Church-Turing Thesis
Definition of Algorithm

- **Algorithm**, aka **procedures, recipes**
  - Collection of simple instructions for carrying out a task
  - Existed for ancient mathematics (finding prime numbers, greatest common divisors)
  - Was not properly defined until the 20th century

- **Hilbert’s Problems**
  - In 1900, David Hilbert gave an address to the International Congress of Mathematicians in Paris
    - Identified 23 problems
  - 10th problem was on algorithms
Polynomials

• Polynomial
  • Sum of terms
    • Each term is a product of variables and constants (coefficients)
  • A root of a polynomial is an assignment of variable values that produces a polynomial value of zero
    • If all values are integers, then it is an integral root

• 10th problem
  • Devise an “algorithm” that tests whether a polynomial has an integral root
  • Hilbert assumed it was possible to find, however this problem is algorithmically unsolvable
    • Proving an algorithm does not exist requires a definition for algorithms
Church-Turing Thesis

- **Church-Turing Thesis** aka **computability thesis**
  - Alonzo Church and Alan Turing, 1936
  - States a function is computable with an algorithm iff it is computable by a Turing machine

- The 10\textsuperscript{th} problem was finally shown to have no algorithm in 1970 by Yuri Matijasevic

- Redefine problem
  - Let \( L = \{ p \mid p \text{ is a polynomial with an integral root} \} \)
  - Problem becomes if \( L \) is decidable
L is not decidable

- TM M recognizes L:
  - M = “On input <p>: where p is a polynomial over the variable x.
    1. Evaluate p with x set successively to 0,1,-1,2,-2,...
    2. Accept if any p(x) = 0
  - If p has a root, M will find it
    - If it does not, M will run forever

- Can convert M to a decider D, if we can find bounds to x.
  - Will always halt if bounds exists.
  - However, these bounds do not exist.
    - M cannot convert to D, L is not decidable, algorithm does not exist by Church-Turing thesis, 10th problem is not solvable.
Descriptions of TM

- Future chapters will not necessarily need detailed descriptions of TM.

- Multiple ways to describe TMs
  - Formal Description
    - All details related to TM and described previously by 7-tuple
    - Lowest-level
  
  - Implementation Description
    - English descriptions of transitions and configurations
    - Mid-level

- High-level Description
  - Just describes algorithm, not implementation details (no details on how TM functions)
Every TM receives a string as an input.
- If we want to input some other kind of object, it MUST be represented by a string.

If we encode objects into a string, we use the following notation
- $O$ is encode the string $<O>$
- $O_1, O_2, \ldots, O_n$ is encoded as $< O_1, O_2, \ldots, O_n >$

We assume that the TM is able to properly encode the objects as strings