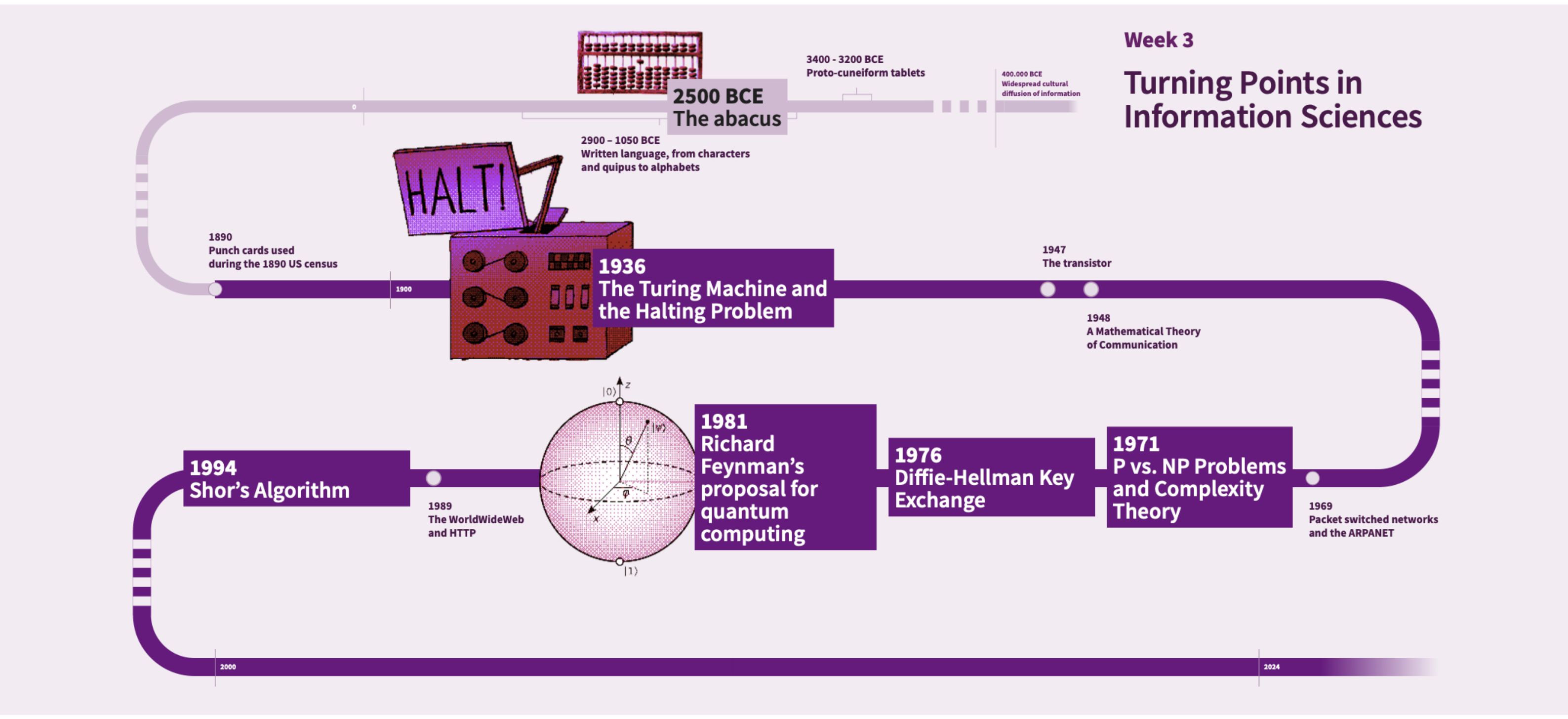
Turning Points in the Information Sciences
Part 3: From Computing to Quantum Computing

Ulle Endriss and Christian Schaffner
University of Amsterdam


Week 3
Turning Points in Information Sciences
400.000 BCE
Widespread cultural diffusion of information
3400 - 3200 BCE
Proto-cuneiform tablets
2500 BCE
The abacus
2900 – 1050 BCE
Written language, from characters and quipus to alphabets
0
1890
Punch cards used during the 1890 US census
1900
HALT!
1936
The Turing Machine and the Halting Problem
1947
The transistor
1948
A Mathematical Theory of Communication
1969
Packet switched networks and the ARPANET
1971
P vs. NP Problems and Complexity Theory
1976
Diffie-Hellman Key Exchange
1981
Richard Feynman's proposal for quantum computing
1989
The WorldWideWeb and HTTP
1994
Shor's Algorithm
2000
2024

Outline

In earlier lectures you learned about how to *gather* information, how to *represent* it, how to *transmit* it, and eventually how to *quantify* it. Now we want to *process* information, transforming a piece of information we are given into a new piece of information we care about. In other words, we want to *compute*. This part of the course consists of a series of three lectures, each addressing a fundamental question regarding the nature of computation:

(1) What can be computed (*at all*)?

(2) What can be computed *efficiently*?

(3) What can be computed *on a quantum computer*?

1 What can be computed (at all)?

Intuitively speaking *computing* means transforming a given input, which might be a natural number or a finite string of symbols, into a desired output. Examples include checking whether a given number is prime or putting a given list of words in alphabetical order. In the first lecture we will turn this intuitive idea into a precise definition that captures any kind of computation, and we will see that there are fundamental limits to what can be computed.

The first part of the lecture is about *Turing Machines*. A Turing Machine is an idealised form of a computer that, at least in principle, can do anything a real-world computer can do or ever will be able to do. This level of abstraction allows us to reason about computing in very general terms. An important take-away message will be that, at some level, it does not matter what kind of computer you have or what kind of programming language you use. What can and cannot be computed does not, at least not in principle, depend on such details.

The second part of the lecture will be about the limits of computing. We will hear about the *Halting Problem*, which asks whether a given program P will halt when executed with a given piece of data x as input, or whether it will keep running forever—seemingly the simplest question one can ask about a program. Surprisingly though, it is impossible to write a program that could be used to answer this question for every possible program P and every possible input x. This is so independently of the kind of technology we might have access to, now or in the distant future. Maybe even more surprisingly, there is a fairly simple argument for why solving this problem really is a mathematical impossibility.

The origin of both of these concepts, Turing Machines and the Halting Problem, can be traced back to Alan Turing's paper on "computable numbers" and the "*Entscheidungsproblem*" [the decision problem], which he published in 1936 at the age of 24. The publication of this paper is widely regarded as marking the birth of the scientific discipline of Computer Science, and thus a significant turning point for the Information Sciences more generally.

Resources and further reading

- Have a look at Turing's **original paper**: Alan M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936. (You should have free access when using the UvA VPN.)

  Of course, reading a paper that will soon be 100 years old is never easy, certainly not when it's a paper that was written for fellow experts working on what was probably the hottest topic in Mathematics of its day. But you might still recognise some of the ideas discussed in class—there using modern terminology. And it is always interesting to see what a paper of such historical importance actually looks like.

- If you want to find out more about the *Grundlagenkrise*, the existential crisis that shook the world of Mathematics in the early decades of the 20th century and that provided the backdrop against which Turing wrote his paper, there's a great **graphic novel** telling that story: Apostolos Doxiadis and Christos Papadimitriou. *Logicomix: An Epic Search for Truth*. Bloomsbury, 2009. (Available at the UvA Library, in Dutch translation.)

- If you want to find out more about Turing's life, there are lots of good books out there, some with more and some with less technical content. Probably the best-known **biography** is this one, which also does a good job at explaining some of Turing's scientific contributions: Andrew Hodges. *Alan Turing: The Enigma*. Princeton University Press, 2014. First edition from 1983. (Available at the UvA Library.) The Hollywood blockbuster The Imitation Game from 2014 was based on this book.

2 What can be computed efficiently?

Even if we know that something can be computed in principle, this does not necessarily mean that we can also compute it in practice. To achieve the latter, certainly once problems get large in size, we require algorithms that are *efficient*. In the second lecture, we will offer a formal definition of what it means for a problem to be efficiently solvable, and we will see that for certain problems designing an efficient algorithm might be very difficult—and maybe even impossible. We will see that such limitations can have both good and bad consequences.

We will start with two problems that look very similar to one another and that both involve (weighted) *graphs*, which can be used, for instance, to model a network of cities and the roads connecting them. The *Travelling Salesperson Problem* (TSP) asks whether we can find a tour visiting all cities that does not exceed a given length, while the *Shortest Path Problem* (SPP) asks whether we can find a path from one specific city to another city that does not exceed a certain length. Despite close to 100 years of trying, so far nobody has found an efficient algorithm for the TSP. Here, "efficient" means that there exists a polynomial function f such that the runtime of the algorithm will be at most $f(n)$ hours when the size of the input problem is n (where, in the case of the TSP, n is the number of cities). In contrast, in the 1950s, Edsger W. Dijkstra, the most famous Dutch computer scientist and the only Dutch winner of the Turing Award[1] to date, designed a simple and efficient algorithm for the SPP.

Whether or not it is in fact *impossible* to design an efficient algorithm for the TSP is not known, and this question is intimately connected to the most famous—and some say the most important—open problem in all of Computer Science. This is the P *vs* NP problem. P is the set of all computational problems that can be *solved* in polynomial time, while NP is the set of all computational problems for which a suggested solution can be *verified* in polynomial time. If there exists an efficient algorithm for the TSP, then P $=$ NP, and if there does not exist such an algorithm, then P $\neq$ NP. There's a 1,000,000 dollar reward for figuring this out. Formulating and realising the importance of this problem in the early 1970s has been a major turning point for Computer Science.

While for problems such as the TSP, we would very much like to find efficient algorithms, for some other problems the fact that designing efficient algorithms is very difficult is a blessing. A prime example can be found in cryptography, where such difficult problems form the basis for the security of systems. A particular example we will discuss is the *Diffie-Hellman key exchange protocol*, which marks a turning point in cryptographic history, namely the invention of *public-key cryptography*. Using this scheme, Alice and Bob are able to communicate over an open channel with no prearrangement, but keep their information secret from any potential eavesdropper Eve. The security provided by this protocol is based on the assumption that Eve cannot efficiently solve a particular number-theoretic problem that we will explain, namely the problem of how to compute discrete logarithms in cyclic groups.

Resources and further reading

- Have a look at Dijkstra's **original paper** on the SPP: Edsger W. Dijkstra. A Note on two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959. Actually, the paper talks about algorithms for two different problems in graph theory. Can you decipher which of the two is the SPP?

  The paper is extremely short. Dijkstra was famous for writing very short and very clear papers. He also was famous for writing lots of handwritten notes documenting his ideas, which computer scientists across the globe used to photocopy and and send around by snail mail (remember, much of this happened well before the Internet). You can find out more at the E.W. Dijkstra Archive maintained at the University of Texas at Austin.

- Researchers at the University of Waterloo in Canada computed an optimal bike tour visiting all 57,912 national monuments in the Netherlands, a great example for a real-

[1]The Turing Award, named after Alan Turing, is the highest distinction in Computer Science, often referred to as the "Nobel Prize of Computer Science", as Computer Science is not one of the fields that are covered by the official Nobel Prizes.

world TSP instance. They discuss some of the **algorithmic techniques** required to tackle such a demanding problem. The first student to complete the tour wins a chocolate bar.

- In case you want to claim the 1,000,000 dollar reward for resolving the P *vs* NP problem, one of the **Millennium Prize Problems**, it's a good idea to first check the official rules of the competition on the website of the Clay Mathematics Institute.
- Here's a good **popular science book** that introduces the P *vs* NP problem and then discusses its enormous significance not only to the Information Sciences but to society at large: Lance Fortnow. *The Golden Ticket: P, NP, and the Search for the Impossible*. Princeton University Press, 2013. (Available online through the UvA Library.)
- The **1976 breakthrough article** New Directions in Cryptography by Whitfield Diffie and Martin E. Hellman marks a turning point in cryptographic history, namely the invention of public-key cryptography. In 2015 they received the Turing Award for this achievement.

3 What can be computed on a quantum computer?

While classical computers use bits (0 and 1) as the basic unit of computation, quantum computers use quantum bits (or qubits) which can be in *superposition* of multiple states simultaneously. This allows quantum computers to carry out many computations in parallel in the different branches of the superposition. However, in order to obtain a final classical result, the superposition state needs to be measured which causes only a random branch to survive. In order to harness the power of this different computational model, quantum computers need to make clever use of *interference* effects, so that only the desired outcomes remain before the measurement.

In this lecture, we will take a computer-science approach to quantum computing, illustrating the fact that these two crucial ingredients of quantum mechanics (superposition and interference) can be described and understood in terms of simple matrix and vector operations.

Quantum computers hold the potential to revolutionise the field of computation, as they seem to be able to solve certain problems more efficiently than the best known classical algorithms, with major consequences for cryptography, simulation of quantum systems, and optimisation problems. Importantly, this speed-up is not the result of higher clock speeds (in fact, quantum computers tend to be rather slow devices compared to modern computers), but stems from the change of the underlying computational model (using qubits and matrix-vector operations allowed by quantum mechanics) which allows us to solve certain problems using fewer steps.

The most famous quantum algorithm was discovered by Peter Shor in 1994. Shor's algorithm can factor large integer numbers in quantum polynomial time, while the best known classical algorithm for this problem runs in sub-exponential time. A variant of Shor's algorithm can also efficiently solve the discrete-logarithm problem, which is widely used on the internet to secure our communication, as explained in the previous lecture. Therefore, when large enough quantum computers come around, the currently used cryptographic schemes will become insecure.

However, it remains a formidable practical and theoretical challenge to build quantum hardware that is good enough and large enough to be of practical value. Physical qubit systems are inherently unstable and noisy, but it is known in theory how to use quantum error correction to cope with noise and imperfect operations.

The main take-home messages of this lecture are:

- **Quantum Speedups:** For specific computational tasks, quantum computers can perform calculations in significantly fewer steps, by making clever use of the quantum effects of *superposition* and *interference*. Despite the fact that quantum computers will be rather *slow* in terms of clock rate, this reduced number of steps allows them to solve these problems faster than any classical super computer.
- **Quantum Algorithms:** Algorithms such as Shor's for factoring large numbers and Grover's for searching unsorted databases exemplify tasks where quantum computers can possibly outperform classical approaches.
- **Impact on Cryptography:** Large-scale quantum computers will break the currently used public-key cryptography (as introduced in the previous lecture). Therefore, one needs to upgrade our current cryptography to quantum-safe variants.

Resources and further reading

- **Richard Feynman's Vision (1981):** Proposed the idea of simulating quantum physics with computers that operate on quantum-mechanical principles. You can read a copy of the original paper, or see it put into context in a recent overview article by John Preskill.
- Peter Shor's original article from 1994: `https://arxiv.org/abs/quant-ph/9508027`
- Quantum Quest is a webclass developed by two QuSoft researchers to introduce quantum computing to high-school students.
- The open-source book by Koen Groenland entitled Introduction to Quantum Computing for Business gives a non-technical but scientifically sound overview of the current state of quantum technology.
- De nationale quantum cursus is a quantum course for the general public. Unfortunately, it's only available in Dutch for now.