

# P vs. NP Classes

Prof. (Dr.) K.R. Chowdhary  
*Email: kr.chowdhary@iitj.ac.in*

Formerly at department of Computer Science and Engineering  
MBM Engineering College, Jodhpur

Thursday 13<sup>th</sup> November, 2025

K.R. Chowdhary

# Theory of Computation

Automata, Formal Languages, Computation and Complexity

Focuses on pedagogy in its writing, that represents a refreshing approach

Ensures comprehensive and enjoyable learning

Undergone a rigorous classroom testing



© 2025

Get 20% off with this code: **SPRAUT**

Available on Springer Nature Link

[link.springer.com/book/  
9789819762347](https://link.springer.com/book/9789819762347)

Please note that promotional coupons are only valid for English-language Springer, Apress, and Palgrave Macmillan books & eBooks and are redeemable on [link.springer.com](https://link.springer.com) only. Titles affected by flood book price laws, forthcoming titles and titles temporarily not available on Springer Nature Link are excluded from promotions, as are reference works, handbooks, encyclopedias, subscriptions, or bulk purchases. The currency in which your order will be invoiced depends on the billing address associated with the payment method used, not necessarily your home currency. Regional VAT/tax may apply. Promotional prices may change due to exchange rates. Promotions are valid for individual customers only. Bookellers, book distributors, and institutions such as libraries and corporations, please visit [springernature.com/contact-us](https://springernature.com/contact-us). Promotions do not work in combination with other discounts or gift cards.

# Complexity of computation

- **Adding two n-digit numbers:**  $n + 1$  steps usually. But if we look at minor steps then  $5n + 1$  steps ( $n$  additions of digits,  $n$  additions of carry,  $n$  comparisons if sum of two digits is greater than 10,  $n$  steps to print lower digit,  $n$  steps to save carry). The last step is for carry save from last sum.

Even further smaller steps are taken, it comes out to be  $an + b$ , where  $a, b$  are constants, not dependent on  $n$ . Thus time complexity of add, is  $\theta(n)$ .

- **Multiplication:** To multiply  $x$  and  $y$ , one approach is to add  $x$  to 0,  $y$  times. If both numbers are  $n$  digit long, then  $\theta(n \cdot 10^n)$ .

Other method is  $\theta(n^2)$  complexity. The best known algorithm for multiplication is  $\theta(n^{1.1})$ .

- **Factoring:** of  $n$  digit number. Some times not well defined, as  $1001 = 77 \times 13$  or  $91 \times 11$ . To factor  $Z$  we need to divide it by range 2 to  $Z - 1$ . If  $|Z| = n$ , complexity is  $10^n$ . No solution like,  $\theta(n)$  or  $\theta(n^c)$ , where  $c$  is constant, is available.

# Complexity terms

- **T(n)**: Time complexity of standard Turing Machine. Function  $T(n)$  is called *time-constructible* if there exists a time-bound Deterministic TM that with input makes  $|w| = n$  moves.
- **T(n)**: Nondeterministic Turing machine Time complexity.
- **S(n)**: Space complexity of standard Turing Machine. Function  $S(n)$  is called space-constructible, if there exists a space-bound standard TM, that for each input of length  $n$ , requires exactly  $S(n)$  space.
- **DTIME(T(n))**: class of languages that have deterministic time complexity of  $O(T(n))$ .
- **NTIME(T(n))**: class of languages that have nondeterministic time complexity of  $O(T(n))$ .

# The class P

- Let  $L \subseteq \Sigma^*$ . L is polynomial if membership of  $w$  can be determined in polynomial function of  $n$ , where  $|w| = n$ . Polynomial time is in terms of number of transitions in TM. L is decidable in polynomial time if standard std TM  $M$  can decide L in  $tc_M \in O(n^r)$ , where  $r$  is natural number, not related to  $n$ . The family of L is Class-P.
- A language accepted by multi-tape TM in time  $O(n^r)$  is accepted by STD TM in time complexity  $O(n^{2r})$ , which is also polynomial. This invariant shows the robustness of TM.
- $P$ : Class of membership problems for the languages in

$$\bigcup_{P(n)} DTIME(P(n)); \text{ where } P(n) \text{ is polynomial in } n.$$

- 1 Acceptance of palindromes: Output is YES if  $w \in \Sigma^*$  is palindrome, else NO. Complexity Class=P.
- 2 Path problem in directed graphs: Input is  $G=(V,E)$ . Output is YES if there is a path from  $v_i$  to  $v_j$  in the graph, else NO. Complexity class=P, as complexity =  $O(n^2)$  due to Dijkstra's algorithm.
- 3 Derivability in CNF: Input: CNF  $G$ ,  $w$ , output=Yes, if  $S \Rightarrow^* w$  else No. Complexity: P= yes.

# The Class NP

- **Definition:** A language is in NP iff it is decided by some NDTM in polynomial time. NDTM guesses the alternatives.
- Polynomial solution for these are not known to exist.
- in *NDTM* the solution is selected nondeterministically rather than systematically examining all the possibilities.  $\therefore P \subseteq NP$ , because a P problem is also NP.
- *NP* : The class of membership problems for languages in

$$\bigcup_{P(n)} NTIME(P(n))$$

## Examples:

1. **SATISFIABILITY problem:** Input Boolean expression  $u$  in CNF, Output = YES if there is an assignment that satisfies  $u$  else NO. Complexity: In P - Unknown, in NP - YES.
2. **Hamiltonian path problem:** Input directed graph  $G$ , Output: YES if there is a single cycle that visits all nodes, No other wise. Complexity: P-unknown, NP- YES. Hamiltonian path problem is in NP, but its solution can be verified in P.
3. **Subset sum problem:** Input: Set  $S$ , number  $k$ , output: Yes if there is  $P \subseteq S$ , whose total is  $k$ , else No. Complexity: P - unknown, NP -

# Primality test and Compositeness

- $PRIMES = \{x | x \text{ is prime}\}$ ,  $COMPOSITS = \{y | y \text{ is Composite number}\}$ ,  $\therefore, PRIMES = \overline{COMPOSITS}$ ,  $\therefore$ , if COMPOSITS is NP then PRIMES is  $Co - NP$  (Complement of NP).
- COMPOSITNESS can be determined by NDTM by guessing Nondeterministically.  
COMPOSITNESS is in NP but its solution can be verified in P time.
- **Fermat's Little theorem for primality test:** If  $p$  is prime and  $a$  is integer, then:  
 $a^p \equiv a \pmod{p}$ , i.e.,  $a^p - a$  is evenly divisible by  $p$ . This problem is NP because exponential component  $a^p$ .
- Example:  $2^{11} - 2$  is divisible by 11.
- Sets of primes are in NP but not in NP-complete, similarly the COMPOSITS. The language of PRIMES is  $NP \cap Co - NP$ , and hence of COMPOSITS also.
- Because, if that is not the case then  $NP = Co - NP$ .

## Theorem

*COMPOSITS are NP*

## Proof.

- 1 *Input on NDTM =  $p$ ,  $|p| = n$ . Guess a factor  $f$  of at most  $n$  bits ( $f \neq 1, f \neq p$ ). This part is non-deterministic. The time taken by any sequence of choice is  $O(n)$ .*
- 2 *Divide  $p$  by  $f$ , check if remainder is 0. Accept if so. Part 2 is deterministic  $O(n^2)$  on STD 2-tape TM.*



- **Definition:** If there is a polynomial time algorithm for one NP-problem, then all NP problems are solvable in P time, are called NP-complete.

This is because, if  $A$  is NP-complete, then all NP-problems are reducible to it. And, if  $A \in P$ , then all those NP are  $P$ .

Adv: 1. if one can be solved, then all rest are automatically solved,  
2. One may choose only one of the most appropriate NP problem for solution.

# Complexity classes-Time

Class	Machine	Time constraint
DTIME( $f(n)$ )	DTM	$f(n)$
P	DTM	$\text{poly}(n)$
NTIME( $f(n)$ )	NDTM	$f(n)$
NP	NDTM	$\text{poly}(n)$
EXPTIME	DTM	$2^{\text{poly}(n)}$

- *EXPTIME* : The class of membership problems for this languages in

$$\bigcup_{P(n)} \text{DTIME}(2^{P(n)}).$$

- Satisfiability is NP-complete. A Boolean expression  $\phi = \{\bar{x} \wedge y\} \vee \{x \wedge \bar{z}\}$  is satisfiable for  $x=0, y=1, z=0$ , as it evaluates  $\phi$  to 1 (TRUE).
- SAT is languages of all satisfiable formulas,  $\text{SAT} = \{ \langle \phi \rangle \mid \phi \text{ is satisfiable Boolean formula} \}$ . Cook-Levin theorem links the complexity of SAT problem to complexities of all problems in NP.

- **Polynomial time reducibility:** If problem A reduces to problem B, then solution of B can be used to solve A.
  1. **Definition:** A function  $f : \Sigma^* \rightarrow \Sigma^*$  is polynomial time computable if some polynomial TM M, which when started with input  $w$ , halts with  $f(w)$  on tape.
  2. **Definition:** Language A is polynomial reducible to lang. B, expressed as  $A \leq_P B$ , if  $\exists$  a polynomial function  $f$  such that  $f : \Sigma^* \rightarrow \Sigma^*$  for every  $w \in A \iff f(w) \in B$ .
  3. To test whether  $w \in A$ , we use the reduction  $f$  to map  $w$  to  $f(w)$  and then test whether  $f(w) \in B$ ?
  4. If one language is polynomial time reducible to a language already known to have polynomial time solution, we obtain a polynomial solution to original.

## Theorem

If  $A \leq_P B$  and  $B \in P$ , then  $A \in P$ .

## Proof.

- Let  $M$  be polynomial time algorithm deciding  $B$ , and  $f$  be polynomial time reduction from  $A$  to  $B$ . We describe polynomial time algorithm for  $M'$  for  $A$  as follows:
- $M' =$  Input  $w$ , step 1. compute  $f(w)$  on TM  $R$  (reducer for  $f$ ), step 2. Run  $M$  on input  $f(w)$ .  $\therefore$ ,  $M'$  is polynomial because each of above steps are polynomial (Note: Composition of two polynomial functions is polynomial).



# NP-Complete and NP-Hard

- **Definition: NP-Complete:** A language  $B$  is NP-complete if it satisfies two-conditions: (1)  $B \in NP$ , (2) Every  $A \in NP$  is polynomial time reducible to  $B$ , i.e.,  
$$B \in NP \wedge \forall A : A \in NP \wedge A \leq_P B \Rightarrow B \in \text{NP-Complete.}$$
- A language  $Q$  is **NP-hard** if every  $L \in NP$  is polynomially reducible to  $Q$ .
- $\forall L : L \in NP \wedge L \leq_P Q \Rightarrow Q \in NP - \text{hard.}$
- The NP-hard problem that is also NP is called NP-complete.
- *Co-NP* is complement of *NP*,  $\therefore$ , *Co-NP* is set of all the complements of all the *NP* problems.

## Theorem

*If  $B$  is NP-Complete and  $B \in P$ , then  $P = NP$ .*

## Proof.

- *If  $B$  is NP-Complete then every problem in NP is polynomially reducible to  $B$ . Since  $B \in P$ ,  $\therefore$ , every NP problem is polynomially reducible to  $B$ , which is  $P$ . Hence, every NP is  $P$ , i.e.  $P = NP$ .*



Once we get NP-Complete, other NP problems can be reduced to it. However, establishing first NP-Complete problem is difficult.

## Theorem

*If  $B \in \text{NP-Complete}$  and  $B \leq_P C$  for  $C \in \text{NP}$ , then  $C \in \text{NP-Complete}$ .*

## Proof.

- *We must show that every  $A \in \text{NP}$  is polynomially reducible to  $C$ .*
- *Because  $B$  is NP-Complete,  $\therefore$ , every  $A \in \text{NP}$  is polynomially reducible to  $B$ . (as per property of NP-Complete). And  $B$  in turn is polynomially reducible to  $C$  (given).*
- *Because the property of polynomial is closed under the composition, We conclude that every  $A \in \text{NP}$  is polynomially reducible to  $C$ . Therefore  $C$  is NP-Complete.*



## Theorem

*SAT is NP-Complete.*

## Proof.

**Proof Idea:** *It is easy to show that SAT is NP, the hard part is to show that any language in NP is polynomially reducible to SAT.*

- *∴, we construct a polynomial time reduction for every  $A \in NP$  to SAT.*
- *Reduction for a language  $A$  takes input  $w$  and produces Boolean formula  $\phi$  that simulates the NP machine for  $A$  on input  $w$ .*
- *If machine accepts,  $\phi$  has a satisfying assignment, that corresponds to accepting computation, otherwise NO.*
- *∴,  $w \in A$  iff  $\phi$  is satisfiable.*

**NP-Complete problems:** *3-SAT, Hamiltonian path problem, subset construction problem.*



# Space - Complexity

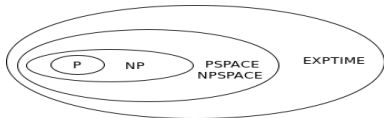
- **S(n):** The function  $S(n)$  is called space constructible if there exists an  $S(n)$  space-binded Det. TM that for each input  $|w| = n$  requires exactly  $S(n)$  space.  $\therefore, S(n) =$  Space complexity of a Det. Turing Machine.
- **DSPACE(S(n)):** class of languages that have deterministic space complexity of  $O(S(n))$ .
- **PSPACE:** The class of membership problems for the languages decidable in polynomial space on deterministic TM:

$$PSPACE = \bigcup_k DSPACE(n^k)$$

class	machine	Space constraint
DSPACE(f(n))	DTM	f(n)
L	DTM	$O(\log n)$
PSPACE	DTM	poly(n)
EXSPACE	DTM	$2^{\text{poly}(n)}$
NSPACE(f(n))	NDTM	f(n)
NL	NDTM	poly(n)
NEXSPACE	NDTM	$2^{\text{poly}(n)}$

# Space - Complexity

- $DSPACE(f(n)) = \{L \mid L \text{ is decidable by } O(f(n)) \text{ space on DTM}\}$ .
- $NSPACE(f(n)) = \{L \mid L \text{ is decidable by } O(f(n)) \text{ space on NDTM}\}$
- **Savitch's Theorem:** If a *NDTM* uses  $f(n)$  space, it can be converted into a *DTM* that uses  $f^2(n)$  space.
- As per Savitch's theorem:  $PSPACE = NSPACE$ ,  
 $EXSPACE = NEXSPACE$ .
- For *NDTM*, if  $f(n)$  is maximum number of tape-cells scan in any branch of computation, then its complexity is  $f(n)$ .
- *SAT* which is *NP-Complete* in time, is linear space. (because is reusable).
- $PSPACE = NSPACE$ ,  $P \subseteq PSPACE$ .  $NP \subseteq NSPACE$ ,  
 $\therefore NP \subseteq PSPACE$ .
- $\therefore P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$ .





Chowdhary, K.R. (2025). Decidability, Undecidability, and Unsolvability. In: Theory of Computation. Springer, Singapore.  
[https://doi.org/10.1007/978-981-97-6234-7\\_12](https://doi.org/10.1007/978-981-97-6234-7_12)