CSCI 161
Introduction to Computer Science

UNIVERSITY of PUGET SOUND
Est. 1888

Department of Mathematics
and Computer Science

Lecture 1
Course overview &
What Are Algorithms?

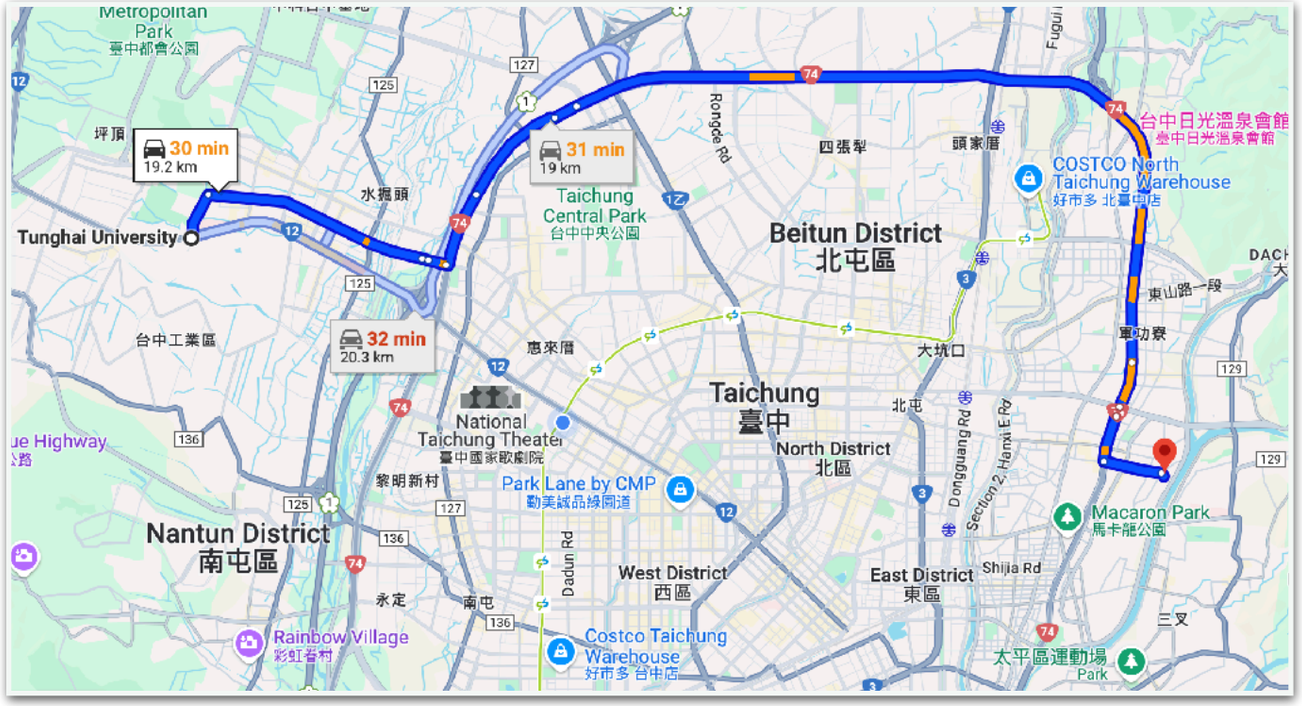
Important

- Two important webpages to bookmark.
  - Course Calendar: tinyurl.com/chiuTHU
    - Notes and code examples
  - iLearn: ilearn.thu.edu.tw
    - Assignment submission

- How to reach me
  - Email: davidchiu@go.thu.edu.tw
  - Line:

Origins in Taichung

- Spent early childhood in Taiwan 1980 - 1985.



About Me

- Moved to Ohio in 1989.
- PhD, Ohio State University in 2010.
- Assistant Professor at Washington State University, 2010-2014
- Full Professor at University of Puget Sound, 2014-now

Ohio State University

Washington State University, Vancouver

My College Campus

- University of Puget Sound (Tacoma, WA)
  - ~ 2000 undergraduate students
  - 60 km south of Seattle, WA (30 minute drive)
  - "Sound?" What sound?

University of Puget Sound Campus

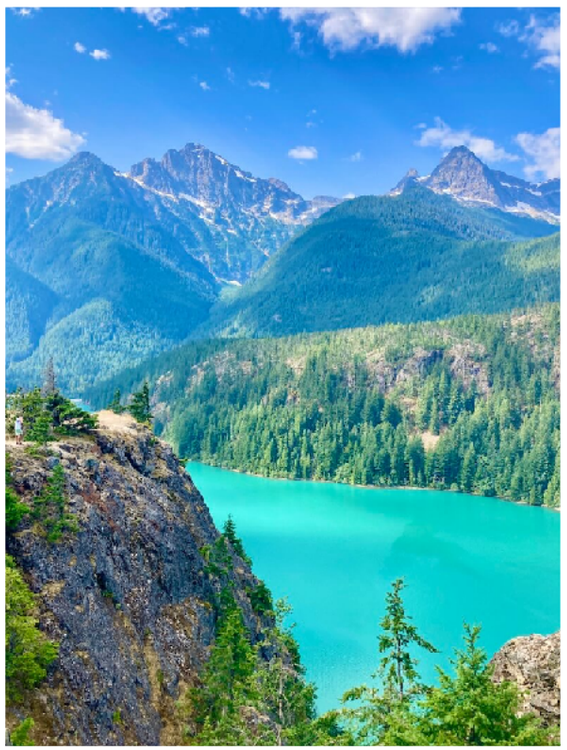
Tacoma, Washington

Seattle, Washington

Puget Sound Surroundings

- Washington state has 3 National Parks
  - Mount Rainier National Park
  - Olympic National Park
  - North Cascades National Park

Gig Harbor, Washington



North Cascades

Olympic

Mount Rainier

Who Are We?

- Please share
  - Your preferred name?
  - Your major?
  - Have you done any kind of programming before?
  - Finally, answer **any one** of the following:

    - What's your go-to comfort food or snack?
    - If you could instantly become an expert in one thing, what would it be?
    - What's a movie or show you could rewatch forever?
    - Do you collect anything? (cards, pins, books, plants…)
    - What's the most unusual thing you've ever eaten?

Grading

- Breakdown
  - 35% Lab Assignments
  - 35% Homework assignments
  - 20% Weekly Quizzes (Fridays)
  - 10% Participation, calculated as follows:
    - $\frac{meetingDays - absentDays}{meetingDays} \times 100$
    - and,
    - $2\ lateDays = 1\ absentDay$

Lab Assignments (Small Group of 2-3)

- Almost daily, done in class. May finish outside of class
  - 2 points = Completed
  - 1 points = Incomplete
  - 0 points = Did not show up, or did not submit
  - Due at 11:59pm, same day.

- Policies
  - Bring your laptop **and** charger to class each day.
  - Strongly encouraged to work with a friend or two.
    - Stay on task and don't disrupt others!
  - Stay off phone and social media!

Homework Assignments (Paired)

- Homework Assignments
  - One homework assigned each week
  - You'll be assigned a different partner each time
  - Due at 11:59pm on specified due date
    - Late penalty: $-3^d\,\%$, where d = days late

- Lots of logic puzzles
  - Leads to **creative** and **critical thinking**
  - Work in pairs, but you can brainstorm with others
    - Copy/pasting others' code or code you find on ChatGPT or Web = academic dishonesty!
    - (More on allowed uses of ChatGPT later)

Weekly Quizzes

- Occurs every Friday
  - Emphasizes on that week's course materials
  - Allowed: 1 page of notes and as much scratch paper as you need.

- Format:
  - Length: 20-30 minutes
  - 5-6 questions per quiz
  - Conceptual questions, code-reading questions, and small code writing questions.

Classroom Participation

- Participation is expected
  - **Please raise your hand** to answer questions 
    - (Instead of blurting out answers while others are still thinking.)
  - I sometimes do small-group work
  - I may ask you to do some "code alongs" with me

- Be attuned to how you present yourself to others
  - For many here, programming is completely new.
  - Don't try to impress others.

Course Policies

- Don't be late.
- Class Disruption
  - Put phone on silent
  - Put laptops away, unless instructed to take them out
    - (Take notes on paper)

- Cheating
  - It's never been easier to compare assignments with ones from the past
  - OK to brainstorm, but *you must write your own code*
  - 0 on assignment + formal report to Tunghai University

AI Policy: Do

- **Do** use it to be your personalized tutor, and that is as far as it should go. Use it to explain concepts you don't fully grasp.
  - Prompt: *"**Without giving me code, explain …**"*

- **Do** use it to explain code to you. Is there a piece of code we went over in class that's hard to grasp? Paste it, and have it explain line-by-line as well as holistically.

- **Do** use it to explain errors to you. Paste your code and the errors you get when compiling or running it. Give hints on what might be the issue.

- **Do** use it to explain any math concepts that you may need to know.

AI Policy: Don't

- **Don’t turn in** anything that was generated by these tools. Copying-and-pasting AI output is considered plagiarism and will be treated as such.

- **Don’t underestimate** how easy it is for us to detect cases where students are turning in code written by generative AI tools.

- **Don’t forget** that you *still* need to demonstrate proficiency on all your exams to pass the course.

Outline

- Course Syllabus
- What Is Computer Science?
  - What are algorithms?
- Conclusion

What Is Computer Science (CS)?

- What is computer science?

"Computer Science is no more about computers than astronomy is about telescopes."

- CS pioneer, Edsger Dijkstra (1930-2002)

- CS is **not:**
  - Using computers competently
  - Building, repairing, troubleshooting computers
  - Coding
  - *(Though, you'll become proficient in the above skills via exposure)*

What Are Algorithms?

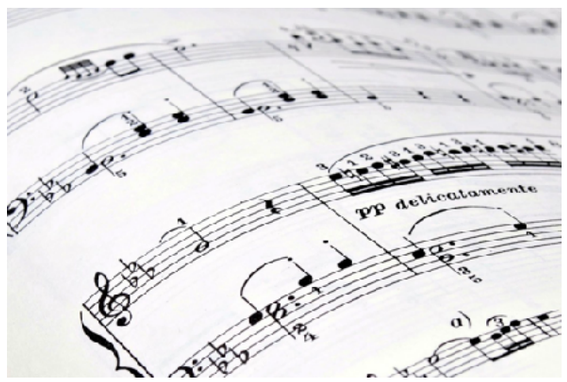
- An *"Algorithm"* is a finite, well-defined sequence of instructions that, when followed (with or without input), produces an output and solves a problem.
  - "finite?"
  - "well-defined?"
  - "input?"
  - "output?"

- What are some algorithms that you apply in your everyday lives?

Real Life Algorithms

- We *"program or code"* + *"execute"* algorithms *all the time!*
  - A recipe to bake an apple pie (*recording* + *baking*)
  - Sheet music (*composing* + *playing*)
  - Formula to find the area of a circle (*recording* + *calculating*)
  - Instructions to set up your wifi

- **Can you think of more?**
- **Key:** algorithms are precise, pre-defined mechanical processes
  - Anyone (or, any *thing*) can carry it out!



Writing an Algorithm

- Pair up with another student, and work together
- **Task:**
  - On a sheet of paper, write an algorithm to purchase a drink at a vending machine.
- **Think critically about:**
  - What inputs, if any, are needed?
  - The desired output?

Class Discussion

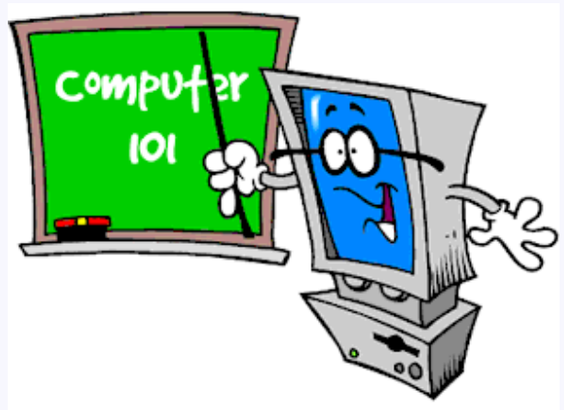
- Did any of your steps assume background knowledge or human intuition?
- Was every instruction unambiguous? Could anyone follow it exactly?
- What would happen if it followed your steps literally?
- What does your algorithm say if the machine is out of stock?
- What if the user doesn’t have enough money?
- What if the selected drink gets stuck — how should the system respond?

Characteristics of Algorithms

- Characteristics of algorithms
  - An algorithm solves a well-defined problem
    - But *some* problems can't be solved algorithmically, so you should know that algorithms (and computers) have limits!
  - There can be *many* algorithms that solve the *same* problem
    - Some algorithms are faster, more elegant and aesthetically pleasing than others.
    - *This is the art of CS!*
  - Algorithms are repeatable and measurable
    - Given the same input, algorithm produce the same result every time.
    - Its performance, energy, space, can all be measured!
    - *This is the science of CS!*

Definition: What Is Computer Science?

- Computer Science is the *study of algorithms*, including their
  - Formal and mathematical properties,
  - Hardware realizations,
  - Linguistic realizations, and
  - Applications

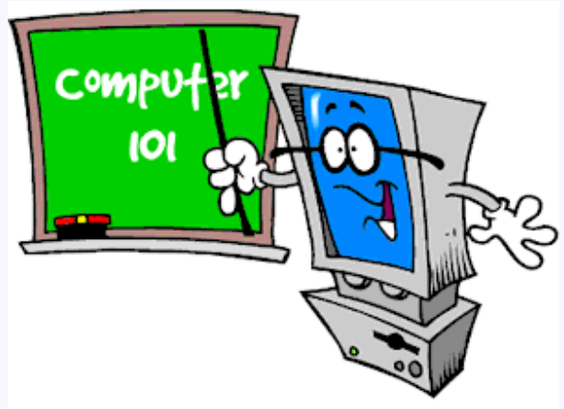


Definition from: Schneider and Gersting. "An Invitation to Computer Science."

- Computers are just tools (hardware) that can carry out algorithms!
  - "Computer Science" is a misnomer
    - It's like calling Chemistry, "Beaker Science"*!*

Definition: What Is Computer Science?

- Computer Science is the *study of algorithms*, including their
  - **Formal and mathematical properties,**
  - Hardware realizations,
  - Linguistic realizations, and
  - Applications



Definition from: Schneider and Gersting. "An Invitation to Computer Science."

- Computers are just tools (hardware) that can carry out algorithms!
  - "Computer Science" is a misnomer
    - It's like calling Chemistry, "Beaker Science"*!*

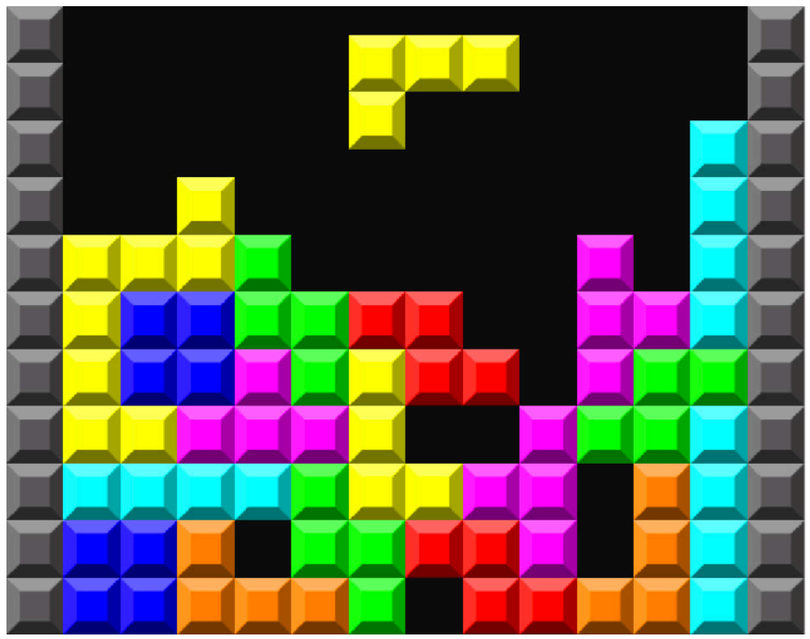
Formal and Mathematical Properties

- There are two fundamental questions in CS theory:
  - Given a yes/no problem, is the problem:
    - Decidable? Can it even be *solved* algorithmically?
    - Hard? If it's solvable, how long would it take to solve it?

- Decidability examples:
  - "Is *P* the shortest path from city *A* to city *B*?"
    - This problem is **decidable**!
  - "Are [3,29,2,0,57] tomorrow's lottery numbers?"
    - This problem is **not decidable!**
    - (Can't solve this one with algorithms)

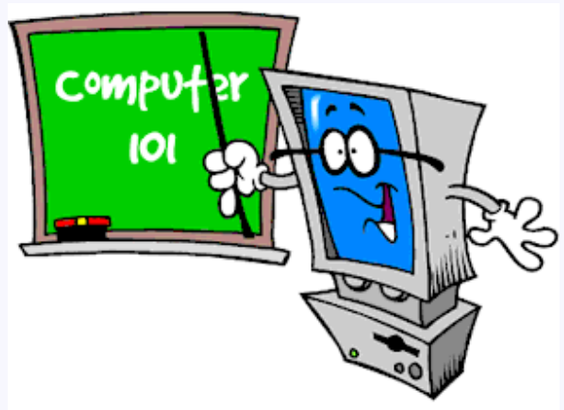
Formal and Mathematical Properties (2)

- Complexity (Hardness): Can a decidable problem be solved in a reasonable amount of time?
  - Determining the shortest path between City *A* and City *B* is considered easy.
  - "Given a sequence of pieces in Tetris, what's the best way to arrange them?" Hard
    - No known algorithm exists that can answer this question before the end of humanity.



What Is Computer Science? (Cont.)

- Computer science is the *study of algorithms*, including their
  - Formal and mathematical properties,
  - **Hardware realizations,**
  - Linguistic realizations, and
  - Applications



Definition from: Schneider and Gersting. An Invitation to Computer Science.

The First Computers... Humans

- So, *how do we <u>execute</u> an algorithm?*
  - The original computer hardware were... human!
    - "Computers" were a job title even up until the 1970s. (Watch "Hidden Figures")

1700s - 1800s: Origins of Computing

- Importance of data tables
  - Used for engineering, studying astronomy, ...
  - Calculations were done by hand. Lacked precision and accuracy

Example: Table of Logarithms

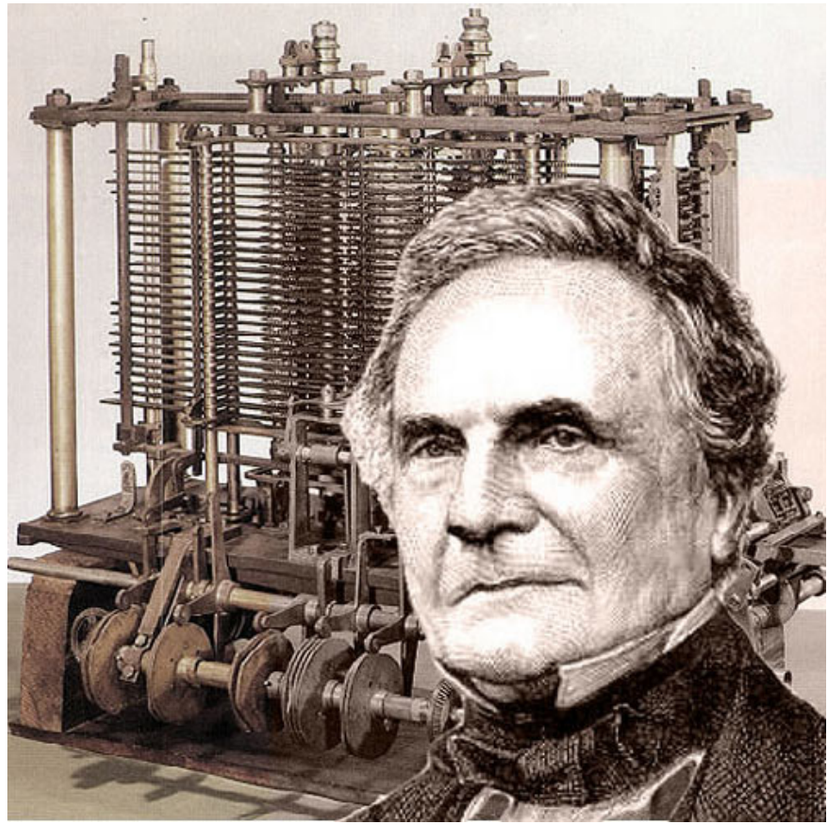
82

N.	Log.	D.	N.	Log.	D	N.	Log.	D.
7110	3.85187		7140	3.85370		7170	3.85552	
7111	3.85193	6	7141	3.85376	6	7171	3.85558	6
7112	3.85199	6	7142	3.85382	6	7172	3.85564	6
7113	3.85205	6	7143	3.85388	6	7173	3.85570	6
7114	3.85211	6	7144	3.85394	6	7174	3.85576	6
7115	3.85217	6	7145	3.85400	6	7175	3.85582	6
7116	3.85224	7	7146	3.85406	6	7176	3.85588	6
7117	3.85230	6	7147	3.85412	6	7177	3.85594	6
7118	3.85236	6	7148	3.85418	6	7178	3.85600	6
7119	3.85242	6	7149	3.85425	7	7179	3.85606	6
7120	3.85248	6	7150	3.85431	6	7180	3.85612	6
7121	3.85254	6	7151	3.85437	6	7181	3.85618	6
7122	3.85260	6	7152	3.85443	6	7182	3.85625	7
7123	3.85266	6	7153	3.85449	6	7183	3.85631	6
7124	3.85272	6	7154	3.85455	6	7184	3.85637	6
7125	3.85278	6	7155	3.85461	6	7185	3.85643	6
7126	3.85285	7	7156	3.85467	6	7186	3.85649	6
7127	3.85291	6	7157	3.85473	6	7187	3.85655	6
7128	3.85297	6	7158	3.85479	6	7188	3.85661	6
7129	3.85303	6	7159	3.85485	6	7189	3.85667	6
7130	3.85309	6	7160	3.85491	6	7190	3.85673	6
7131	3.85315	6	7161	3.85497	6	7191	3.85679	6
7132	3.85321	6	7162	3.85503	6	7192	3.85685	6
7133	3.85327	6	7163	3.85509	6	7193	3.85691	6
7134	3.85333	6	7164	3.85516	7	7194	3.85697	6
7135	3.85339	6	7165	3.85522	6	7195	3.85703	6
7136	3.85345	6	7166	3.85528	6	7196	3.85709	6
7137	3.85352	7	7167	3.85534	6	7197	3.85715	6
7138	3.85358	6	7168	3.85540	6	7198	3.85721	6
7139	3.85364	6	7169	3.85546	6	7199	3.85727	6
7140	3.85370	6	7170	3.85552	6	7200	3.85733	6

83

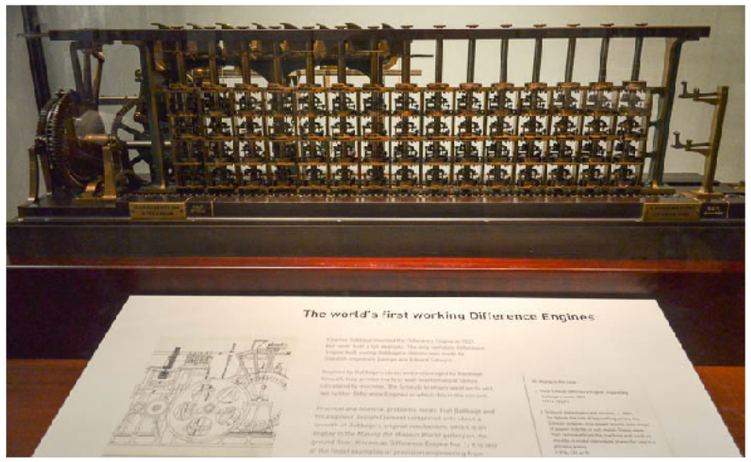
N.	2. 0′ 0″ Log.	D.	N.	2. 0′ 30″ Log.	D.	N.	2. 1 0 Log.	D.
7200	3.85733		7230	3.85914		7260	3.86094	
7201	3.85739	6	7231	3.85920	6	7261	3.86100	6
7202	3.85745	6	7232	3.85926	6	7262	3.86106	6
7203	3.85751	6	7233	3.85932	6	7263	3.86112	6
7204	3.85757	6	7234	3.85938	6	7264	3.86118	6
7205	3.85763	6	7235	3.85944	6	7265	3.86124	6
7206	3.85769	6	7236	3.85950	6	7266	3.86130	6
7207	3.85775	6	7237	3.85956	6	7267	3.86136	5
7208	3.85781	6	7238	3.85962	6	7268	3.86141	6
7209	3.85788	7	7239	3.85968	6	7269	3.86147	6
7210	3.85794	6	7240	3.85974	6	7270	3.86153	6
7211	3.85800	6	7241	3.85980	6	7271	3.86159	6
7212	3.85806	6	7242	3.85986	6	7272	3.86165	6
7213	3.85812	6	7243	3.85992	6	7273	3.86171	6
7214	3.85818	6	7244	3.85998	6	7274	3.86177	6
7215	3.85824	6	7245	3.86004	6	7275	3.86183	6
7216	3.85830	6	7246	3.86010	6	7276	3.86189	6
7217	3.85836	6	7247	3.86016	6	7277	3.86195	6
7218	3.85842	6	7248	3.86022	6	7278	3.86201	6
7219	3.85848	6	7249	3.86028	6	7279	3.86207	6
7220	3.85854	6	7250	3.86034	6	7280	3.86213	6
7221	3.85860	6	7251	3.86040	6	7281	3.86219	6
7222	3.85866	6	7252	3.86046	6	7282	3.86225	6
7223	3.85872	6	7253	3.86052	6	7283	3.86231	6
7224	3.85878	6	7254	3.86058	6	7284	3.86237	6
7225	3.85884	6	7255	3.86064	6	7285	3.86243	6
7226	3.85890	6	7256	3.86070	6	7286	3.86249	6
7227	3.85896	6	7257	3.86076	6	7287	3.86255	6
7228	3.85902	6	7258	3.86082	6	7288	3.86261	6
7229	3.85908	6	7259	3.86088	6	7289	3.86267	6
7230	3.85914	6	7260	3.86094	6	7290	3.86273	6

Charles Babbage

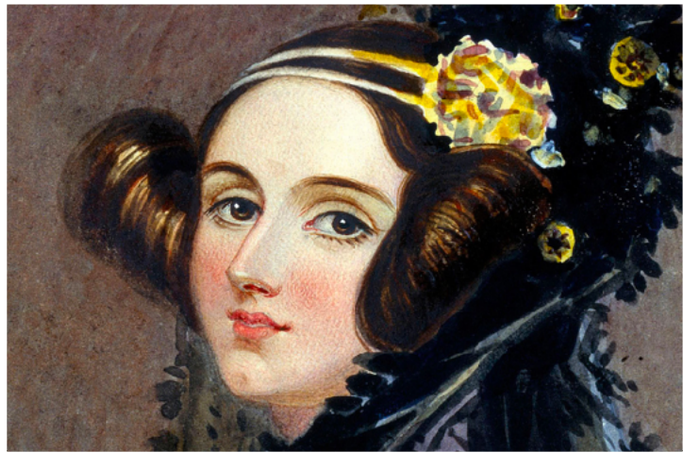


Early Computers: The Difference Engine

- Charles Babbage (1791-1871) and Ada Lovelace (1815-1852)
  - Invented the Difference Engine, a mechanical calculator (~1820s)
  - Watch: Calculating Ada documentary (11:00-13:20 minute mark)



Portion of the Difference Engine
Science Museum in London

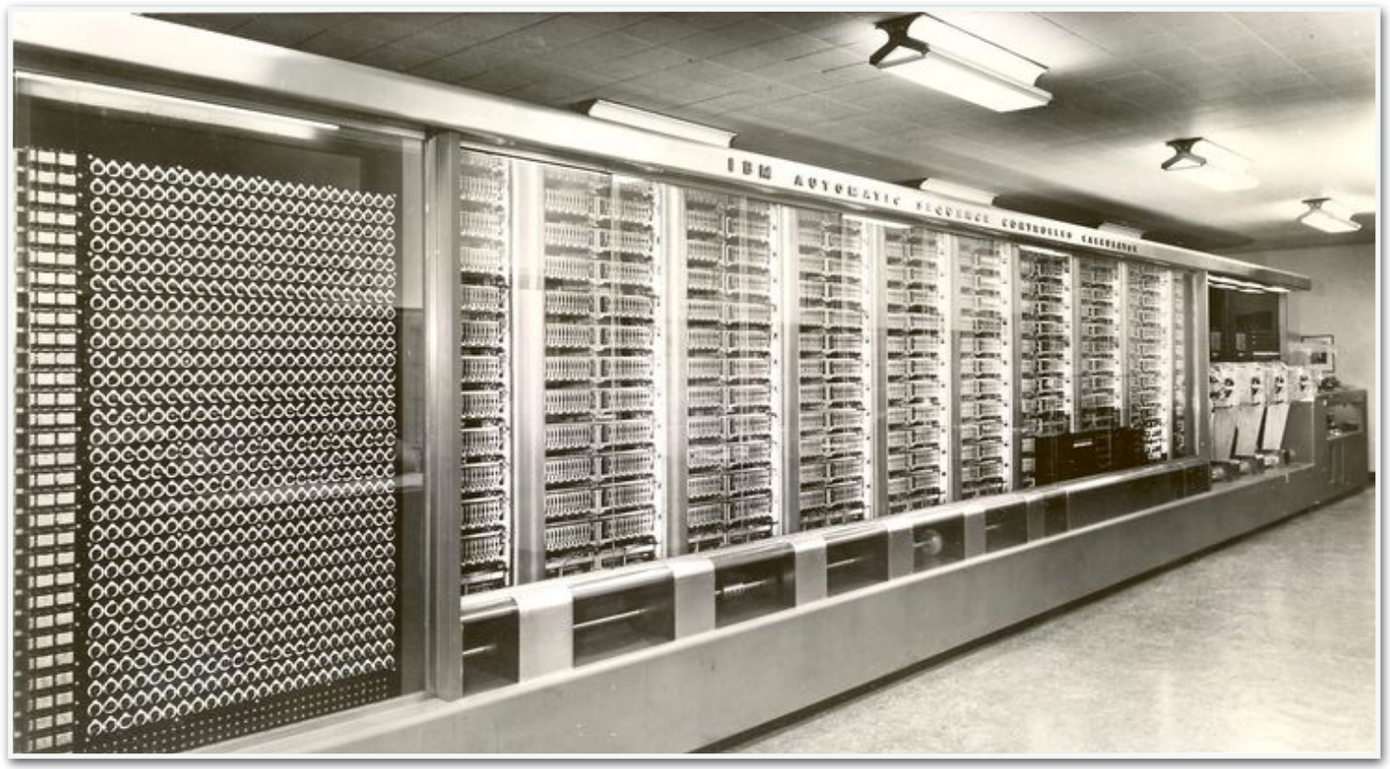


Ada Lovelace:
Published the first ever program!
(For the Difference Engine's successor - Analytical Engine)

Harvard IBM Mark I (1944)

- First fully-automatic, *electro-mechanical* general-purpose computer
  - 16m x 2.5m x 2.5m, weighed 5 tons (5000 kg)
  - 3 adds/subtracts per sec; 1 multiplication in 6 secs
  - Fully programmable!
    - Stopped and waited for "go" signal between operations.
  - No OS, programs on tape fed by humans

- *"Sounded like a room full of people knitting."*



1940s: Harvard IBM Mark I (1944)

- Harvard Mark I programming team was led by **Grace Murray Hopper**
  - Professor of Mathematics @ Vassar College
  - Then, as a US Naval Officer, became the first principle programmer of the Mark I

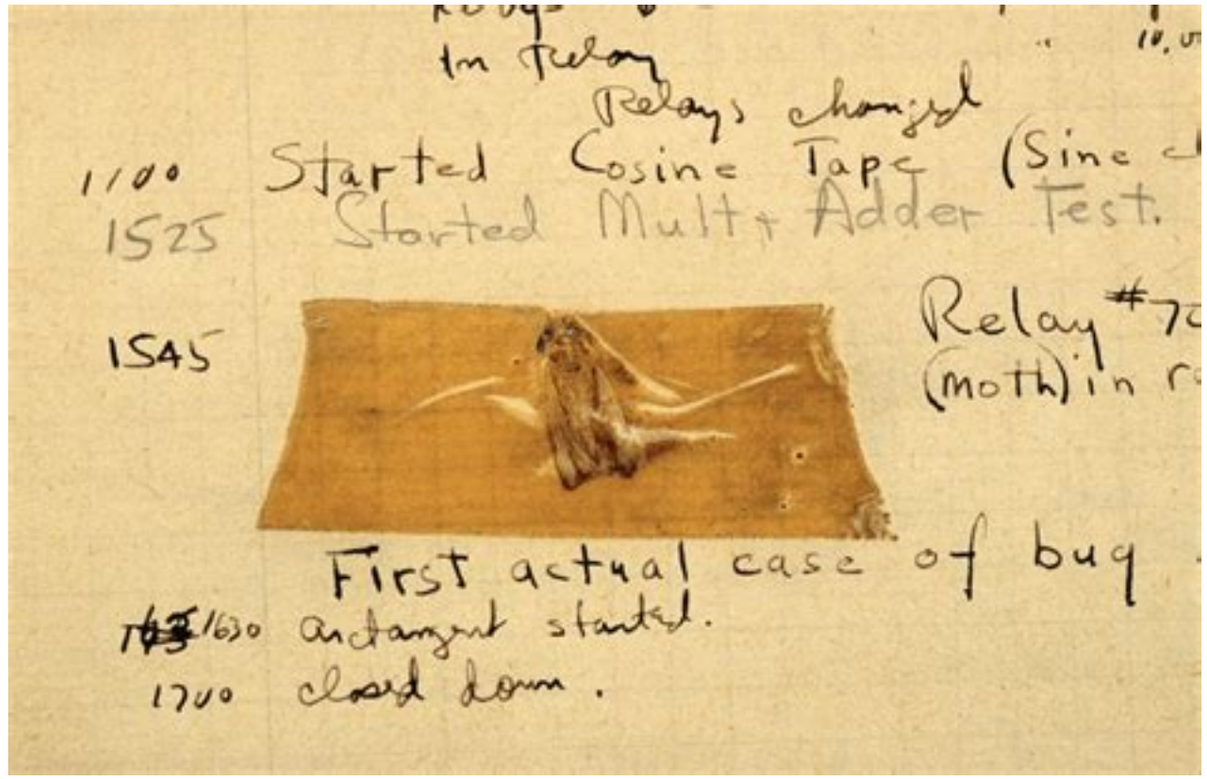
On the Harvard Mark I:

"[it was] the first machine that was built that was supposed to assist the power of man's brain instead of the strength of his arms."

Grace Murray Hopper

Aside: Etymology of Common Terms

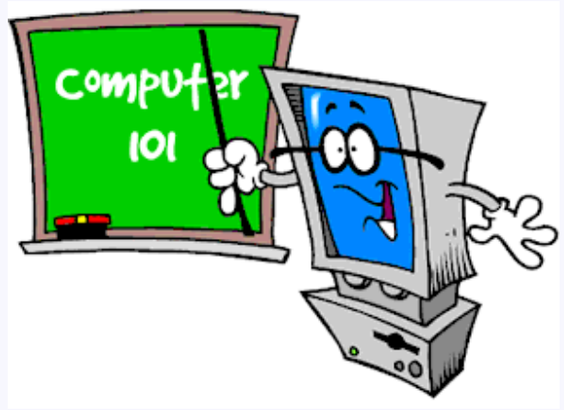
- The term "bug" had been already used in the 1800s to describe flaws in electrical systems.
  - An actual bug (a moth) discovered in Harvard Mark II
  - It did actually cause a mechanical issue in a relay
- "Debugging" was coined under Hopper.



"First actual case of bug being found"

What Is Computer Science? (Cont.)

- Computer science is the *study of algorithms*, including their
  - Formal and mathematical properties,
  - Hardware realizations,
  - **Linguistic realizations**, and
  - Applications



Definition from: Schneider and Gersting. An Invitation to Computer Science.

"The computer is an extremely fast moron. It will, at the speed of light, do exactly what it is told to do--no more, no less."

- Dr. Grace Murray Hopper
  CS Pioneer

Linguistic Realizations of Algorithms

- Let's unpack the quote: "The computer is an extremely fast moron"
- **"extremely fast** moron**"**
  - Today's home computers can do trillions of operations per second!
  - The fastest computer to date (2023) is the *Frontier Supercomputer*
  - At its peak, 1.679 quintillion "primitive operations" per second!
  - https://www.top500.org/

Linguistic Realizations of Algorithms (2)

- Let's unpack the quote: "The computer is an extremely fast moron"

- "extremely fast **moron**"
  - Computers lack *insight*
  - Also, you must tell it what to do *in great detail*, and instructions must be ***very primitive***

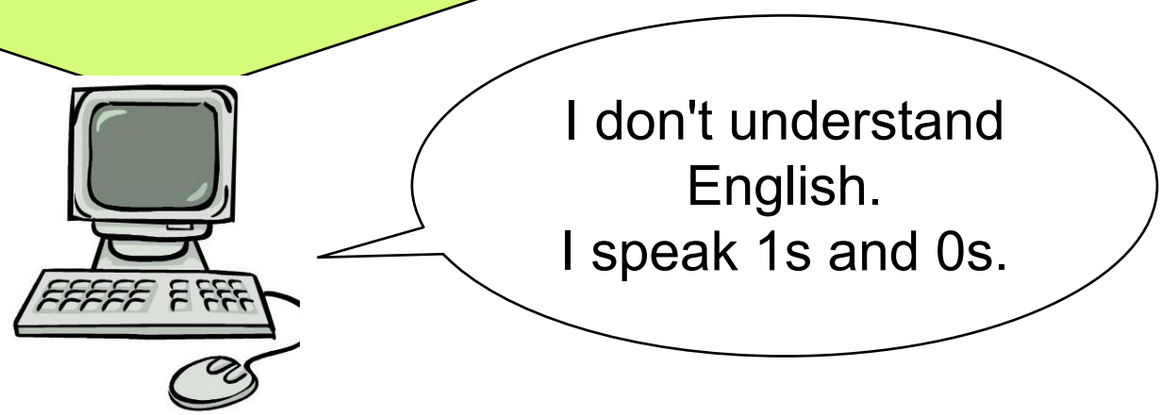
This is what makes programming both frustrating and super rewarding!

Linguistic Realizations of Algorithms (3)

- Just how *primitive* must these instructions be to a computer?

- Consider this: Compute `D = A + (B - 4 * C)`

Instructions to do D = A + (B - 4 * C):
LOAD Contents of A
LOAD Contents of B
LOAD Contents of C
MULT C and 4 and **STORE** in TMP
SUB TMP from B and **STORE** in TMP2
ADD A to TMP2 and **STORE** in D



Even this is not primitive enough!

Linguistic Realizations of Algorithms (4)

- Just how *primitive* must these instructions be to a computer?

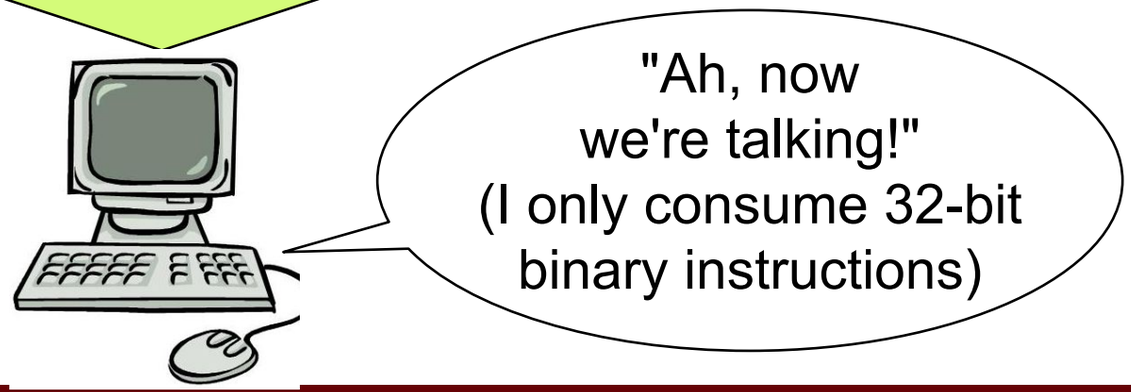
- Consider this: Compute `D = A + (B - 4 * C)`

Instructions to do D = A + (B - 4 * C):

```
01010101110111011001010101000101
01010111011101100101010100010101
01010101110110010101010001010101
01110111011001010101000101010101
11011101100101010100010101010111
01110110010101010001010101011101
```

What each sequence means

⟷	**LOAD**	Contents of A
⟷	**LOAD**	Contents of B
⟷	**LOAD**	Contents of C
⟷	**MULTIPLY**	C and 4 and store in TMP
⟷	**SUB**	TMP from B and store in TMP2
⟷	**ADD**	A to TMP2 and store in D



Linguistic Realizations of Algorithms (6)

Here's an algorithm for how I can solve this problem.

Algorithm Design

Computer Scientist

Express your algorithm using a programming language.

(Known as "coding" or "programming")

Algorithm
(or code, or program)

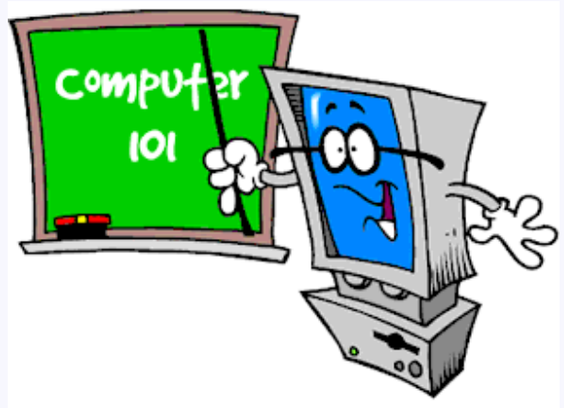
Translate to 0s and 1s

(Known as "compiling")

Computer **executes** your algorithm.
Solves problem, or throws error

What Is Computer Science? (Cont.)

- Computer science is the *study of algorithms*, including their
  - Formal and mathematical properties,
  - Hardware realizations,
  - Linguistic realizations, and
  - **Applications**

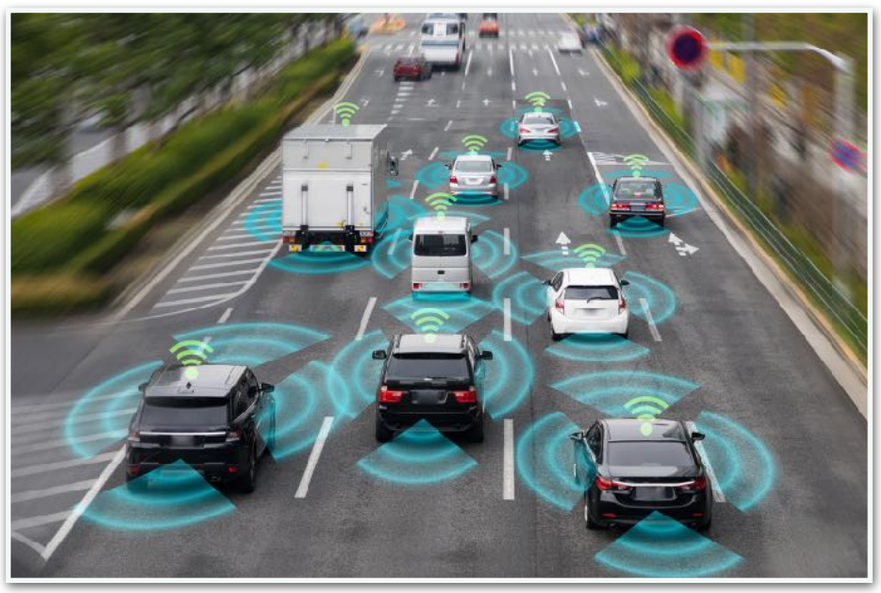


Definition from: Schneider and Gersting. An Invitation to Computer Science.

Applications (Software)

- Next "sea-change" in our lifetime?
  - Generative AI: ChatGPT, Dall-E, Midjourney, etc.
  - Smart "things": homes, AI assistants, etc.
  - Social AI: companions, service-oriented
  - Autonomous cars (death of traffic lights?)

- MIT research: https://www.youtube.com/watch?v=kh7X-UKm9kw



Algs: "... and Applications."

- *Computer applications (programs)* start with an idea...
  - Is the idea solvable with algorithms?
    - If not, stop! No computer-based solution is possible.
  - Can the algorithm be processed quickly?
    - If so, program the algorithm, and improve on it over time!
  - Considerations before deploying an algorithm:
    - Technical: What language should I use? What hardware?
    - Ethical: How could my program be misused?
    - Accessibility: Who can have access to my program?
    - Inclusivity: Is my program inclusive of everyone regardless of ethnic background, economic status?
    - **Read: "Weapons of Math Destruction" book**