Fundamental Computer Science
Sequence 1: Turing Machines
An introduction

Denis Trystram

MoSIG1-M1Info, 2021

About the module FCS

Classes

- 33 hours in total (10 lectures plus a reading session) (half Theory, half Exercises/Practice).
- 5 topics
  1. Universal Computing Model: the Turing Machine
  2. Introduction to Quantum Computing
  3. NP-completeness
  4. Approximation Theory
  5. Introduction to parallel complexity
  6. Alternative model: λ-Calculus

Evaluation

- Exam: 70%
- Reading papers: 30%

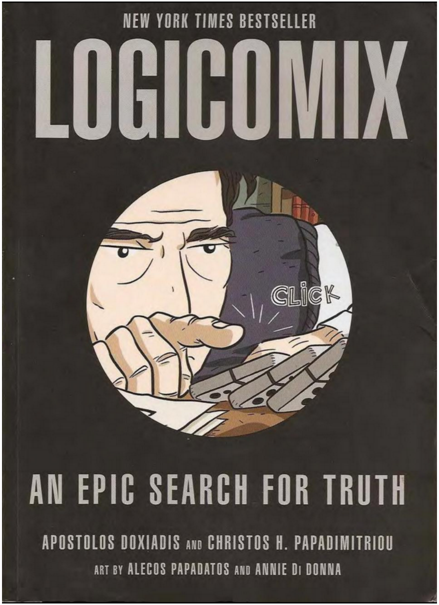
Organization

- Documents available at:
  $http://datamove.imag.fr/denis.trystram/teaching.php$
- Mattermost
  $https://im2ag-tchat.univ-grenoble-alpes.fr/$
- Active participation where some specialized topics are prepared by the students and discussed in class.
  Interactive class through many questions/answers.

References (Books)

- M. Garey and D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman
- P. Wolper, Introduction à la calculabilité, Dunod
- C. Papadimitriou, *Computational Complexity*, Pearson
- A. Rosenberg, The pillars of Computation Theory, Springer
- S. Arora and B. Barak, *Computational complexity – a modern approach*, Cambridge
- V. Vazirani, *Approximation Algorithms*, Springer
- R. Motwani and P. Raghavan, Randomized Algorithms, Cambridge Univ. Press

A comics about the beginning of fundamental CS



Agenda

Objective of the session

Present (and discuss) the universal computational model of **Turing machine**.

Guidelines

- Start by this introduction that present and discuss the concept of *algorithms*
- The main piece of the cake: basic Turing Machines
- Some exercises
- Extensions
  - Classical variants
  - Random access TM
- Non-Deterministic TM
- Three interesting related questions

Preliminary

What is an *Algorithm*?

The first question is to discuss what can be calculated by a *Computer*.

Preliminary

What is an *Algorithm*?
The first question is to discuss what can be calculated by a *Computer*.

Informally:
this is a step-by-step procedure that solves a problem.

Preliminary

What is an *Algorithm*?

The first question is to discuss what can be calculated by a *Computer*.

Informally:

this is a step-by-step procedure that solves a problem.

Desired properties

- clearly defined steps (formalization)
- efficiency
  how many –elementary– steps are needed for solving the problem?
- termination

Short History

- Etymology:
  - Al-Khwārizmī – a Persian mathematician of the 9th century
  - $\alpha\rho\iota\theta\mu$ός – the Greek word that means "number"

- Euclid's algorithm for computing the *greatest common divisor* (3rd century BC)
- End of XIXth century/beginning of XXth century: mathematical formalizations (proof systems, axioms, etc).
  Is there an algorithm for any problem?
- Church-Turing thesis (1930's): provides a formal definition of an algorithm (λ-calculus, Turing machine).
- Entscheidungsproblem (a challenge proposed by David Hilbert 1928): create an algorithm which is able to decide if a mathematical statement is true in a finite number of operations.
  Godel's and Turing's works in the 30ties show that a solution to Entscheidungsproblem does not exist.

A remark

This evolution was done **before** the reality of computers...

Preliminaries

alphabet: a finite set of symbols

- examples: Roman alphabet $\{a, b, \ldots, z\}$, binary alphabet $\{0, 1\}$

Preliminaries

alphabet: a finite set of symbols

- examples: Roman alphabet $\{a, b, \ldots, z\}$, binary alphabet $\{0, 1\}$

string: a finite sequence of symbols over an alphabet

- examples: $science$, 0011101
- ϵ: the empty string
- Σ^*: the set of all strings over an alphabet Σ (including ϵ)

Preliminaries

alphabet: a finite set of symbols

- examples: Roman alphabet $\{a, b, \ldots, z\}$, binary alphabet $\{0, 1\}$

string: a finite sequence of symbols over an alphabet

- examples: $science$, 0011101
- ϵ: the empty string
- Σ^*: the set of all strings over an alphabet Σ (including ϵ)

language: a set strings over an alphabet Σ (i.e., a subset of Σ^*)

- examples: $\emptyset$, Σ, Σ^*
- more examples:

  $L = \{w \in \Sigma^* : w$ has some property $P\}$
  $L = \{w \in \Sigma^* : w = w^R\}$ $\quad$ (w^R = reverse of w)
  $L = \{w \in \{0, 1\}^* : w$ has an equal number of 0's and 1's$\}$
  $L = \{w \in \{1, 2, \ldots, n\} : w$ is a permutation of $\{1, 2, \ldots, n\}$
  corresponding to a Hamiltonian Path in a graph of order n$\}$

Notion of Problem

Define first what is a *problem*:
An **id** (name), the **list of input** (with their coding) and a **question**.

Notion of Problem

Define first what is a *problem*:
An **id** (name), the **list of input** (with their coding) and a **question**.

Decision problem: a problem that can be posed as an yes/no question.

Notion of Problem

Define first what is a *problem*:
An **id** (name), the **list of input** (with their coding) and a **question**.

Decision problem: a problem that can be posed as an yes/no question.

- example:
  Prime
  Given a integer n
  Is n a prime?

Notion of Problem

Define first what is a *problem*:
An **id** (name), the **list of input** (with their coding) and a **question**.

Decision problem: a problem that can be posed as an yes/no question.

- example:
  Prime
  Given a integer n
  Is n a prime?

- another example:
  Hamiltonian Path
  Given a graph $G = (V, E)$
  Is there a permutation π of the vertex set such that
  $(v_{\pi(i)}, v_{\pi(i+1)}) \in E$ for all i, $1 \leq i \leq |V - 1|$?

Beyond decision problems

Optimization: a problem of searching for the best answer

Beyond decision problems

Optimization: a problem of searching for the best answer

- example:
  Given a graph $G = (V, E)$, two vertices $s, t \in V$ and an integer distance $d(e)$ for each $e \in E$
  find the path p between s and t such that the sum of distances of the edges in p is minimized.

  Transform it to a decision problem.

Beyond decision problems

Optimization: a problem of searching for the best answer

- example:
  Given a graph $G = (V, E)$, two vertices $s, t \in V$ and an integer distance $d(e)$ for each $e \in E$
  find the path p between s and t such that the sum of distances of the edges in p is minimized.

  Transform it to a decision problem.

- decision version: Given a graph $G = (V, E)$, two vertices $s, t \in V$, an integer distance $d(e)$ for each $e \in E$ **and an integer** D
  is there a path p between s and t such that the sum of distances of the edges in p is at most D?

Coding

Observation 1:
In most of these lectures we will deal with decision problems

Coding

Observation 1:
In most of these lectures we will deal with decision problems

Observation 2:
A decision problem is defined by the input and the yes/no question

- examples of input:
  - Given a set of numbers $A = \{a_1, a_2, \ldots, a_n\}$
  - Given a graph $G = (V, E)$
  - Given a graph $G = (V, E)$ and a positive weight $w(e)$ for each $e \in E$

Coding

Observation 1:
In most of these lectures we will deal with decision problems

Observation 2:
A decision problem is defined by the input and the yes/no question

- examples of input:
  - Given a set of numbers $A = \{a_1, a_2, \ldots, a_n\}$
  - Given a graph $G = (V, E)$
  - Given a graph $G = (V, E)$ and a positive weight $w(e)$ for each $e \in E$
- $< I >$: string encoding of the input
  - $< a_1, a_2, \ldots, a_n >$
  - $<$ adjacency matrix of $G >$
  - $<$ adjacency matrix of $G, w(e)\ \forall e \in E >$

Coding

Observation 1:
In most of these lectures we will deal with decision problems

Observation 2:
A decision problem is defined by the input and the yes/no question

- examples of input:
  - Given a set of numbers $A = \{a_1, a_2, \ldots, a_n\}$
  - Given a graph $G = (V, E)$
  - Given a graph $G = (V, E)$ and a positive weight $w(e)$ for each $e \in E$
- $< I >$: string encoding of the input
  - $< a_1, a_2, \ldots, a_n >$
  - $<$ adjacency matrix of $G >$
  - $<$ adjacency matrix of $G, w(e)\ \forall e \in E >$
- $|I|$: size of the input (in binary)
  - $\log_2 a_1 + \log_2 a_2 + \ldots \log_2 a_n$
  - $|V|^2$ or $k \cdot |V|$ where k is the average degree
  - $|V|^2 + \sum_{e \in E} \log_2 w(e)$