Introduction to Algorithms

Aniel Nieves-González

Institute of Statistics

Spring 2014

"...algorithms are concepts that have existence apart from any programming language."

—Donald Knuth

What is an algorithm?

- The concept of algorithm is very old. For example: think about Euclid's algorithm to find the greatest common divisor (first described c. 300 BCE).

What is an algorithm?

- The concept of algorithm is very old. For example: think about Euclid's algorithm to find the greatest common divisor (first described c. 300 BCE).
- The word algorithm comes from the last name of Muhammad ibn Musa Al-Khowarizmi (780-850 CE). He was a Persian mathematician .

What is an algorithm?

- The concept of algorithm is very old. For example: think about Euclid's algorithm to find the greatest common divisor (first described c. 300 BCE).
- The word algorithm comes from the last name of Muhammad ibn Musa Al-Khowarizmi (780-850 CE). He was a Persian mathematician .
- Informally: An *algorithm* is a sequence of steps to solve a problem in a finite number of steps.

What is an algorithm?

- The concept of algorithm is very old. For example: think about Euclid's algorithm to find the greatest common divisor (first described c. 300 BCE).
- The word algorithm comes from the last name of Muhammad ibn Musa Al-Khowarizmi (780-850 CE). He was a Persian mathematician .
- Informally: An *algorithm* is a sequence of steps to solve a problem in a finite number of steps.
- More formally: An *algorithm* is a well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time.

Let's take this definition apart:

- Well-ordered collection: it means that the computing agent know which operation goes first.

.

Let's take this definition apart:

- Well-ordered collection: it means that the computing agent know which operation goes first.
- Unambiguous: means that the operation can be understood and carried out by the computing agent without any further explanation.

.

Let's take this definition apart:

- Well-ordered collection: it means that the computing agent know which operation goes first.
- Unambiguous: means that the operation can be understood and carried out by the computing agent without any further explanation.
- Effectively computable (doable): means that there is a computational process that allows the computing agent to compute the operation.

.

Let's take this definition apart:

- Well-ordered collection: it means that the computing agent know which operation goes first.
- Unambiguous: means that the operation can be understood and carried out by the computing agent without any further explanation.
- Effectively computable (doable): means that there is a computational process that allows the computing agent to compute the operation.
- Halts in a finite amount of time: It must terminate (no infinite loop).

.

Let's take this definition apart:

- Well-ordered collection: it means that the computing agent know which operation goes first.
- Unambiguous: means that the operation can be understood and carried out by the computing agent without any further explanation.
- Effectively computable (doable): means that there is a computational process that allows the computing agent to compute the operation.
- Halts in a finite amount of time: It must terminate (no infinite loop).

A procedure that possibly lacks finiteness may be called *computational method*.

A more rigorous definition of algorithm

Definition (Computational method)

A *computational method* is a quadruple (Q, I, Ω, f) in which Q is a set that contains I and Ω. f is a function from Q into itself. Furthermore $f(q) = q, \forall q \in \Omega$. The 4 quantities Q, I, Ω, and f are intended to represent respectively the states of computation, the input, the output and the computational rule.

For each input $x \in I$ the set I defines a *computational sequence*, $x_0, x_1, x_2, \ldots,$ as follows:

$$x_0 = x \text{ and } x_{k+1} = f(x_k) \text{ for } k \geq 0$$

The computational sequence is said to terminate in k steps if k is the smallest integer for which $x_k \in \Omega$ and in this case it is said to produce the output x_k from x.

A more rigorous definition of algorithm

Observe that some computational sequences may never terminate.

Definition (Algorithm)

An *algorithm* is a computational method that terminates in finitely many steps for all $x \in I$.

A more rigorous definition of algorithm

An equivalent way to formulate the concept of computational method (algorithm) is using *Turing machines*.

- At the early part of the XX century David Hilbert (a german mathematician) formulated the problem sometimes known as the “decision problem” (*entscheidungsproblem*). Hilbert asked whether or not there existed some algorithm that in principle could be used to solve certain type of problems in mathematics.

A more rigorous definition of algorithm

An equivalent way to formulate the concept of computational method (algorithm) is using *Turing machines*.

- At the early part of the XX century David Hilbert (a german mathematician) formulated the problem sometimes known as the "decision problem" (*entscheidungsproblem*). Hilbert asked whether or not there existed some algorithm that in principle could be used to solve certain type of problems in mathematics.
- Hilbert expected a yes as an answer to his question, albeit years later (in the 1930's) the answer turn out to be no.

A more rigorous definition of algorithm

An equivalent way to formulate the concept of computational method (algorithm) is using *Turing machines*.

- At the early part of the XX century David Hilbert (a german mathematician) formulated the problem sometimes known as the "decision problem" (*entscheidungsproblem*). Hilbert asked whether or not there existed some algorithm that in principle could be used to solve certain type of problems in mathematics.
- Hilbert expected a yes as an answer to his question, albeit years later (in the 1930's) the answer turn out to be no.
- In order to attack Hilbert's problem Alonzo Church and Alan Turing have to formulate a precise definition of the concept of algorithm.

A more rigorous definition of algorithm

An equivalent way to formulate the concept of computational method (algorithm) is using *Turing machines*.

- At the early part of the XX century David Hilbert (a german mathematician) formulated the problem sometimes known as the "decision problem" (*entscheidungsproblem*). Hilbert asked whether or not there existed some algorithm that in principle could be used to solve certain type of problems in mathematics.
- Hilbert expected a yes as an answer to his question, albeit years later (in the 1930's) the answer turn out to be no.
- In order to attack Hilbert's problem Alonzo Church and Alan Turing have to formulate a precise definition of the concept of algorithm.
- This laid the foundations of the modern theory of algorithms and computer science.

A more rigorous definition of algorithm: Turing machine

A Turing machine consists of two major elements: a tape and a control unit.

1. The *tape* is a sequence of cells that extends to infinity in both directions. Each cell contains a symbol from a *finite* alphabet. There is a tape head that reads and writes to the same cell.
2. The *control unit* contains a *finite* set of states and a *finite* set instructions The instructions can be represented as a 5-tuple. For example, the instruction (i, a, b, L, j) is executed as follows:
   If the current state of the machine is i and if the current symbol in the current tape cell is a, then write b in the current tape cell, move left (L) one cell, and go to state j.

A Turing machine can be illustrated as:

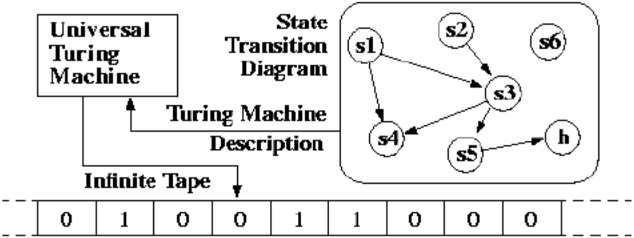

Image from: http://web.mit.edu/manoli/turing/www/turing.gif

Church-Turing thesis

What does it mean that a certain problem is computable?

Theorem (Church-Turing thesis)

Anything that is intuitively computable can be computed by a Turing machine.

A brief remark about computer science.

- Computer science can be thought as the dicipline that studies algorithms, including:

A brief remark about computer science.

- Computer science can be thought as the dicipline that studies algorithms, including:
    1. Their formal mathematical properties. (Determine correctness and efficiency).
    2. Their hardware realizations. (Design machines able to execute algorithms).
    3. Their linguistic realizations. (Design programming languages)
    4. Their applications. (Identify important problems and design correct and efficient algorithms).

Examples of algorithms

- A cooking recipe: check www.epicurious.com
- Calculate the area of a circle of radius r:

Examples of algorithms

- A cooking recipe: check www.epicurious.com
- Calculate the area of a circle of radius r:

Algorithm

1. *Input:* r
2. *set variable area to* $\pi^* r^2$ *where* π^* *is an approximation of* π.
3. *Output: area*

Examples of algorithms

- Add the first n natural numbers: (i.e., compute, $\sum_{i=1}^{n} i$)

Examples of algorithms

- Add the first n natural numbers: (i.e., compute, $\sum_{i=1}^{n} i$)

Algorithm

1. *Input:* n
2. *Set accum to 0 and* i *to 1*
3. *while* $i \leq n$
    - *set accum to accum* $+ i$
    - *increment* i *by 1*
4. *Output: accum*

Examples of procedures that aren't algorithms

- Calculate the area of a circle of radius r:
  1. Input: r
  2. set variable $area$ to πr^2
  3. Output: $area$

  Observe that π is exact!
- Find the nth prime number.
  1. Input: n
  2. Generate a list of all prime numbers $(p_1, p_2, \ldots)$.
  3. Sort that list.
  4. Select the nth item in that list (p_n)
  5. Output: p_n

A brief history of computing: calculators I

Remark: Calculators at the time lack two fundamental characteristics that computers have:

- A component that store information in machine readable form (memory).
- Programmable: An algorithm can be provided in advance to solve some problem.

(1) $\sim$ 1622: Slide rule.

(2) The Pascaline: a mechanical calculator built by Blaise Pascal. It is able to do $+$ and $-$.

(3) Around 1674 Leibnitz constructed what was called the Leibnitz wheel. The device was able to add, substract, multiply, and divide.

A brief history of computing: calculators II

(4) In order to automate the weaving process, J. Jacquard design and built a machine that had the two characteristics described above.

(5) Around 1823 C. Babbage built a machine called the difference engine. The device was able to add, substract, multiply, divide, and solve polynomial equations. It had the following components: mill (ALU), store (memory), operator (processor, it was actually a human), and output (I/O).

A brief history of computing: calculators III

(6) Circa 1890 H. Hollerith was a statistician working at the U.S. Census Bureau. At that time he designed and constructed a machine that could count, tally, and sort. It had all the components of Babbage's Analytic Engine, but still it was no general purpose computer. Later on, Hollerith left the Census Bureau and founded a company that in the 1920's became IBM.

(7) In 1944 Prof. H. Aiken completed the construction of machine designed by him (the project was funded by the U.S. Navy and IBM). It was called Mark I and this was a general-purpose electromechanical programmable computer. First computer to use the binary number system.

A brief history of computing: calculators IV

(8) The first fully electronic general-purpose programmable computer was completed in 1946. It was named ENIAC (Electronic Numerical Integrator and Calculator) The machine was designed by J. Machly and J.P. Eckert from the University of Pennsylvania. The project was funded by that university and the U.S. Army.

(9) Other early examples of computing systems include:
- Atansoff-Berry Computer (ABC system), which was designed to solve systems of linear equations.
- Colossus: A computer designed in the U.K. by A. Turing and his team around 1943. The purpose of rhe computer was to crack the German Enigma Code. This project was shroud in secrecy until not so many years ago.

A brief history of computing: calculators V

- In Germany the Nazis funded the design and construction of a device akin to ENIAC. It was called Z1, and it was designed by K. Zuse.

A brief history of computing: computers

The aforementioned general-purpose computers didn't follow a computing model proposed by Von Neumann. This model is now called the Von Neumann Architecture. The model proposed that not only the data, but also the instructions to be executed by the machine were stored in memory. Von Neumann invented programming as is known today (more on this later).

(10) In 1951 Von Neumann and his team implemented his model. The machine they built was named EDVAC. The commercial version of it was called UNIVAC I.

A brief history of computing: Modern era of computing

(11) First generation: 1950-1957. This is the period of machines like UNIVAC I and IBM 701. They were fully electronic general-purpose programmable computers, but its circuitry was made using vacuum tubes.

(12) Second generation: 1957-1965. This period saw the appearance of solid-state devices like the transistor. Also, the first high-level programming languages: FORTRAN and Cobol were developed in that period.

A brief history of computing: Modern era of computing

(13) Third generation: 1965-1975. In this period appears the integrated circuit. Thus, the circuitry can be minaturized and mass produced by photographically etching the the circuit into a silicon waffer. From computers that filled whole rooms, the computer became a desk-size object. The first mini-computer was born: PDP-1 by DEC.

(14) Fouth generation: 1975-1985. Advances in integrated circuit technology lead to the appearance of the first microcomputer. Thus, desk-size minicomputers became a desktop computer. The first microcomputer available was the Altair 8800 in January 1975. This period also saw the appearance of the first computer networks and of the email as an application. GUI's are also developed during this period.

A brief history of computing: Modern era of computing

(15) Fifth generation: 1985-?. A period in which we have:

- Massive parallel processing capable of millions of operations per second. E.g. top supercomputer in the world is Tianhe-2 (located in China) has a 3,120,000 cores and has a performance of 33862.7 TFlops (more than 1 PetaFlop).
- Smartphone revolution.
- The Internet: integrated global communications.
- Massive storage devices.
- Wireless data communication.

References I

- Donald D. Knuth.
  The art of computer programming. Volume 1: Fundamental Algorithms.
  Third edition. Addison-Wesley, 1997.

- G. Michael Schneider and Judith Gersting.
  An Invitation to Computer Science: C++ Version.
  4th Edition. Thomson: Course Technology, 2006

- James L. Hein
  Dicrete Structures, Logic, and Computability
  Second edition. 2002.