What is Computer Science?

How is it different from Computer Programming?

Ultimately, this is a Philosophical Question!

What is a computer?

What does it mean to compute?
“Computer Science is no more about computers than Astronomy is about telescopes!”

-Edsger W. Dijkstra

“I didn’t say that!”
Given a computational problem:

- Is it possible to *compute* the answer?
- How do you *know*?
- Is it possible to know whether or not we *can* know?
- What *type* of machine could (or could not) perform the computation?
Post’s Correspondence Problem - 1946


Game: https://www3.nd.edu/~dchiang/teaching/puzzles/post.html
    Special thanks to Dr. Chiang for writing this implementation!

An “easy” Decision problem.

How can we write a program to solve this problem?
Given a statement $\phi$ of first order logic and some axioms, is there an algorithm that decides whether $\phi$ is provable from the axioms?
Halting Problem - 1936


In computability theory, the halting problem is the problem of determining, from a description of an arbitrary computer program and an input, whether the program will finish running (i.e., halt) or continue to run forever.
Alan Turing (1912-1954)

Influential in development of Theoretical Computer Science

Formalized concepts of Algorithms and Computation

Developed the idea of the Turing Machine

Portrayed by Benedict Cumberbatch.
Turing Machine

**Powerful** abstraction

Can compute **anything** that is computable

Doesn’t need **electricity**

https://www.youtube.com/watch?v=vo8izCKHiF0

https://www.youtube.com/watch?v=E3keLeMwfHY
&index=2&list=PLF87F259C658F6872&t=0s
Problem:

Turing machines are too powerful!!!

(first world problems)
We Must Restrict Our Definitions!

Turing Machines
Deterministic Context-Free Languages
Pushdown Automata
Context-Free Grammars
Regular Languages
Finite Automata

Begin with the simplest.
Build to the most powerful.
Understand the limitations.
But...

Can a Turing machine solve Post’s Correspondence Problem?

How can you know for sure?