P versus NP

Non-deterministic Turing Machines.
Polynomial Time (P) and Non-deterministic Polynomial Time (NP).
Reducibility and NP-completeness: SAT.
Is NP ⊆ P?

“I can't find an efficient algorithm, but neither can all these famous people.”
Last Time

1. The infinite loop.

2. Recognizable languages (allowing the infinite loop for “reject”).

3. Decidable languages (no-infinite loops allowed).

4. Programs versus computing machines.
   The string (text/mathematical) description of a TM.
   The Universal Turing Machine (UTM).

5. Feeding the string $<TM>$ as input to TM (TM is a Turing machine).

6. The language corresponding to “program verification”
   A simpler language: TMs that reject “themselves”.

7. The simpler language is undecidable.

8. We cannot automate program verification!

9. Challenge problem #2, though it looks harmless, is UNSOLVABLE!
Today: Decidable and Undecidable Problems

1. Non-deterministic Turing Machines.

2. Polynomial: a definition of efficiency that stood the test of time.

3. Non-deterministic Polynomial ($\text{NP}$) versus Deterministic Polynomial ($\text{P}$).
   
   $\text{NP}$ does not mean Non-Polynomial. 😊

4. The $\text{NP}$-complete problem $\text{SAT}$ and the complexity hierarchy.

5. Does $\text{P} = \text{NP}$? That is THE QUESTION.