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

Monday 10^{th} April, 2017

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 if add x to 0, y times. If both numbers are n digit long, then $\theta(n.10^n)$.

  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 91×11. 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

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

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. Deriviability 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 toatl 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 Nondterministically.

  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:

  $a^p \equiv a(mod \text{ p})$, 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 $NP = Co - NP$.

Primality test and Compositeness ...

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.*

$\square$

- **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	poly(n)
NTIME(f(n))	NDTM	f(n)
NP	NDTM	poly(n)
EXPTIME	DTM	$2^{poly(n)}$

- *EXPTIME* : The class of membership problems for this languages in
$$\bigcup_{P(n)} DTIME(2^{P(n)}).$$
- 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 ϕ to 1 (TRUE).
- SAT is languages of all satisfiable formulas, $SAT = \{< \phi > | \phi$ is satisfiable Boolean formula $\}$. Cook-Levin theorem links the complexity of SAT problem to complexities of all problems in NP.

Polynomial reduction

- **Polynomial time reducibility:** If problem A reduces to problem B, then solution of B can be used to solve A.

  1. **Definition:** A function $f : \Sigma^* \to \Sigma^*$ 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 $f : \Sigma^* \to \Sigma^*$ for every $w \in A \Longleftrightarrow f(w) \in B$.

  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.

Polynomial reduction

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).*

$\square$

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.,

  $B \in NP \wedge \forall A : A \in NP \wedge A \leq_P B \Rightarrow B \in$ NP-Complete.
- A language Q is **NP-hard** if every $L \in NP$ is polynomially reducible to Q.
- $\forall L : L \in NP \wedge L \leq_P Q \Rightarrow Q \in NP - hard.$
- 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.

NP-Complete Theorem

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 to B, which is P. Hence, every NP is P, i.e. $P = NP$.*

$\square$

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

NP-Complete Theorem

Theorem

If $B \in$ NP-Complete and $B \leq_P C$ for $C \in NP$, then $C \in$ NP-Complete.

Proof.

- *We must show that every $A \in NP$ is polynomially reducible to C.*
- *Because B is NP-Complete, $\therefore$, every $A \in NP$ 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 $A \in NP$ is polynomially reducible to C. Therefore C is NP-Complete.*

$\square$

Cook-Levin Theorem

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.*

- $\therefore$, *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* ϕ *that simulates the NP machine for A on input w.*
- *If machine accepts,* ϕ *has a satisfying assignment, that corresponds to accepting computation, otherwise NO.*
- $\therefore$, $w \in A$ *iff* ϕ *is satisfiable.*

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

$\square$

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)
EXPSPACE	DTM	$2^{poly(n)}$
NSPACE(f(n))	NDTM	f(n)
NL	NDTM	poly(n)
NEXPSPACE	NDTM	$2^{poly(n)}$

Space - Complexity

- $DSPACE(f(n)) = \{L|L$ is decidable by O(f(n)) space on DTM$\}$.
- $NSPACE(f(n)) = \{L|L$ is decidable by O(f(n)) 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: $PSPACE = NSPACE$, $EXPSPACE = NEXPSPACE$.
- For NDTM, if $f(n)$ is maximum number of tape-cells scan in any branch of computation, then its complexity if $f(n)$.
- SAT which is $NP - Complete$ in time, is linear space. (because is reusable).
- $PSPACE = NSPACE$, $P \subseteq PSPACE$. $NP \subseteq NSPACE$, $\therefore, NP \subseteq PSPACE$.
- $\therefore, P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$.