P vs NP

Lecture 16

Binghui Peng

The Class P

Definition:

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

In words, P is the class of languages decidable in polynomial time.

Why do we care about polynomial time?

1. Polynomial algorithms usually scale reasonably,
2. They are closed under composition,
3. They match the informal notion of efficient computation.

Decision vs. Optimization

Decision: answer **yes/no**.

Optimization: find the answer.

Example: Graph distance

1. Decision: is there an s-t path of length at most B?
2. Optimization: compute the shortest s-t path length.

These are essentially the same in P (up to polylogarithmic factor):

Decision vs. Optimization

Decision: answer **yes/no**.

Optimization: find the answer.

Example: Graph distance

1. Decision: is there an s-t path of length at most B?
2. Optimization: compute the shortest s-t path length.

These are essentially the same in P (up to polylogarithmic factor):

use the decision algorithm + binary search on B.

Examples in P

Question: Which problems are in P?

Examples in P

Question: Which problems are in P?

The following problems are in P.

1. Graph reachability.
2. Shortest path.
3. Sorting.
4. DFA acceptance.
5. Primality testing.

Generation vs. Verification

We have talked this question at the beginning of this semester: Sometimes generating a solution seems hard, while verifying one is easy.

Generation vs. Verification

We have talked this question at the beginning of this semester: Sometimes generating a solution seems hard, while verifying one is easy.

- Count 24
  - Generate an expression that makes 24.
  - Verify that a proposed expression really equals 24.

Generation vs. Verification

We have talked this question at the beginning of this semester: Sometimes generating a solution seems hard, while verifying one is easy.

- Count 24
  - Generate an expression that makes 24.
  - Verify that a proposed expression really equals 24.
- Solve equations (e.g. $x^3y + y^3x + x + y + xy + 1 = 0$)
  - Generate a solution to some equations.
  - Verify that a proposed solution satisfies the equations.

Polynomial-Time Verifiers

A **verifier** for a language L is a polynomial-time algorithm $V(x, c)$ such that:

1. if $x \in L$, then there exists a certificate c with $V(x, c) = 1$;
2. if $x \notin L$, then for every certificate c, we have $V(x, c) = 0$.

The certificate should have polynomial length in $|x|$.

The Class NP

Definition: A language L is in NP if it has a polynomial-time verifier.

Equivalent viewpoint: $L \in \mathrm{NP}$ if some nondeterministic Turing machine accepts L in polynomial time.

$$P \subseteq NP$$

Proof Idea: $\mathrm{P} \subseteq \mathrm{NP}$

Suppose $L \in \mathrm{P}$.

Then there is a polynomial-time decider M for L.

We build a verifier $V(x, c)$ as follows:

1. ignore the certificate c,
2. run M on input x,
3. accept iff M accepts.

So the verifier just uses the decider and does not need any certificate information.

Proof Idea $\mathrm{P} \subseteq \mathrm{NP}$

For the verifier V from the previous slide:

- if $x \in L$, then M accepts x, so $V(x, c) = 1$ for every c;
- if $x \notin L$, then M rejects x, so $V(x, c) = 0$ for every c.

Also, V runs in polynomial time because it just runs M.

Therefore V is a polynomial-time verifier for L, so $L \in \mathrm{NP}$.

SAT

Boolean satisfiability: Given a Boolean formula, decide whether it has a satisfying assignment.

$$\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee x_3)$$

Example satisfying assignment: $x_1 = \mathrm{T},\ x_2 = \mathrm{F},\ x_3 = \mathrm{T}$.

SAT: Boolean formula with at least one satisfying assignment.

$\mathrm{SAT} \in \mathrm{NP}$

Input: a Boolean formula φ.

Certificate: a truth assignment to the variables of φ.

Verifier:

1. evaluate φ under that assignment,
2. accept iff the formula evaluates to true.

This takes polynomial time, so $\mathrm{SAT} \in \mathrm{NP}$.