Theory of Formal Languages and Automata

Lecture 23

Mahdi Dolati

Sharif University of Technology

Fall 2023

December 25, 2023

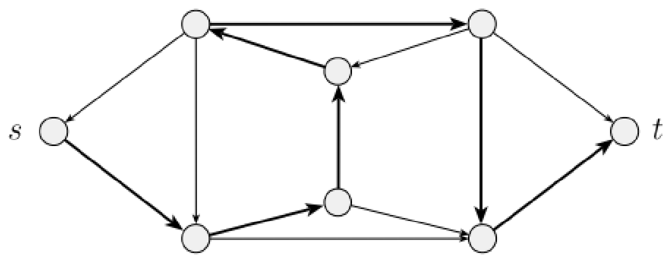
Time Complexity

The Class NP

- We were able to find a polynomial time algorithm for **PATH** problem.
  - Avoided the brute-force search.
- We have not been able to find a polynomial time algorithm for certain other problems,
  - Some of these problems are interesting and useful.

Example: HAMPATH
A Hamiltonian path goes through every node exactly once.

$$HAMPATH = \{\langle G, s, t\rangle \mid G \text{ is a directed graph with a Hamiltonian path from } s \text{ to } t\}.$$



We don’t know why we were unsuccessful in finding polynomial time algorithms for certain problems.

- Maybe these problems have polynomial time algorithms. Or, they are intrinsically difficult.
- We know the complexity of many of such problems are **linked**. A polynomial time algorithm for one such problem can be used to solve an entire class of problems.
- They have a feature called polynomial verifiability.
  - One can verify a membership in polynomial time:
    - Provide the solution to them.

- Definition:

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. A language A is ***polynomially verifiable*** if it has a polynomial time verifier.

- We call c a certificate or proof of membership in A.
- Since the verifier runs in polynomial time, the certificate has polynomial length (in the length of w).
- Observe that a Hamiltonian path from s to t is a certificate for the HAMPATH problem.

- Definition:

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. A language A is ***polynomially verifiable*** if it has a polynomial time verifier.

- Some problems may not be **polynomially** verifiable:
- Example:
  - $\overline{HAMPATH}$
  - Need to verify a nonexistence.

- Definition:

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

- NP: Nondeterministic polynomial time.
- Alternative way to characterize problems in NP (NP-problems): Solvable in polynomial time with a nondeterministic Turing machine.
- **Example**: HAMPATH: All stages run in poly. time.

N_1 = "On input $\langle G, s, t\rangle$, where G is a directed graph with nodes s and t:

1. Write a list of m numbers, $p_1, \ldots, p_m$, where m is the number of nodes in G. Each number in the list is nondeterministically selected to be between 1 and m.
2. Check for repetitions in the list. If any are found, *reject*.
3. Check whether $s = p_1$ and $t = p_m$. If either fail, *reject*.
4. For each i between 1 and $m - 1$, check whether (p_i, p_{i+1}) is an edge of G. If any are not, *reject*. Otherwise, all tests have been passed, so *accept*."

Theorem

A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine.

- Proof Idea: Convert a polynomial time verifier to an equivalent polynomial time nondeterministic Turing machine and vice versa:
  - TM simulates the verifier by guessing the certificate.
  - The verifier simulates the TM by using the accepting branch as the certificate.

Theorem
A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine.

- Proof: Let A be an NP-problem.
  - N: A nondeterministic Turing machine (NTM),
  - V: A polynomial time verifier. V is a TM that runs in time n^k.
  - Given V, construct N:

N = "On input w of length n:
1. Nondeterministically select string c of length at most n^k.
2. Run V on input $\langle w, c \rangle$.
3. If V accepts, *accept*; otherwise, *reject*."

Theorem

A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine.

- Proof Cont.: Let A be an NP-problem.
  - N: A nondeterministic Turing machine (NTM),
  - V: A polynomial time verifier. V is a TM that runs in time n^k.
  - Given N, construct V:

V = "On input $\langle w, c\rangle$, where w and c are strings:

1. Simulate N on input w, treating each symbol of c as a description of the nondeterministic choice to make at each step (as in the proof of Theorem 3.16).
2. If this branch of N's computation accepts, *accept*; otherwise, *reject*."

Theorem

A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine.

- Proof Cont.: Let A be an NP-problem.
  - N: A nondeterministic Turing machine (NTM),
  - V: A polynomial time verifier. V is a TM that runs in time n^k.
  - Given N, construct V:

V = "On input $\langle w, c\rangle$, where w and c are strings:

1. Simulate N on input w, treating each symbol of c as a de tion of the nondeterministic choice to make at each step (the proof of Theorem 3.16).
2. If this branch of N's computation accepts, *accept*; otherwi e, *reject*."

Given the correct certificate, V runs the accepting branch of N and accepts.

Time Complexity
The Class NP

- Definition:

NTIME$(t(n)) = \{L \mid L$ is a language decided by an $O(t(n))$ time nondeterministic Turing machine$\}$.

- Definition:

$$\text{NP} = \bigcup_k \text{NTIME}(n^k).$$

Time Complexity

The P versus NP Question

- We learned that:

  P = the class of languages for which membership can be *decided* quickly.

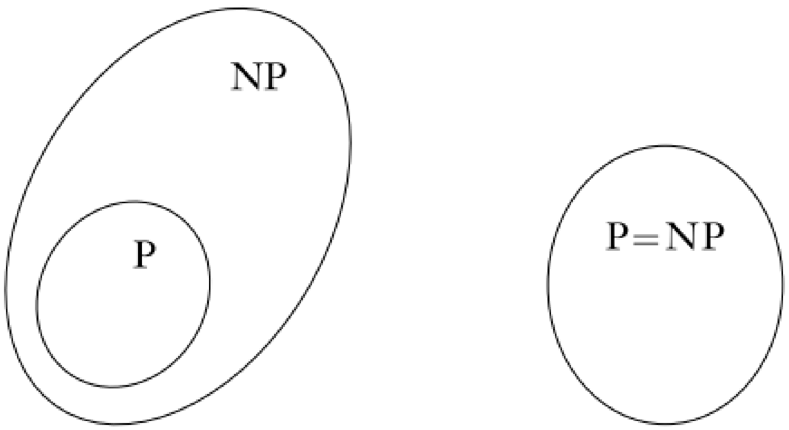
  NP = the class of languages for which membership can be *verified* quickly.

- Polynomial verifiability seems more powerful than polynomial decidability.
- However, we have been unable to **prove** the there is a problem that has a polynomial verifier but it has not a polynomial decider.
- P=NP? is one of the greatest unsolved problems in theoretical computer science.
- Most researchers believe P≠NP.

Time Complexity

The P versus NP Question

- P=NP?



- The Clay Mathematics Institute (a US$1 million prize):
  - Birch and Swinnerton-Dyer conjecture,
  - Hodge conjecture,
  - Navier–Stokes existence and smoothness,
  - **P versus NP problem**,
  - Riemann hypothesis,
  - Yang–Mills existence and mass gap, and
  - Poincaré conjecture