P and NP

Last modified: 2026-10-30 13:54:17 (PDT)

Some problems have fast algorithms; for others, the best known algorithms take time that grows exponentially with the size of the input. This page defines the class P of problems with fast algorithms, and the class NP of problems whose yes answers can be checked fast, measuring speed with asymptotic notation.

Decision problems and running time

Definition 0.1 (Decision problem)**.** A **decision problem** asks a yes-or-no question about an input. Write each input as a binary string $x \in \{0,1\}^*$, the set of finite strings of binary digits, and write $|x|$ for its length. The decision problem is then the set $L \subseteq \{0,1\}^*$ of inputs whose answer is yes, and an algorithm **decides** L if, on every input x, it halts and outputs 1 if $x \in L$ and 0 if $x \notin L$. Arora and Barak (2009, chap. 1) call such a set L a *language*.

Definition 0.2 (Running time)**.** Let T be a function from the nonnegative integers to the nonnegative integers. An algorithm **runs in time** $T(n)$ if on every input x it halts within $T(|x|)$ steps. It runs in **polynomial time** if it runs in time $T(n)$ with $T(n) = O(n^c)$ for some constant c, where O is big-O notation.

The class P

Definition 0.3 (The class P)**. P** is the set of decision problems that some algorithm decides in polynomial time (Arora and Barak 2009, sec. 1.6).

Remark 0.1 (What counts as an algorithm and a step). Arora and Barak (2009, chap. 1) make "algorithm" precise as a Turing machine, and count one step as one move of the

machine. These notes have not yet defined Turing machines, so read "algorithm" as any program for an ordinary computer, with one step per basic operation on a bounded number of bits. The two models can simulate each other with at most a polynomial increase in running time (Arora and Barak 2009, chap. 1), so they agree on which problems are in P.

Definition 0.4 (The reachability problem PATH)**.** The decision problem **PATH** asks: given a graph G with m vertices, stored as its $m \times m$ adjacency matrix, and two vertices s and t, is there a path from s to t?

Example 0.1 (Reachability is in P)**.** Breadth-first search decides PATH. It starts at s, and, for each vertex it reaches, scans that vertex's row of the matrix for neighbors not yet reached. It handles each vertex at most once and scans each row in $O(m)$ steps, so it takes $O(m^2)$ steps in all, and it answers yes exactly when it reaches t. The input has length at least m^2, so this running time is $O(n)$ in the input length n, and PATH is in P (Definition 0.3).

The class NP

Definition 0.5 (The class NP)**.** A decision problem L is in **NP** if there are a polynomial q and an algorithm V, the *verifier*, which takes two binary strings x and u as input and runs in time polynomial in $|x|$, such that for every $x \in \{0,1\}^*$,

$$x \in L \quad \text{if and only if} \quad V(x,u) = 1 \text{ for some } u \in \{0,1\}^{q(|x|)}.$$

A string u with $V(x,u) = 1$ is a *certificate* for x (Arora and Barak 2009, sec. 2.1).

Remark 0.2 (NP means checkable, not solvable). For a problem in NP (Definition 0.5), a yes answer always has a short proof that can be checked quickly, though nothing says the proof is easy to find.

Definition 0.6 (The independent set problem INDSET)**.** An *independent set* in a graph is a set of vertices no two of which are joined by an edge. The decision problem **INDSET** asks: given a graph G with m vertices, stored as its adjacency matrix, and an integer $k \geq 0$, does G have an independent set of k vertices?

Example 0.2 (Independent set is in NP)**.** Let x be an input to INDSET, with a graph of m vertices. Its adjacency matrix has $m^2 \geq m$ entries, so $m \leq |x|$. Take $q(n) = n$ in

Definition 0.5, so a certificate u has $|x|$ binary digits. Its first m digits mark the vertices of a proposed independent set, and the verifier ignores the rest. The verifier checks that k of those m digits are 1 and that no two marked vertices are adjacent. That takes $O(m^2)$ lookups in the adjacency matrix, which is polynomial in $|x|$. G has an independent set of k vertices exactly when some such u passes, so INDSET is in NP (Definition 0.5). No polynomial-time algorithm that decides INDSET is known. Arora and Barak (2009, sec. 2.1) use the same example.

Theorem 0.1 (P is contained in NP)**.** *Every decision problem in P (Definition 0.3) is in NP (Definition 0.5).*

Proof

Proof. Let an algorithm A decide L in polynomial time. Take $q(n) = 0$, so the only certificate is the empty string, and let the verifier $V(x, u)$ ignore u and return $A(x)$. V runs in polynomial time, and $V(x, u) = 1$ exactly when $x \in L$. □

Remark 0.3 (The P versus NP question). Whether P = NP is open: no one has shown that some problem in NP lies outside P, or found polynomial-time algorithms for every problem in NP (Arora and Barak 2009, sec. 2.7). Most researchers expect the classes to differ, which would mean that, for some problems, checking a proposed answer is much easier than finding one.

References

Arora, Sanjeev, and Boaz Barak. 2009. *Computational Complexity: A Modern Approach*. Cambridge University Press. https://doi.org/10.1017/CBO9780511804090.