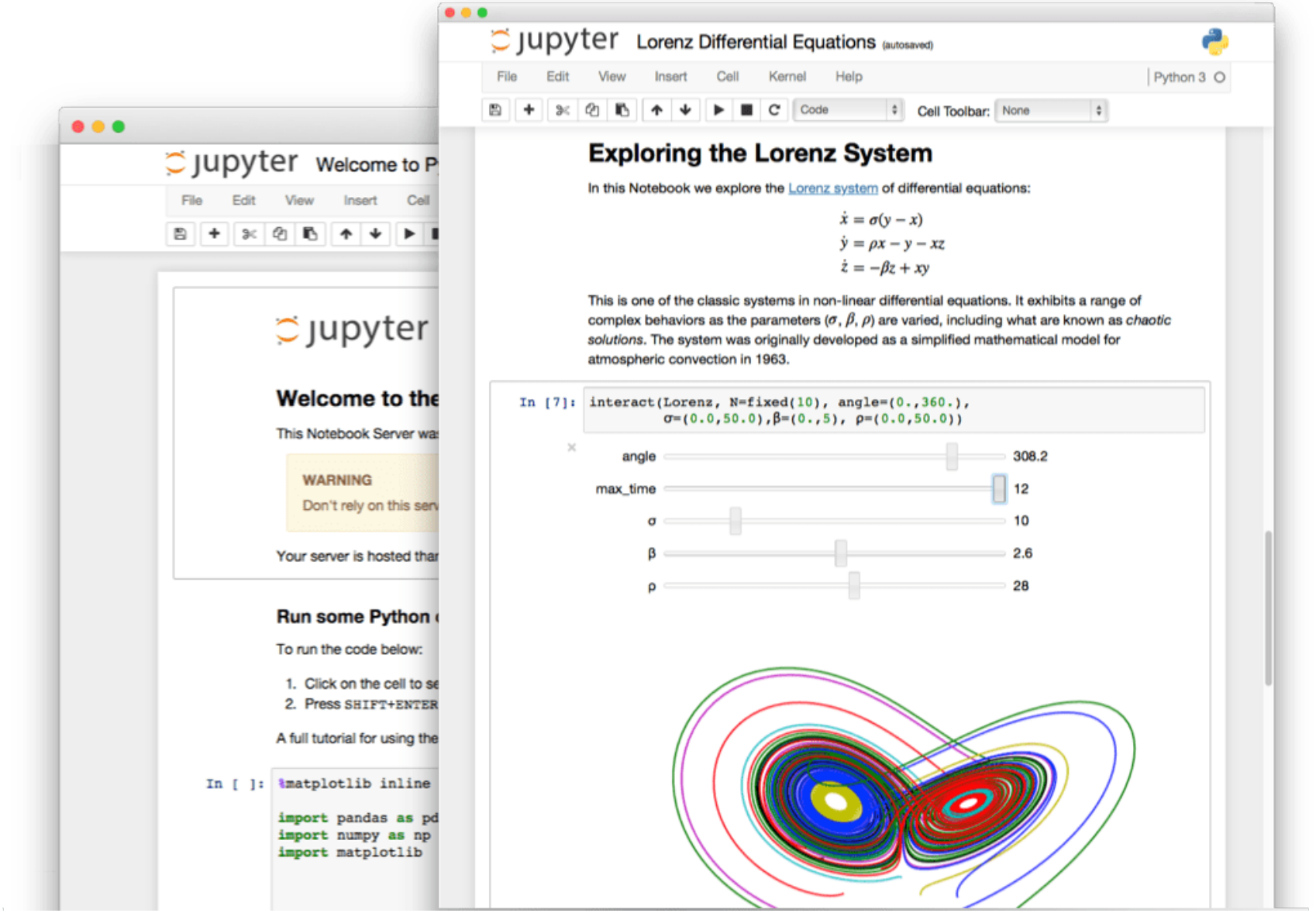
ANACONDA: Install



Spyder: Python Integrated Development Environment (IDE)



Jupyter Notebooks



Road map

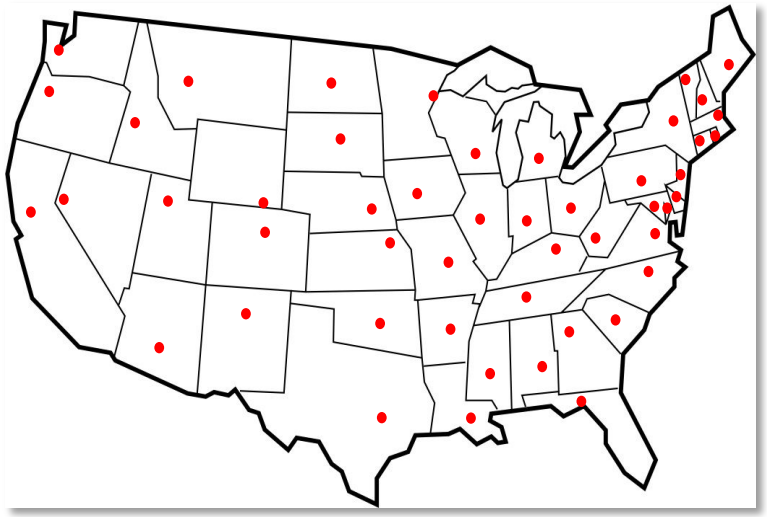
- General overview: what this course is about
- Utility of the course
- Logistics: classes, labs, homework, exams, grading, website, books, piazza, rules
- Software for python programming
- More about the need to use efficient algorithms + computers: a practical example

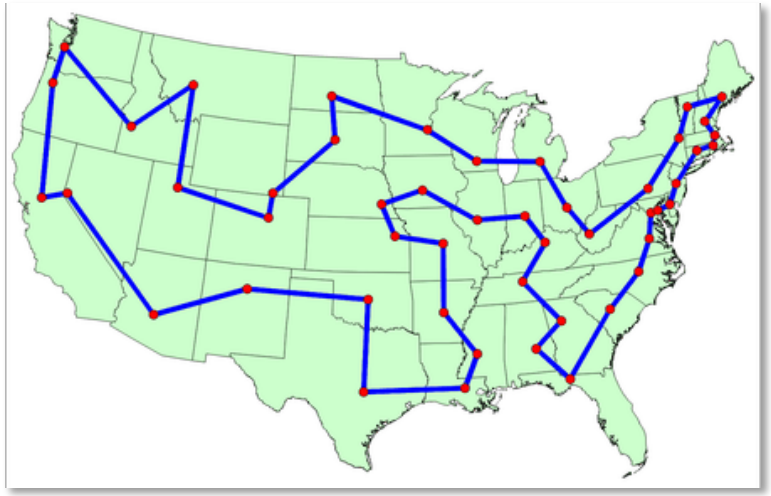
Are the (usual) problems really so difficult?

- Do we need efficient algorithms?
  - Do we really need computers?
    - Can we get the flavor of how difficult solving typical problems can be?
      - The following 10 slides illustrate these aspects considering one of the most important and well-studied problems in computer science, the **Traveling Salesman Problem** (TSP)

The (very famous) Traveling Salesman Problem (TSP)

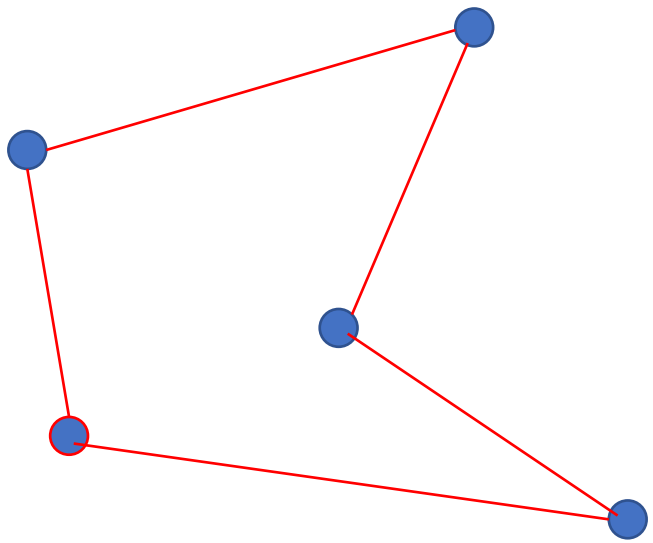
- ✓ Given a set of n sites (e.g., cities) along with the cost of travel between each pair of them, the TSP is to find the cheapest way of visiting all the n sites once and only once, and returning to the starting point (**Closed tour**)
- ✓ Common cost measure is distance: Tour of shortest length

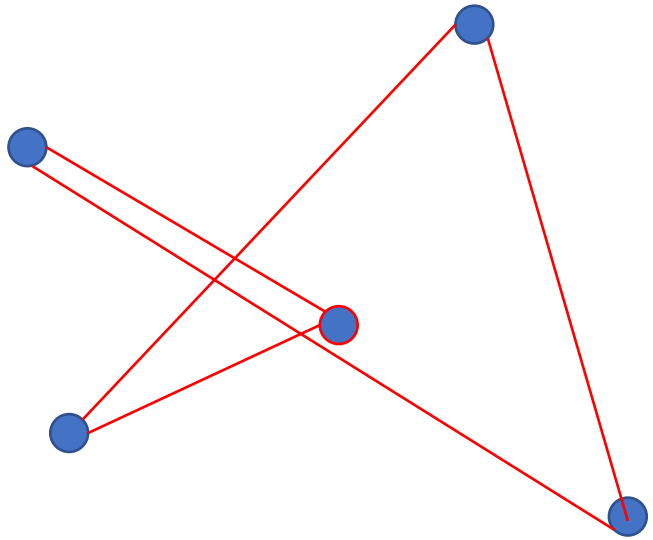


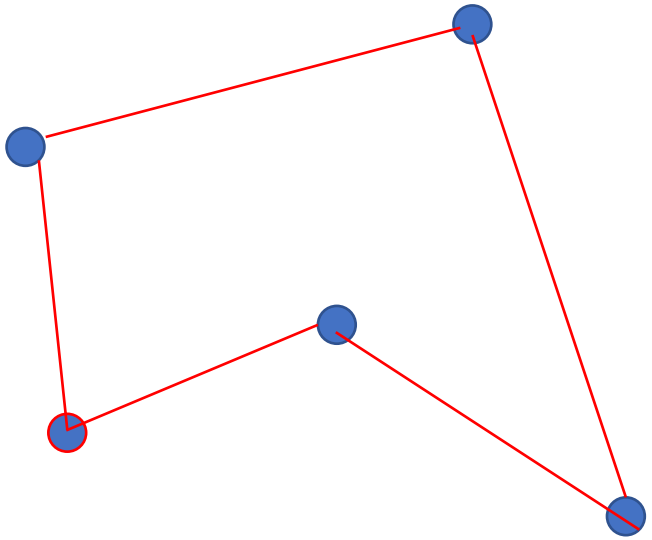


- ❖ The TSP is around since the 18th century, since it pertains to the practical optimization of the traveling paths of merchants, traders, postmen, ...

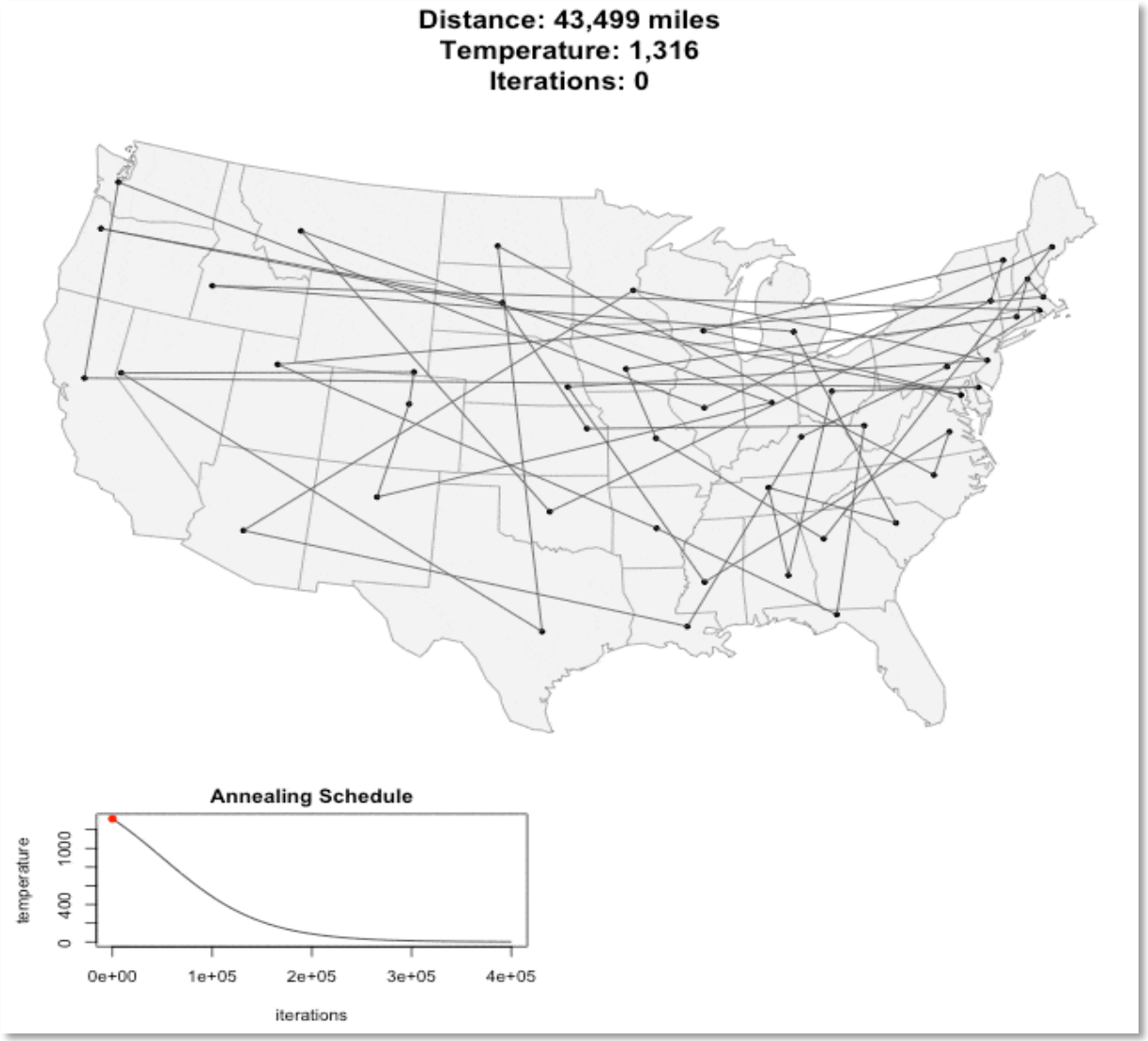
TSP: 5-sites, example of solutions with different costs

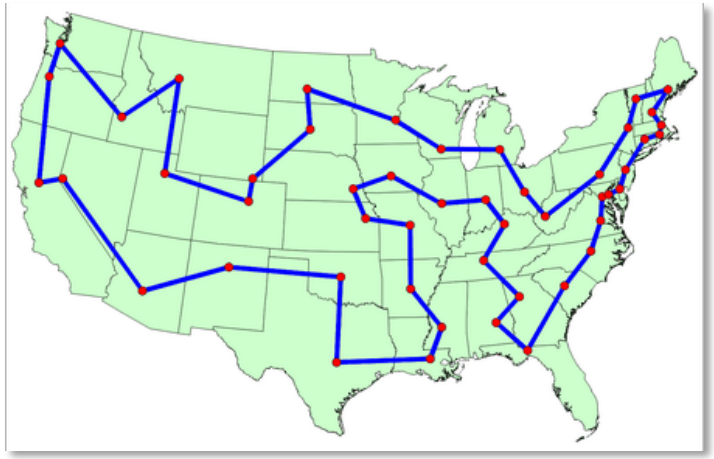






TSP among the capital cities of US states: many possible solutions!

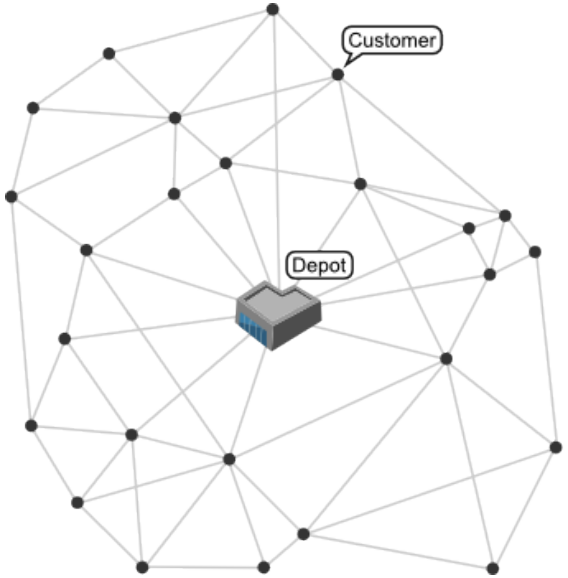




The solution of minimal traveling distance!

TSP is an abstract model of many real-world problems!

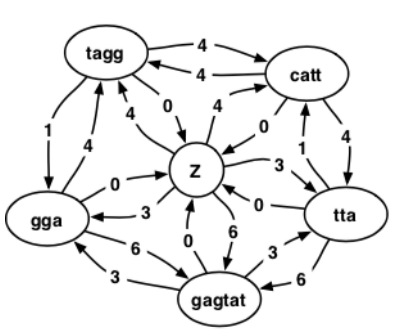
(Bus) route optimization

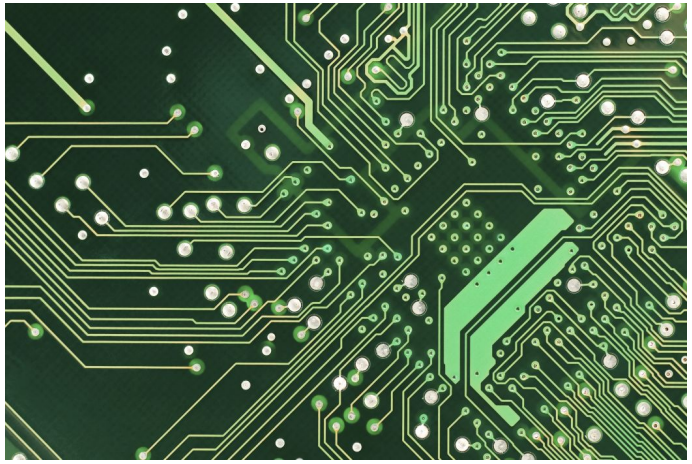


Delivery / Pick-up / Monitoring:

- Pizzas, Water
- Postal mail
- Express courier
- Flora, fauna
- …

Genome sequencing

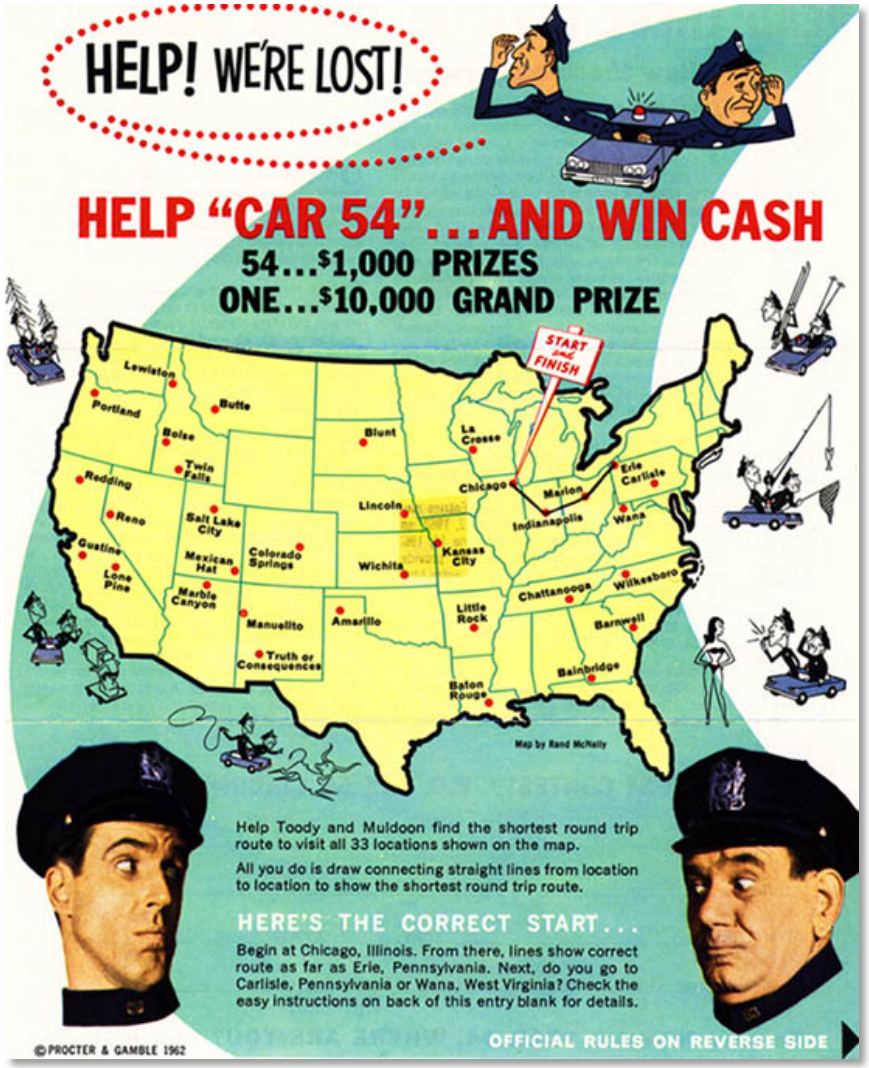




Microchips manufacturing

A big prize for solving a TSP instance, in 1962

✓ Procter & Gamble, 1962, 33-city contest in USA

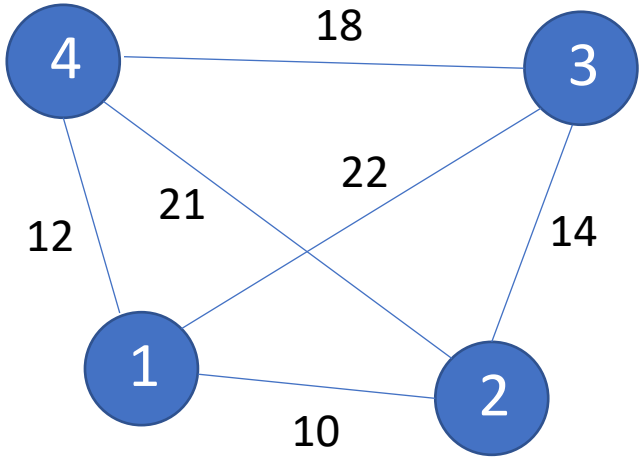


- 10,000 $ (1962) → 500,000 $ (2020)
- Winner: Gerald Thompson, CMU! ☺

Is it so difficult?

How many feasible solutions are there? (we need the best one!)

- A TSP with $n = 4$ sites to visit: how many possible solutions?



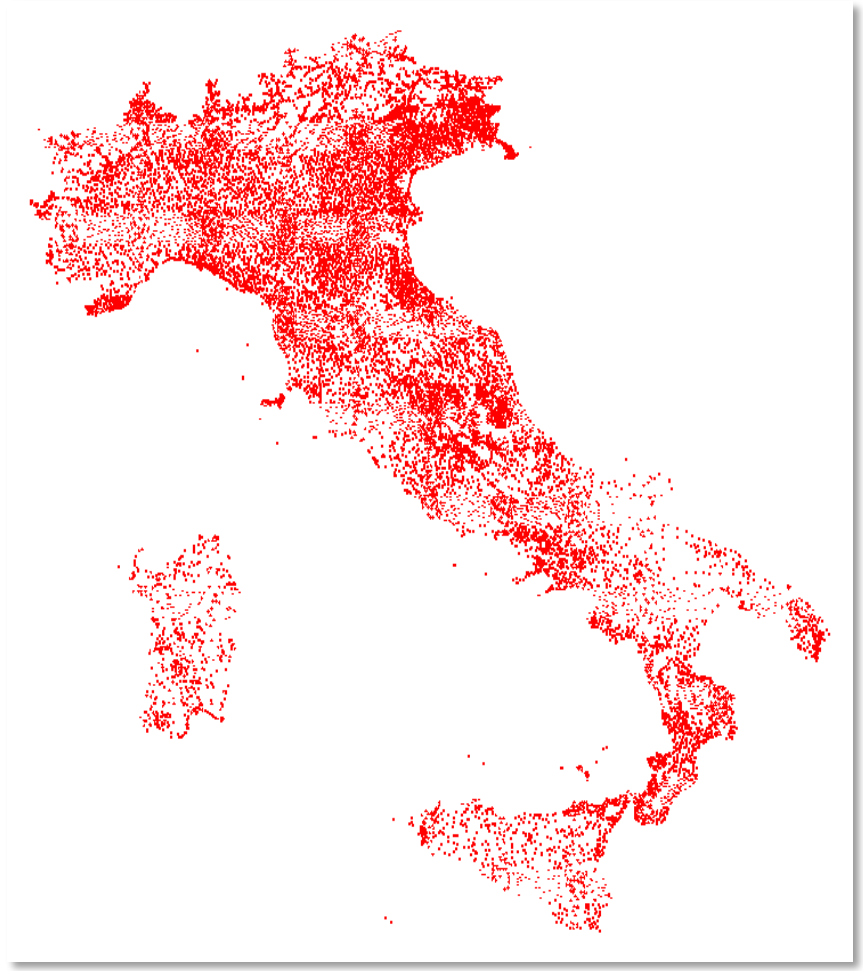
Optimal tour?

1234	2134	3124	4123
1243	2143	3142	4132
1324	2314	3214	4213
1342	2341	3241	4231
1423	2413	3412	4312
1432	2431	3421	4321

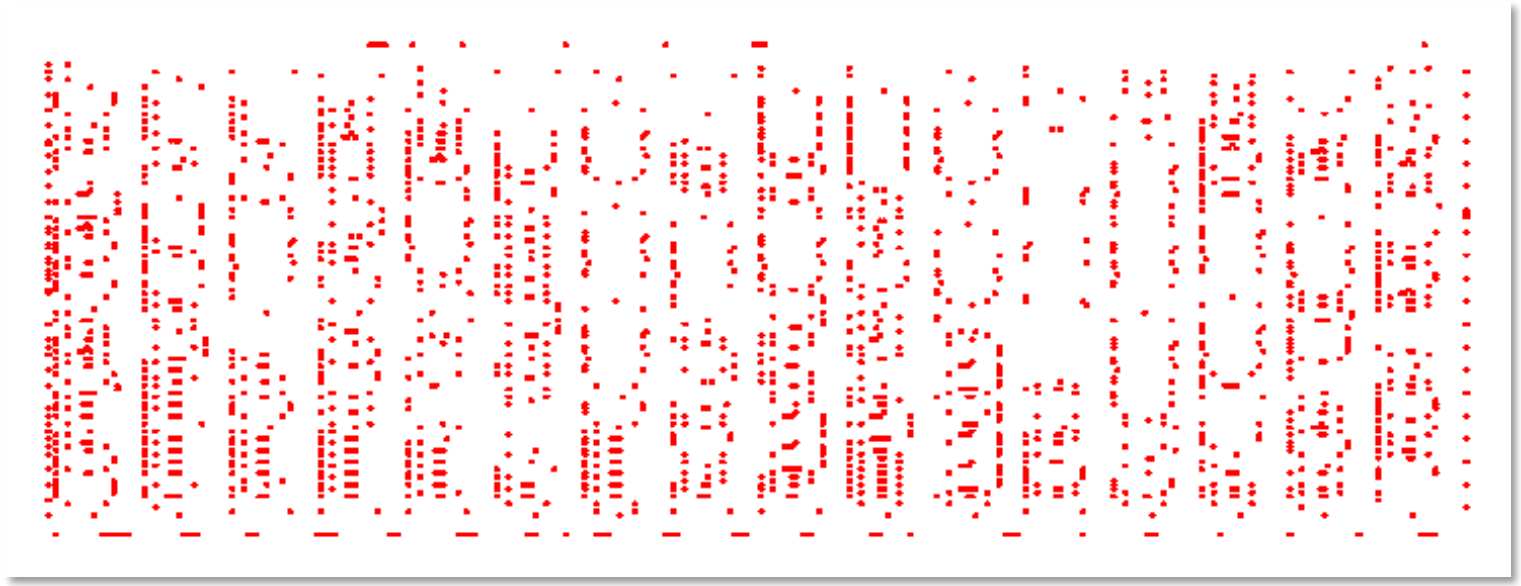
$24 = 4 \cdot 3 \cdot 2 \cdot 1 = 4!$

Number of feasible solutions: $\approx n!$

A very difficult problem when the number of sites is large



16,862 cities



21,144 sites

Optimal tour? By hand? → No way!

Factorial explosion, need for smart and efficient algorithms

Number of feasible solutions: $\approx n!$

- $n = 100 \rightarrow 100! \approx 9 \cdot 10^{157}$
- Number of atoms in the universe: $\approx 10^{80}$
- Fastest computer in the world (IBM AC922): $\approx 10^{15}$ FP operations per sec
- If it generates one solution per operation (not true): $\approx 10^{142}$ sec
- $\approx 10^{7}$ sec in one year $\rightarrow \approx 10^{135}$ years!
- Age of the universe: $\approx 10^{9}$ years …

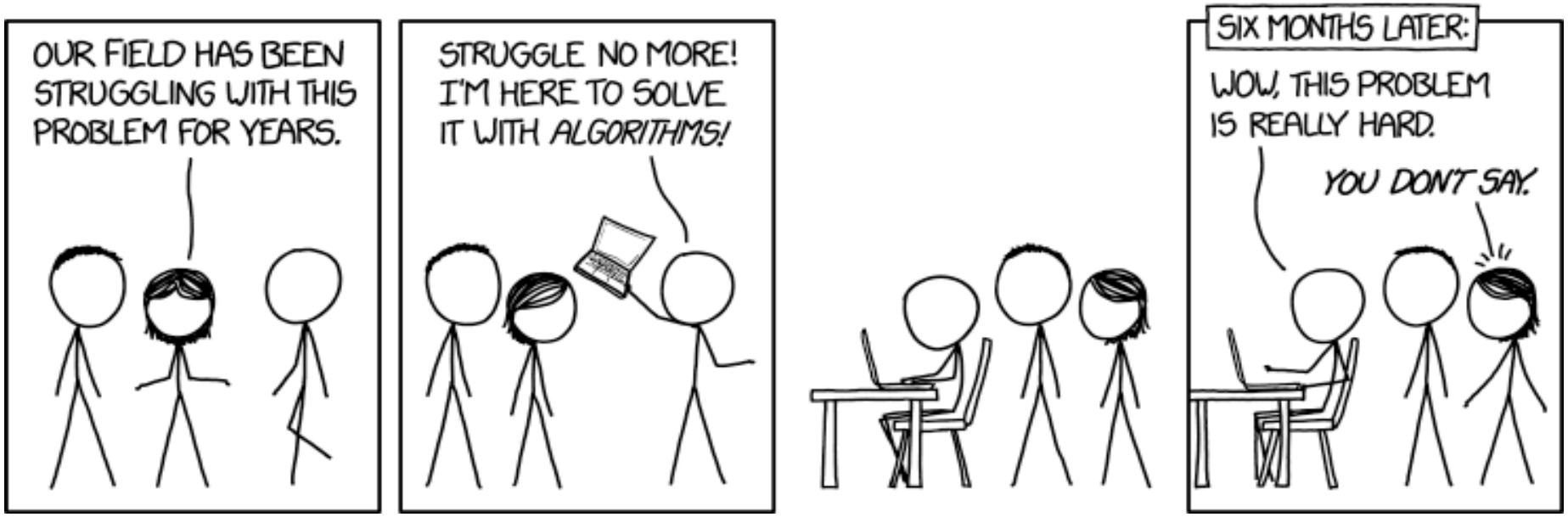
✓ Brute force (complete enumeration of all solutions) is not feasible!

✓ We need to find out smart (and formally sound) ways to search the solution set

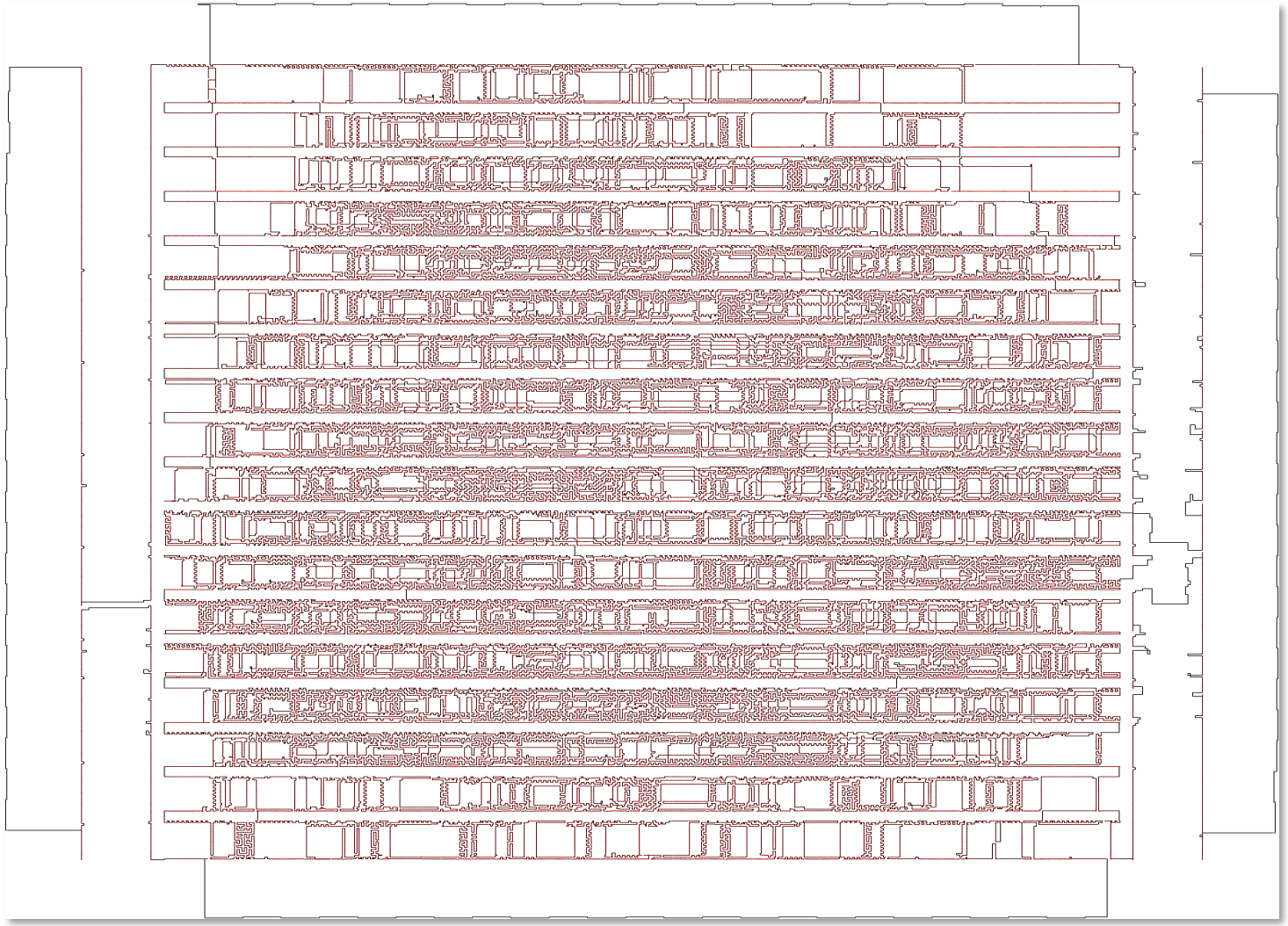
→ Intelligent & Efficient algorithms!!

NP-Hardness (just to let you know that such a thing does exist!)

- ✓ BTW, It doesn't matter how intelligent the algorithm is, in the worst case (i.e., for *some* problem instances) the time to find the optimal solution for a given TSP instance will explode exponentially with n ☹
  - ✓ NP-hardness (conjecture)



Largest solved (with guarantees of optimality of the solution found)



- Largest TSP instance solved to optimality with guarantees: 85,900 sites, from VLSI

The algorithm is smart: it exploits the structure of the problem for searching among all the possible solutions without searching among all of them 😉

Summary

- **Overview of the topics**: a journey into computation
  - Problem solving
  - Designing and understanding algorithms
  - Fundamentals programming constructs
  - Core techniques for computational problem-solving
  - Python as programming language
  - Tools (graphics, data manipulation, data publishing)
  - Applications

- **Action strategies**: first think, reason, abstract and model, design the algorithm, implement it by starting with a little chunk of code for a part of it, test it out, revise it, keep adding pieces of code, test with different inputs, understand what you've done!

- **The good student**: Actively participates to lectures + READs the slides + READs technical notes + READs the book chapters + Asks questions + Tries out code, tries out code, tries out code …. ☺