CS301: Theory of Computation
Sections 7.2 & 7.3 - P versus NP

Daniel White

Gettysburg College
Computer Science

Spring 2024

We have established the differences between the time complexities of problems solved by single/multitape and non/deterministic TMs.

$$\text{single-tape TM} \xleftrightarrow{\text{polynomial}} \text{multitape TM}$$

$$\text{deterministic TM} \xleftrightarrow{\text{exponential}} \text{nondeterministic TM}$$

We will consider polynomial differences in running time to be "small", whereas exponential differences are considered "large".

exercise: Give n so that 10^n seconds exceeds the age of the universe.
exercise: Give n so that n^2 seconds exceeds the age of the universe.

Definition 7.12 (pg. 286)

P is the class of languages that are decidable in polynomial time on a deterministic single-tape Turing machine. In other words,

$$P = \bigcup_{k \geq 1} \text{TIME}(n^k).$$

- P is invariant for models of computation that are polynomially equivalent to the deterministic single-tape TM (*i.e.* can simulate one another with only polynomial time increase), and
- P (roughly) corresponds to the class of problems that are realistically solvable on computers.

Theorem 7.14 (pg. 288)

PATH $\in$ P, where
PATH $= \{\langle G, s, t\rangle \mid G$ is directed graph with path from s to $t\}$.

Proof. On input $\langle G, s, t\rangle$,

1. Place a mark on node s.
2. Repeat the following until no additional nodes are marked:
3. Scan all edges of G. If (a, b) is found going from marked a to unmarked b, mark b.
4. If t is marked, *accept*. Otherwise, *reject*.

exercise: Finish this proof by establishing time complexity.
exercise: Bad approach that yields exponential complexity?

homework: 7.8

Theorem 7.15 (pg. 289)

RELPRIME $\in$ P, where
RELPRIME $= \{\langle x, y\rangle \mid x$ and y are relatively prime$\}$.

Proof. We build R which uses the following subroutine. Define the Euclidean algorithm E, where on input $\langle x, y\rangle$,

1. Repeat until $y = 0$:
2. Assign $x \leftarrow x \pmod{y}$.
3. Exchange x and y.
4. Halt with x on tape.

exercise: Finish this proof.
exercise: What would it mean to instead "brute force" RELPRIME? What is the time complexity in that situation?

Definition 7.18 (pg. 293)

A **verifier** for a language A is an algorithm V, where

$$A = \{w \mid V \text{ accepts } \langle w, c\rangle \text{ for some string } c\}.$$

We measure the time of a verifier only in terms of the length of w, so a **polynomial time verifier** runs in polynomial time in the length of w. Language A is **polynomially verifiable** if it has a polynomial time verifier.

e.g. Consider $A = \{\langle n\rangle \mid n \text{ is composite}\}$. A verifier for A would be V that checks if some $1 < c < n$ is a divisor of n.

A verifier uses additional information, represented by c, to verify $w \in A$. This information is called a **certificate** or **proof**.

Definition 7.19 (pg. 294)

NP is the class of languages that have polynomial time verifiers.

The class NP contains many important problems of practical interest. The term NP comes from **nondeterministic polynomial time**; see below.

Theorem 7.20 (pg. 294)

A language is in NP if and only if it is decided by some nondeterministic polynomial time Turing machine.

exercise: Prove Thm 7.20. *(pgs. 294 - 295)*
exercise: Do you think that $\mathrm{P} \subseteq \mathrm{NP}$? What about $\mathrm{P} = \mathrm{NP}$?

homework: 7.6, 7.7, & 7.16

Theorem 7.24 (pg. 296)

CLIQUE $= \{\langle G, k\rangle \mid G$ undirected graph with k-clique$\} \in$ NP

Proof. We present a verifier V for CLIQUE. On input $\langle\langle G, k\rangle, c\rangle$

1. Test whether c is subgraph of G.
2. Test if c has k nodes and is complete.
3. If both pass, *accept*; otherwise, *reject*.

exercise: Finish proof; establish that $V \in$ P.
exercise: Do you think CLIQUE $\in$ P?
exercise: Prove instead using a nondeterministic machine.

Theorem 7.25 (pg. 297)

$\text{SUBSUM} = \{\langle S, t\rangle \mid \exists R \subseteq S \text{ where } \sum_{r\in R} r = t\} \in \text{NP}$

Proof. We present a verifier V for SUBSUM. On input $\langle\langle S, t\rangle, c\rangle$

1. Test whether c is subset of S.
2. Test if elements of c sum to t.
3. If both pass, *accept*; otherwise, *reject*.

exercise: Finish proof; establish that $V \in$ P.
exercise: Do you think SUBSUM $\in$ P?
exercise: Prove instead using a nondeterministic machine.

homework: Know CLIQUE and SUBSUM proofs

P = languages where membership can be decided quickly

NP = languages where membership can be verified quickly

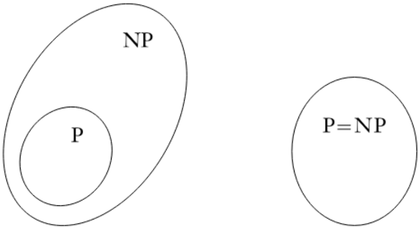

FIGURE 7.26
One of these two possibilities is correct

The question of P $=$ NP is one of the greatest unsolved problems in computer science and mathematics. If these classes are equal, then any quickly verifiable problem also has a quick solution.

exercise: Loosely speaking, what are the implications of P $=$ NP in, say, cryptography and artificial intelligence?