1 Quiz

Give an implementation level algorithm that decides the language

\[ L = \{ w : w \text{ has only two zeros and } |w| \text{ is odd} \} \quad (1) \]

M = "On input w:
1. Scan tape and mark any zeros. If more than two zeros, reject.
2. Go back to the beginning of the tape and if symbol is not marked zero, cross every other character. If one is marked zero, skip.
3. If tape contains odd number of crossed off symbols go to step 2, else reject.
4. If at the end of the tape everything is crossed off or marked, accept."

2 Equivalent Computation Models

- Turing Machine, k-Tape Turing Machine, Nondeterministic Turing Machine, C++ Programs, etc.
- If we define the notion of an algorithm with any of these, we can use any of them to give an algorithm.
- An Algorithm is a sequence of simple instructions for carrying out some task. (Note in class: finite sequence, but does not necessarily have to be)
  - Precisely defined in 1900’s
- On August 8, 1900 at the International Congress of Mathematics in Paris, David Hilbert gave a speech outlining 23 mathematical problems that would shape the 20th Century
  - Problem 10: Devise an algorithm to test whether a polynomial has an integral root
- A Polynomial is a sum of terms where each term has a coefficient and variables
Example:
\[ f(x, y) = 3x^2 + 2y^2 + x + 3 \]  \hspace{1cm} (2)

Root: The roots of a polynomial are the values of the variable that cause the polynomial to evaluate to zero.

If all variable settings are integers, then it is an integral root.

Example: Pythagorean Theorem
\[ a^2 + b^2 = c^2 \]  \hspace{1cm} (3)

Some Polynomials have them some don’t

- Intuitive notion of algorithm is good for finding solutions, but not for proving no algorithm exists to do a task.
- Definition in 1936 by Alan Turing and Alonzo Church
  - Turing Machines and λ-Calculus are equivalent (proven by Alan Turing)
  - Turing-Complete

Church-Turing Thesis (M152) - Every computational process that is intuitively considered to be an algorithm can be converted to a Turing Machine.

- We define an algorithm to be a Turing Machine

\[
D = \{ p : p \text{ is a polynomial with integral roots} \} \]  \hspace{1cm} (4)

- Hilbert’s 10th Problem: Is D decidable?

Simpler \( D_1 = \{ p : p \text{ is a polynomial over } x \text{ with integral roots} \} \)  \hspace{1cm} (5)

- \( D_1 \) is recognizable

\[ M_1 = " \text{On input } < p > \text{ where } p \text{ is a polynomial over } x:\]

1. Evaluate \( p \) with \( x \) set successively to values 0, 1, -1, 2, -2, 3, -3 . . . N. If at any point the polynomial evaluates to zero, accept.”

- \( D \) is also recognizable with a similar machine.

- Both of these machines are recognizers, but not deciders.

  - For \( D_1 \), we can convert \( M_1 \) to a decider.
    * If no root exists, reject.
\[ f(x) = c_1x^n + c_2x^{n-1} + \ldots + c_nx + c_{n+1} \quad (6) \]

Let \( C_{\text{max}} \) be the largest absolute value of \( c_i \)

\[ |X_i| < (n + 1) \frac{C_{\text{max}}}{|c_1|} \quad (7) \]

\( M_1 = "\text{On input } < p > \text{ where } p \text{ is a polynomial over } x: \)

1. Evaluate \( p \) with \( x \) set successively to values 0, 1, -1, 2, -2, 3, -3 . . . \( N \). If at any point the polynomial evaluates to zero, accept.

\[ \text{where } |N| = \left\lceil (n + 1) \frac{C_{\text{max}}}{|c_1|} \right\rceil \quad (8) \]

If after \( N, -N \), no root found, reject.”

**Example:**

\[ A = \{ < G > : G \text{ is a connected undirected graph} \} \quad (9) \]

TM that decides \( A \):

\( M = "\text{On input } < G >, \text{ the encoding of the graph: } \)

1. Select the first vertex of \( G \) and mark it.
2. Repeat until no new vertices are marked.
3. For each vertex in \( G \), mark it if it is connected to a marked node.
4. Scan all vertices to check if they’re all marked. If yes, accept; otherwise, reject.”