Applied Algorithms

CSCI-B505 / INFO-I500

Lecture 1.

Introduction to Applied Algorithms and Course Overview

M. Oğuzhan Kulekci

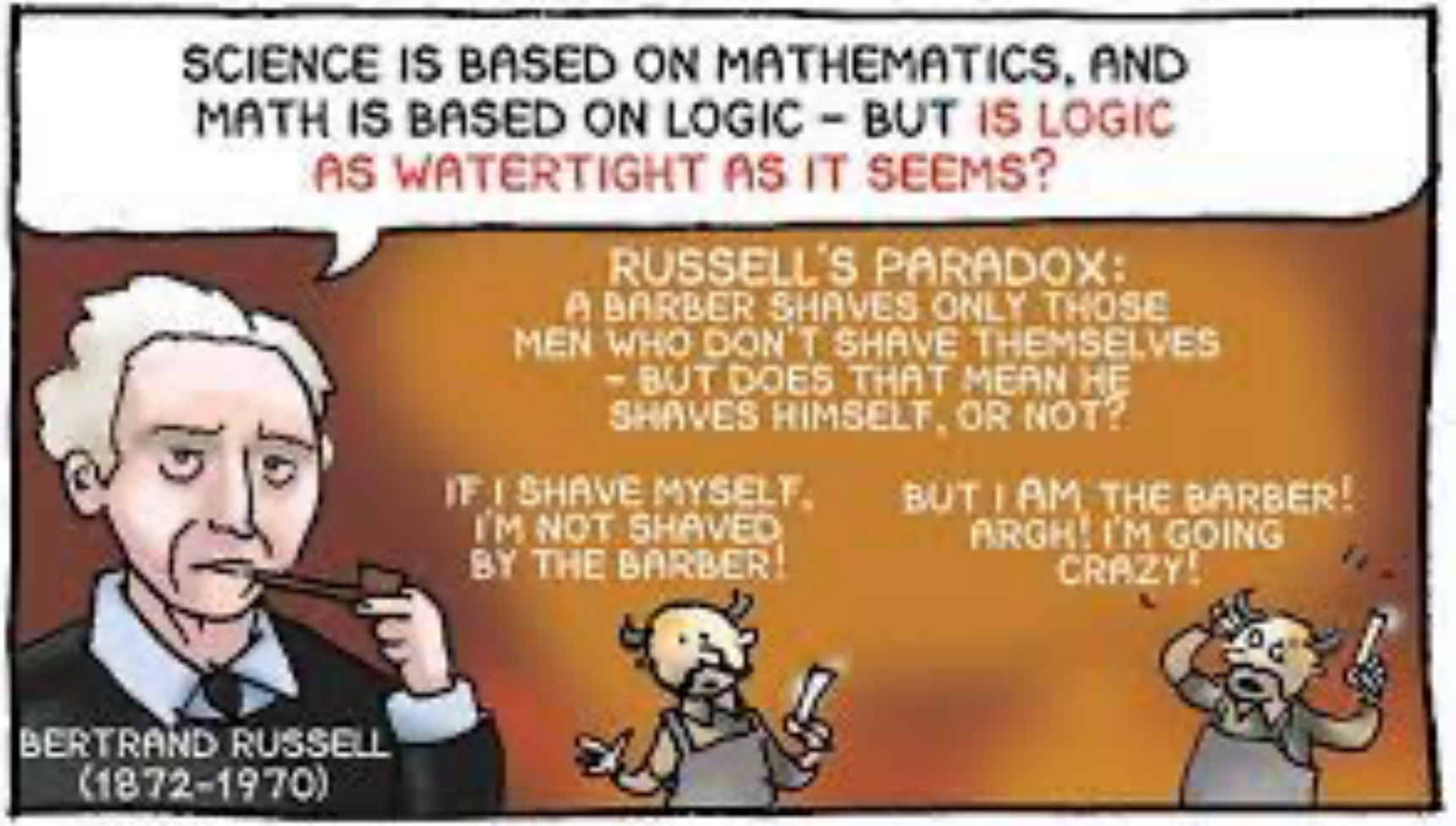
Georg Cantor
1845-1917

$$
\begin{array}{ll}
s_1 & = 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\ldots \\
s_2 & = 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\ldots \\
s_3 & = 0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\ldots \\
s_4 & = 1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\ldots \\
s_5 & = 1\,1\,0\,1\,0\,1\,1\,0\,1\,0\,1\ldots \\
s_6 & = 0\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\ldots \\
s_7 & = 1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\ldots \\
s_8 & = 0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\ldots \\
s_9 & = 1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\ldots \\
s_{10} & = 1\,1\,0\,1\,1\,1\,0\,0\,1\,0\,1\ldots \\
s_{11} & = 1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,0\ldots \\
\vdots & \quad \vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\ddots
\end{array}
$$

$$
s = 1\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1\ldots
$$

There are uncountable sets

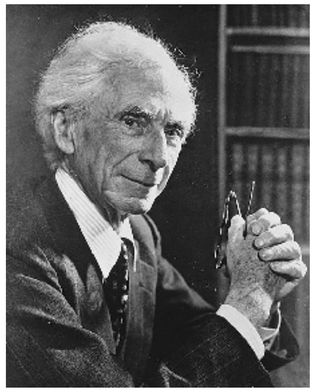
THE QUESTION THAT LED TO THE DEVELOPMENT OF COMPUTING MACHINES ! *(I.M.O.)*



Mathematical Roots of Computing 1870 - 1940

Georg Cantor
1845-1917

SET THEORY - 1874



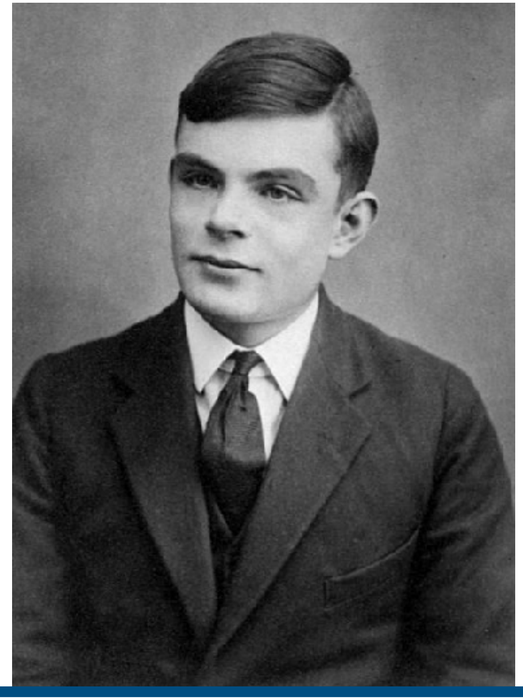
Bertrand Russell
1872-1970

RUSSEL PARADOX - 1901

David Hilbert
1862-1943
MATHEMATICAL LOGIC - 1921

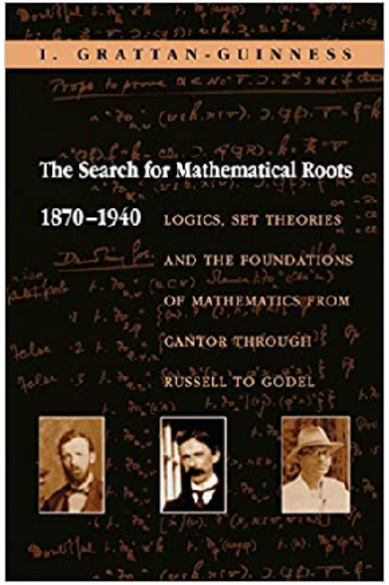
Kurt Gödel
1906 - 1978

Incompleteness - 1931



Alan Turing
1912-1954

Undecidability - 1936

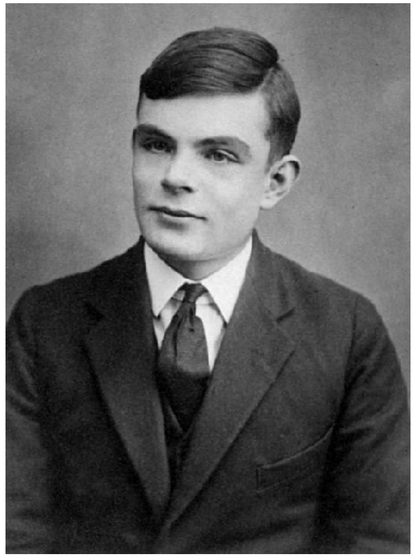


FIRST PROGRAMMABLE COMPUTING MACHINES 1940 - 1950

Alanzo Church
1903 - 1995

$\lambda - calculus$

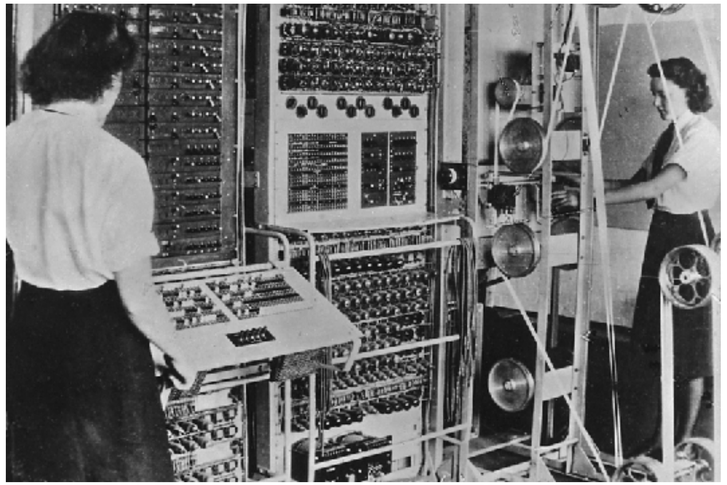
1936



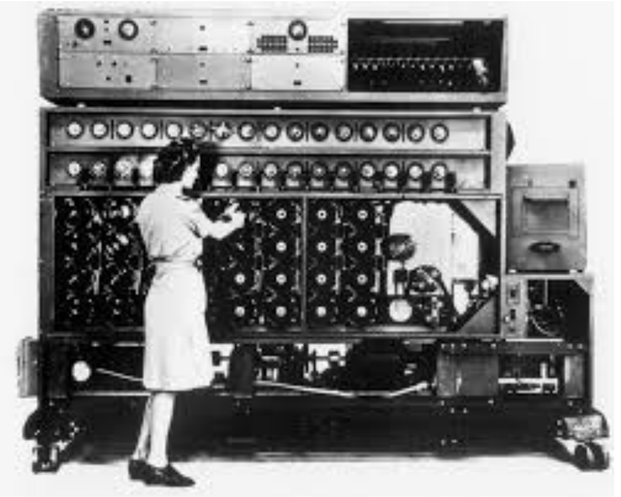
Alan Turing
1912-1954

Turing Machine

1936



COLOSSUS



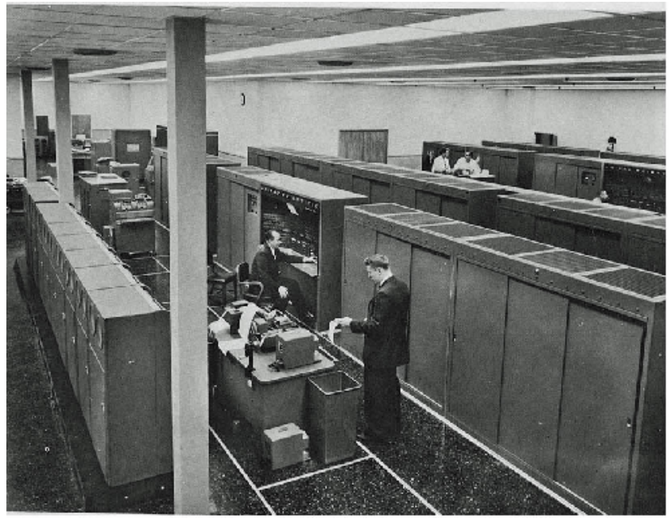
BOMBE

1942 - 1945
BLETCHLEY PARK - LONDON, UK

ENIGMA
II. WORLD WAR
CRYPTO MACHINE

Church-Turing Thesis:
A function on the natural numbers can be calculated by an effective method if and only if it is computable by a Turing machine

EARLY YEARS 1950 - 1960



UNIVAC 1103 (1953)

Computers appeared as new commercial devices, and ignited the “computing business”, (have a look to https://en.wikipedia.org/wiki/Timeline_of_computing for more details…)

Computer Programming:

Machine Codes → Assembly Language → Modern programming languages FORTRAN’54, ALGOL’60, *(later PASCAL’70, C’73, C++’82, JAVA’91, PYTHON, …)*

TOWARDS A NEW SCIENTIFIC DISCIPLINE (1960 - 1970)

ON THE COMPUTATIONAL
COMPLEXITY OF ALGORITHMS

BY

J. HARTMANIS AND R. E. STEARNS

I. **Introduction.** In his celebrated paper [1], A. M. Turing investigated the computability of sequences (functions) by mechanical procedures and showed that the set of sequences can be partitioned into computable and noncomputable sequences. One finds, however, that some computable sequences are very easy to compute whereas other computable sequences seem to have an inherent complexity that makes them difficult to compute. In this paper, we investigate a scheme of classifying sequences according to how hard they are to compute. This scheme puts a rich structure on the computable sequences and a variety of theorems are established. Furthermore, this scheme can be generalized to classify numbers, functions, or recognition problems according to their computational complexity.

Hartmanis & Stearn'1965:
Time and space complexity analysis of algorithms

How can we measure the performance of an algorithm?

PATHS, TREES, AND FLOWERS

JACK EDMONDS

2. Digression. An explanation is due on the use of the words "efficient algorithm." First, what I present is a conceptual description of an algorithm and not a particular formalized algorithm or "code."

For practical purposes computational details are vital. However, my purpose is only to show as attractively as I can that there is an efficient algorithm. According to the dictionary, "efficient" means "adequate in operation or performance." This is roughly the meaning I want—in the sense that it is conceivable for maximum matching to have no efficient algorithm. Perhaps a better word is "good."

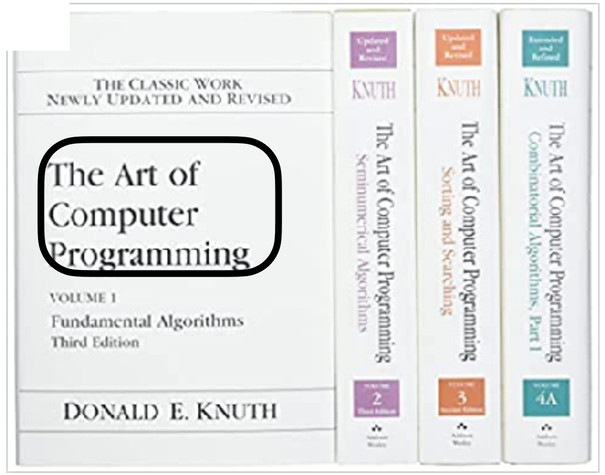
I am claiming, as a mathematical result, the existence of a *good* algorithm for finding a maximum cardinality matching in a graph.

There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether *or not* there exists an algorithm whose difficulty increases only algebraically with the size of the graph.

The mathematical significance of this paper rests largely on the assumption that the two preceding sentences have mathematical meaning. I am not prepared to set up the machinery necessary to give them formal meaning, nor

Edmunds'65:
Definition of Efficient Algorithm

When does an algorithm is assumed to be "efficient" ?



1962(start)-1971

Elegance in computing ?

Elegance *is beauty that shows unusual* ***effectiveness*** *and* ***simplicity****….*

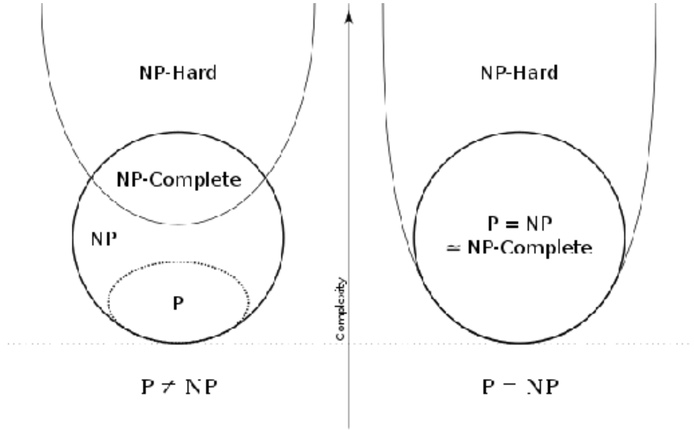
PEN & PAPER ERA IN ALGORITHMS & THEORY (up to late 80s)

Stephan Cook
1971 (3-SAT)

Leonid Levin
1971

Richard Karp
1972

21 new hard problems



Some computational problems are really hard,
with amazing relations between them

Developments in many sub-divisions of computer science: **parallel algorithms**, **external memory**, **randomization**, **approximation**, etc….

GAP BETWEEN THEORY AND PRACTICE (up to late 90s)

! Personal computers !
Everyone can purchase.

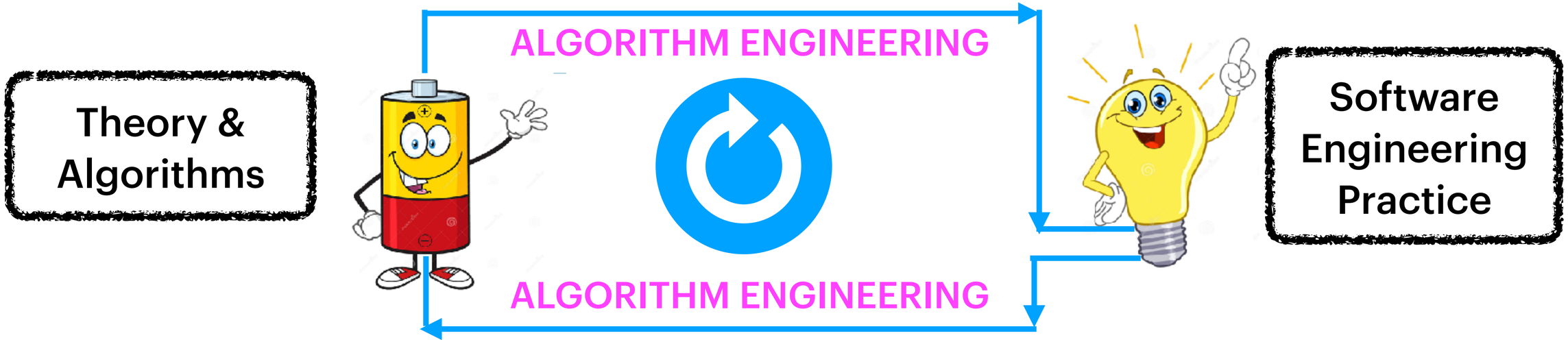
Software industry developing very fast ! More and more digitization in daily life…

- Theory and practice were flowing in different rivers.
- Practice was not in big need for theory, and theory was not much interested in practice.
- However, the gap between theory and practice became apparent.

ALGORITHM ENGINEERING PARADIGM (2000 - …)

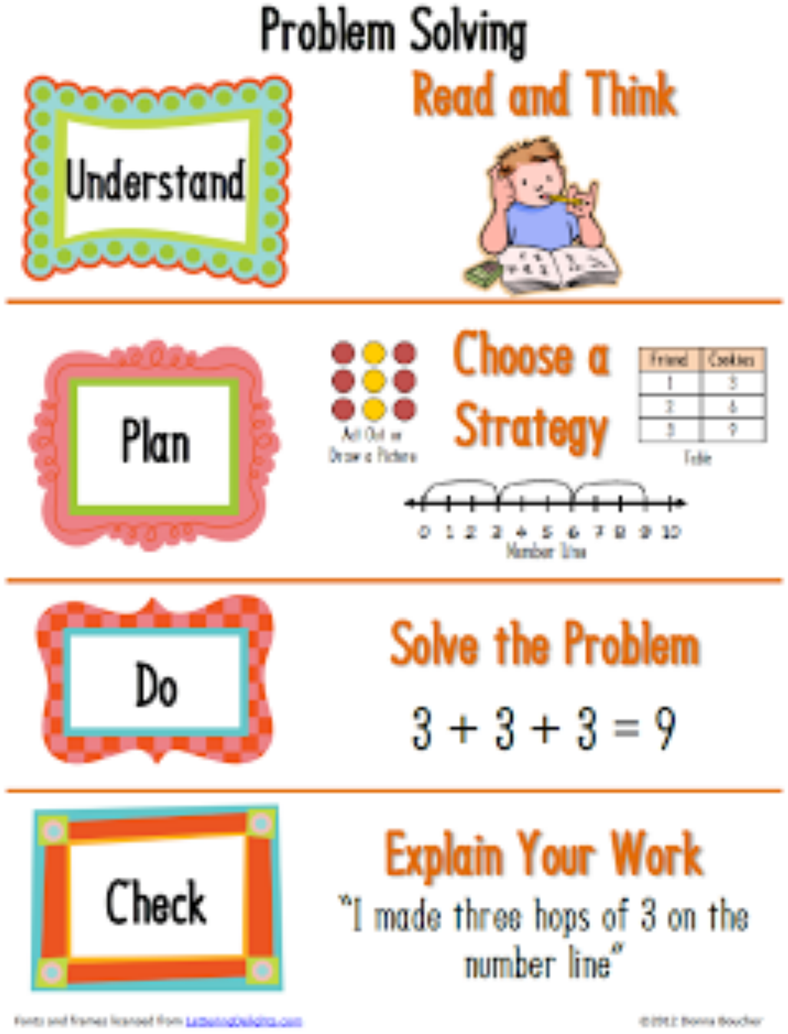
Previous hypotheses , which were assumed to be unrealistic, are turning into realities due to :

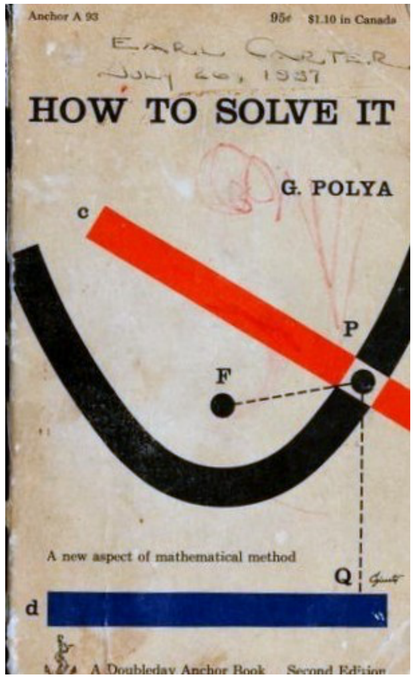
- Data deluge and ever increasing digitization
- Advances on computing platforms and processor technology
- Increasing importance of theoretical results in practical applications



Sound understanding of algorithms and data structures is essential for devising efficient solutions of computational challenges.

Is it sufficient ? (Another story)





The Aim of This Course

To improve our algorithm design and analysis skills and increase our awareness on integrating them into applications.

IMPORTANT NOTICE: We assume

- **you know the basic data structures and algorithms at the undergraduate level**
- **you know how to program in Python on Jupyter Notebook environment**
- **you have the basic probability and discrete math knowledge at the undergraduate level**

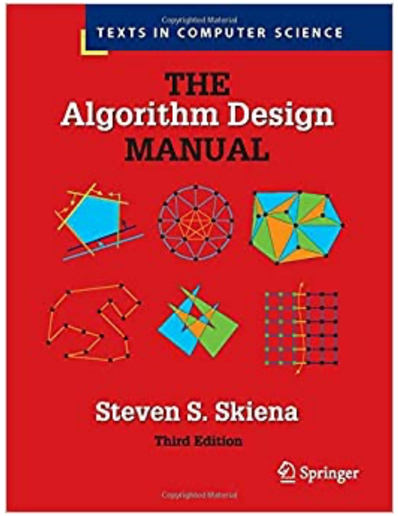
This course does not aim to teach any programming.

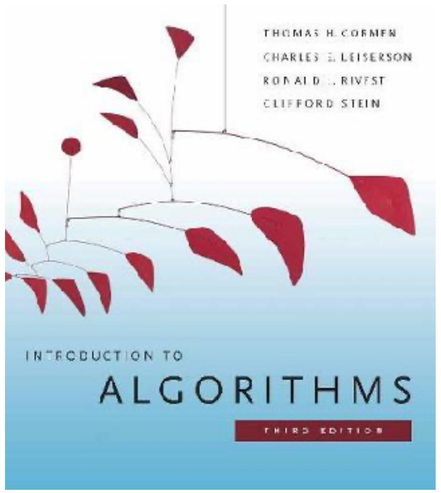
We will be doing programming a lot during this course,

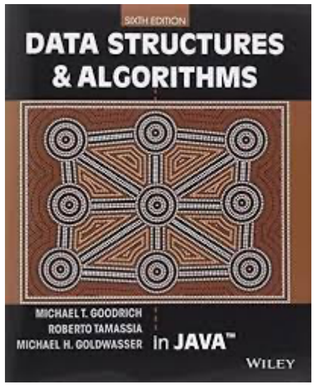
but do not have a special purpose to teach it.

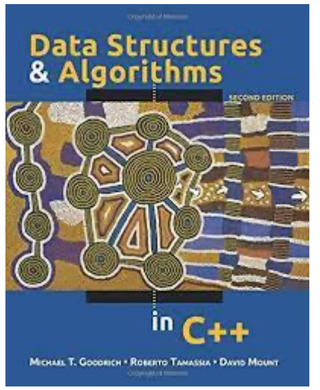
Textbooks & Learning Materials

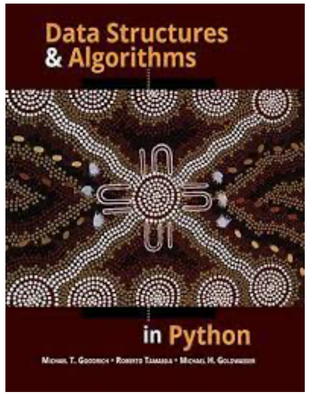
This course does not aim to teach any programming,
but to improve your skills in algorithms, particularly on real-life scenarios !











There are many great algorithms textbooks you can refer to, but particularly I would like to mention the following

- **The Algorithm Design Manual by Steven Skiena**
- **Introduction to Algorithms, Cormen et al.**
- **Data Structuctures and Algorithms in Java / Python / C/C++, Goodrich et al.**
- **Papers and other reading materials will be provided during the course**

Course Organization

Principal Instructor:

M. Oğuzhan Külekci
Email: okulekci@iu.edu
Office: Luddy - 2030
Office Hours:
Monday - Wednesday
11:00 - 12:00

Head of Teaching Assistants is
Jenil Jignesh Gandhi
jegandhi@iu.edu
Any questions/concerns about the course should be addressed to Jenil first.

The information of the all TAs will be announced this week !!!

Course Organization

- **PLEASE FOLLOW YOUR REGISTERED SECTION.**
  - **Each student is registered to a specific LAB session and you can ONLY attend to your section, NOT OTHERS!**
- **THE RULES ARE STRICT, VERY, REALLY !!!**

Course Organization

- **OFFICE Hours: (exact timing will be announced this week)**

  We will provide you office hours every hour of every weekday!

  Please do not hesitate to contact with the TAs for your questions.

Office hours will be held at Luddy 2014.

This is the discussion area at the second floor of Luddy.
Please go and check this place.
The AI/TAs will be waiting you to help on your questions an the mentioned times.

Some Notes on Course Execution

- There will be 6 homework assignment, which will require pen&paper work or programming or both.
- You will be doing programming exercises during the LAB sessions, that aims to improve your understanding via examples. Attend the LAB, put your effort to accomplish the task.
- You can visit the TAs on the specified office hours without any prior appointment.
- Auto-grader will be used to evaluate HWs. You will need to submit your HW to Canvas and also to Auto-Grader.
- **No bargaining on the grades !**

Grading

- Homeworks 45%
    - Late submissions: First 24 hours **for one homework** is free, after that 20% penalty will be applied for each 24 hours.
    - **Python** will be used for the programming exercises
- Midterm — I: 20%
- Final: 25%
- LAB performance %10

PLAGIARISM IN THE SUBMITTED HWs WILL BE MONITORED RIGOROUSLY !

Tentative Topics

1. Algorithm Analysis and Asymptotic Notation
2. Review of the besic data structures, e.g., array, linked list, stack, queue, tree
3. Recursion
4. Amortized Analysis
5. Divide & Conquer paradigm
6. Dynamic Programming
7. Priority Queues and heaps
8. Huffman coding
9. Sorting & Selection algorithms
10. Greedy algorithms
11. Graph algorithms
12. Randomized algorithms
13. Hash algorithms
14. Streaming algorithms

Questions, Comments ?

- Please see syllabus for more details