Computational Complexity

P, NP, NP-Hard and NP-Complete

Efficiency • Verification • Polynomial-time reductions

Learning outcomes

Measure efficiency

Distinguish decidability from tractability and classify running times.

Understand complexity classes

Define P, NP, NP-hard and NP-complete using decision problems.

Apply reductions

Prove NP-completeness using a worked reduction from 3-SAT to CLIQUE.

From computability to complexity

Computability

Can an algorithm halt with the correct answer on every input?

Complexity

How much time or space does an algorithm need as input length grows?

Worst-case time

T(n) = maximum number of steps over inputs of length n. Measure n in encoding symbols, not numerical value.

Polynomial vs. exponential growth

Input length n	n^2 steps	2^n steps
5	25	32
10	100	1,024
20	400	1,048,576
30	900	1,073,741,824

Polynomial time: $O(n^k)$ for some fixed k. Exponential growth quickly becomes impractical.

Class P: polynomial-time decisions

Formal definition

P = {L : a deterministic Turing machine decides L in polynomial time}.

Examples

DFA membership, graph reachability, and shortest-path decision problems with nonnegative edge weights.

Interpretation

P is a mathematical model of efficient computation; a high-degree polynomial may still be impractical.

Class NP: polynomial-time verification

Formal definition

$L \in NP$ if $x \in L$ exactly when some certificate c of polynomial length makes a polynomial-time verifier accept (x,c).

Equivalent characterization

A nondeterministic Turing machine decides L in polynomial time on every branch.

Important

NP means nondeterministic polynomial time, not non-polynomial time.

Worked example: CLIQUE verification

Decision problem

Given an undirected graph G and integer k, is there a set of k pairwise adjacent vertices?

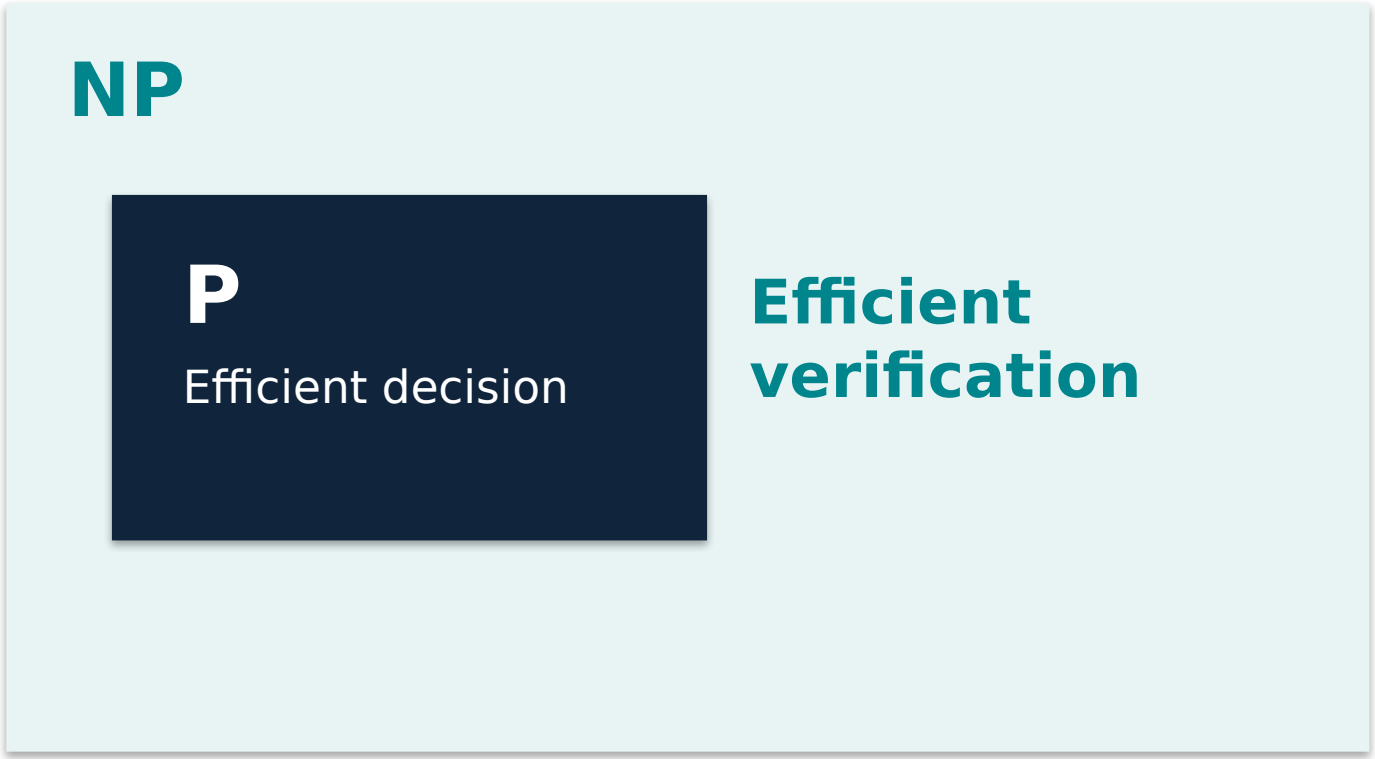
Certificate

List k distinct vertices. For edges AB, AC, BC, CD and k = 3, the certificate {A,B,C} is valid.

Verification

Check distinctness and all $k(k-1)/2$ pairs using an adjacency matrix. This is polynomial in the graph encoding.

P versus NP: the open question



$P \subseteq NP$

Is $P = NP$?

Unknown. The drawing illustrates the possible strict-containment case.

Polynomial-time mapping reductions

Definition

$A \leq_p B$ if a polynomial-time computable function f satisfies $x \in A$ iff $f(x) \in B$.

Consequence

If $A \leq_p B$ and $B \in P$, then $A \in P$: compute f(x), then decide B.

Direction matters

To show B is hard, reduce a known hard problem A TO B, not the reverse.

NP-hard and NP-complete

NP-hard

B is NP-hard if every $A \in NP$ reduces to B in polynomial time. B need not belong to NP.

NP-complete

B is NP-complete if $B \in NP$ and B is NP-hard.

Consequence

If any NP-complete problem is in P, then $P = NP$. Otherwise no NP-complete problem is in P.

Relationship among complexity classes

Membership in NP

Polynomial-size certificates
Polynomial-time verification

NP-hardness

All NP problems reduce
to this problem in polytime

Both conditions ⇒ NP-complete

SAT, 3-SAT and Cook–Levin

SAT

Given a Boolean formula, is there a truth assignment that makes it true?

3-SAT

SAT restricted to conjunctions of clauses with three literals each. A truth assignment is a polynomial-size certificate.

Foundational results

Cook–Levin establishes SAT as NP-complete; 3-SAT is also NP-complete by polynomial reduction. Full proofs omitted.

Reduction: 3-SAT to CLIQUE

Input

A 3-CNF formula with m clauses. Construct a graph with one vertex for each literal occurrence.

Edges

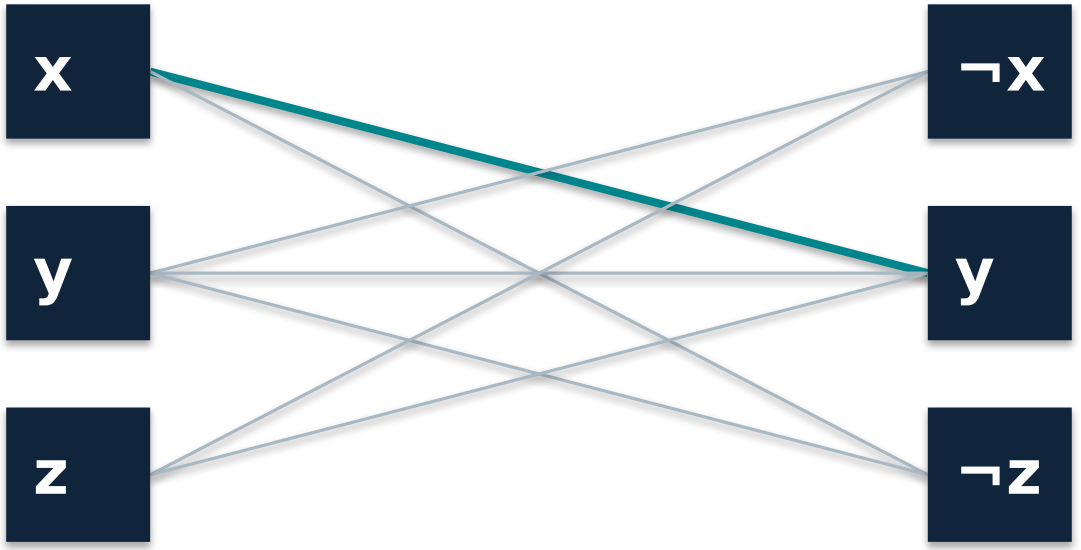
Connect vertices from different clauses unless their literals are complementary (x and ¬x). No edges within a clause.

Output

Ask whether the graph contains a clique of size m. At most 3m vertices and $O(m^2)$ edges: polynomial construction.

Worked reduction: a two-clause formula

$$(x \vee y \vee z) \wedge (\neg x \vee y \vee \neg z)$$



$m = 2$

Clique: x from C_1, y from C_2.

Set x = y = true.

Highlighted edge selects compatible literals from distinct clauses.

Reduction correctness and conclusion

If formula is satisfiable

Select one true literal from each clause. No selected pair is complementary; all selected vertices form an m-clique.

If an m-clique exists

It selects exactly one literal per clause. No selected literals conflict; assign them true and extend arbitrarily.

Conclusion

3-SAT $\leq_p$ CLIQUE, so CLIQUE is NP-hard. Its polynomial verifier proves CLIQUE $\in$ NP; hence CLIQUE is NP-complete.

Decision vs. optimization problems

Traveling Salesperson (decision)

Given weighted graph and bound B, is there a tour of total weight at most B? Standard decision TSP is NP-complete.

Traveling Salesperson (optimization)

Find a minimum-cost tour. This optimization problem is NP-hard; NP itself is defined for decision languages.

Practical response

Exact exponential algorithms, approximation, heuristics or special-case algorithms may still be useful.

Classroom exercises

Exercise 1

If $A \leq_p B$ and A is NP-complete, what can be concluded about B? What additional fact proves B is NP-complete?

Exercise 2

Explain why a proposed Hamiltonian cycle can be verified in polynomial time.

Exercise 3

If an NP-complete problem has a polynomial-time algorithm, what follows about P and NP?

Solutions and recap

Exercise 1

B is NP-hard by transitivity. Prove $B \in NP$ to establish NP-completeness.

Exercise 2

Check that every vertex occurs once and every consecutive pair, including last to first, is connected.

Exercise 3

$P = NP$. Remember: $P \subseteq NP$; hardness comes from reductions; completeness additionally requires membership in NP.