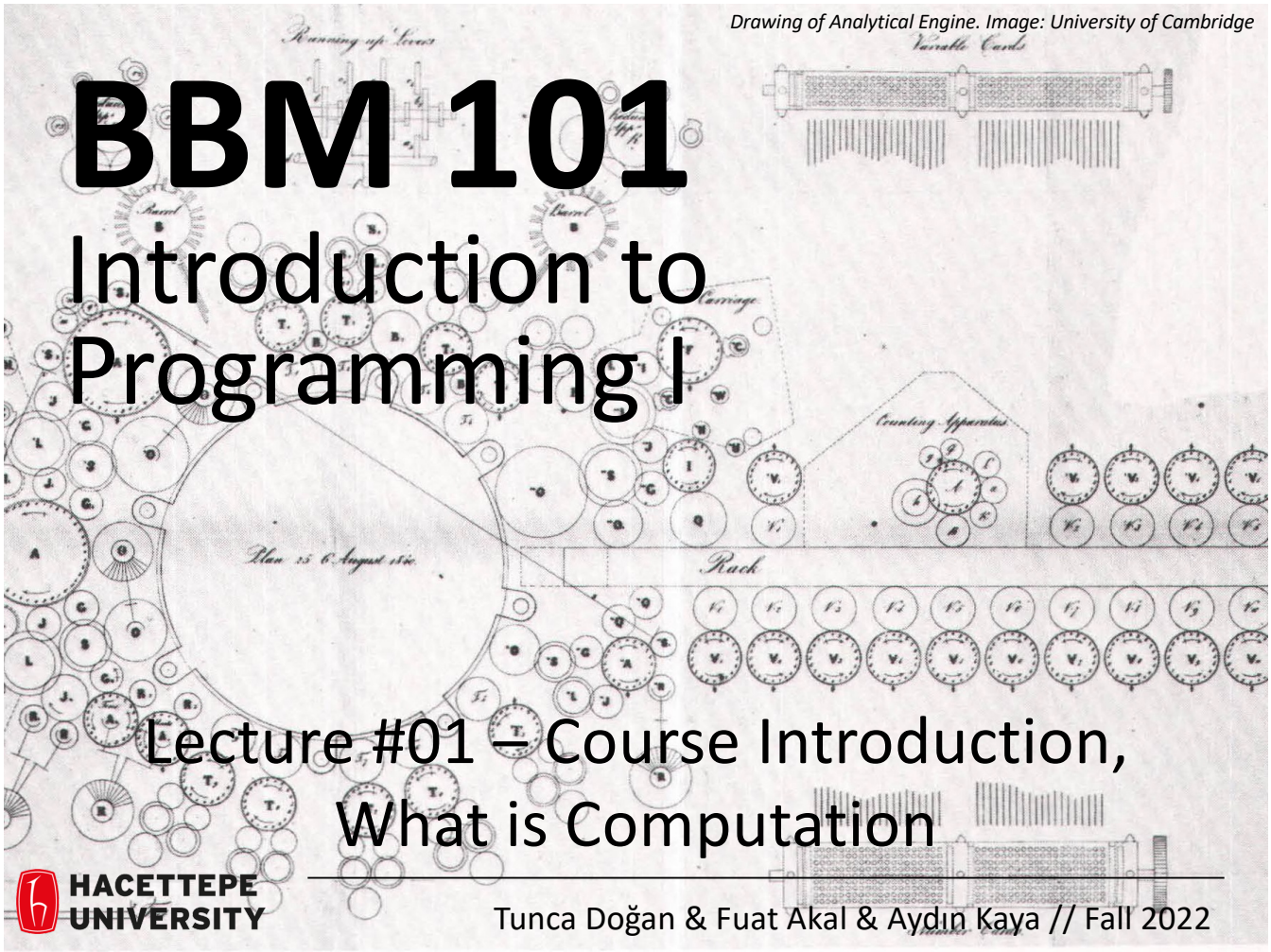


Drawing of Analytical Engine. Image: University of Cambridge

BBM 101

Introduction to Programming I

Lecture #01 – Course Introduction, What is Computation

HACETTEPE UNIVERSITY

Tunca Doğan & Fuat Akal & Aydın Kaya // Fall 2022

Lecture Overview

- Course introduction
- What is computation

Disclaimer: Much of the material and slides for this lecture were borrowed from
— Ruth Anderson, Michael Ernst and Bill Howe's CSE 140 class

Course Information

Course Staff

- **Instructors:** Tunca Doğan, Fuat Akal, Aydın Kaya



- **Teaching Assistants:**
  - Burcu Yalçıner
  - Hayriye Çelikbilek
  - Nebi Yılmaz



Do not hesitate to ask TAs for help!

About BBM 101

- This course teaches core programming concepts with an emphasis on data manipulation tasks from science, engineering, and business

- **Goal** by the end of the semester: Given a data source and a problem description, you can independently write a complete, useful program to solve the problem

- **BBM103 Introduction to Programming Laboratory I**
  - Students will gain skills to apply the concepts to real world problems

Learning Objectives

- Computational problem-solving
  - Writing a program will become your "go-to" solution for data analysis tasks.

- Basic Python proficiency
  - Including experience with relevant libraries for data manipulation, scientific computing, and visualization.

What This Course is <u>not</u>

- A "skills course" in Python
  - …though you'll become proficient in the basics of the Python programming language
  - …and you will gain experience with some important Python libraries

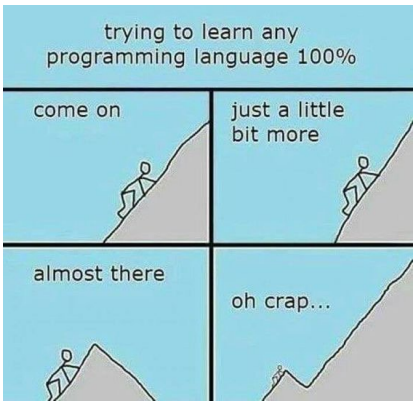


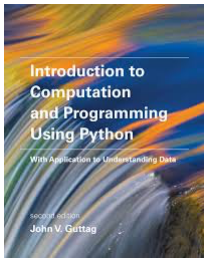
Image credit: Google+ user nixCraft

- A "project" course
  - the assignments are "real," but are intended to teach specific programming concepts
- A "software engineering" course
  - Programming is the starting point of computer science and software engineering

Communication

- Website: `http://web.cs.hacettepe.edu.tr/~bbm101/`
- See the website for all administrative details
- Read the handouts and required texts, *before* the lecture
- Take notes!
- Always use your department account while sending email! <u>Change your e-mail in BILSIS immediately</u>!
- Follow the course in Piazza
  `https://piazza.com/hacettepe.edu.tr/fall2022/bbm101`
  `https://piazza.com/hacettepe.edu.tr/fall2022/bbm103`

Text Books



- The Python Tutorial, available from the Python website.
  - This is good for explaining the nuts and bolts of how Python works.
- Introduction to Computation and Programming Using Python, Second Edition, John V. Guttag, MIT Press, August 2016
- Think Python, 2nd edition
  - Freely available online in HTML and PDF.
  - Also available for purchase as a printed book, but don't buy the first edition.
  - This book introduces more conceptual material, motivating computational thinking.
- There is an interactive version of "How to Think Like a Computer Scientist" (the first edition of "Think Python"), which lets you type and run Python code directly while reading the book.

Grading Policy

- Grading for BBM 101 will be based on
  - class participation (5%)
  - two midterm exams (25% and 30%)
  - final exam (40%)
    - You must achieve at least 30% of the final exam
- In BBM 103, the grading will be based on
  - five assignments (5% +10% + 20% + 20% + 20% = 75%)
  - six quizzes (5 * 5% = 25%)
    (the lowest 1 quiz grade will be dropped)

Attendance

- Attendance to the lectures is mandatory.
- A student who does not attend the lectures more than 4 weeks will fail BBM101 directly with an F1 grade.
- A student who does not attend more than 3 recitation sessions or does not submit more than 1 assignment will fail BBM103 directly with an F1 grade.

Academic Integrity

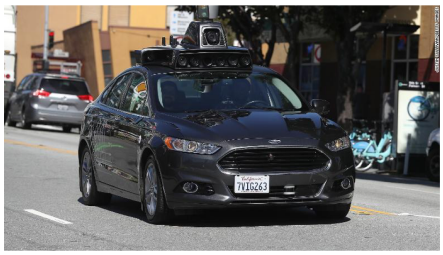
- Honest work is required of a scientist or engineer.
- Collaboration policy on the course web. **Read it!**
  - Discussion is permitted.
  - **Carrying materials from discussion is not permitted.**
  - Everything you turn in must be your own work.
    - Cite your sources, explain any unconventional action.
  - **You may not view others' work.**
  - If you have a question, ask.
- We trust you completely.
- But we have no sympathy for trust violations – nor should you!

How to Succeed

- No prerequisites
- Non-predictors for success:
  - Past programming experience
  - Enthusiasm for games or computers
- Programming and data analysis are challenging
- Every one of you can succeed
  - There is no such thing as a **"born programmer"**
  - Work hard
  - Follow directions
  - Be methodical
  - *Think* before you act
  - Try on your own, then ask for help
  - Start early

Computer Programming

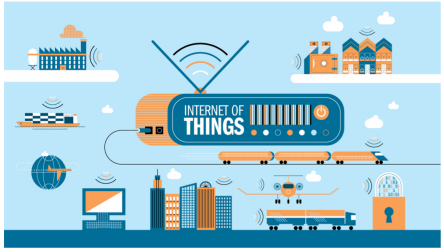
Computers are now everywhere!



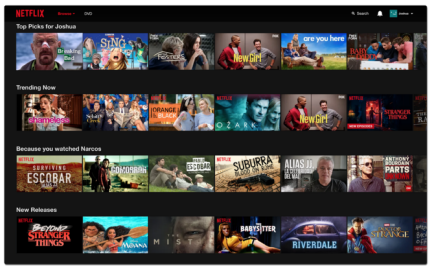
Self Driving Cars

Medical Diagnosis and Imaging

Entertainment



Internet of Things



Recommendation Systems

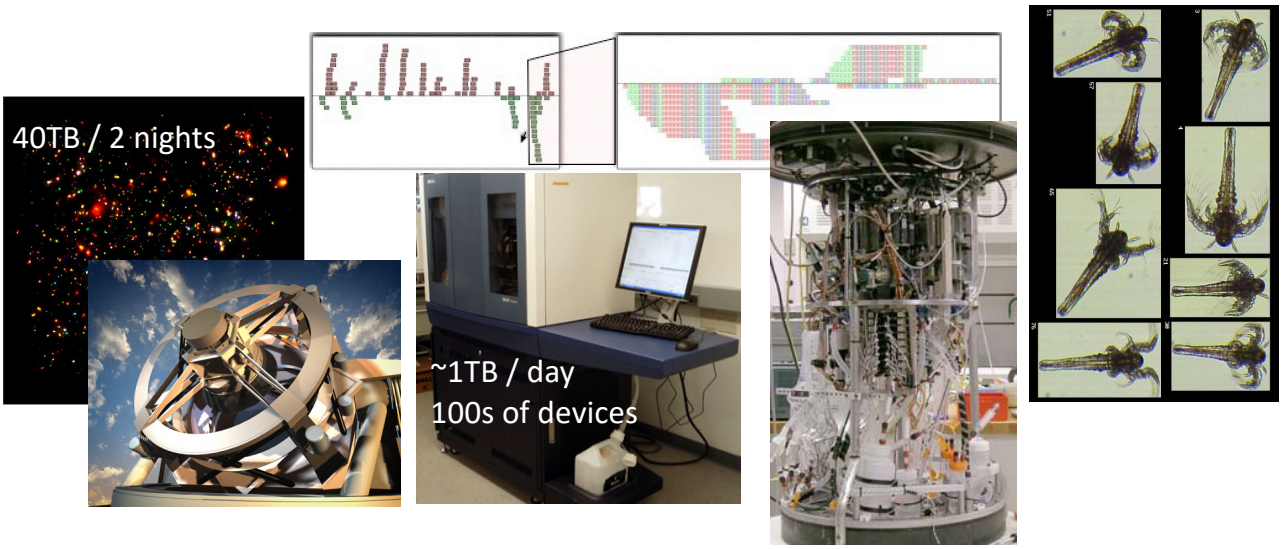
Finance

All of Science is Reducing to Computational Data Manipulation

Old model: "Query the world" (Data acquisition coupled to a specific hypothesis)

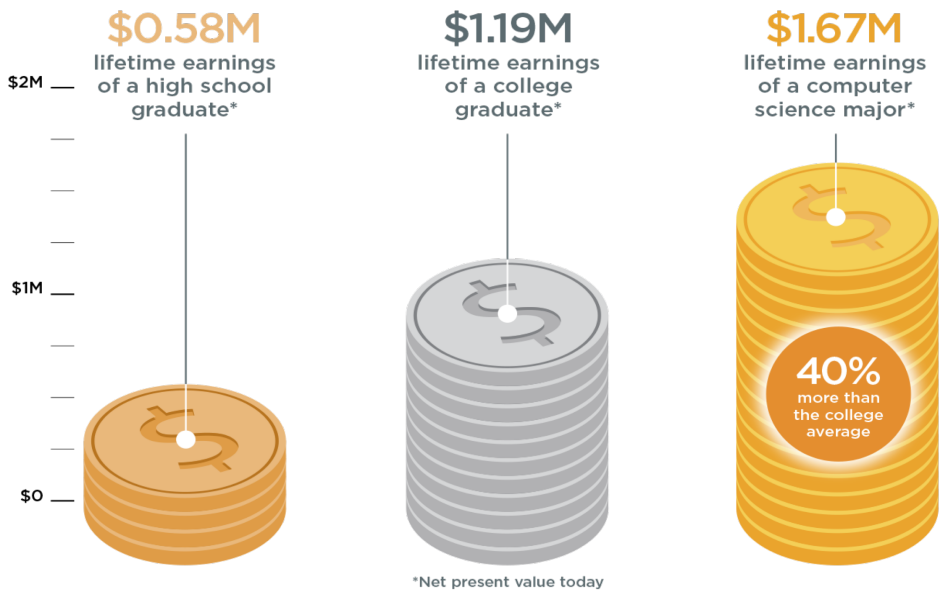
New model: "Download the world" (Data acquisition supports many hypotheses)

- Astronomy: High-resolution, high-frequency sky surveys (SDSS, LSST, PanSTARRS)
- Biology: lab automation, high-throughput sequencing,
- Oceanography: high-resolution models, cheap sensors, satellites



Some statistics (from U.S.)

The value of a computer science education

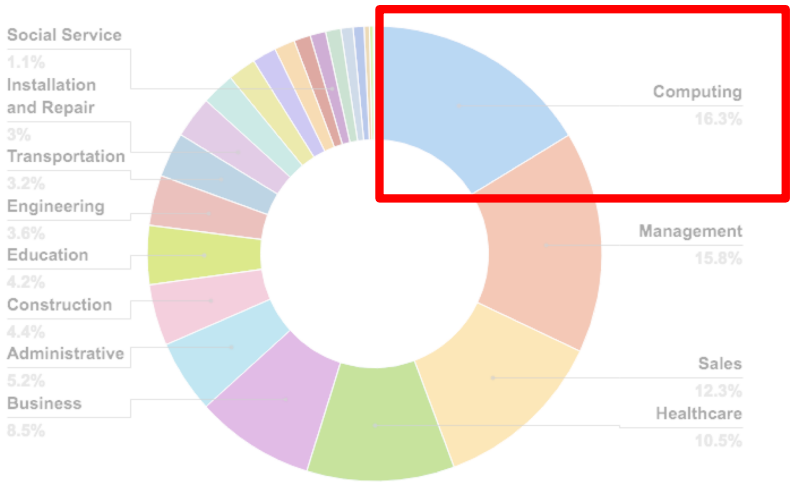


Source: Brookings

Slide credit: code.org

Some statistics (from U.S.)

Computing jobs are the #1 source of new wages in the United States

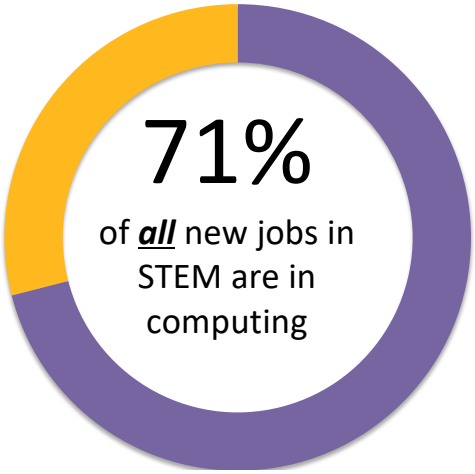


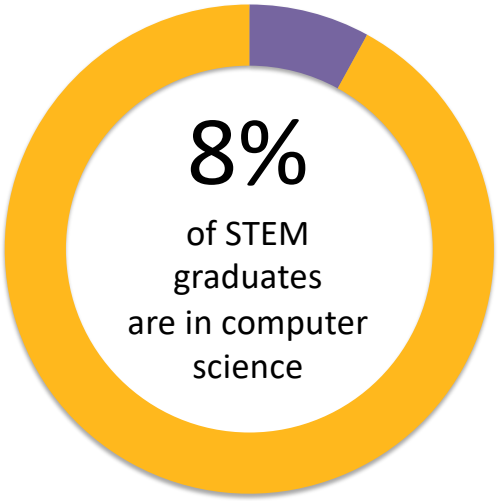
500,000 current openings: These jobs are in ***every*** industry and ***every*** state, and they're projected to grow at twice the rate of all other jobs.

Slide credit: code.org

Some statistics (from U.S.)

The STEM* problem **is in computer science:**





Sources: Bureau of Labor Statistics, National Center for Education Statistics

Slide credit: code.org

*STEM = Science, Technology, Engineering, and Math

What is meant by computation?

Some may think:

Computer science is just about learning technology

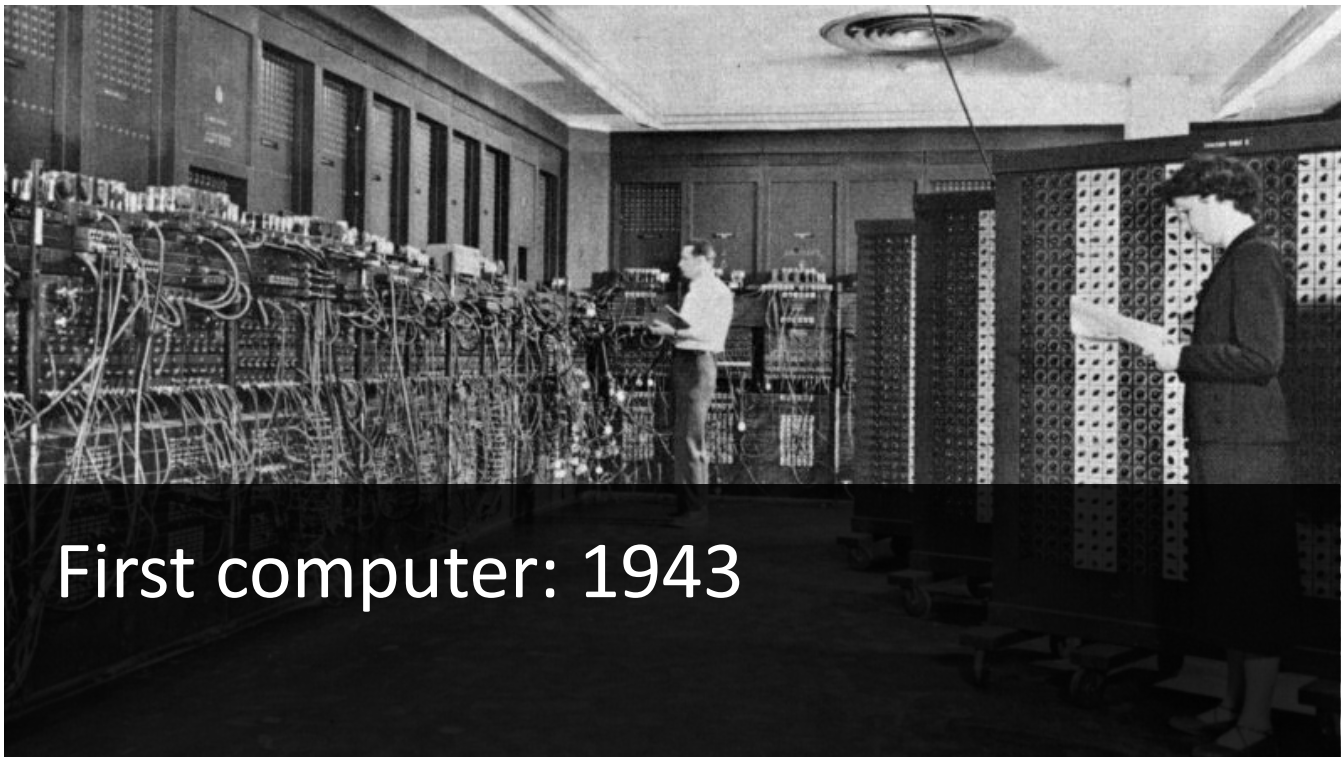
Slide credit: code.org

Some may think:

~~Computer science is just about learning technology~~

Computer science is about logic, problem solving, and creativity

Slide credit: code.org



Slide credit: code.org

Slide credit: code.org

What is a Computer?

- A device that executes a sequence of computations and instructions.
- Modern computers are electronic and digital.

Programs

- These sequences of instructions and computations is called a **program**.
- We will be designing programs in this course.
- These programs will be based on **algorithms**.
  - **Algorithm** - a step-by-step problem-solving procedure.

Where did the Term 'Computer' Originate?

- The definition from The Oxford Dictionary:

  ***"Computer (noun)**. A person who makes calculations, especially with a calculating machine."*

Courtesy of the Library of Congress

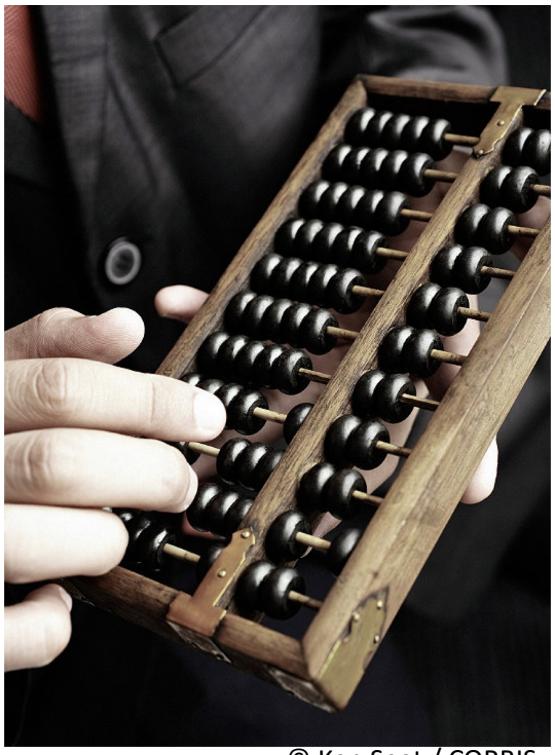
Fixed Program Computers

- Developed to solve a specific problem (set).

- Very old roots, old perspectives, …
  - Abacus
  - Antikythera Mechanism
  - Pascaline
  - Leibniz Wheel
  - Jacquard's Loom
  - Babbage Difference Engine
  - The Hollerith Electric Tabulating System
  - Atanasoff-Berry Computer (ABC)
  - Turing Bombe
  - etc.

Abacus (500 BC)

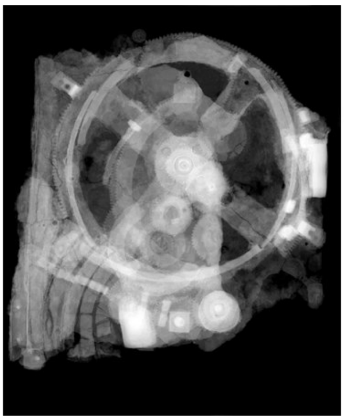
- First pocket calculator
- Still used by businessmen in Asia.



© Ken Seet / CORBIS

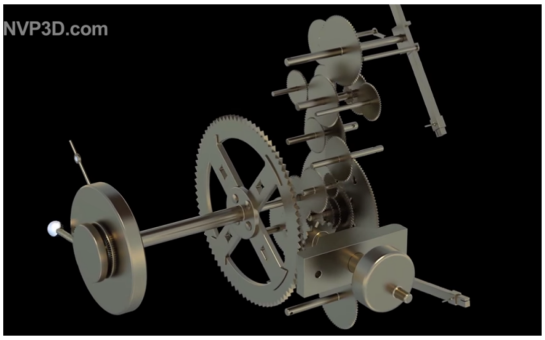
Antikythera Mechanism (100 BC)

- First analog computer
- An ancient mechanical computer designed to calculate astronomical positions



© Antikythera Mechanism Research Project

© Rien van de Weygaert



Pascaline (1642)

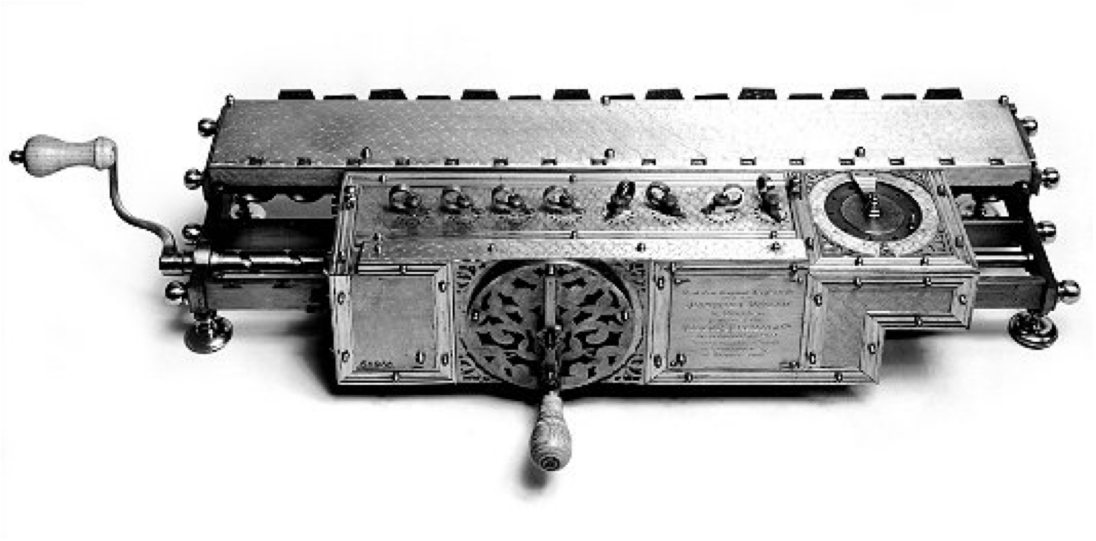
- Blaise Pascal, 1642
- A mechanical calculator for performing two arithmetic operations: addition and subtraction

© Britannica

© Mark Richards

Leibniz Wheel (1694)

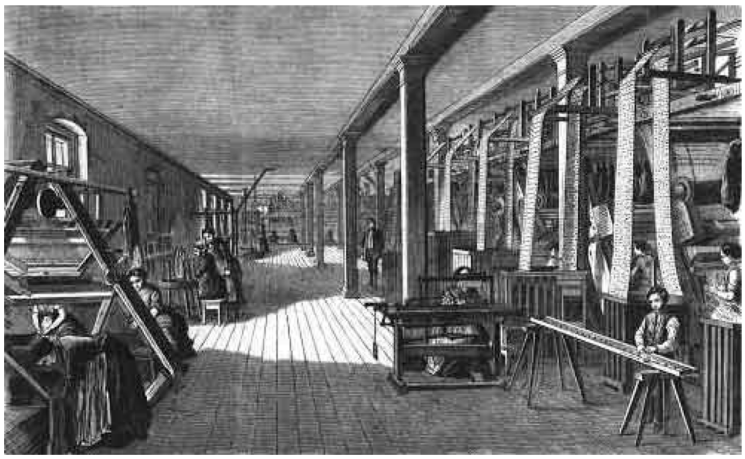
- Gottfried Wilhelm von Leibniz, 1694
- A mechanical calculator for performing all four arithmetic operations: addition, subtraction, multiplication and division



Courtesy of the Deutsches Museum, München

Jacquard's Loom (1801)

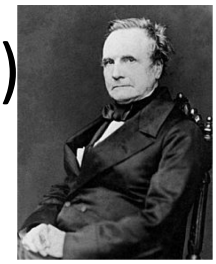
- Developed in 1801 by Joseph-Marie Jacquard.
- The loom was controlled by a loop of punched cards.
- Holes in the punched cards determined how the knitting proceeded, yielding very complex weaves at a much faster rate



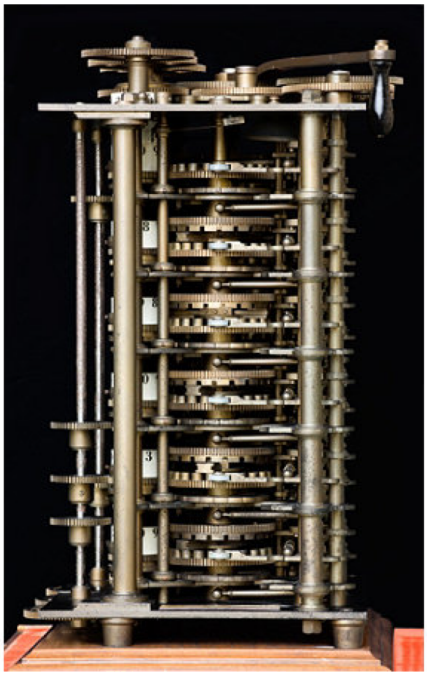
A Jacquard Loom workshop - Germany, 1858.

Adopted from: T. Cortina

Babbage Difference Engine (1832)



- Charles Babbage, 1832
- A mechanical calculator designed to tabulate polynomial functions (can be used for solving polynomial equations, curve fitting, etc.)
- A working difference engine was built in 1991 to celebrate the 200th anniversary of Babbage's birth (London Science Museum).
- It could hold 8 numbers of 31 decimal digits each and could thus tabulate 7th degree polynomials to that precision.



© Mark Richards

The Hollerith Electric Tabulating System

- **1880 Census.** Took 1,500 people 7 years to manually process data.
- **Herman Hollerith.** Developed counting and sorting machine to automate.
  - Use punch cards to record data (e.g., gender, age).
  - Machine sorts one column at a time (into one of 12 bins).
  - Typical question: how many women of age 20 to 30?

Hollerith tabulating machine and sorter

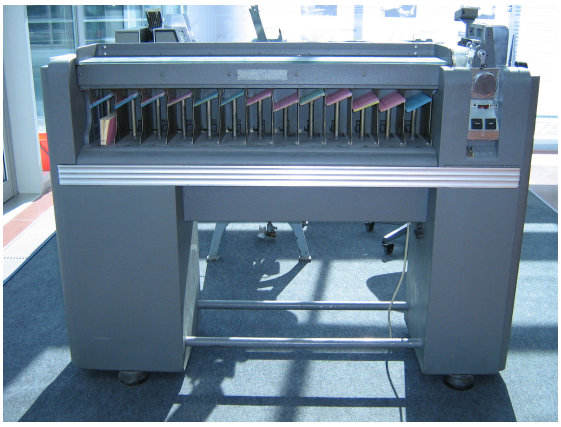
punch card (12 holes per column)

- 1890 Census. Finished months early and under budget!

Adopted from: Sedgewick and Wayne

Modern Punch Cards

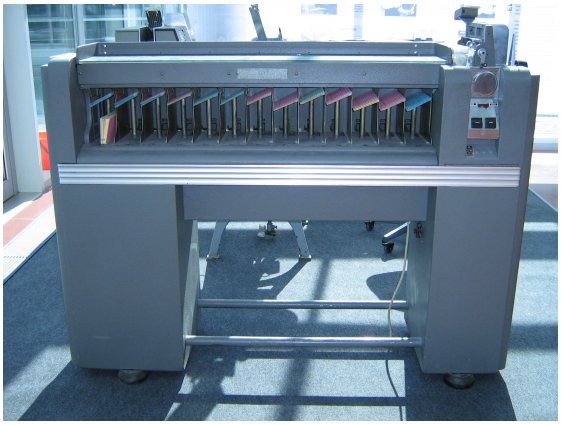
- Punch cards. [1900s to 1950s]
  - Also useful for accounting, inventory, and business processes.
  - Primary medium for data entry, storage, and processing.
- Hollerith's company later merged with 3 others to form Computing Tabulating Recording Corporation (CTRC); the company was renamed in 1924.



Adopted from: Sedgewick and Wayne

Modern Punch Cards

- Punch cards. [1900s to 1950s]
  - Also useful for accounting, inventory, and business processes.
  - Primary medium for data entry, storage, and processing.
- Hollerith's company later merged with 3 others to form Computing Tabulating Recording Corporation (CTRC); the company was renamed in 1924.

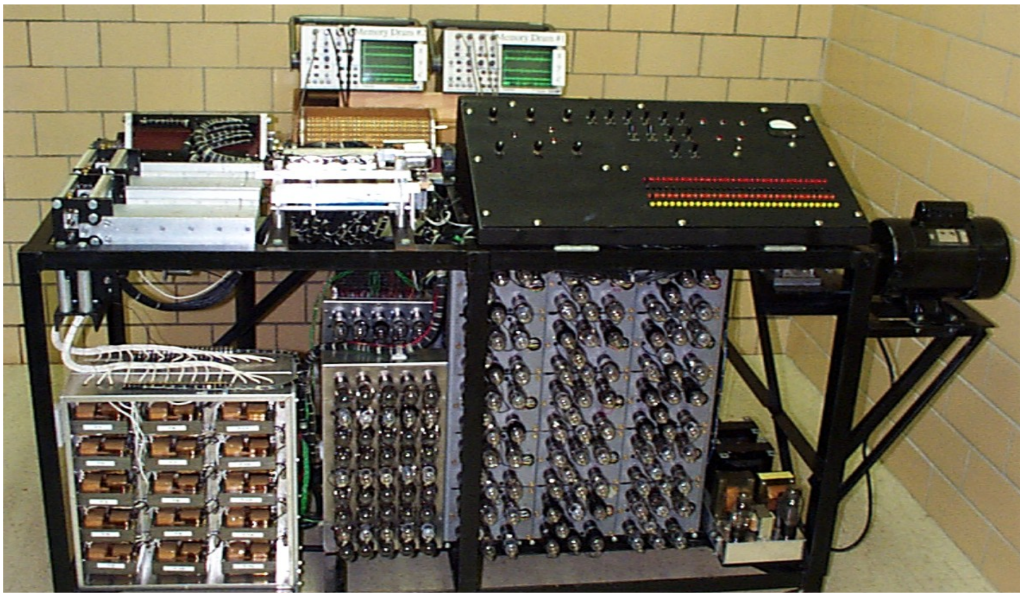


IBM 80 Series Card Sorter, 1949
(650 cards per minute)

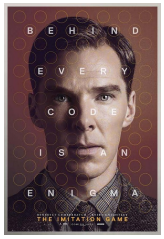
Adopted from: Sedgewick and Wayne

Atanasoff-Berry Computer (ABC) (1939)

- John Vincent Atanasoff and Clifford Berry, 1939-1942
- One of the first electronic digital computing devices
- Designed to solve a system of linear equations

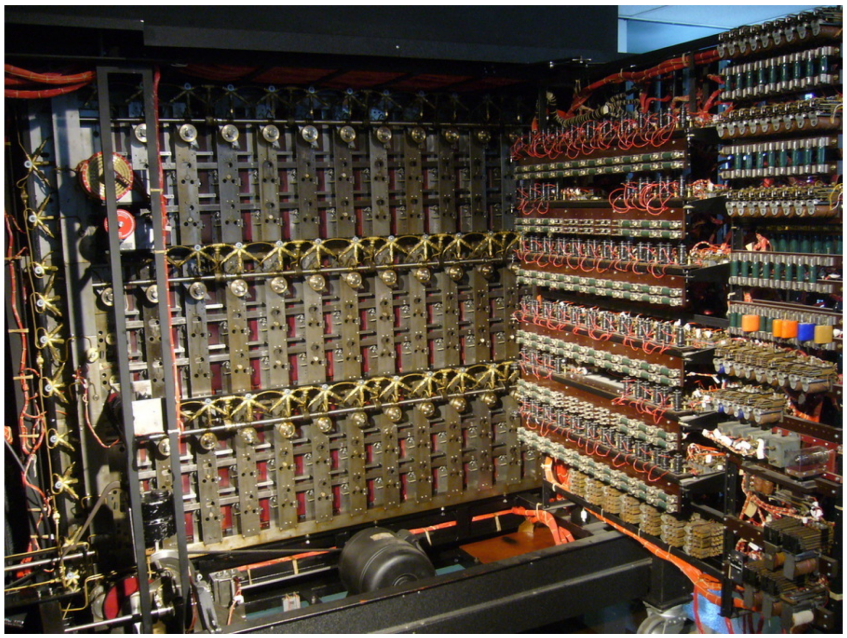


Turing Bombe (1941)

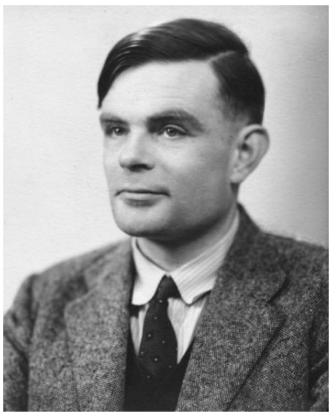


- Alan Turing, 1939
- Developed to crack German Enigma codes during WW II.

Enigma machine in use



Alan Turing



- 1912-1954
- Considered the “father” of modern computer science.
- Presented formalisms for the notions of computation and computability in the 1930’s.
- Worked at Bletchley Park in Great Britain during WWII to develop Collossus to help break the German Enigma Code.
- Developed the notion in 1950 of a test for machine intelligence now called the Turing Test.
- The Turing Award, the highest award in computing, is named in honor of Alan Turing.

Slide credit: G. Kesden

Stored Program Computers

- Problem solving

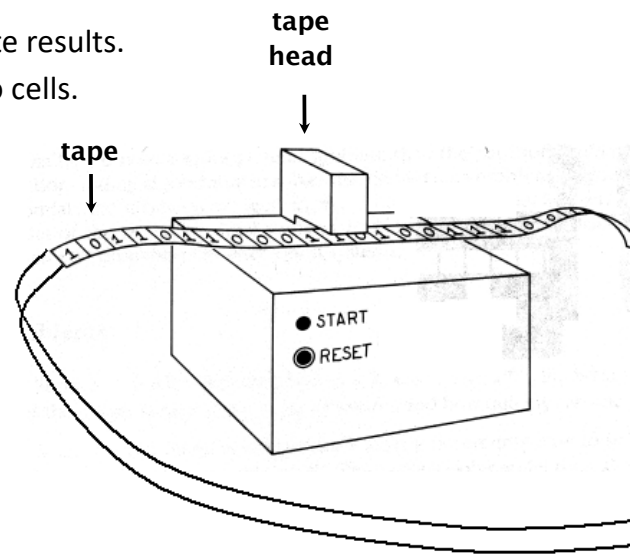
- What if input is a machine (description) itself?

- Universal Turing machines
  - An abstract general purpose computer

Universal Turing Machines

- Tape
  - Stores input, output, and intermediate results.
  - One arbitrarily long strip, divided into cells.
  - Finite alphabet of symbols.

- Tape head
  - Points to one cell of tape.
  - Reads a symbol from active cell.
  - Writes a symbol to active cell.
  - Moves one cell at a time.



					tape head								
tape	…	#	1	1	**0**	0	+	1	0	1	1	#	…

- Is there a more powerful model of computation? No!

Most important scientific result of 20th century?

Adopted from: Sedgewick and Wayne

Universal Turing Machines

- Tape

https://www.youtube.com/watch?v=E3keLeMwfHY

- Is there a more powerful model of computation? No!

Most important scientific result of 20th century?

Questions About Computation

- What is a general-purpose computer?
- Are there limits on the power of digital computers?
- Are there limits on the power of machines we can build?


David Hilbert


Kurt Gödel


Alan Turing


Alonzo Church


John von Neumann

Church-Turing Thesis (1936)

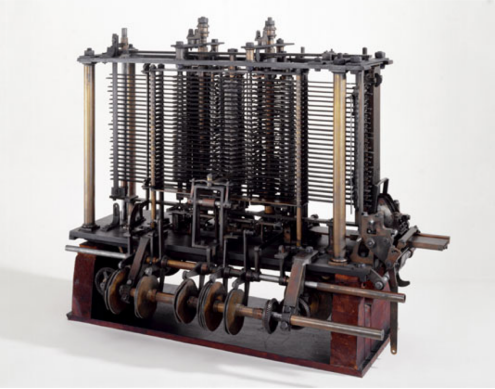
Turing machines can compute any function that can be computed by a physically harnessable process of the natural world.

- **Remark.** "Thesis" and not a mathematical theorem because it's a statement about the physical world and not subject to proof.
- Use simulation to prove models equivalent.
  - Android simulator on iPhone.
  - iPhone simulator on Android.
- **Implications.**
  - No need to seek more powerful machines or languages.
  - Enables rigorous study of computation (in this universe).
- **Bottom line.** Turing machine is a simple and universal model of computation.

Adopted from: Sedgewick and Wayne

Babbage's Analytical Engine (1834, 1836)

- Designed around 1834 to 1836
  - was to be a universal machine capable of any mathematical computation
  - embodies many elements of today's digital computer
  - a control unit with moveable sprockets on a cylinder that could be modified
  - separated the arithmetic operations (done by the mill) from the storage of numbers (kept in the store)
    - store had 1000 registers of 50 digits each
  - Babbage incorporated using punched cards for input
    - idea came from Jacquard loom
- Never built by Babbage due to lack of funds and his eventual death in 1871



Adopted from: T. Cortina

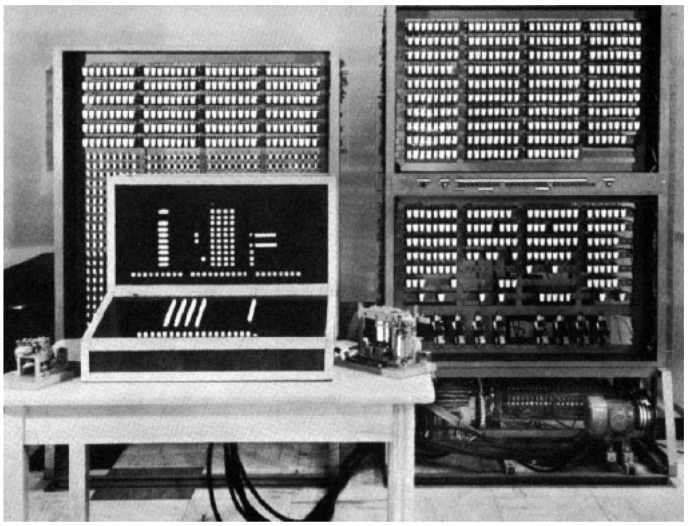
Ada Lovelace

- 1815-1852
- Daughter of poet Lord Byron
- Translated Luigi Menabrea's article on Babbage's Analytical Engine to English
  - Quadrupled its length by adding lengthy notes and detailed mathematical explanations
- Referred to as the world's first programmer
  - Described how the machine might be configured (programmed) to solve a variety of problems.

Slide credit: G. Kesden

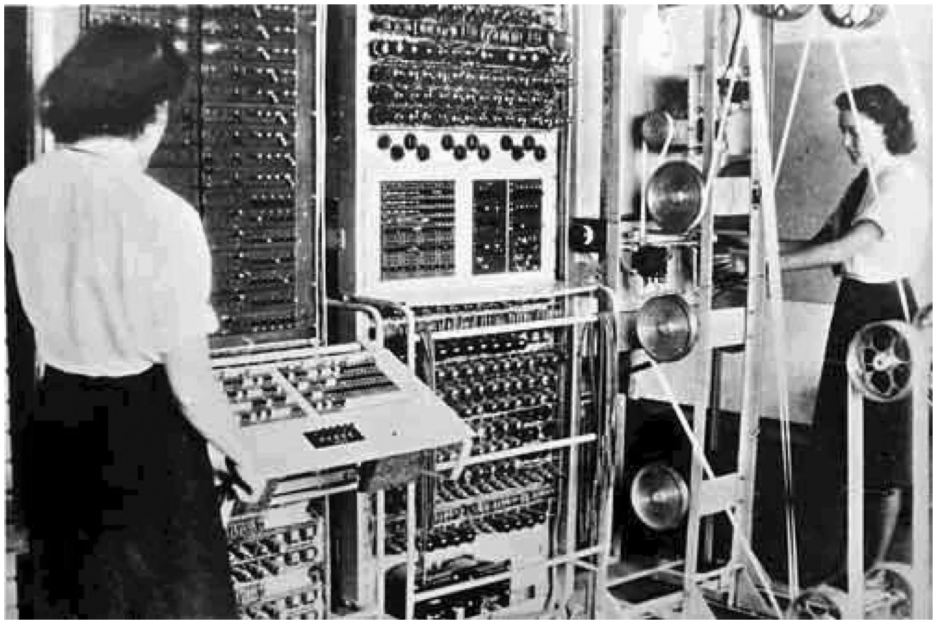
The Zuse Z3 Computer (1941)

- Konrad Zuse, 1941
- The original Z3 was destroyed in a bombing raid of Berlin in 1943.
- Zuse later supervised a reconstruction of the Z3 in the 1960s (currently on display at the Deutsches Museum in Munich)



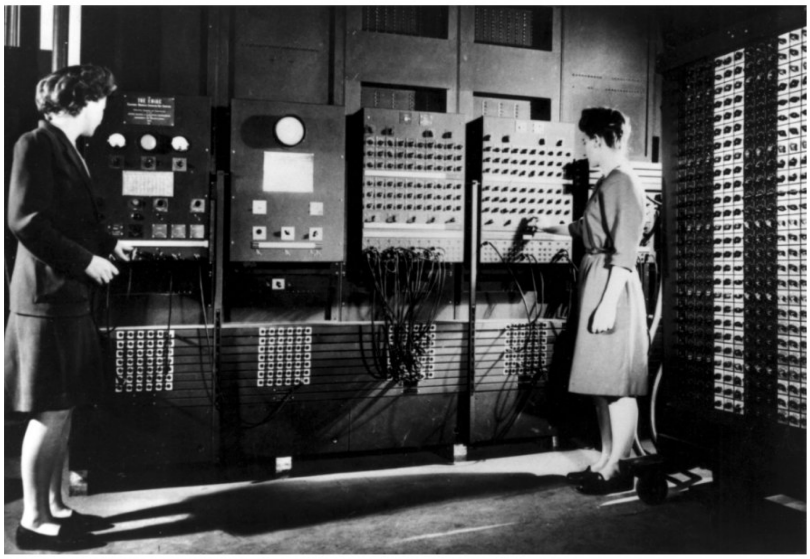
Colossus Mark 1 (UK,1944)

- The world's first electronic digital computer with programmability.



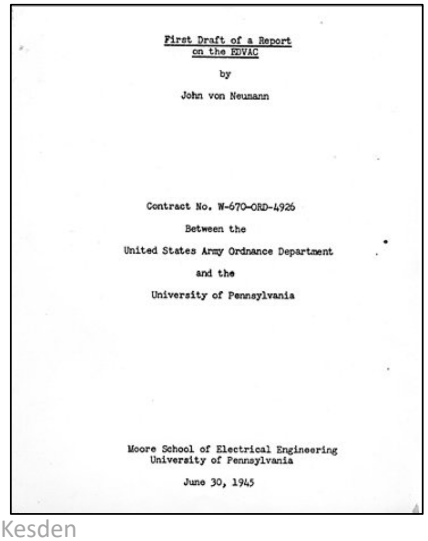
ENIAC (Mauchly and Eckert, USA, 1946)

- The first large-scale general-purpose electronic computer without any mechanical parts.
- Designed to calculate artillery firing tables for the United States Army's Ballistic Research Laboratory



EDVAC (von Neuman, USA, 1951)

- Unlike the ENIAC, it uses binary rather than decimal numbering system
- Instructions were stored in memory sequentially with their data
- Instructions were executed sequentially except where a conditional instruction would cause a jump to an instruction someplace other than the next instruction



First Draft of a Report
on the EDVAC

by

John von Neumann

Contract No. W-670-ORD-4926

Between the

United States Army Ordnance Department

and the

University of Pennsylvania

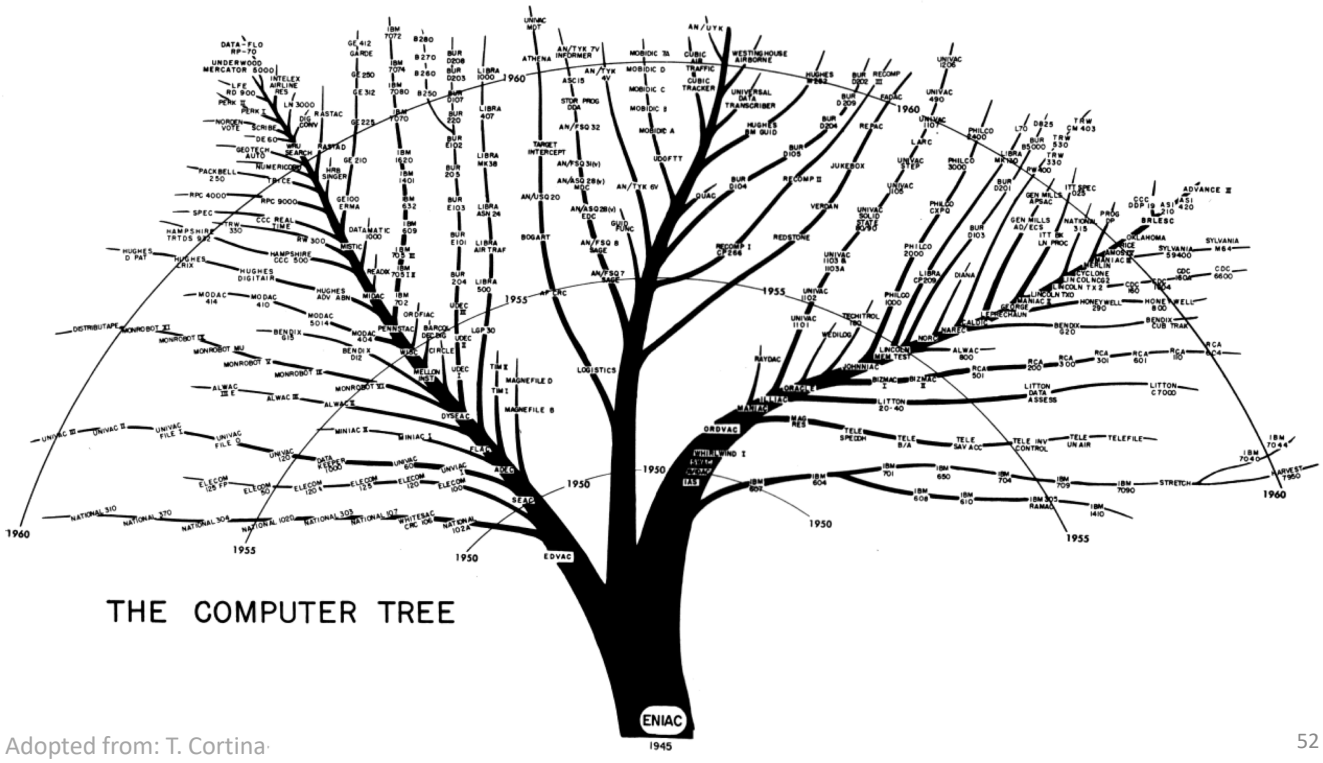
Moore School of Electrical Engineering
University of Pennsylvania

June 30, 1945

Adopted from G. Kesden

The Computer Tree

- http://ftp.arl.mil/~mike/comphist/61ordnance/chap7.html



Adopted from: T. Cortina

What is Knowledge?

- **Declarative knowledge**
  - Axioms (definitions)
  - Statements of fact

"y is the square root of x if and only if y*y = x"

does not help to find the square root!

What is Knowledge? (cont'd.)

- **Declarative knowledge**
  - Axioms (definitions)
  - Statements of fact

"y is the square root of x if and only if y*y = x"

does not help to find the square root!

- **Imperative knowledge**
  - How to do something
  - A sequence of specific instructions (what computation is about)

Babylonian method

Get x as an input

1. Begin with an arbitrary positive number y_0 *(an initial guess)*
2. If $y_n^2 \approx x$, stop *(found the solution - y_n)*
   Else let $y_{n+1} = (y_n + x/y_n)/2$ *(use the arithmetic mean to approximate the geometric mean)*
3. Repeat step (2)

What is Knowledge? (cont'd.)

- **Another example** – Estimating greatest common divisor (gcd)

Declarative definition
"d is the gcd of a and b if and only if d is the largest possible integer satisfying a = d*x and b = d*y with x and y being two positive integers"

Imperative definition: Euclid's formula
Get 2 positive integers a and b, a>=b as input

1. Divide a by b, call the remainder R
2. If R = 0, stop
   Else let a = b and b = R
3. Repeat step 2

(found the solution - b)

Use Euclid's formula to compute gcd(48,18).

What's in Computer Science?

- Abstraction

- Problem Solving!
- Artistic, Creative.
  - e.g. Digital Media, Electronic Music, Games, Animation.

- Science.
  - e.g. Understand and model reality.

Algorithms: Takeaway

- **Definition:** An *algorithm* is a recipe for solving a problem.

- Computer science is (loosely) the study of algorithms.

Algorithms: Takeaway

- **Definition:** An *algorithm* is a recipe for solving a problem.

- Computer science is (loosely) the study of algorithms.

- That is, computer science is the study of *automated methods of solving problems.*

Algorithms: Takeaway

- **Definition:** An *algorithm* is a recipe for solving a problem.
- Computer science is (loosely) the study of algorithms.
- That is, computer science is the study of *automated methods of solving problems.*
- **Programs are ways of carrying out algorithms!!!**

Problem Specification

- **A specification defines a problem**

Problem Specification

- **A specification defines a problem**
- **An algorithm solves a problem**

Problem Specification

- *INPUT: Some stuff!*
- *OUTPUT: Information about the stuff!*

Problem Specification Examples

- *INPUT: Two numbers, X and Y.*
- *OUTPUT: A single number, Z, such that Z = X + Y.*

Problem Specification Examples

- *INPUT: Some doctor's knowledge about cancer.*
- *OUTPUT: Cure to cancer*

Problem Specification Examples

- *INPUT: The Internet*
- *OUTPUT: The winner of the 2020 election*

Problem Specification Examples

- *INPUT: Map of solar system, description of physical laws, summary of current technology.*
- *OUTPUT: A method for colonizing Mars.*

Problem Specification Examples

- *INPUT: Data from the stock market.*
- *OUTPUT: Correct predictions about the market.*

What is an Algorithm?

- A procedure for solving a mathematical problem in a finite number of steps that frequently involves repetition of an operation.
- Step-by-step method for accomplishing some task.
- Algorithms are expressed using Pseudocodes.

Pseudocode

- English (or any) language constructs modelled to look like statements in most programming languages.
- Steps presented in a structured manner (numbered, indented, and so on)
- No fixed syntax for most operations is required

Pseudocode -2

- Less ambiguous and more readable than natural language.
- Emphasis is on process, not notation.
- Well-understood forms allow logical reasoning about algorithm behaviour.
- Can be easily translated into a programming language.
- A bridge between problem owner and problem solver (programmer).

Types of algorithmic operations

- Sequential operations
- Conditional operations
- Iterative operations

All of the programming languages contain these atomic operations.

Sequential operations

- Input operations
  - To receive data values from the outside world
  - Example
    - Get a value for ***w***, the weight of a person.
    - Get a value for ***h***, the height of a person.
- Computation operations
  - To set the value of "variable".

  (V*ariable: Named storage location that can hold a data value)*
  - To make an arithmetic expression.
  - Example
    - Set the value of **BMI** with **h / (w * w)**
- Output operations
  - To send results to the outside world for display
  - Example
    - Print the value of ***BMI*** *(body mass index)*

Example – Add two numbers

1. Start
2. Get number a
3. Get number b
4. Compute c as b+a
5. Print c
6. End

Example – Area of Rectangle

1. Start
2. Get number b as base of rectangle
3. Get number h as height of rectangle
4. Compute area <= b * h
5. Print area
6. End

Conditional operations

- Ask questions and choose alternative actions based on the answers.
  - Example
    - if *x* is greater than 100 then
      print *x*
      else
      add 100 to x

- There might be a more complicated question. The answer should be logical (True or False)
  - Example
    - if (*x* is greater than 100) and (y is equal to 200) ….

Example- Get two numbers, print the largest

1. Start
2. Get first number as f
3. Get second number as s
4. If f > s
   print f
5. Else
   print s
6. End

Iterative operations

- Perform "looping" behavior, repeating actions until a continuation condition becomes <u>false</u>.

- The repetition of a block of instructions
  - Examples
    - while $j > 0$ do

      set s to $s + a_j$

      set j to $j - 1$
    - repeat

      print a_k

      set k to $k + 1$

      until $k > n$

Example – Find the largest number in a list with positive numbers

1. Start
2. Largest = 0
3. While there is an item to check

   If item > largest

   largest = item
4. Print largest
5. End

Conditional and Iterative Operations

- Components of a loop
  - Continuation condition
  - Loop body
- Infinite loop
  - The continuation condition never becomes false
  - An error

Summary

- **What is computation?**
  - History of computing
  - What is a computer?
  - What is knowledge?
  - What is a program?
  - What is an algorithm?

The Birth of the Computer

- A TED talk given by George Dyson

http://www.ted.com/talks/george_dyson_at_the_birth_of_the_computer.html

The Map of Computer Science

https://www.youtube.com/watch?v=SzJ46YA_RaA