北京航空航天大學
BEIHANG UNIVERSITY

Introduction to Computer Science and Programming

Lecture 1

Sebastian Wandelt

Beihang University

Outline

- Introduction
  - About you and us
  - Administrative issues
  - What is Computer Science?
  - A (very) brief history of Computing
  - Outline of this Course
  - Algorithms and Programming Languages

About you and us

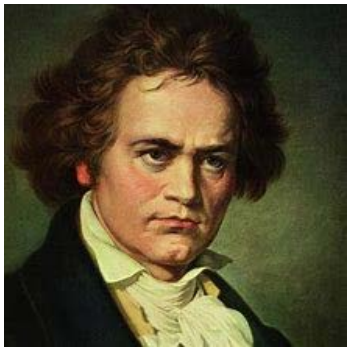
About you?

- What is your future career goal?
- How do you use computer technology?
- What do you expect from attending this course?
- What else do you want everybody to know?

Whenever you see these three, you should (wake up and) participate! ☺

About us (Sebastian Wandelt)

- Name: Sebastian Wandelt
- Chinese Name: 小塞
- Nationality: German



About us (Sebastian Wandelt)

- 2001 - 2004: Berlin School of Economics and Law
  Degree: **B.Sc. in Computer Science**
- 2004 - 2006: Dresden University of Technology
  Degree: **M.Sc. in Computer Science**
- 2006 - 2011: Hamburg University of Technology
  Degree: **Ph.D. in Computer Science**
  Research/Teaching on: Logic, Databases, Software Engineering
- 2011 - 2016: Humboldt-University of Berlin
  Position: **Postdoc**
  Research/Teaching on: Algorithms, Databases, Information fusion
- Since March 2016: Beihang University
  Position: **Professor**, **青年千人计划**
- (Since September 2017: Teaching CS@SGE)

About us (Teaching assistants)

Name: Louis
Chinese Name: **林威**
louislin@buaa.edu.cn

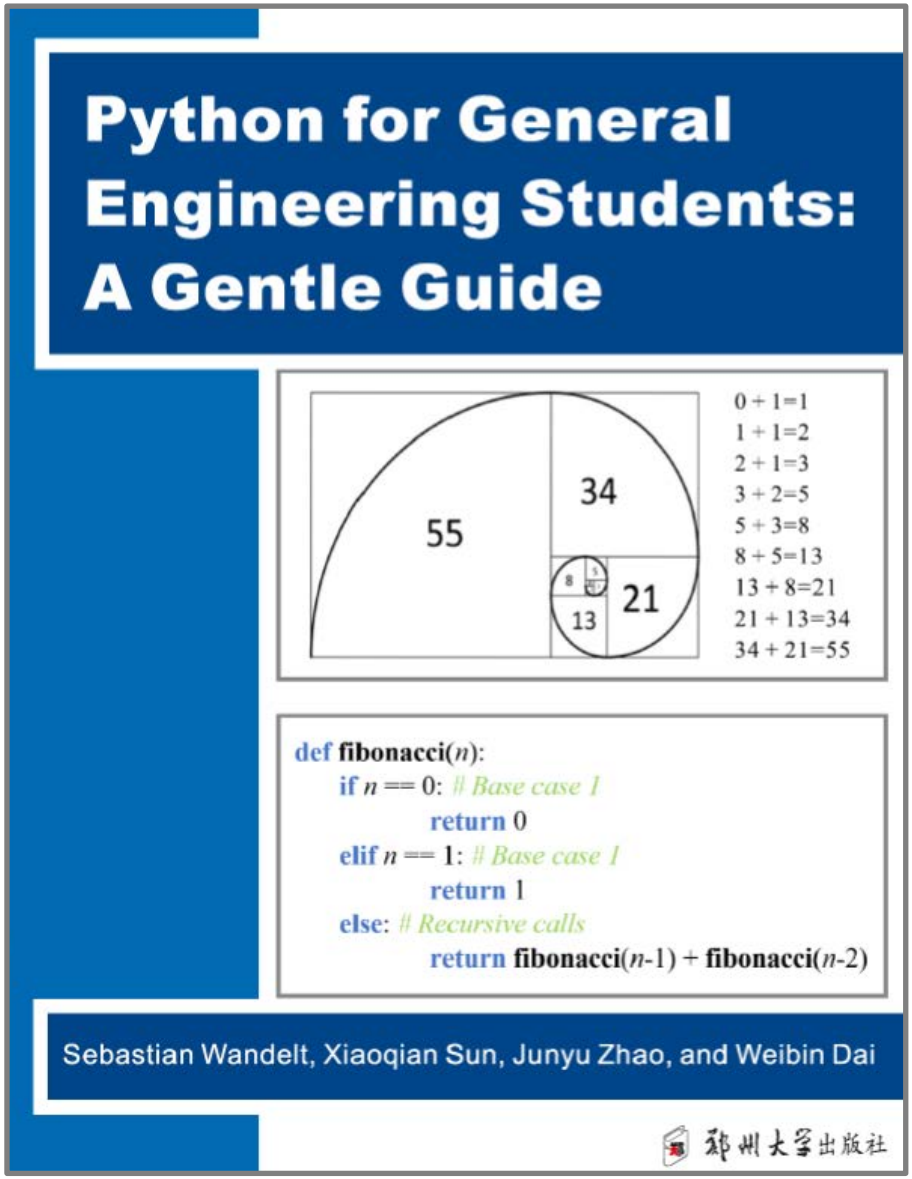
Name: Donovan
Chinese Name: **丁屹达**
19351040@buaa.edu.cn

While both TAs are just one year older than you, both of them are perfectly qualified to be in charge of this course on Computer Science.
Treat them with respect, they deserve it!

Administrative issues

Course material

- At one point, we decided to write a book which covers all essential topics of the lecture.
- This is the major reference, and the lectures are roughly in the same order as the chapters in the book.



Administrative

- Lectures:
  - Every Monday 8:00-9:35AM
- Lab classes:
  - Every Tuesday 7:00-8:35PM or 8:40-10:15PM
  - Usually some mix of paper/pencil exercises and laptop work
  - Bring your laptop (and recharger)!
- Examination type:
  - Written examination
  - Closed-book (=no material allowed)
- Final score:
  - Based on performance in lecture, lab classes, homwork and final examination

Homework

- Homework:
  - Usually one or two tasks
  - It will be uploaded to the website together with the lecture pdf
- Please hand in the Python file to the email address icsp2020@126.com **before** the lab class on Tuesday.
  - The subject of the email and the name of the python file should both be HWX_StudentID_Name,
  - For instance: HW1_20351001_小明
- The first homework will be assigned after Lecture 2.

Administrative

- Course web page:
  - http://m3nets.de/teaching/SGE2020/

Advices

- The course starts easy going, but becomes quite involved after the first four weeks.
- Reading the course book is strongly encouraged. It is not required to read book chapters before the lecture, but you should use the book as reference for keeping the pieces together and also for learning the right (English) **terminology**. We wrote this book for a reason!

About you (again)

- How many of you have a laptop?
- How many of you have used a programming language before?
- Have you heard of C/C++? Did you use it?
- Have you heard of Python? Did you use it?

What is CS?

TASK!

- Please describe what is Computer Science in your opinion!

Whenever you see these three, you should (wake up and) participate! ☺

What is computer science (CS)?

Some misconceptions.

- **Misconception 1**:
  - I am good with Windows, and can surf the net with ease, so I really know CS very well.
- **Misconception 2**:
  - Last night I did 20 head shots in Counter Strike, so I know CS.
- **Misconception 3**:
  - Computer science is the study of how to write computer programs only, but nothing else.
- **Misconception 4**:
  - Computer science is the study of the uses and applications of computers and software only.

What is computer science (CS)?

Some misconceptions.

- **Misconception 1**:
  - ~~I am good with Windows, and can surf the net with ease, so I really know CS very well.~~
- **Misconception 2**:
  - ~~Last night I did 20 head shots in Counter Strike, so I know CS.~~
- **Misconception 3**:
  - Computer science is the study of how to write computer programs ~~only, but nothing else~~.
- **Misconception 4**:
  - Computer science is the study of the uses and applications of computers and software ~~only~~.

What is computer science?

Computer Science,
= Telescope science ?

"Computer science is no more about computers than astronomy is about telescopes"

-- Edsger W. Dijkstra

"The computer is not our object of study, It's our observational instrument"

What is computer science?

- **Computer science is mainly about the study of algorithms, including:**
  - Their formal and mathematical properties
  - Their linguistic realizations
  - Their applications to real problems

TASK!

- Please describe what is **Programming** in your opinion!

Terminology (=some important keywords)

- **Algorithm:** A set of steps that defines how a task is performed
- **Program:** One representation of an algorithm
- **Programming:** The process of developing a program
- **Software:** Programs and algorithms
- **Hardware:** Physical equipment

Algorithms

- Informally, an algorithm is ...

A well-defined computational procedure that takes some values as ***input*** and produces some values as ***output***.

A sequence of computational steps that transform the ***input*** into ***output***.

A simple example: Euclid‘s algorithm

Description: This algorithm assumes that its input consists of two positive integers and proceeds to compute the greatest common divisor of these two values.

Procedure:

Step 1. Assign M and N the value of the larger and smaller of the two input values, respectively.

Step 2. Divide M by N, and call the remainder R.

Step 3. If R is not 0, then assign M the value of N, assign N the value of R, and return to step 2; otherwise, the greatest common divisor is the value currently assigned to N.

Algorithms

- Some definitions ...

An algorithm is said to be ***correct*** if, for every input instance, it halts with the correct output.

A correct algorithm ***solves*** the given computational problem.

Focus will be on correct algorithms; incorrect algorithms can sometimes be useful.

Algorithm specification may be in English, as a computer program, even as a hardware design.

TASK!

- Find the maximum difference in age between any pair of students in the room!

Efficient algorithms / Hard problems

- Efficiency is a very important concept in CS
- Usual measure of efficiency is speed
  - How long does an algorithm take to produce its result?
  - Define formally measures of efficiency
- Problems exist that, in all probability, will take a long time to solve
  - Exponential complexity
  - NP-complete problems
- Problems exist that are even unsolvable

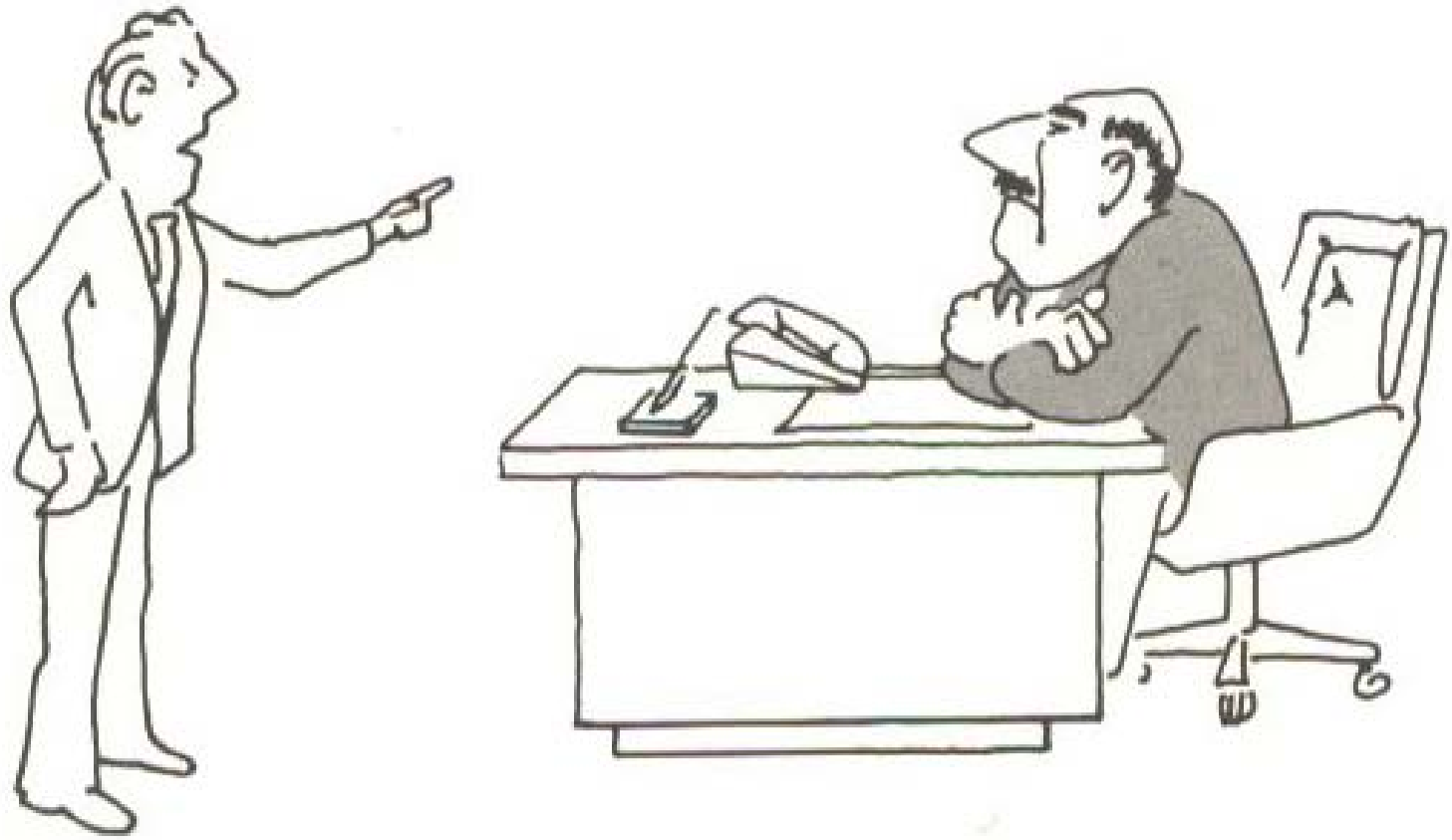
TASK (extended)!

- Find the maximum difference in age between any pair of students in the room, with minimum amount of interaction required.
  - Interaction: One person asking one question!

About algorithm properties

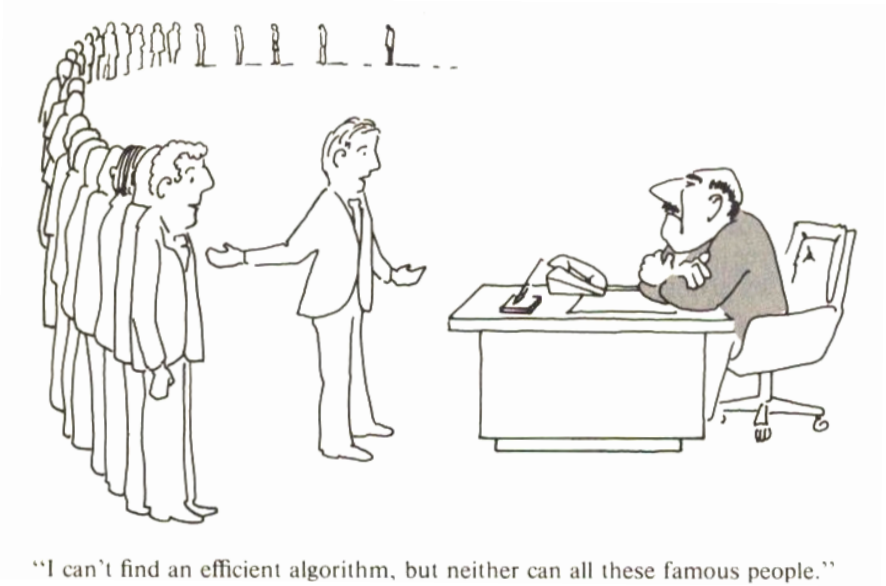
"I can't find an efficient algorithm, but neither can all these famous people."

About algorithm properties



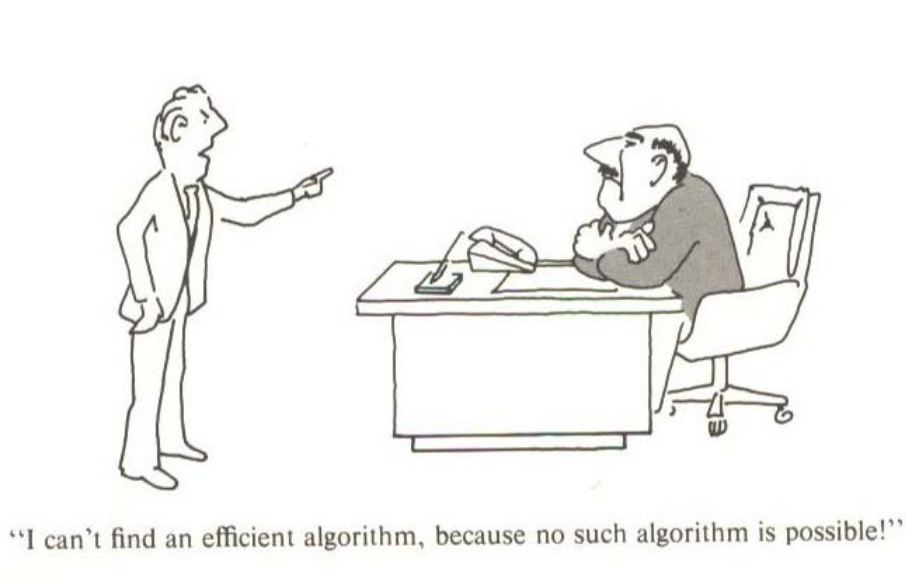
"I can't find an efficient algorithm, because no such algorithm is possible!"

About algorithm properties



"I can't find an efficient algorithm, but neither can all these famous people."

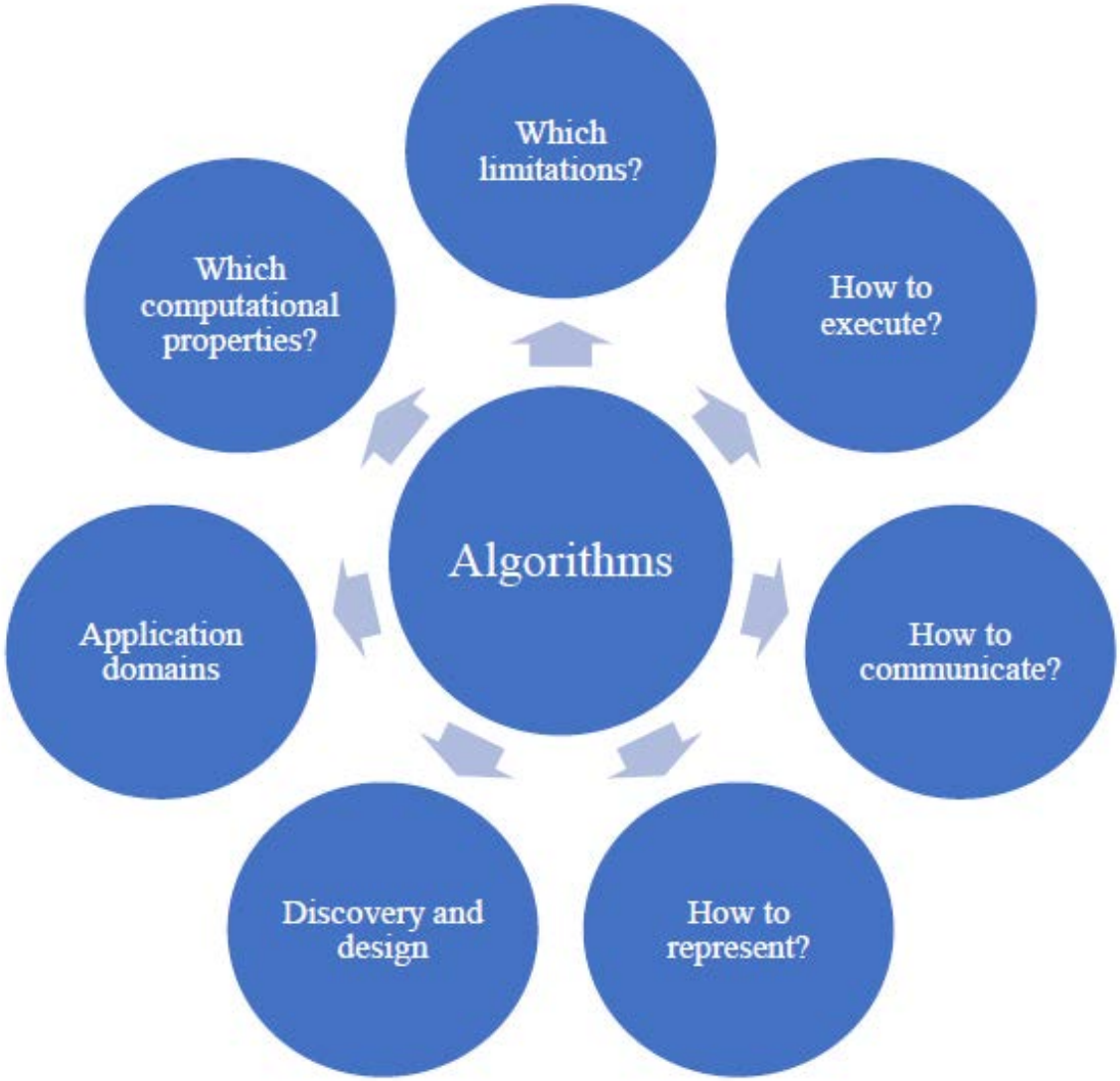
Pseudo science



"I can't find an efficient algorithm, because no such algorithm is possible!"

Computer Science

Summary: The central role of algorithms



A brief history of Computing

Question

- When (=time) do you think the history of Computer Science started?

The very roots of CS

The great thinkers of CS are:

- **Euclid, 300 BC**
- **Bhaskara, 6th century**
- **Al Khwarizmi, 9th century**
- **Fibonacci, 13th century**
- **Babbage, 19th century**
- **Turing, 20th century**
- **von Neumann, Knuth, Karp, Tarjan, …**

The roots of computing

- 1945: John von Neumann defines his architecture for an “automatic computing system”
  - Basis for architecture of modern computing
    - Computer accepts input
    - Processes data using a CPU
    - Stores data in memory
      - Stored program technique, storing instructions with data in memory
    - Produces output

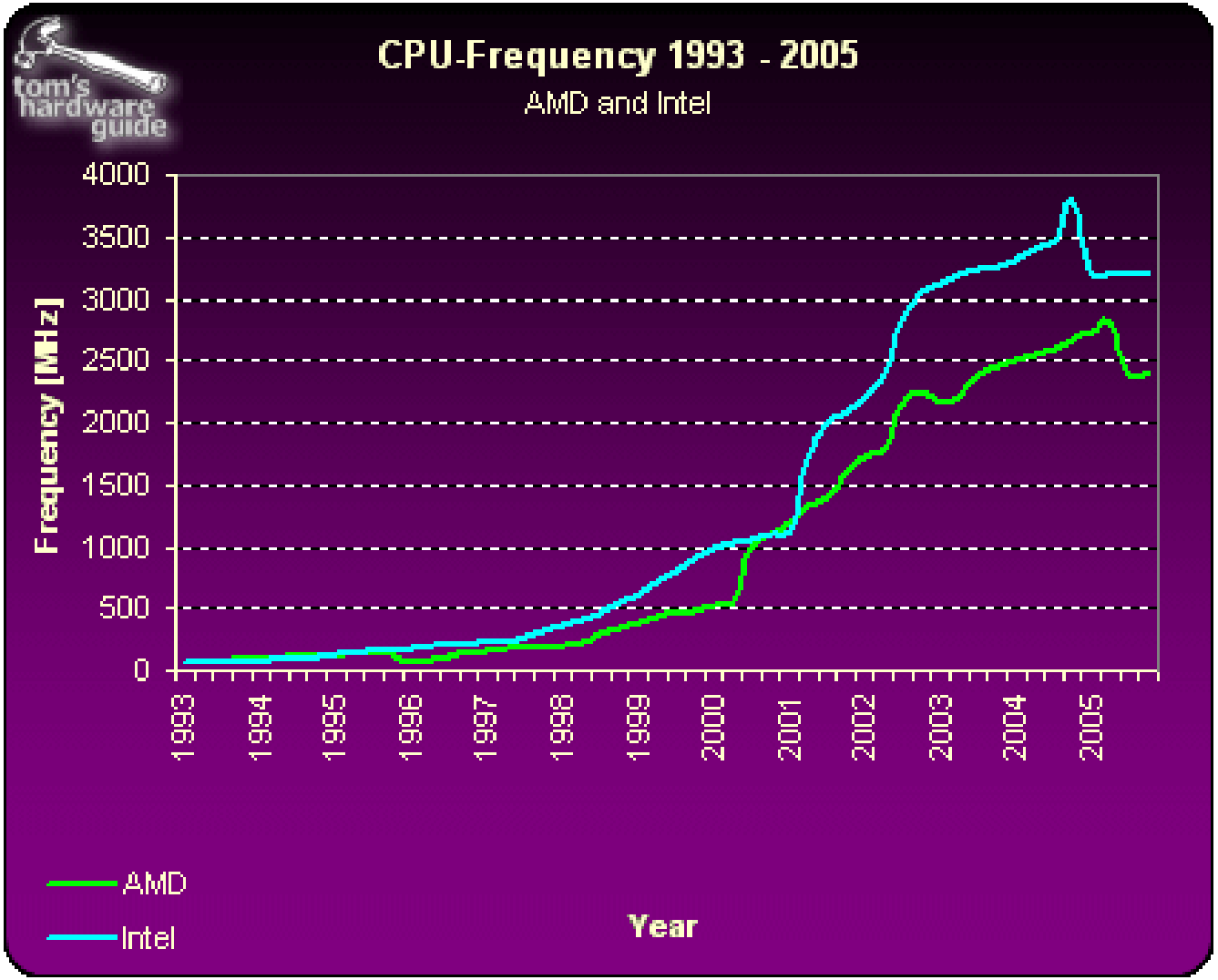
Personal Computer Era

- First microprocessor, Intel 4004 in 1971
- MITS Altair “kit” in 1975
- Apple in 1976
- IBM PC in 1981 using 8086
- Macintosh in 1984, introduced the GUI (Graphical User Interface) we still use today

Two fundamental trends with computers

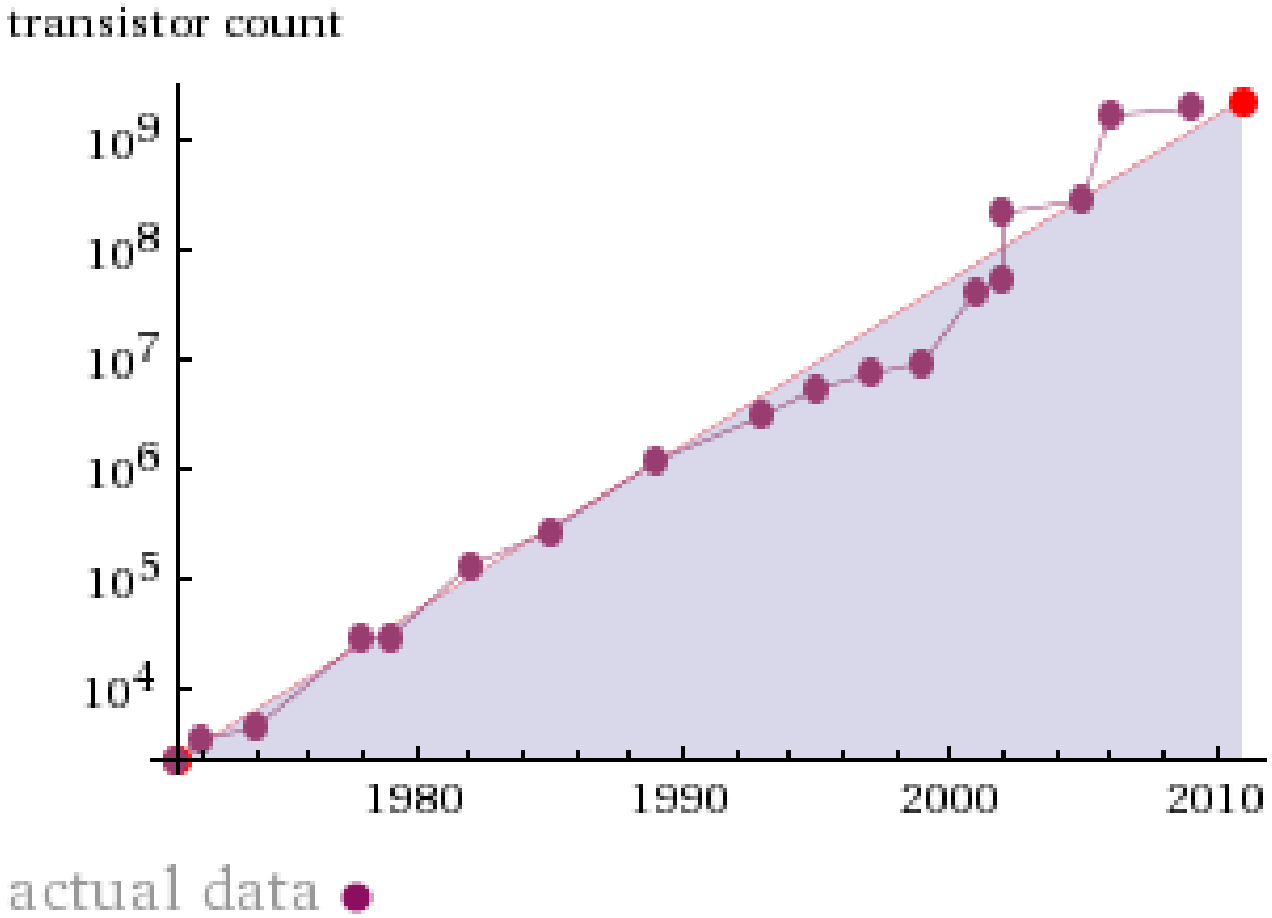
- Faster
- Smaller

CPU clock speeds

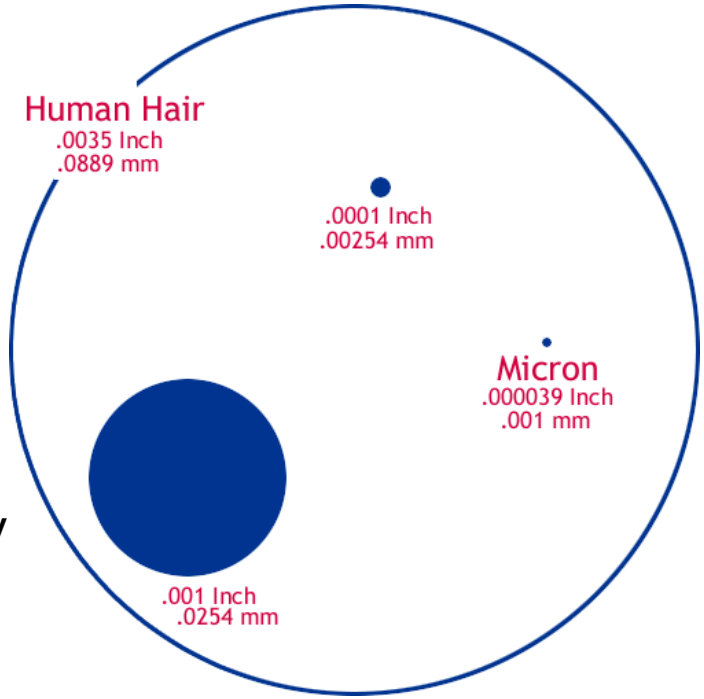


Moore‘s law

- 1965: Computing power doubles ~ every 18 months



Size matters?



- Human Hair: 100 microns wide
  - 1 micron is 1 millionth of a meter
- Bacterium: 5 microns
- Virus: 0.8 microns
- Early microprocessors: 10-15 micron technology
- 1997: 0.35 Micron
- 1998: 0.25 Micron
- 1999: 0.18 Micron
- 2001: 0.13 Micron
- 2003: 0.09 Micron
- 2007: 0.065 Micron
- 2009: 0.045 Micron
- 2017: 0.014 Micron
- Physical limits ?

Today: Internetworking/Cloud Era?

- Computer as communication device across networks
- World Wide Web, Internet
- Publishing, data sharing, real-time communications
- Seamless devices

Outline of CS in your study

General tracks in CS (computer science)

Theoretical computer science

- Algorithms and data structures
- Theory of computation
- Programming language theory
- Information and coding theory
- Formal methods
- Logic
- Machine learning
- Automata theory
- Computational Geometry

Applied computer science

- Computer architecture
- Computer networks
- Computer graphics
- Computer intelligence (AI)
- Databases
- HCI
- Software engineering
- Big Data
- Intelligent Systems

Your schedule: First semester!

Theoretical computer science

- **Algorithms and data structures**
- **Theory of computation**
- **Programming language theory**
- Information and coding theory
- Formal methods
- Logic
- Machine learning
- Automata theory
- Computational Geometry

Applied computer science

- **Computer architecture**
- Computer networks
- Computer graphics
- Computer intelligence (AI)
- Databases
- HCI
- **Software engineering**
- Big Data
- Intelligent Systems

Your schedule: Second semester ...

Theoretical computer science

- Algorithms and data structures
- Theory of computation
- Programming language theory
- Information and coding theory
- Formal methods
- Logic
- **Machine learning**
- Automata theory
- Computational Geometry

Applied computer science

- Computer architecture
- Computer networks
- Computer graphics
- **Computer intelligence (AI)**
- **Databases**
- HCI
- Software engineering
- **Big Data**
- **Intelligent Systems**

Outline of the lecture

Introduction to Computer Science and Programing

Outline of the lecture

Lecture	Date	Topics
1	9/21/2020	Introduction: Administrative issues, Algorithms, Programming
2	9/28/2020	Python: Overview, Syntax, Simple datatypes, Input/Output, Conditions
3	10/12/2020	Python: Composite data structures, Loops
4	10/19/2020	Python: Functions, Recursion, Files, Exceptions
5	10/26/2020	Python: Classes and objects
6	11/2/2020	Complexity Analysis: Big O, Recursion trees
7	11/9/2020	Complexity Analysis: Solving recurrences, Master method
8	11/16/2020	Sorting: Problem, Naive solution, Lower bounds
9	11/23/2020	Sorting: Mergesort, Quicksort
10	11/30/2020	Searching: Concept, Binary search trees, Complexity
11	12/7/2020	Searching: Hashing
12	12/14/2020	Searching: Hashing 2
13	12/21/2020	TBD/overflow
14	1/4/2021	TBD/overflow
15	1/11/2021	TBD/overflow
16	1/18/2021	Summary

Overview: Python

```
>>> name = "Lucy"
>>> print("Hello " + name)
Hello Lucy
```

- We will use Python to understand different programming concepts:
  - All examples will be written in Python
  - Your assignments will be written in Python
- Some advantages of Python
  - Free
  - Powerful, many extensions
  - Widely used (Google, NASA, Yahoo, Electronic Arts, Linux operating system scripts etc.)
- What you will learn in this course?
  - Designing and implementing (=coding) small programs yourself
  - Automatize repetitive tasks
  - Maintaining your program, finding bugs

Overview: Complexity Analysis

- Problems can be put into complexity classes
  - Complexity: complex (or not)?
  - There are simple problems and rather (provably) hard problems
  - Examples?
- Algorithms also come with worst-case complexities
  - There are efficient and inefficient algorithms
  - Examples?
- Combining hard problems with inefficient algorithms for solving them leads to unfeasible computations
- What you will learn in this course?
  - Compute the inherent complexity of a problem
  - Estimate the actual complexity of a solution algorithm
  - Taking into account scalability from the beginning

Overview: Sorting

- Given a list of elements, say L=[1,5,3,7,9,8,4], rearrange the list in increasing order
  - Obvious result: [1,3,4,5,7,8,9]
- Sorting if often used in CS education for learning about algorithms
  - Easy to understand, many different algorithms for solving
- What you will learn in this course?
  - The strengths and weaknesses of multiple sorting algorithms
  - How to implement sorting algorithms efficiently
  - Understanding list data structures
  - Understanding upper and lower bounds for complexity

Overview: Searching

- Given a list of elements, say L=[1,5,3,7,9,8,4], check whether the list contains an element, say 8
  - Obvious result: yes!
- Searching is another frequently used operation in many programs, with different levels of complexity
  - Examples?
- What you will learn in this course?
  - The strengths and weaknesses of multiple search algorithms
  - Finding elements in lists in less than linear time
  - Using tree data structures
  - Understanding upper and lower bounds for complexity

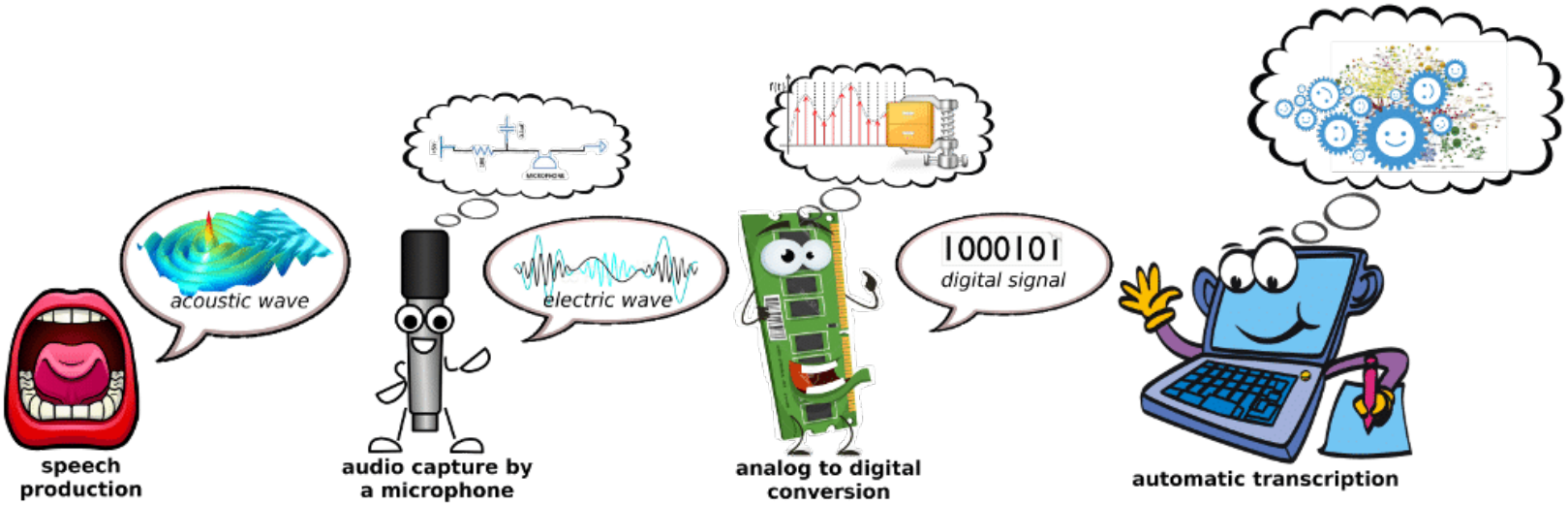
Overview: Graphs

- Model elements and their relationships as a graph (or network)
  - Examples?
- A fundamental data structure in CS applications
- Almost anything can be understood as a network (nowadays)
- What you will learn in this course?
  - How to model problems using graphs
  - Implementing graph algorithms efficiently
    - E.g., exploration, shortest paths, weakly/strongly connected components

Algorithms and Programming Languages

Can computers understand humans?

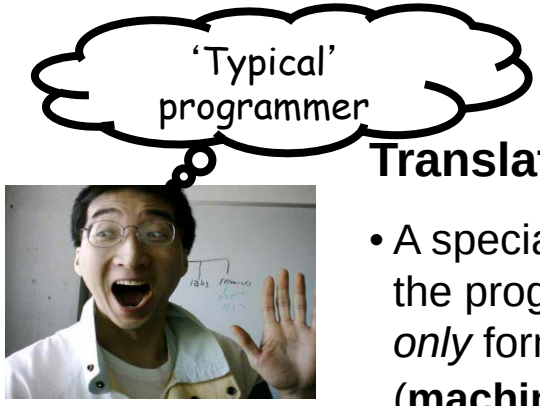
Can computers understand humans?



Algorithms

- General concept for describing the solution of a problem
  - A computational procedure ...
- Many properties
  - Termination
  - Correctness / completeness
  - Time complexity
- Computer Science is about the formal study of algorithms
- Programs are implementations of algorithms
  - Algorithm: **Abstract**
  - Program: **Specific / Concrete**
- A computer can execute a program!

Creating A Computer Program



Translation

- A special computer program (translator) translates the program written by the programmer into the *only* form that the computer can understand (**machine language/binary**)

Program Creation

- A person (programmer) writes a computer program (series of instructions).
- The program is written and saved using a text editor/IDE.
- The instructions in the programming language (**e.g., Python**) are high level (look much like a human language).

```
a=2
print(a*3.1415926)
```

Execution

- The machine/binary language instructions can now be directly executed by the computer.

```
10000001
10010100 10000100
10000001 01010100
```

Types of Translators

1) **Interpreters** (e.g., Python is an interpreted language)
    - Each time the program is run the interpreter translates the program (translating a part at a time).
    - If there are any translation errors during the process of interpreting the program, the program will stop execution right when the error is encountered.

2) **Compilers** (e.g., 'C', C++ are compiled languages)
    - Before the program is run the compiler translates the program all at once.
    - If there are *any translation errors* during the compilation process, no machine language executable will be produced (nothing to execute)
    - If there are *no translation errors* during compilation then a machine language program is created which can then be executed.

Python History

From:
http://www.python.org/~guido/

"Gawky and proud of it."

- Developed in the early 1990s by Guido van Rossum.
- Python was designed with a tradeoff in mind (from "*Python for everyone*" (Horstman and Necaise):
  - Pro: Python programmers could quickly *write programs* (and not be burdened with an overly difficult language)
  - Con: Python programs weren't optimized to *run* as efficiently as programs written in some other languages.
- Some advantages (from Python dot org)
  - Free
  - Powerful
  - Widely used (Google, NASA, Yahoo, Electronic Arts, some Linux operating system scripts etc.)
- Named after a British comedy "Monty Python's Flying Circus"
  - Official website (Python the programming language, not the Monty Python comedy troop): http://www.python.org

Can you understand this small program?

program1.py

```
a=2
b=3
print(a*b)
```

Can you understand this small program?

program2.py

```
while True:
    print("Keep learning")
```

Running Programs

program1.py

```
a=2
b=3
print(a*b)
```

- In general, three different ways:
  - Run whole script
    - Execute: python program1.py
    - Preset file written in text editor
  - Interactive environment
    - Execute: python
    - Type script line by line and inspect variables
    - Program is essentially developed over time
    - Convenient for simple debugging, testing, exploring data
  - Integrated development environment (IDE)
    - Open IDE, load file, run
    - Examples: Spyder, Ninja IDE, and many more
    - Very convenient for graphical debugging and testing
- Let's have a quick look at these ...
- We will do more of this in the lab class on Tuesday!

What should you know after this lecture?

- This course targets to teach you on the intersections of:
  - Algorithms, data structures, and programming
- Computer Science is not about computers only
- Algorithms are at the core of many CS concepts
- Algorithms have properties
  - Correctness
  - Efficiency / Time complexity
- It is, in general, useless to design bad algorithms
- Algorithms (abstract) vs. implementations (concrete)
- We will use Python

What will you learn in the next few lectures?

- During the next few weeks we will look at the programming language Python
  - Understand Syntax and Semantics
  - Design small programs by yourself
  - Debug and maintain programs
- Afterwards we will start with the theoretical part

What will happen on Tuesday?

- You will get used to the lab environment
- Understand some very basic concepts of Python
- The real introduction comes in the next **lecture**
  - If, until then, you want to prepare something, try to play around with Python a little bit

- Please download the following file before the lab class:

https://jaist.dl.sourceforge.net/project/winpython/WinPython_3.8/3.8.5.0/Winpython64-3.8.5.0.exe

Open questions?
Suggestions from your side?

Thank you very much!