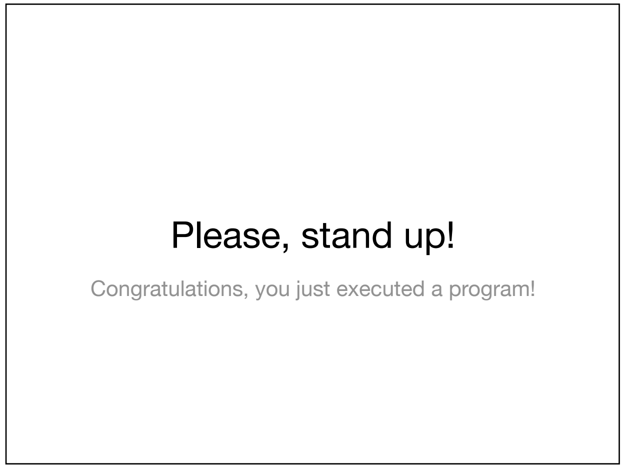
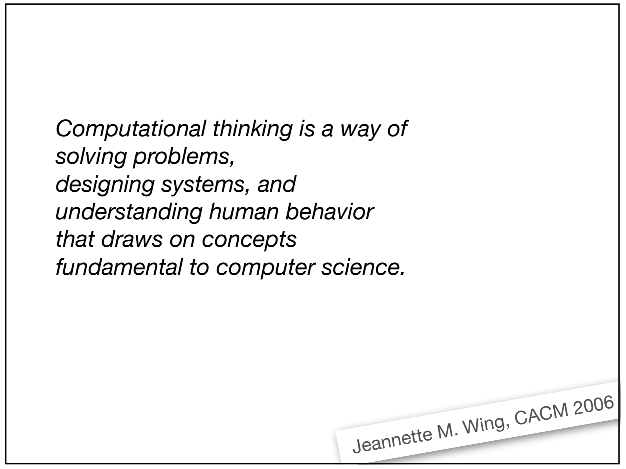
Computational thinking

www.tudorgirba.com



Please, stand up!

Congratulations, you just executed a program!

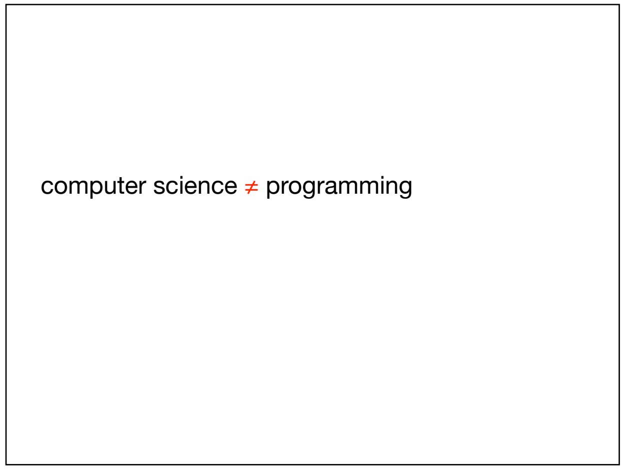


Computational thinking is a way of
solving problems,
designing systems, and
understanding human behavior
that draws on concepts
fundamental to computer science.

Jeannette M. Wing, CACM 2006

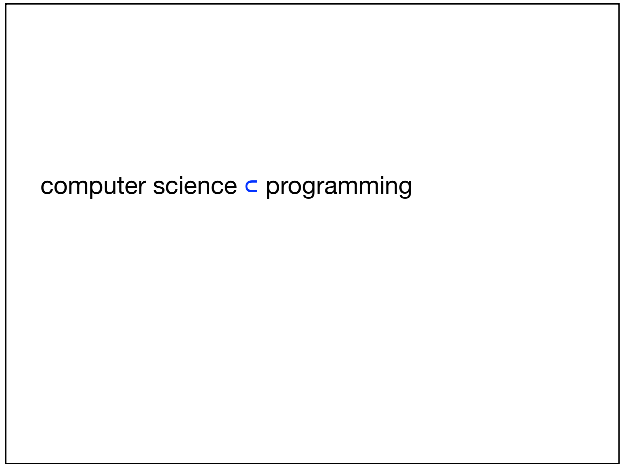
In German: “Rechenbetontes Denken” or “Berechnungsbetontes Denken”

“Computational thinking will be a fundamental skill used by everyone in the world by the middle of the 21st Century.” - Jeannette M. Wing



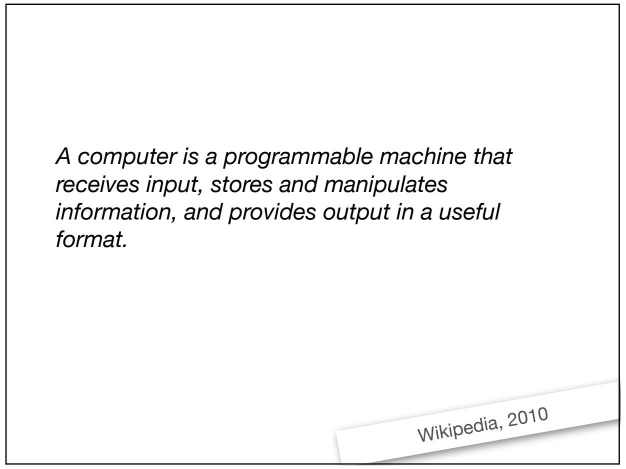
computer science $\neq$ programming

The common take on computer science is to make it a synonym to software programming or hacking. Computer science is hardly that.



computer science $\subset$ programming

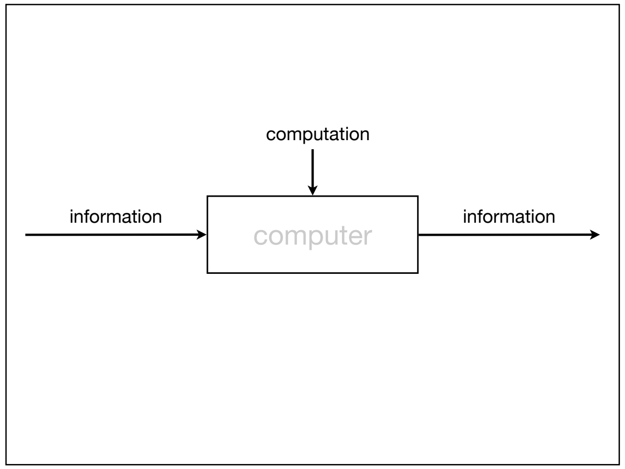
Some mastery of technology is of course important, but it is not the whole, any more than economics is just about arithmetic or bookkeeping.



A computer is a programmable machine that receives input, stores and manipulates information, and provides output in a useful format.

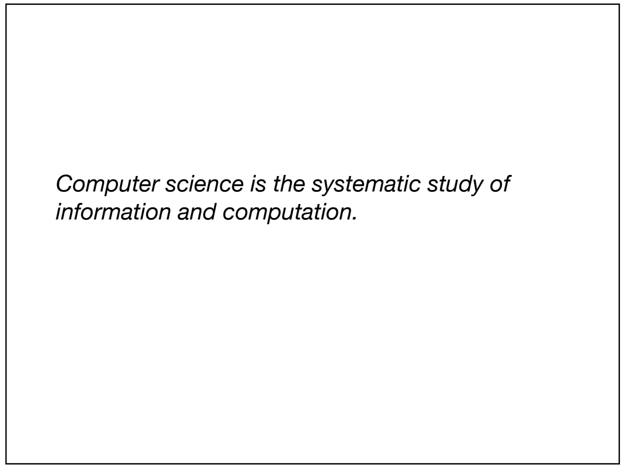
Wikipedia, 2010

Before going forward, we need to step back and ask what a computer is. A computer:
- is a machine of some sort
- computes programs
- manipulates information



While a computer is a machine, computer science is not about computers.

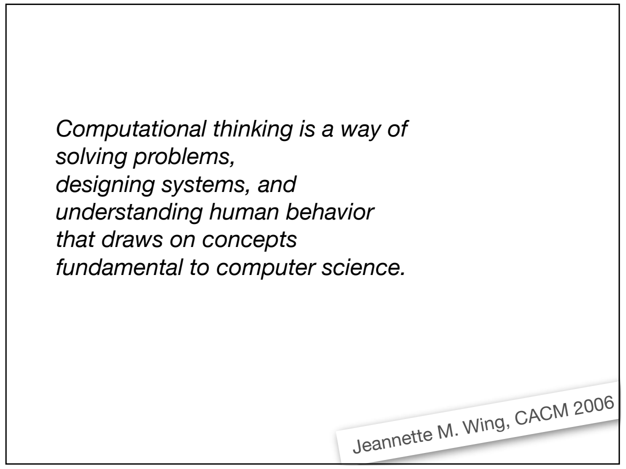
“Computer science is no more about computers than astronomy is about telescopes.” Edsger Dijkstra



Computer science is the systematic study of information and computation.

Computer Science also requires many different kinds of skills:
abstraction, communication, reading, math, modeling, planning

Users (or Stakeholders) have real problems in real or virtual domains.
A computer scientists develops solutions with the help of technology.
Mastery of technology is important, but just as important is the facility to understand the real-world domains and to map to technological domains.



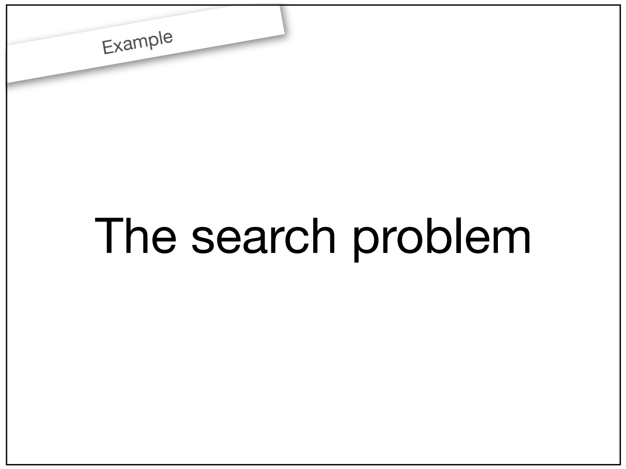
*Computational thinking is a way of
solving problems,
designing systems, and
understanding human behavior
that draws on concepts
fundamental to computer science.*

Jeannette M. Wing, CACM 2006

In German: “Rechenbetontes Denken” or “Berechnungsbetontes Denken”

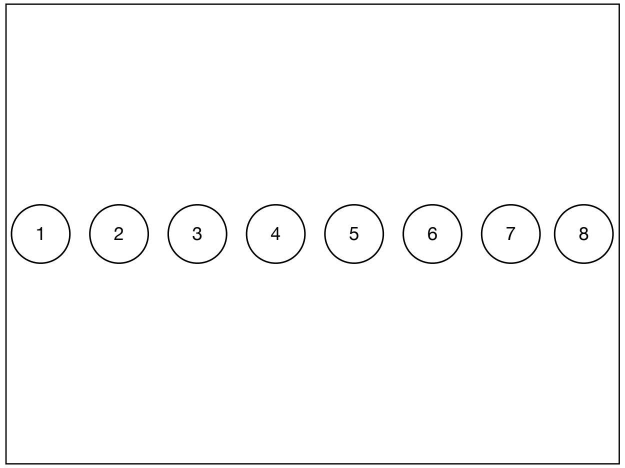
“Computational thinking will be a fundamental skill used by everyone in the world by the middle of the 21st Century.” - Jeannette M. Wing

“Computing is not about computers any more. It is about living.” Nicholas Negroponte


Example

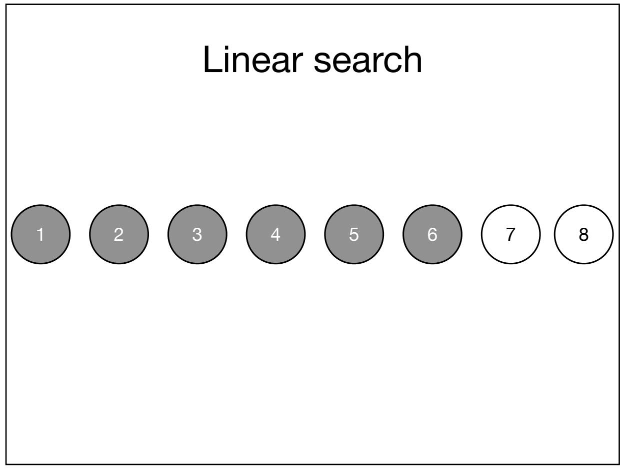
The search problem

Abstractions allow us to deal with complexity by stripping away needless detail. The first step to take is to describe the problem in abstract terms. An example of such an abstracted problem is the search problem.



For example, how to find an item within a given list? There can be multiple situations when we need to solve this problem:
- find a name in a phone book
- find a book in a library

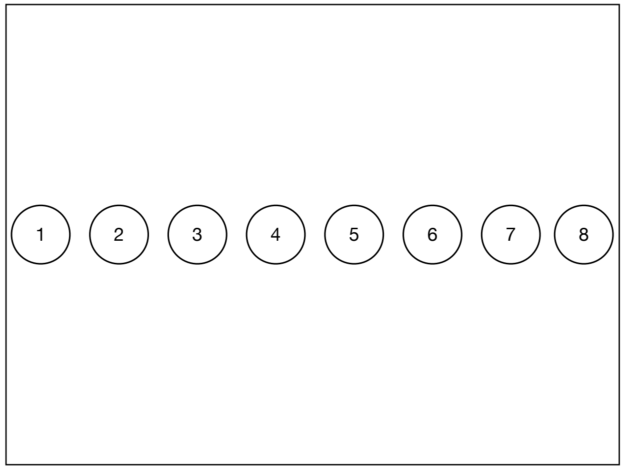
All these problems can be abstracted away into a simpler representation. For example we can formulate the problem like: how to find a certain number (=item) out of a given set of numbers (= different items)?


Linear search

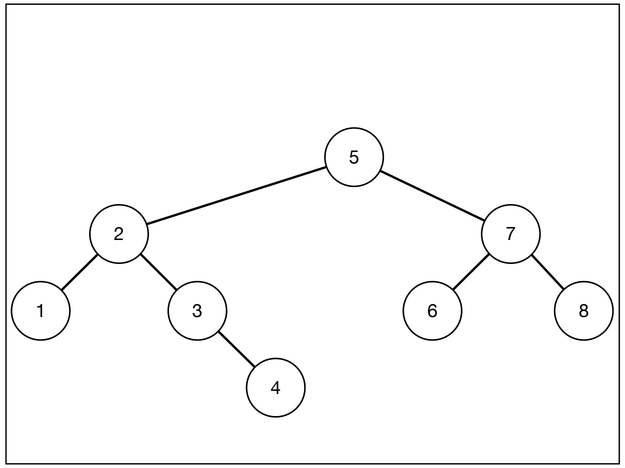
The simplest way to approach the problem would be to check all the given numbers one by one until we find the one we are looking for.

For example, when looking for number 6 out of 8 numbers, we can take them one by one starting from the left and make our way to the right.

This is called a linear search.

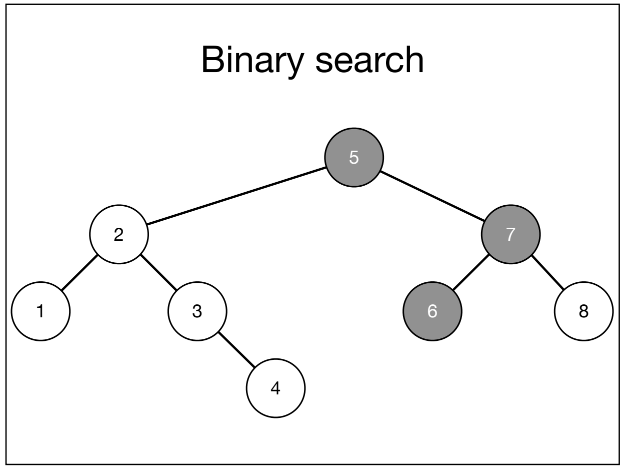


But, how else could we approach the problem?



If we assume that there is a certain order in the number arrangement we can design a different algorithm:
- given a list of numbers check the number in the middle and see if the number we find is equal, less or greater than the one we look for
- if it's equal we stop, if it's less we repeat the same step for the numbers on the left, if it's greater we repeat for the numbers on the right

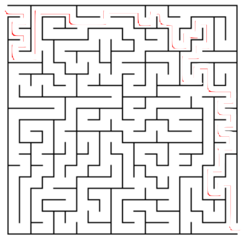
To make the search easier, we can represent the data as a binary tree. In this case, each number will split the structure: those smaller are on the left subtree, those larger are on the right subtree.



Binary search

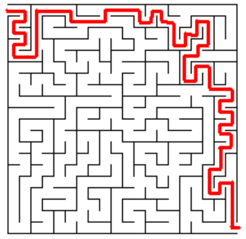
This approach is called the binary search, which is faster than the linear search.

This is an example of how representing the information in a different way leads to a better overall solution.



Finding your way is another classic problem that we can find when:
- searching through a maze;
- looking for something you lost (keys, glasses)
- trying out winning strategies.

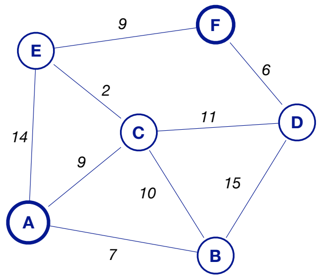
Backtracking



This problem can be solved with backtracking.
Backtracking requires you to keep track of every decision you make.
If you discover you made an error, you backtrack to the last decision you make and change it.
Essentially you are searching depth-first through a tree of possibilities.

A Stack is a useful data structure for keeping track of progress, where the last decision is always on top of the stack.
The stack is one of the most basic programming abstractions.

http://video.google.com/videoplay?docid=5487963406635625680#



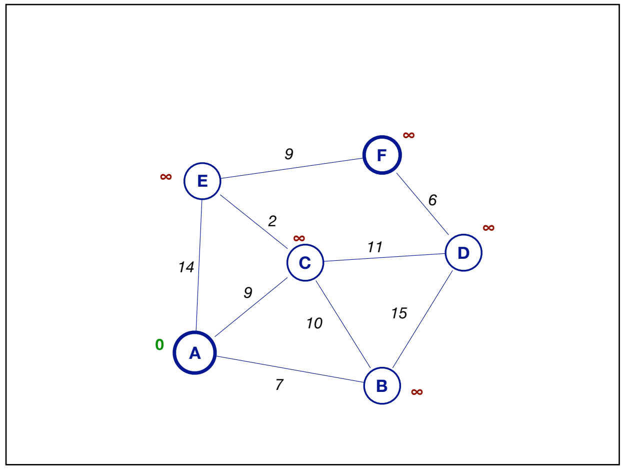
http://scg.unibe.ch/download/lectures/ei/01ComputationalThinking.pptx

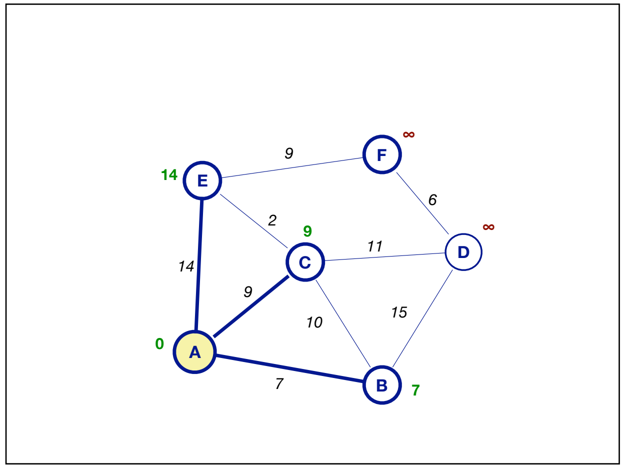
A similar problem is to find the shortest path in a space of connected places. Such a space can be represented as a graph in which each two nodes are connected via edges that have different weights or costs. For example, such a problem can help us find:
- the cheapest route between two places
- the fastest route between two places

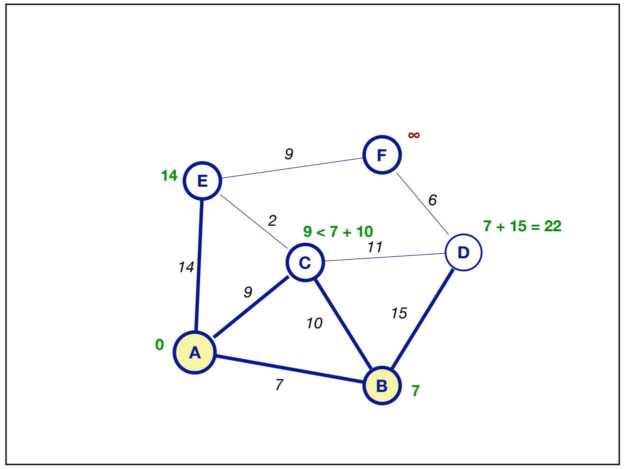
One possibility would be to take the backtracking approach, try all possibilities and then pick the shortest. However, that is not necessarily optimal.

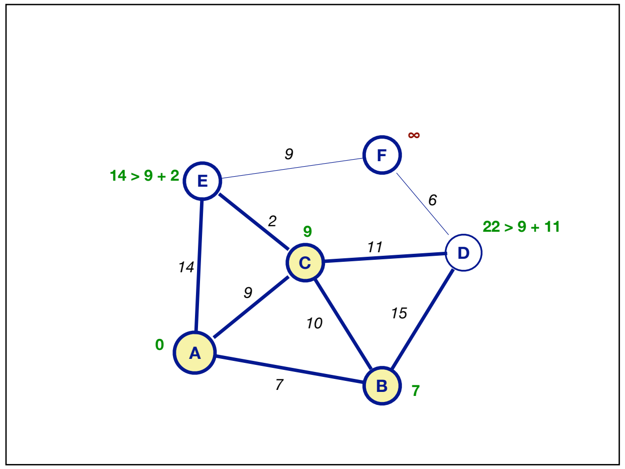
Example taken from:
http://scg.unibe.ch/download/lectures/ei/01ComputationalThinking.pptx

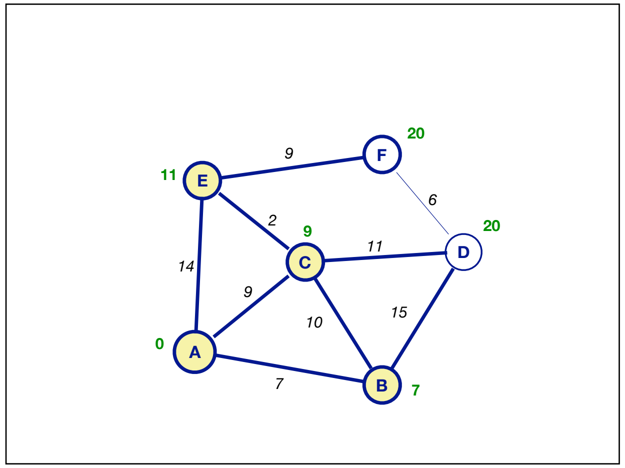
Another way is to start from the start node and then traverse each reachable nodes from that node to compute intermediary costs. At each step we check if the temporary cost in the target node is less than the cost of reaching the current node plus the direct way. If it’s less, we replace the temporary cost with the new one.

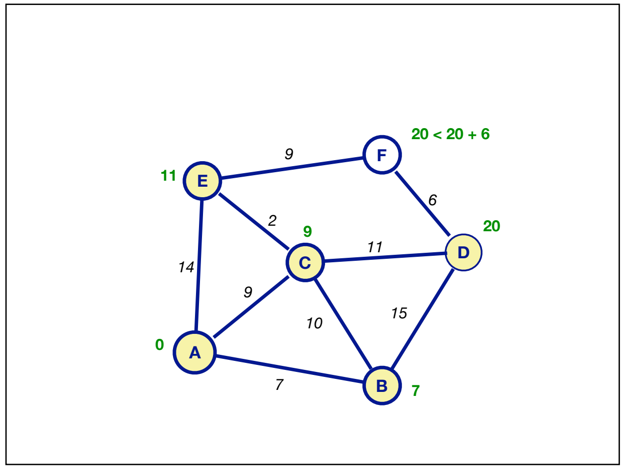




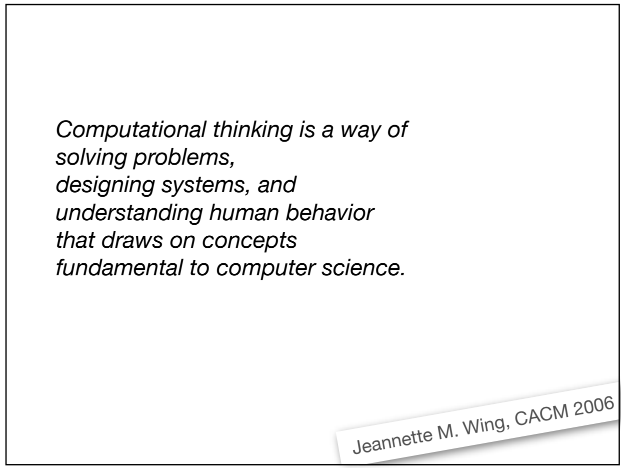








Approaching the problem with a different computation can lead to a different result.



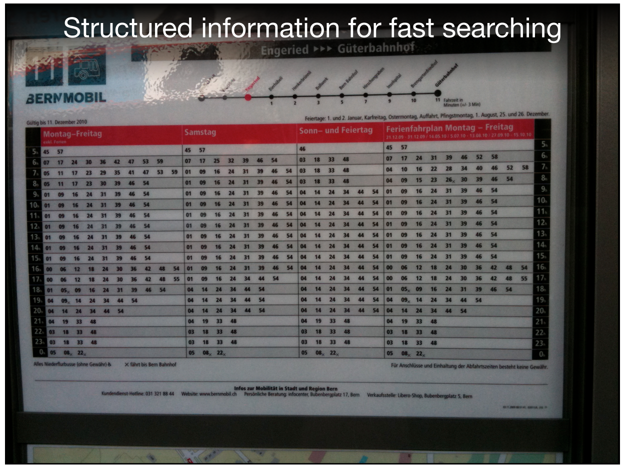
Computational thinking is a way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science.

Jeannette M. Wing, CACM 2006

In German: “Rechenbetontes Denken” or “Berechnungsbetontes Denken”

“Computational thinking will be a fundamental skill used by everyone in the world by the middle of the 21st Century.” - Jeannette M. Wing

Structured information for fast searching



The bus schedule is organized in a way that makes searching for the next bus fast.

Reading from a common resource

The board in the middle of the train station is a common resource that can be read from in the same time by everyone.

This is possible because one person reading from the board does not impede another one to read from the same board.

Abstractions allow us to deal with complexity by stripping away needless detail.
They also provide us with some nice properties. Queues ensure fairness.
But any organization that we deal with (like a bank) provides us with a simple interface and a much more complex implementation.
Abstractions only really exist in our heads. A queue, or even a bank, does not exist in reality, only by convention.
What exists is only a physical manifestation (a bunch of people in a line, a building) but that is not the abstraction.

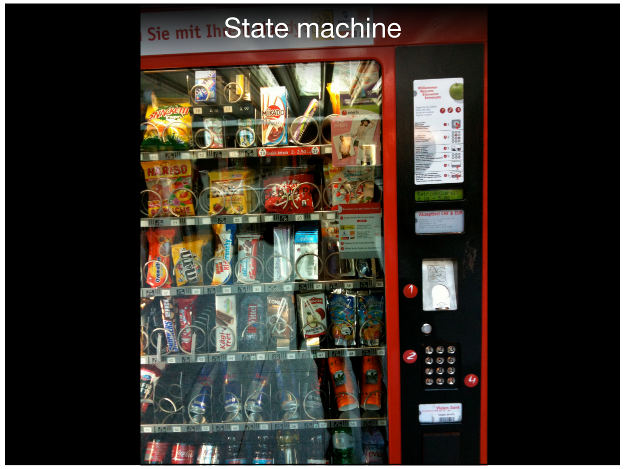
All computations make heavy use of abstraction to cope with complexity.

http://www.ac-nancy-metz.fr/enseign/anglais/Henry/bus-queue.jpg

When we can identify clusters of distinct concerns we can enhance our system to serve multiple stakeholders at the same time. In this case, those that need fruits will most likely not need pastries, and thus we can form two parallel queues.

We see this in any organization. We can add more factory workers to increase production.
This works only if the tasks are truly independent. Sometimes, adding more members to a team may actually slow down work!

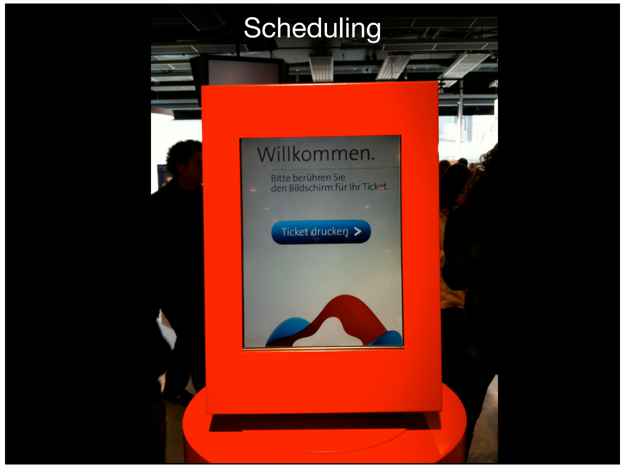
Computer servers work this way – a farm of servers can be scaled up to handle more queries.



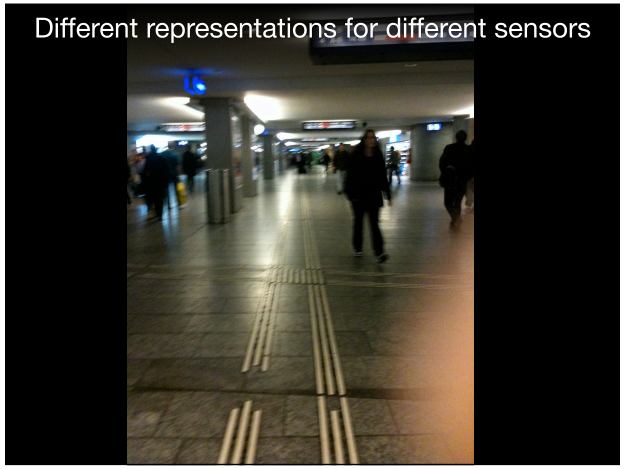
The instructions on the right describe the behavior of the state machine.

We are used to manipulating instructions as data.
We give or receive instructions separately from acting on them.
Instructions can be translated, analyzed, formatted, and processed before being performed.

Computers must also process programs in many stages before they can be executed.
Even running programs may be analyzed and optimized on-the-fly.

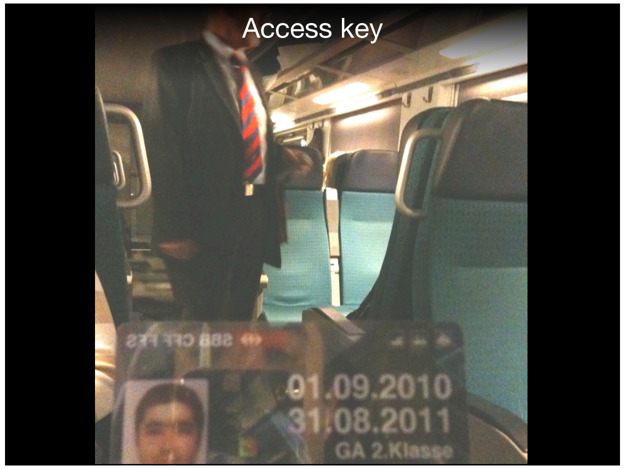


This machine provides a different implementation of a queue, but besides that it also enables better scheduling of resources by tracking the average time it takes from the moment of registration until the moment of serving. If the time goes above a certain threshold, another clerk can be added to the serving team.

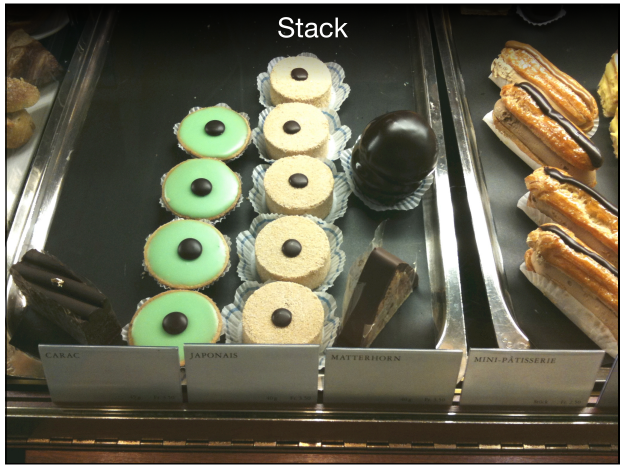


While most of the time we will only notice the visual signs, the tracking lines from the ground are useful for people that function without relying on visual sensors.

The same information can be represented in multiple ways, each way being useful for a different machinery with a pertaining a different kind of computation.



Keys enables us to check whether the checked party is allowed to access the resources in question.



Stack is similar to a queue in that it also contains a list of elements. However, as opposed to a queue, a stack works by the "last in / first out" principle.

Translated into the situation from the picture, the cakes at the bottom stayed the longest in display.

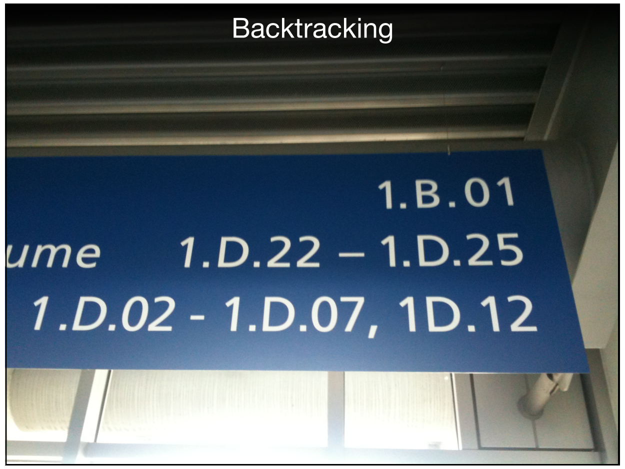
All cars, trams, busses, and pedestrians want access to the same road. For all of them to find their way we need to ensure that they both do not crash into each other and do not wait forever.

Concurrency problems can basically be divided into safety problems (nothing bad should happen) and liveness (something good should happen. Traffic accidents are bad. Progress is good. (Traffic jams are safe but not live.)

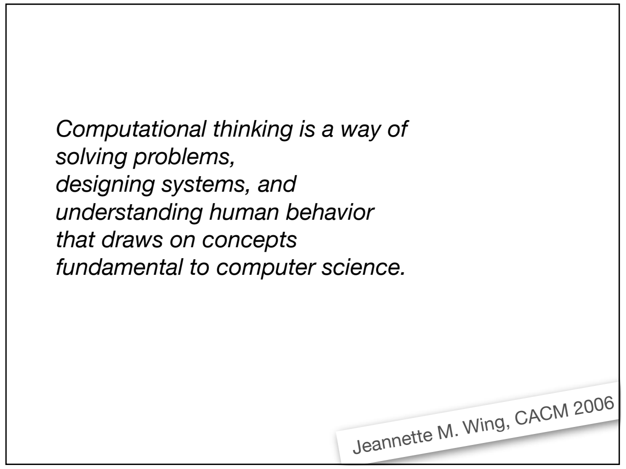
Traffic lights and roundabouts, together with proper rules and scheduling, can yield good solutions to managing concurrent traffic. Bad designs can create traffic jams or even accidents.

Safety and liveness problems arise because of need for access to shared resources (roads, intersections …).

Backtracking



We use backtracking all the time, even if we sometimes take shortcuts.

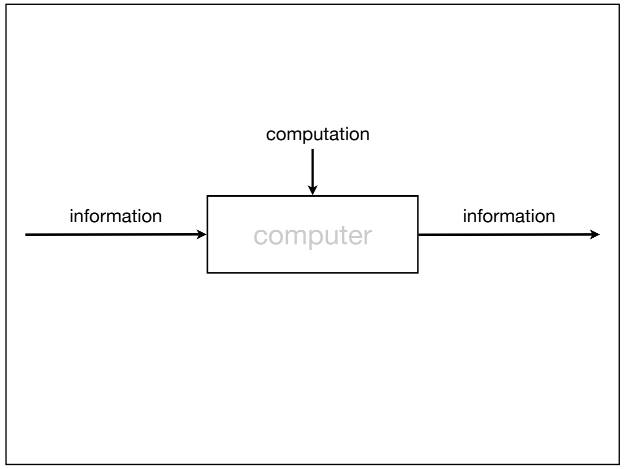


Computational thinking is a way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science.

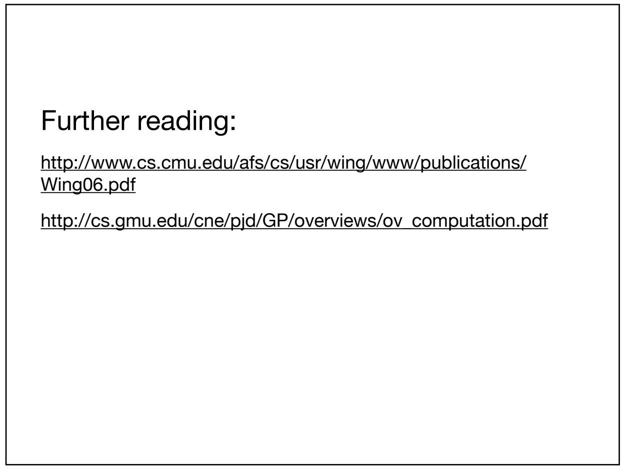
Jeannette M. Wing, CACM 2006

In German: “Rechenbetontes Denken” or “Berechnungsbetontes Denken”

“Computational thinking will be a fundamental skill used by everyone in the world by the middle of the 21st Century.” - Jeannette M. Wing



Computer science is not about computers, it is about computation and information.

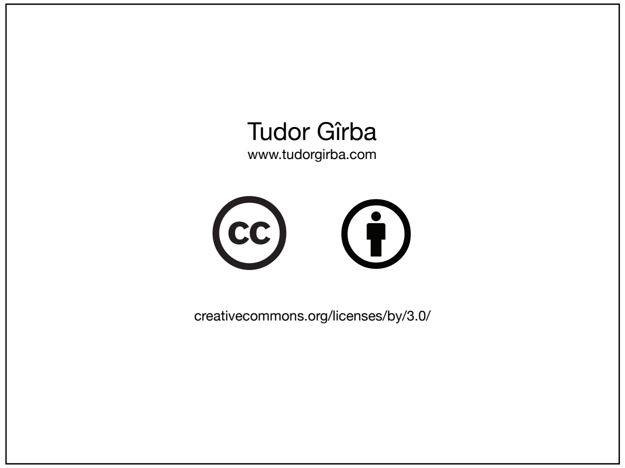


Further reading:

http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf

http://cs.gmu.edu/cne/pjd/GP/overviews/ov_computation.pdf

Some examples taken from:
http://scg.unibe.ch/download/lectures/ei/01ComputationalThinking.pptx



Tudor Gîrba
www.tudorgirba.com

creativecommons.org/licenses/by/3.0/