Complementary Lecture Notes (1)
Background for Computability

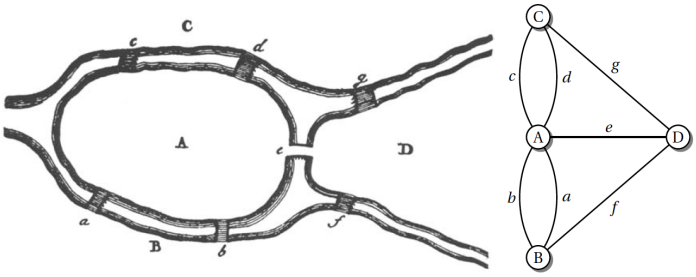
MSc in Physics Engineering - Quantum Computation, L. S. Barbosa, 2024-25

Summary
(1) An invitation to computability and computational complexity: Eulerian and Hamiltonian paths.
(2) Computation and mathematics: Babbage's calculators and Hilbert's *Entscheidungsproblem*
(3) A brief background tour: Sets, functions, relations.

---

1 Computability and complexity: Eulerian and Hamiltonian paths

Euler's problem (1736): crossing the bridges of Königsberg



A walk through a graph that crosses each edge once, starting and finishing at the same place.

- **Exhaustive search**: computation time exponential on the number of bridges.

  > *As far as the problem of the seven bridges of Königsberg is concerned, it can be solved by making an exhaustive list of possible routes, and then finding whether or not any route satisfies the conditions of the problem. Because of the number of possibilities, this method of solutions would be too difficult and laborious, and in other problems with more bridges, it would be impossible.*

- **Euler's discovery**: computation time scales linearly, ... and offers a useful explanation:

  The existence of a path was shown equivalent to a much simpler property: if a path is going to cross every link exactly once, then each node within the path must have an even number of links attached to it. Actually, whenever one enters the node by one link, one needs to leave it by another, so the node needs two links if one visits it once, four if visited twice, and so on. The only nodes that can have an odd number of links attached to them are the nodes where the walk starts and ends (if they are distinct).

The theorem: *A connected graph contains an Eulerian path if and only if every vertex, with the exception of the vertices where the path starts and ends, has even degree.*

Exercise 1

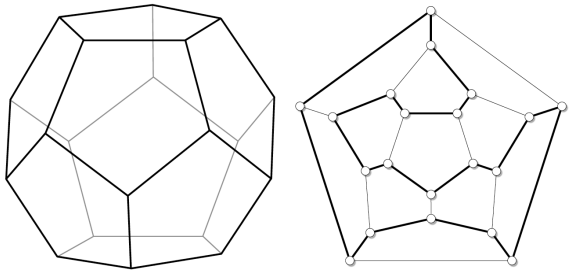
Can you find a rigorous proof of Euler's theorem? Based on the insight you may take from the proof, can you design a simple algorithm that constructs an Eulerian path in a given graph?

Hamilton's problem (1859): walk around the edges of a dodecahedron

Finding an Hamiltonian path, which visits each vertex once, is not known to be reductable to an easier problem. Thus, the best possible algorithm scales exponentially on the number n of nodes, requiring an execution time proportional to 2^{kn}, for k a constant.

Exercise 2

Does the actual value of k actually matter?



While **finding** a Hamiltonian path seems to be hard, **checking** whether a given path is Hamiltonian is easy: simply follow the path vertex by vertex, and check that it visits each vertex once. This turns out to be an important observation for the future.

2 Computation and mathematics

Let us forget about how an algorithm scales, and ask instead if, for any problem, there is an algorithm that solves it given a finite, but arbitrary, amount of time.

Basic ingredients of *computation*:

- **Algorithm**: specification, in a precise language, of a way to carry out some calculation or solve a problem.
- **Processor**: a programmable device that can carry out any given algorithm.

From an *engineering* point of view, attempts to build a universal (i.e. programmable) calculation machine can be traced back to Charles Babbage's *Difference engine* and *Analytical engine*: *I wish to God that these calculations had been executed by steam!* (Babbage, 1821)

> *The Analytical Engine does not occupy common ground with mere calculating machines. In enabling mechanism to combine together general symbols in successions of unlimited variety and extent, a uniting link is established between the operations of matter and the abstract mental processes of the most abstract branch of mathematical science...* (Ada, Countess of Lovelace)

The notion of an *algorithm*, as an object of specific study and a well-defined mathematical object, arose only in the beginning of the 20th century, associated to the famous David Hilbert's 10th problem:

> *Specify a procedure which, in a finite number of operations, enables one to determine whether or not a given Diophantine equation with an arbitrary number of variables has an integer solution.* (1900)

More generally,

> *The Entscheidungsproblem is solved if one knows a procedure that allows one to decide the validity of a given logical expression by a finite number of operations* (1928)

For example, give an algorithm to express, for example, Fermat's last theorem and determine, in a finite amount of time, whether it holds or not:

$$\neg\exists_{x,y,z,\in\mathbb{N}-\{0\},n\geq 3}.\ x^n + y^n = z^n$$

If the answer to this question was positive, this would pave the way to the *mechanization of mathematics*.

The general context was the quest for an *axiomatic foundation for mathematics*, i.e. the attempt to reduce all of mathematics to set theory and logic, creating a formal system powerful enough to prove all the mathematical facts.

This programme became problematic and lead to paradoxes — cf the Russell's paradox — *the set of sets that are not elements of themselves is not a well-defined mathematical object*. Later, in 1931, Gödel proved that no formal system can provide a complete foundation for mathematics, thus putting an end to Hilbert's programme.

On the other hand, the quest for a universal processor and for a suitable notion of program lead to

- a number of definitions of computability: a procedure is *computable* if it can be i) encoded in a Turing machine, ii) expressed in the λ-calculus, iii) formulated as a partial recursive function;
- and a to fundamental conjecture: the *Church-Turing thesis* that all these notions are equivalent and capture anything that could be reasonably called a computation.

In 1936, A. Turing formulated a problem — the *Halting Problem* (*does a given program ever terminate?*) and showed it *uncomputable*, or, what is the same, *undecidable*.

Curiously enough, the techniques used for understanding Russell's paradox and the Halting problem are similar: it is the self-referential nature of computation which leads to both undecidable problems and unprovable truths (as we will explore in the next lecture).

For the moment, let us revisit a few notions and techniques of set theory.

3 Sets, functions and relations

To recall

- Function $f : A \longrightarrow B$. Composition.
- Notation A^B to represent the space of functions from B to A.
- Injective, surjective and bijective functions.
- Powerset ($\mathcal{P}(A)$ or $\mathbf{2}^A$).
- Russell's paradox.
- Binary relations; $\mathbf{2}^{A\times B} \cong \mathbf{2}^{A^B}$.
- Equivalence relations. Orders.

Finite and infinite sets

- equicardinality *vs* isomorphism.
- finite *vs* infinite.
- countable *vs* uncountable:
  A set A is *countable* iff there is an injective function $f : A \longrightarrow \mathbb{N}$.
  It is *infinite countable* if f is a bijection.

Ranking cardinality

$$|A| \leq |B| \text{ iff there is an injection } f : A \longrightarrow B$$

Relation $\leq$ above is a total order. Note that proving antisymmetry (i.e. the Cantor-Bernstein-Schroeder theorem) and totality (which requires the axiom of choice) is extremely hard. Let us discuss some applications.

Theorem

$\mathbb{N}$ and $\mathbb{Z}$ have the same cardinal

Proof (hint)

Consider $h : \mathbb{Z} \longrightarrow \mathbb{N}$ defined as follows and show it is a bijection:

$$h(x) = \begin{cases} 2x & \Longleftarrow x > 0 \\ -2x + 1 & \Longleftarrow \text{otherwise} \end{cases}$$

□

Theorem

$\mathbb{N}$ and $\mathbb{N} \times \mathbb{N}$ have the same cardinal

Proof

Look for a bijection between $\mathbb{N}$ and $\mathbb{N} \times \mathbb{N}$. Let's see some (of several) possibilities:

Enumerate all pairs of numbers which sum $0, 1, 2, \cdots$:

(0,0)
(0,1) (1,0)
(0,2) (1,1) (2,0)
(0,3) (1,2) (2,1) (3,0) $\cdots$
$\vdots$

For every sum n there are only finitely many, actually $n + 1$ pais (i, j) that sum n. I.e. for every number n, one gets all the pairs which sum n. On the other hand, since every pair of numbers (i, j) has a finite sum it will appear somewhere on this list. This defines a bijection

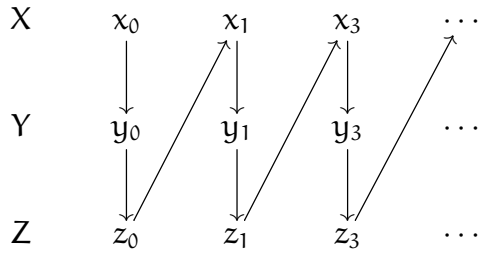
$$\begin{aligned} (0,0) &\mapsto 0 \\ (0,1) &\mapsto 1 \\ (1,0) &\mapsto 2 \\ (0,2) &\mapsto 3 \\ (1,1) &\mapsto 4 \\ &\cdots\cdots \end{aligned}$$

□

Theorem

The union of a finite number of countably infinite sets is countably infinite.

Proof (hint)



□

Theorem

$|\mathbb{N}| < |\mathbb{R}|$

Proof

To show that $|\mathbb{N}| \leq |\mathbb{R}|$ is trivial: function $h(n) = n$ is injective.

The difficult part is to prove that $\mathbb{N} \neq \mathbb{R}$. Let us prove an even stronger statement: that there is no surjection from $\mathbb{N}$ to $[0,1[$. Consider an arbitrary function $h : \mathbb{N} \longrightarrow [0,1[$ with which one may enumerate an infinite sequence of real numbers

$$r_0, r_1, r_2, \cdots$$

making $r_i = h(i)$.

To show that h in not surjective, we have to find a real x such that $r_n \neq x$ for all $n \in \mathbb{N}$. Let us build x as an infinite dizime

$$0.x[0]x[1]x[2]\cdots$$

such that

$$x[i] \;=\; \begin{cases} 1 \;\Leftarrow\; r_i[i] = 0 \\ 0 \;\Leftarrow\; \text{otherwise} \end{cases}$$

Observe that any real $h(n)$ differs from number x exactly in position n, and conclude that x does not belong to the image of h.

□

Theorem

$|\mathbb{N}| < |2^{\mathbb{N}}|$

Proof

If both sets were the same size, it would be possible to put them in one-to-one correspondence. In other words, there would be a way to list all subsets as $S_0, S_1, \cdots$ and so on, such that for every $S \subseteq \mathbb{N}$, then $S = S_i$, for some $i \in \mathbb{N}$.

In a table list the elements of S

$$\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & \cdots \\ 1 & 1 & 1 & 1 & 1 & \cdots \\ 1 & 0 & 1 & 0 & 1 & \cdots \\ 0 & 0 & 1 & 1 & 0 & \cdots \\ \vdots & & & & & \end{array}$$

Now, take the diagonal of this table, and flip all its bits. This gives us a new set D. Since $i \in D$ iff $i \notin S_i$, then $D \neq S_i$, for all i. Therefore, table D is not complete. Any correspondence between $\mathbb{N}$ and $2^{\mathbb{N}}$leaves at least one subset out.

Both theorems use a popular proof principle: *diagonalization*. Along the same path, the argument can be generalised:

Cantor Theorem

For any set X, $|X| < |2^X|$.

Proof

Function

$$\eta : x \longrightarrow \{x\}$$

is trivially injective. However there is no surjection $h : X \longrightarrow 2^X$.

To argue by contradiction, suppose such a function h exists and consider a set

$$W = \{x \in X \mid x \notin h(x)\}$$

If h is indeed surjective it must exist an element $w \in X$ such that $h(w) = W$ and w may or may not belong to $h(w)$.
These two cases are as follows as both lead to a contradiction:

- $w \in h(w)$ but then $w \notin W$,
- $w \notin h(w)$ but then $w \in W$

which invalidates our assumption that h is a surjection.

□

NOTE: Unsolvable problems. This theorem sheds light on the limits of computability: *there are more problems that we might want to solve than there are programs to solve them, even though both are infinite*.

To see this, restrict your attention to one type of problem: deciding whether a string has some property (e.g. having even length, being a palindrome, or a legal Haskell program). A property can be identified with the set of strings that happen to share it. Clearly, the number of possible programs is no bigger than the number of strings, while the number of sets of strings is strictly greater.

This shows the existence of unsolvable problems, i.e. problems that can be formulated but not possibly solved.

Another important principle to reason about cardinalities is the following:

Theorem: the pigeonhole principle

Let m objects be distributed into n containers. If $m > n$, then some container contains at least two objects

Or

If f is a function from a set S to a smaller set R, then there must be some pair x, y such that $f(x) = f(y)$.

I.e. if S pigeons nest in a smaller set of nests, at least one pair of pigeons must nest together.

Proof

By contrapositive: let us show that if every container contains at most one object, then $m \leq n$.

If c_i is the number of objects in container i, then

$$m = \sum_{i=1}^{n} c_i$$

but, every container contains at most one object, we get

$$m = \sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} 1 = n$$

□

Applications: Given a large enough number of objects with a bounded number of properties, eventually at least two of them will share a property.

Ex. 1
Suppose that every point in the real plane is coloured either red or blue. Then for any distance $d > 0$, there are two points exactly distance d from one another that are the same color.

Ex. 2
Similarly, there are at least two Portuguese citizens with exactly the same number of individual hair in their heads.

Ex. 3
For any natural number n, there is a nonzero multiple of n whose digits are all 0s and 1s.