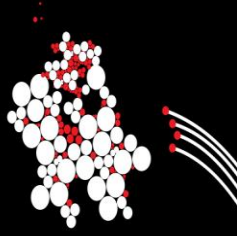




UNIVERSITY OF TWENTE.



Algorithmics

What is it about:
thoughts, ideas and hardly any computer code

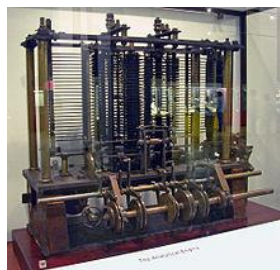



1



Almost 203 years ago

The famous English poet Lord George Byron and his wife Anne Milbanke became the proud parents of *Augusta Ada Byron* Countess of Lovelace, now known as **Ada Lovelace**.



Ada later became well-known for her work with the *Analytical Machine*, a mechanical device proposed in 1837 and almost constructed by *Charles Babbage*, and she is generally considered **the first machine programmer** for that reason.



UNIVERSITY OF TWENTE.

2

2



The algorithm and -ics

Algorithm: a (perceived) recipe to solve a computational problem

Perceived recipe: the intellectual idea of how to do it, without actually having written it up.

Computer program: realization of the recipe in some language, that allows the machine to actually do it

Computational problem: a problem that appears to be solvable by a computer
(And this is a much wider definition than what you would think at first ...)



UNIVERSITY OF TWENTE.

3

3



Lentil soup with cumin

The first meal we ate on Moroccan soil was lentil soup. We had just taken the ferry from Algieras to Tangiers and had arrived slightly weary and hungry from the day's travelling. We soon stumbled across a dark room filled with a few tables and a few people who kept one eye on their food, the other on us. There was only one thing on offer - lentil soup. We sat down and were immediately presented with a bowl of this steaming soup, a dish of harissa, some juicy, oily black olives and a couple of rounds of Moroccan bread. The soup was made from lentils subtly flavoured with cumin. It was a great introduction to Moroccan food: simple, exotically spiced and very satisfying, just what we felt like eating. We make this soup in the restaurant and sometimes add spinach for variety.

Serves 4

4 tablespoons olive oil
1½ large Spanish onion, thinly sliced
4 garlic cloves, thinly sliced
1½ rounded teaspoons cumin seeds, roughly ground
250g lentils (brown, green or yellow)
1.75 litres cold water
250g spinach braised with olive oil (see page 234), roughly chopped
1 lemon, quartered, or 150g yoghurt spiced with ¾ teaspoon ground cumin and salt
sea salt and black pepper

In a large saucepan, heat the oil over a medium heat. Add the onion with a pinch of salt and cook for about 10 minutes, stirring occasionally, until sweet and golden. Now add the garlic and cumin and fry for another minute, followed by the lentils and water. Bring to the boil, reduce the heat to a gentle simmer, and cook for about 20 minutes or until the lentils are soft. Remove from the heat and blend all the ingredients either in a food processor or by hand until almost smooth. Return to the pan along with the spinach, and season with salt and pepper. If the soup is too thick, simply add more water and adjust the seasoning. We serve this soup with lemon or yoghurt, and olives. [Harissa](#) (see page 234) and Moroccan bread (see page 20).

- 76 - Soups -

The Moro Cookbook

▪ Gentle **introduction** and **motivation** for having this recipe in this book

▪ The **ingredients**

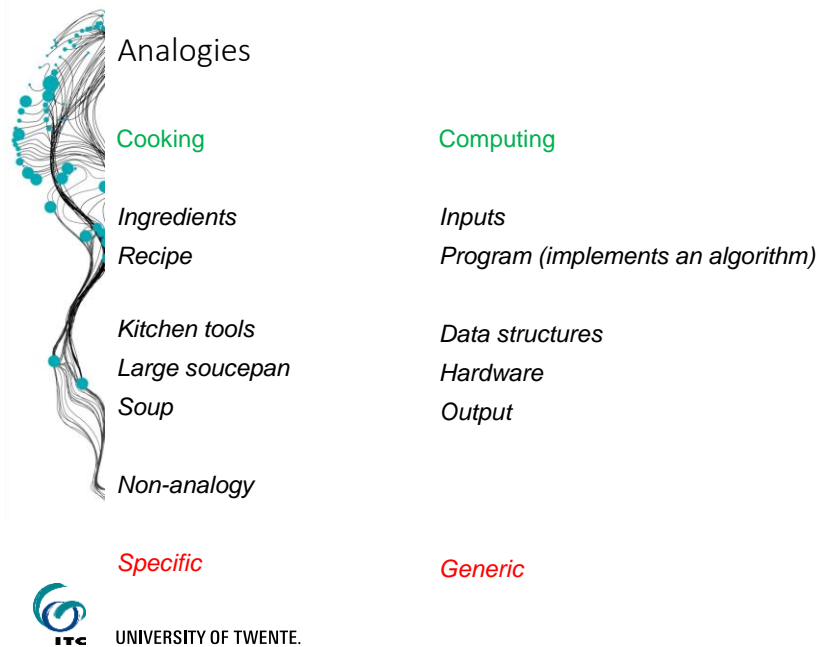


▪ The cooking **instructions**

▪ **References** to other recipes

4

4



5

5

Computational problems

Across the board:

- Fly an aircraft, or write a book
- Operate a power station
- Recognize a song on the radio
- Translate a business letter into Swahili
- Making sure your phone call reaches the called person
- Forecast the amount of rain for tomorrow
- Atmospherically correct a satellite image
- Determine the optimal routes for a fleet of DHL trucks
- Find a similar photo on the internet
- Trace the evolutionary history of the human genome
- Win a chess game

UNIVERSITY OF TWENTE.

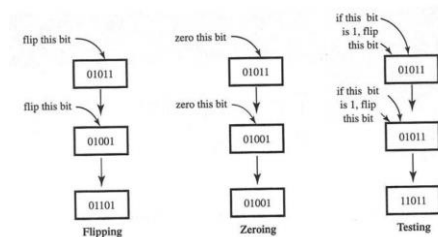
6

6

Solvability

Computers (in all disguises) and the *software* (algorithms) they run are remarkably capable of solving hard problems. All of this with an extremely limited repertoire of basic functions:

- Memory registers that hold bits
- Capability to flush bits in and out of memory
- Some extremely basic functions to flip, shove, set, and test bits
- And yes: *speed* in doing these things



UNIVERSITY OF TWENTE.

7

7

Solvability

Yet, some computational problems are known to be:

- **Unsolvable** (e.g., the Halting Problem and many others)
- **So difficult** that one cannot expect to have an algorithm that allows waiting for them to finish on arbitrary inputs (e.g., NP-hard problems)



Alan designed the perfect computer



UNIVERSITY OF TWENTE.

8

8



Foundations of Computer science

- Aims to **study characteristics of algorithms**, *independent of used hardware and programming language.*
- The **Turing Machine** as a universally accepted abstract model of a computer (other useful models do exist)

Important characteristics:

- Will the algorithm **terminate** with output?
- Does the algorithm always give a **correct** result?
- Is the algorithm **efficient** in use of time and memory?



UNIVERSITY OF TWENTE.

9

9



Complexity

Algorithmic complexity has two orthogonal forms:

- **Time complexity:** *how much running time* will the algorithm require to come to a solution?
- **Data complexity:** *how much memory space* does the algorithm need to come to a solution?

Machine specs are in the way ...

We are *not* after actual numbers, but rather after formulas for these that are expressed in terms of “*the size of the problem.*”

Often data complexity can be compromised in favour of time complexity, and *vice versa*.



UNIVERSITY OF TWENTE.

10

10



The size of the problem

Size of the problem: that/those characteristic number(s) that summarize how “big” the input is.



Examples:

- To sort a deck of cards: *the number N of cards*
- To traverse a graph: *the number N of graph vertices and also the number M of graph edges*
- To determine whether K is a prime number: *K itself, or the number N of bytes needed to encode K (very large numbers need more bytes)*
- To classify a satellite image: *the number N of pixel rows and number M of pixel columns*
- To translate a letter: *the number N of characters in it.*



UNIVERSITY OF TWENTE.

11

11



Complexity formulas

We may capture:

- The *worst case* (gives upper bound, but this may be really rare!)
- The *average case*
- (The *best case*, but this is not usually of much interest. You want to know what happens on a rainy day.)

Big-O notation is used to capture a general impression of how the algorithm behaves. It captures a notion of *upper bound*. And is especially important when the problem grows in size.

(There is also Big- Ω notation which captures a notion of lower bound.)



UNIVERSITY OF TWENTE.

12

12



Complexity formulas

Big-O notation

Algorithm has **$O(n)$ time complexity**: the running time grows linearly with n (i.e., with the size of the problem)

Algorithm has **$O(n^2)$ time complexity**: the running time grows quadratically with n .

Can also be expressed as an $O(n,m)$ formula for problems with two size parameters. And so forth.

The **complexity class P**: algorithms for which the time complexity is expressed as a polynomial in the size of the problem. Otherwise, i.e., if worse: the algorithm has **exponential** complexity.



UNIVERSITY OF TWENTE.

13

13



Time complexity growth

If your $O(f(n))$ algorithm solves the case for $n=10$ in one minute then you can expect to solve for $n=\dots$ in a day/year/century.

$f(n)$	Number of data items processed per:			
	1 minute	1 day	1 year	1 century
n	10	14,400	$5.26 \cdot 10^6$	$5.26 \cdot 10^8$
$n \log n$	10	3,997	883,895	$6.72 \cdot 10^7$
$n^{1.5}$	10	1,275	65,128	$1.40 \cdot 10^6$
n^2	10	379	7,252	72,522
n^3	10	112	807	3,746
2^n	10	20	29	35



UNIVERSITY OF TWENTE.

14

14



Common complexity formulas

- And how they behave ...

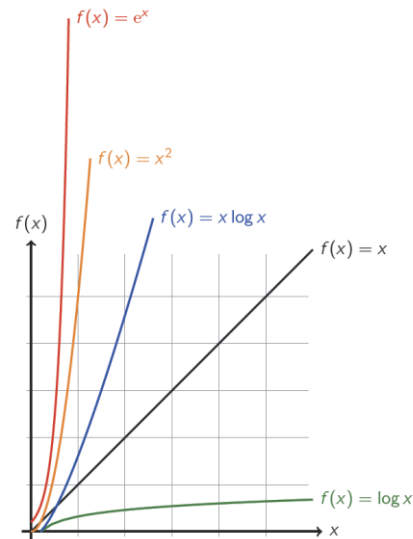
- When is the complexity $f(n)$ of an algorithm in the class of $g(n)$?

$$f(n) \in O(g(n)) \Leftrightarrow$$

$$\exists C > 0, \forall k \in \mathbb{N} \mid f(k) < C \cdot g(k)$$



UNIVERSITY OF TWENTE.



15

15



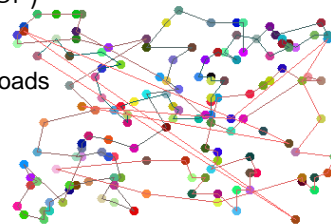
The Travelling Salesman problem (TSP)

Assume we have N cities, and assume direct roads exist between each pair of cities with a known travel distance.

Suppose you must:

- Start in one city and also end there
- Visit every other than the start city exactly once

Problem: what is the shortest route that achieves this?



This problem is known to be “**NP-hard**,” which means that

- It is amongst the hardest problems that have a polynomial-time solution, or worse even:
- It has *no* polynomial-time solution even.

It is unknown which of the two is the case!



UNIVERSITY OF TWENTE.

16

16

Algorithms existed way before computers

- Εὐκλείδης (Euclid) of Alexandria (323-283 BC) devised an algorithm to determine the *greatest common divisor* GCD of two natural numbers.
- Remember: $\text{gcd}(25,15)=5$.

Euclid's algorithm in functional notation:

$\text{gcd} :: \text{int} \rightarrow \text{int} \rightarrow \text{int}$

$\text{gcd}(n, 0) \mid n > 0 = n$

$\text{gcd}(n, m) \mid m > n = \text{gcd}(m, n)$

$\text{gcd}(n, m) \mid \text{otherwise} = \text{gcd}(n \bmod m, m)$



So, $\text{gcd}(52,46)=\text{gcd}(46,6)=\text{gcd}(6,4)=\text{gcd}(4,2)=2$

UNIVERSITY OF TWENTE.

17

17

Simpler algorithms

- **Given** a collection C of natural numbers, find the biggest one.
- **Assume** A is addressable as a list: A[i] is the i-th number in A, and A has n elements.

Pseudocode algorithm (in two coding styles):

imperative style

```

input A[n] : [int]
declare foundmax:int
foundmax = A[1]
for i = 2 to n:
    if A[i] > foundmax
    then foundmax = A[i]
output foundmax
  
```

$\text{findmax} :: [\text{int}] \rightarrow \text{int}$

$\text{findmax } [e] = e$

$\text{findmax } (\text{hd}: \text{tl})$

$\mid \text{hd} > \text{findmax } \text{tl} = \text{hd}$

$\mid \text{otherwise} = \text{findmax } \text{tl}$

functional style

$O(n)$ time complexity



UNIVERSITY OF TWENTE.

18

18

bubble sort

Sorting: Specifically,

Sorting is a *classical* CS study problem.

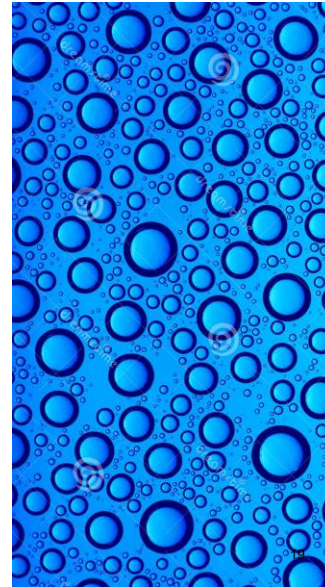
Bubble sort intuition:

- Go from left to right in the list and swap any pair that is in wrong order
- Continue doing this until you have not swapped anything anymore
- After step n you no longer need to look at last n items because they are already sorted.

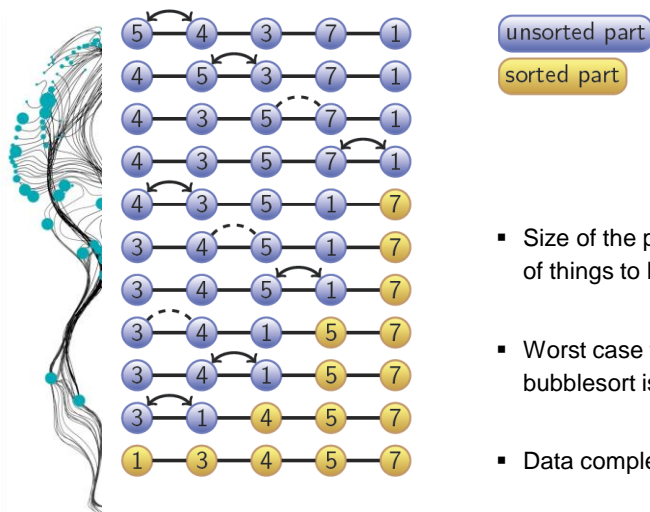
Demo: <https://visualgo.net/bn/sorting>



UNIVERSITY OF TWENTE.



19



- Size of the problem is the number n of things to be sorted.
- Worst case time complexity of bubblesort is $O(n^2)$
- Data complexity is $O(n)$

For later: Try write out bubblesort in pseudocode or even in Python



UNIVERSITY OF TWENTE.

20

20

bigocheatsheet.com

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	$O(n \log(n))$	$O(n \log(n))$	$O(n^2)$	$O(\log(n))$
Mergesort	$O(n \log(n))$	$O(n \log(n))$	$O(n \log(n))$	$O(n)$
Timsort	$O(n)$	$O(n \log(n))$	$O(n \log(n))$	$O(n)$
Heapsort	$O(n \log(n))$	$O(n \log(n))$	$O(n \log(n))$	$O(1)$
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$
Shell Sort	$O(n)$	$O((n \log(n))^2)$	$O((n \log(n))^2)$	$O(1)$
Bucket Sort	$O(n+k)$	$O(n+k)$	$O(n^2)$	$O(n)$
Radix Sort	$O(nk)$	$O(nk)$	$O(nk)$	$O(n+k)$



UNIVERSITY OF TWENTE.

21

21

Algorithmic strategies

Divide-and-conquer To solve a problem it is divided in two (or even more) similar-but-smaller, non-overlapping problems, which are separately solved (in the same way) and then their solutions are merged into an overall solution.

Dynamic programming To solve a problem faster, intermediate results are remembered when they are deemed to be reusable within the algorithm later.

Heuristic algorithms Especially applied to hard problems, apply rules of thumb that are not proven to give the best solution, but that will typically give a rapid result that is acceptable.



UNIVERSITY OF TWENTE.

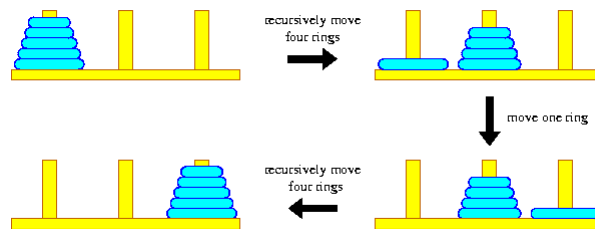
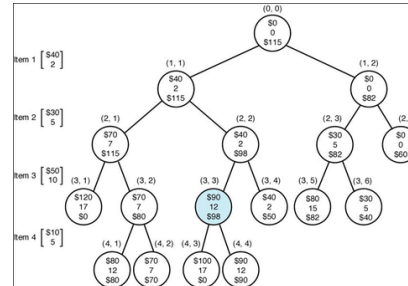
22

22

Algorithmic strategies

For study later:

- **Brute-force**
- **Branch-and-bound**
- **Recursive**
- **Backtracking**



UNIVERSITY OF TWENTE.

23

23

Algorithmic strategies

- **Greedy algorithm** A heuristic algorithm that always makes the locally optimal choice *in the hope* that that leads to a global optimal solution eventually.
- The *greedy traveling salesman solution* applies the heuristic that the next city to be visited is the closest that has not yet been visited.



UNIVERSITY OF TWENTE.

24

24