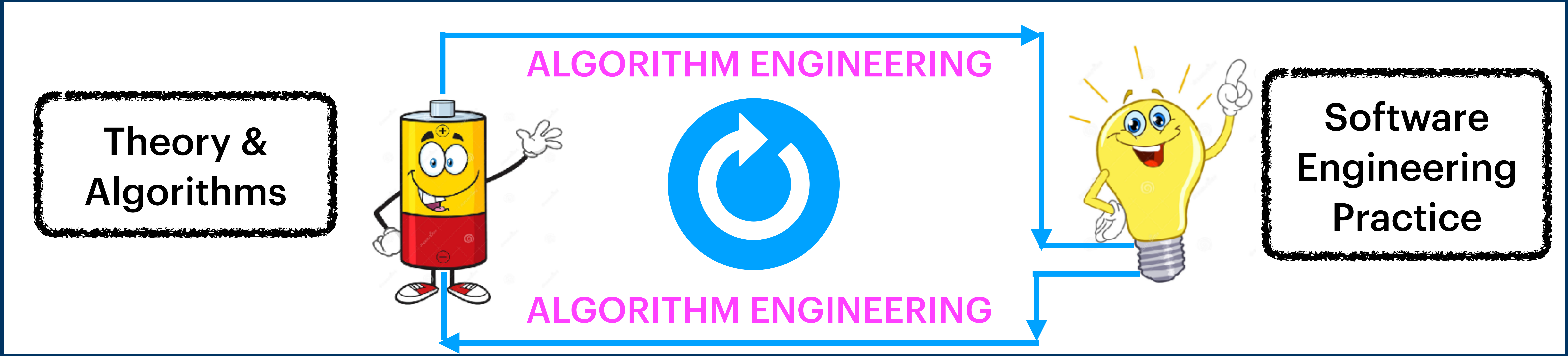
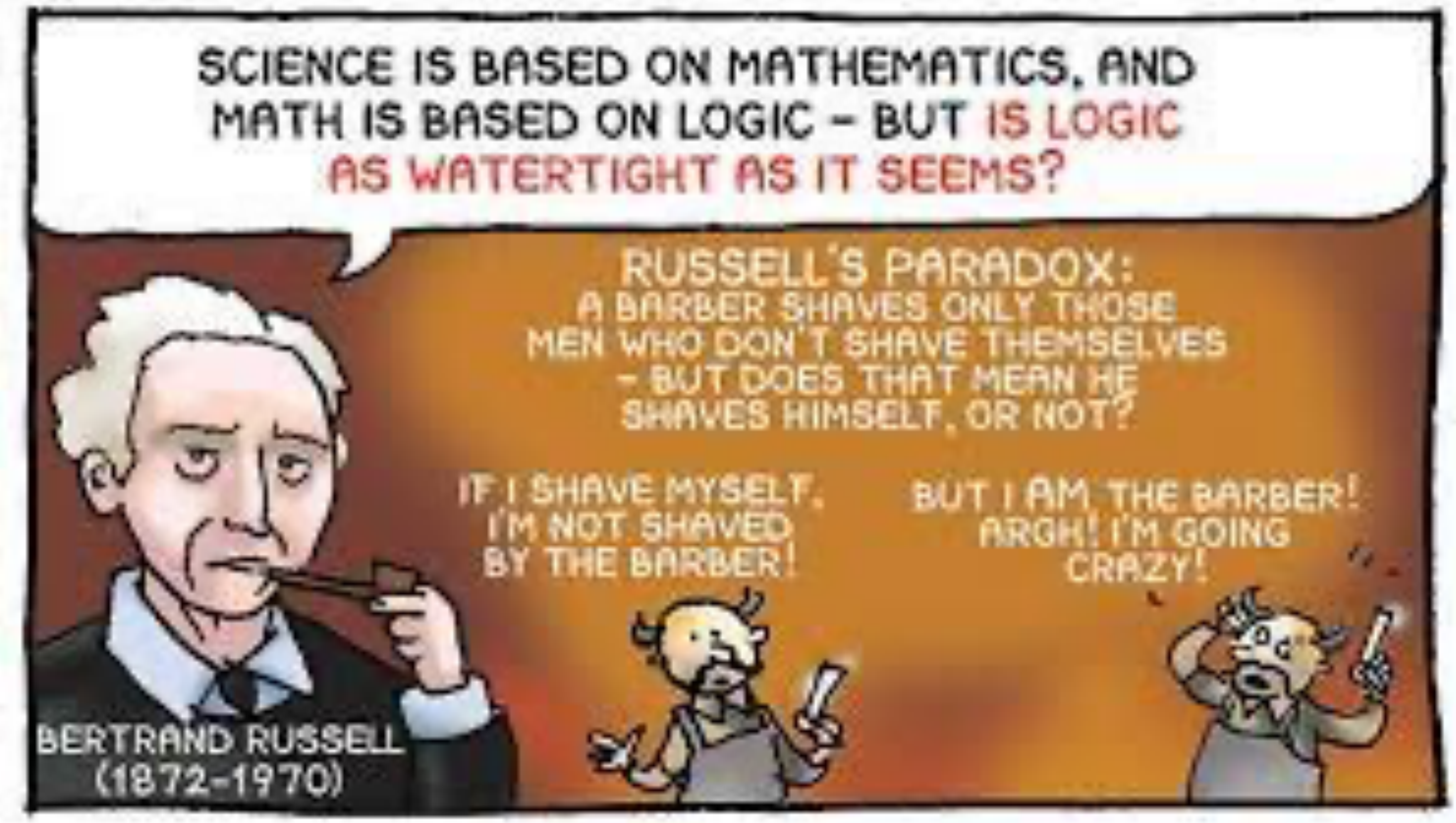


ALGORITHM ENGINEERING

Lecture 0:
A Short Tale of Modern Computing and Rise of Algorithm Engineering

M. Oğuzhan Külekci - kulekci@itu.edu.tr

THE QUESTION THAT EVENTUALLY LED TO THE DEVELOPMENT OF COMPUTING MACHINES ???

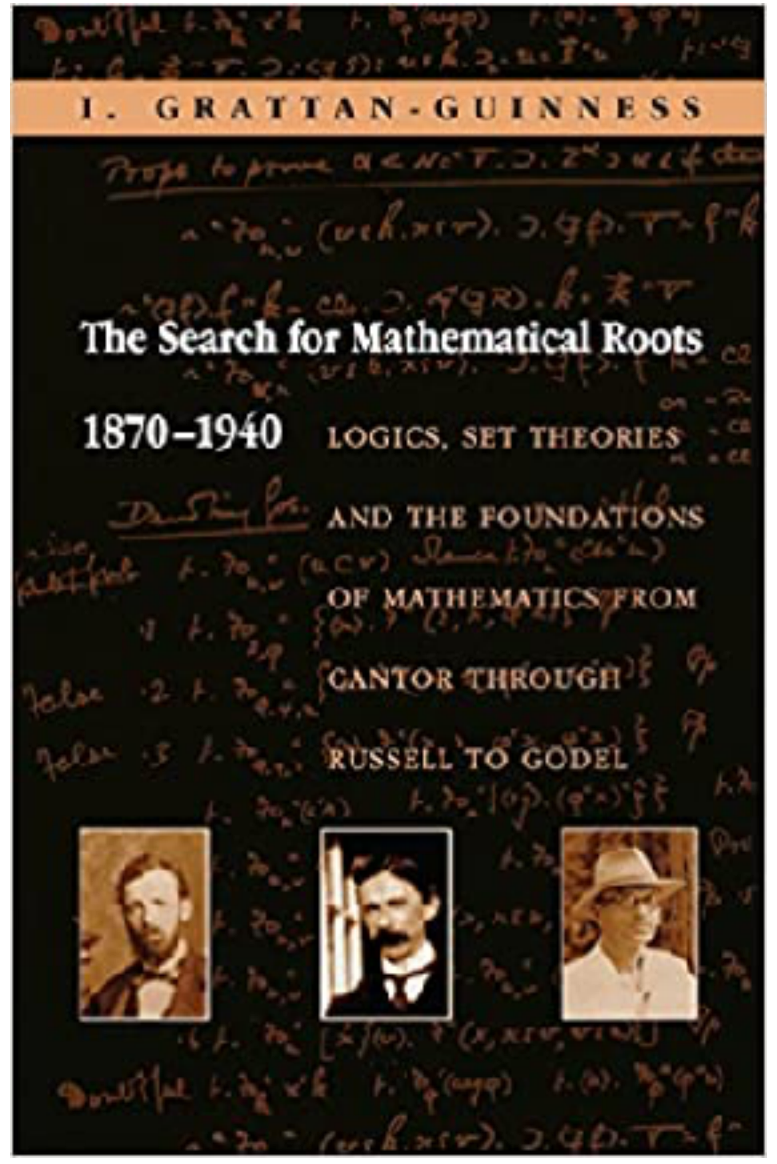


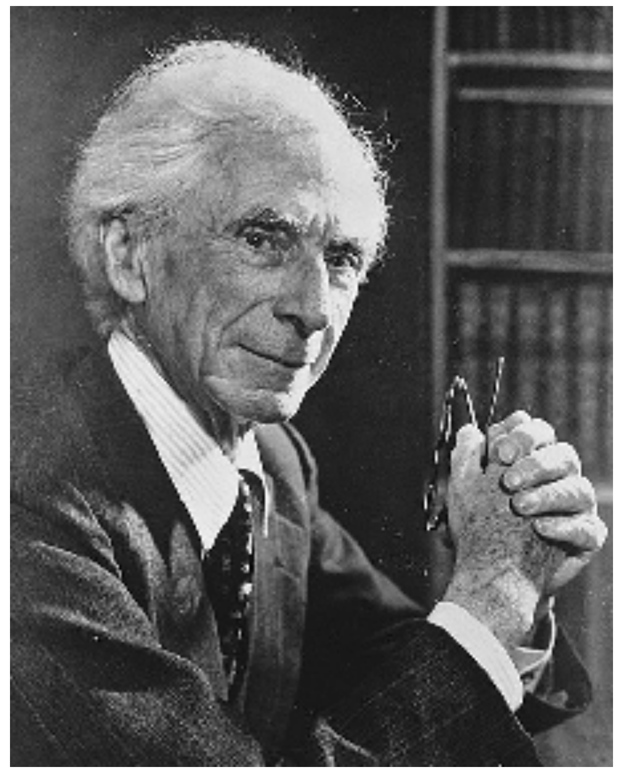
Mathematical Roots of Computability 1870 - 1940

Georg Cantor
1845-1917

SET THEORY - 1874

David Hilbert
1862-1943
MATHEMATICAL LOGIC - 1921



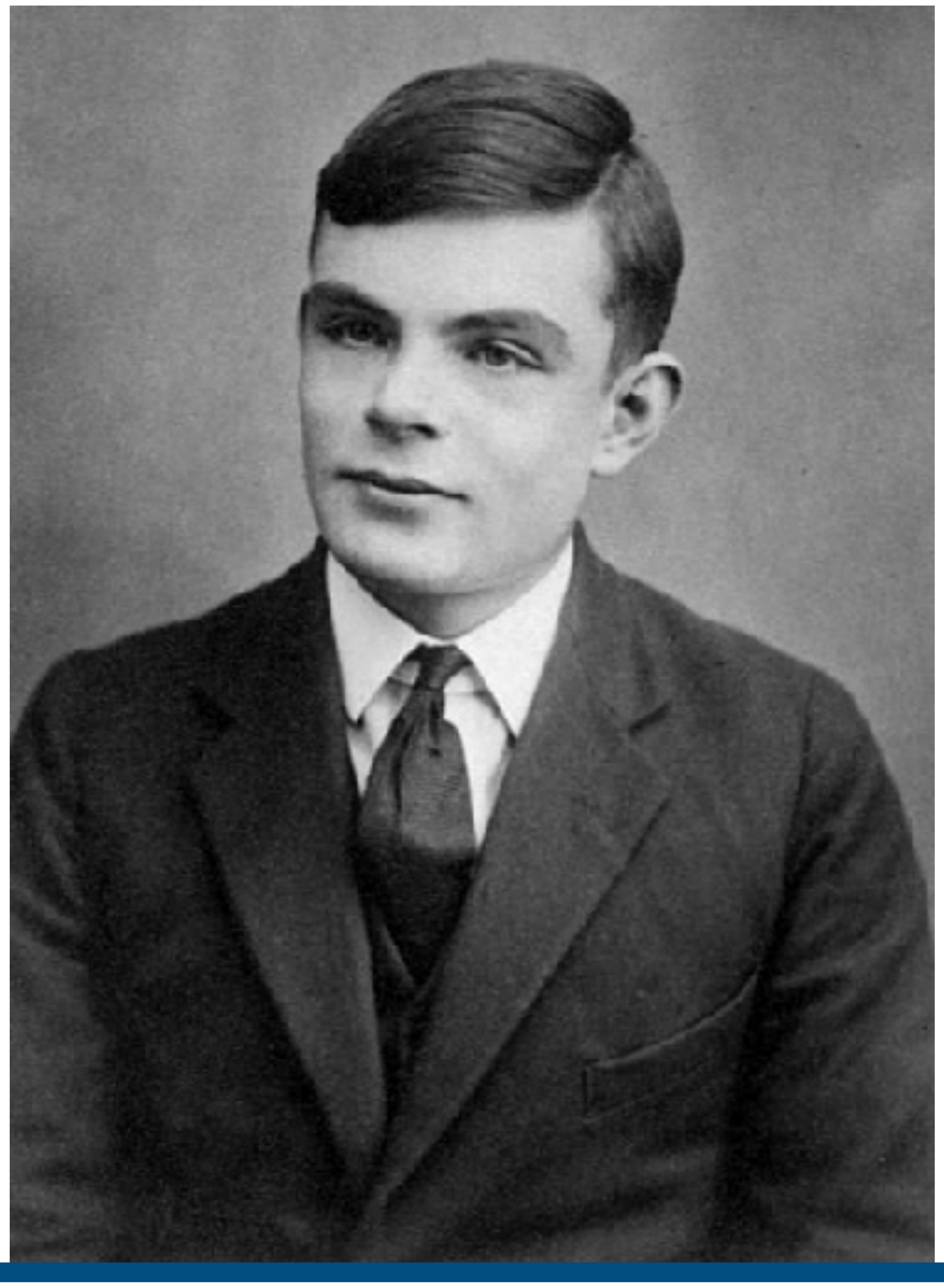


Bertrand Russell
1872-1970

RUSSEL PARADOX - 1901

Kurt Gödel
1906 - 1978

Incompleteness - 1931



Alan Turing
1912-1954

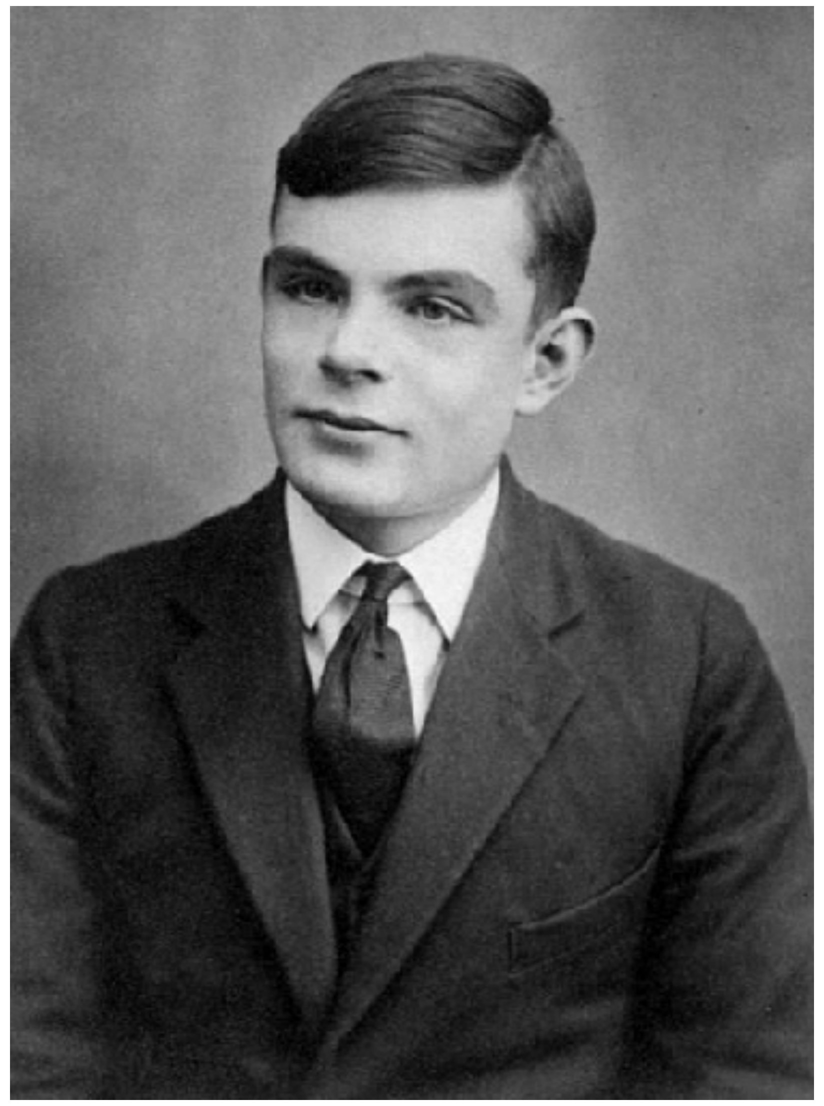
Undecidability - 1936

FIRST PROGRAMMABLE COMPUTING MACHINES 1940 - 1950

Alanzo Church
1903 - 1995

$\lambda - calculus$

1936

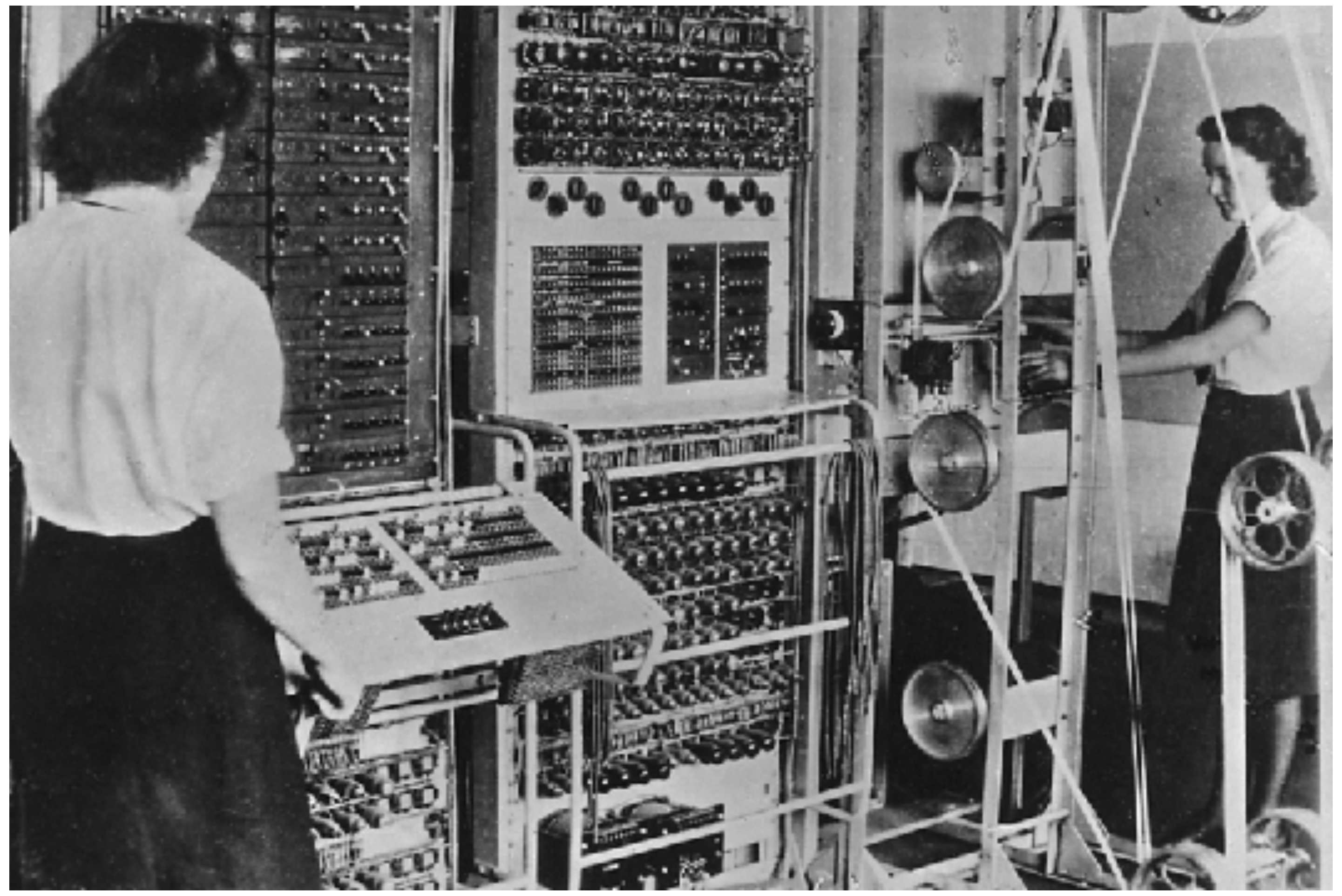


Alan Turing
1912-1954

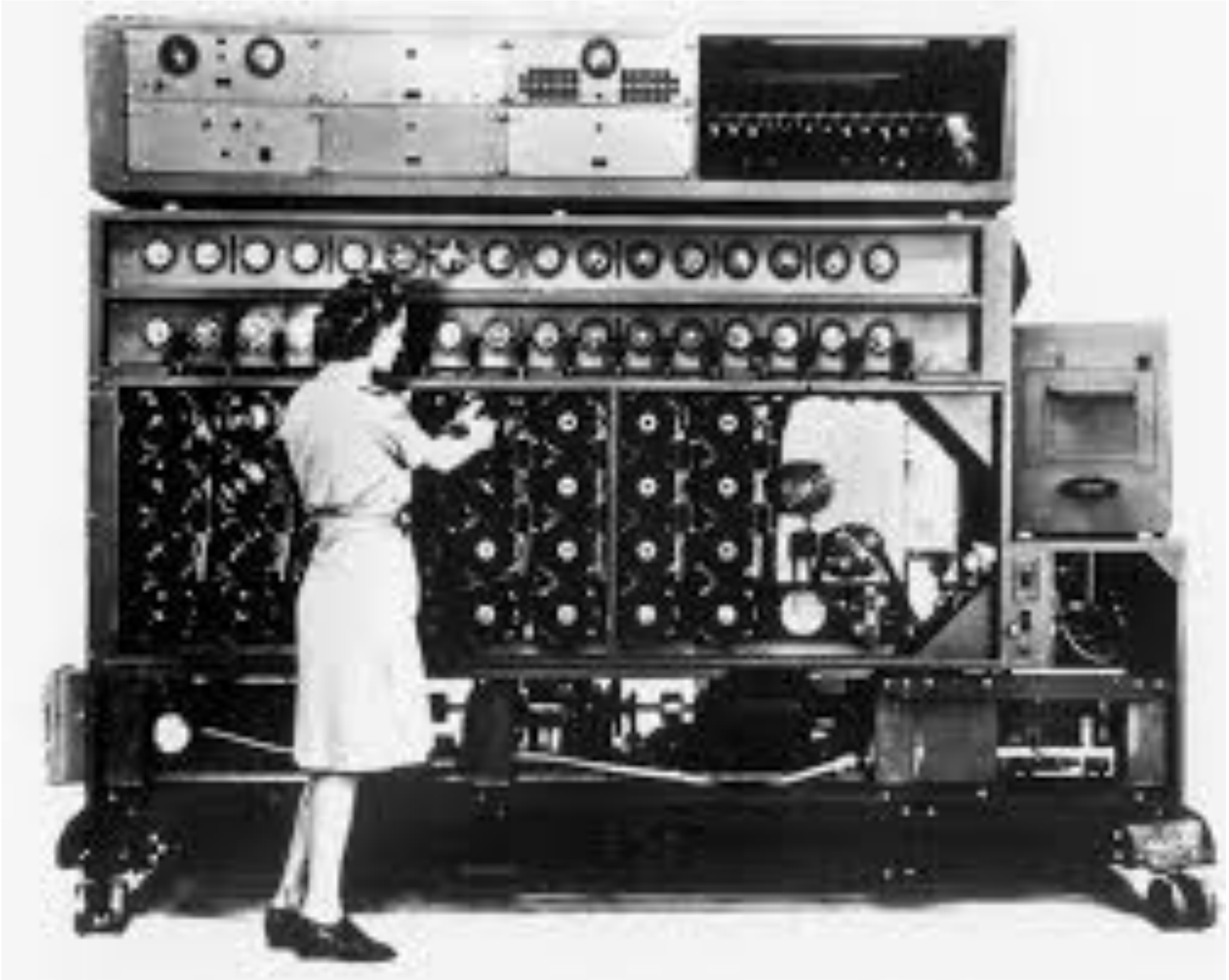
Turing Machine

1936

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



COLOSSUS



BOMBE

1942 - 1945
BLETCHLEY PARK - LONDON, UK

ENIGMA
II. WORLD WAR
CRYPTO 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’9, …)*

TOWARDS A NEW SCIENTIFIC DISCIPLINE (1960 - 1970)

ON THE COMPUTATIONAL COMPLEXITY OF ALGORITHMS

BY

J. HARTMANIS AND R. E. STEARNS

1. 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

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

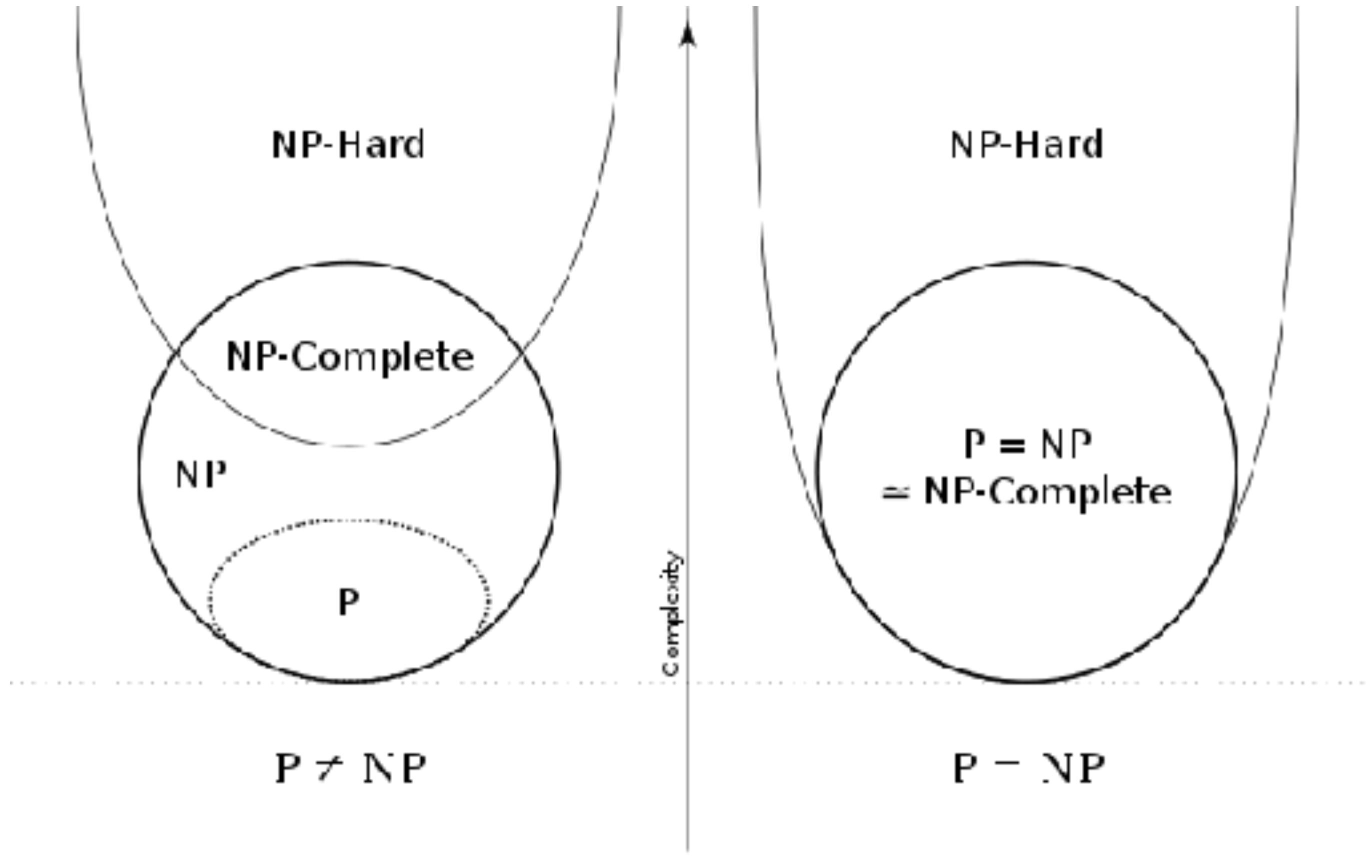
1962(start)-1971

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,
and many 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.

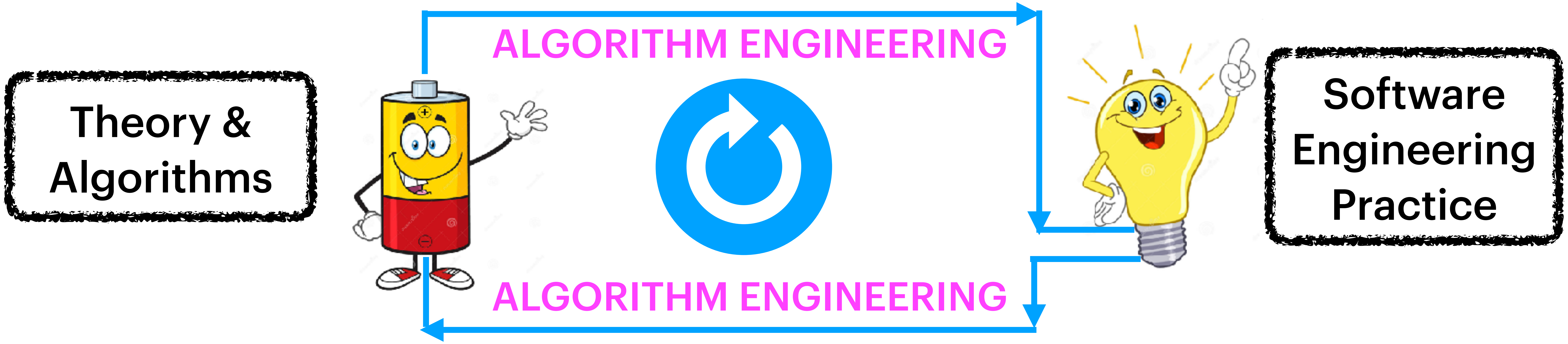
Practice was not in big need for theory, and theory was not much interested in practice.

However, the gap between theory and practice became apparent.

ALGORITHM ENGINEERING PARADIGM (2000 - …)

Previous hypotheses and assumptions turning into realities due to mainly :

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



Workshop on Algorithm Engineering

Venice, Italy

September 11-13, 1997

Now, it is ESA- Track B, experimental algorithms

ALENEX 99

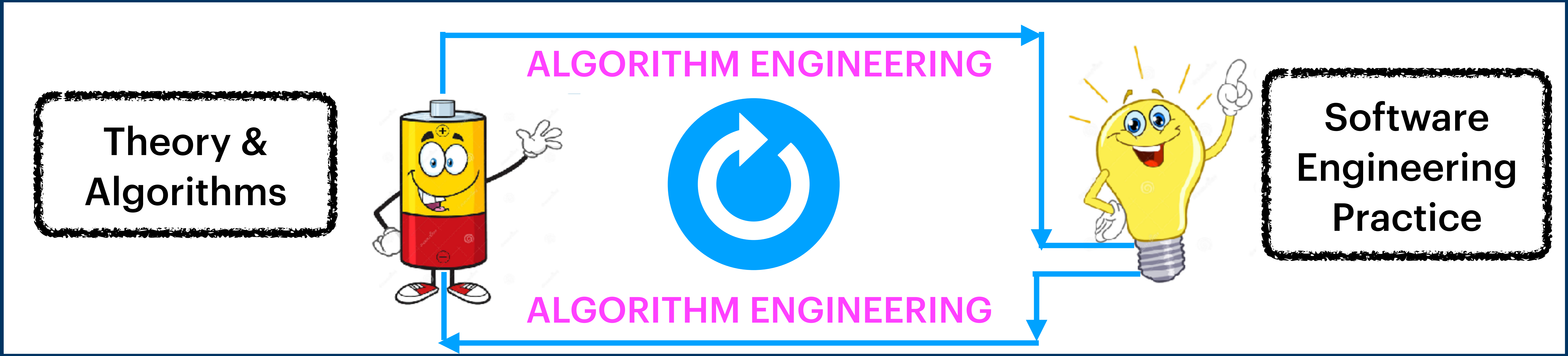
Workshop on Algorithm Engineering and Experimentation

January 15-16, 1999

Omni Hotel, Baltimore, Maryland

Workshop on Experimental Algorithms - WEA, (2001, Riga)
Now, Symposiumn on Experimental Algorithms

NEXT LECTURE ...



ALGORITHM ENGINEERING

Lecture 1:
Introduction to Experimental Algorithms

M. Oğuzhan Külekci - kulekci@itu.edu.tr