Space Complexity

Space complexity is a function describing the amount of ~~memory~~ memory (space) an algorithm takes in terms of the amount of i/p of the algorithm.

Definition:

Let M be a deterministic TM that halts on all i/p s. The space complexity of M is the function $f: N \to N$, where $f(n)$ is the max^m no. of cells of tape & M scans any i/p of length n.

If the space complexity of M is $f(n)$, we can say that M runs in space $f(n)$.

We estimate the space complexity of TM by using asymptotic notation.

Let $f: N \to R^+$ be a fn.

The space complexity classes can be defined as follows —

$Space = \{ L \mid L$ is a language decided by an $O(f(n))$ space deterministic TM$\}$

$Space = \{ L \mid L$ is a language decided by an $O(f(n))$ space non-deterministic TM$\}$

PSpace ⇒ is the class of languages that are decidable in polynomial space on a deterministic TM.

$\therefore \quad PSpace = \bigcup_k Space(n^k)$

Date :
Page no.

Savitch's Theorem (Related to space complexity)

According to this theorem, a deterministic machine can simulate non-deterministic machines by using a small amount of space.

A TM that uses $f(n)$ space can be converted to deterministic TM that uses $f^2(n)$ space.

Hence, Savitch's theorem states that, for any fn.

$$f : N \to R^+, \text{ where } f(n) \geq n$$

$$NSpace(f(n)) \subseteq Space(f(n))$$

Cook's Theorem:

The satisfiability problem (SAT) is NP-complete.

What is SAT?

A propositional logic formula ϕ is called satisfiable if there is some assignment to its variable that makes it evaluate to true.

→ $P \wedge q$ is satisfiable (if $P=1$ & $q=1$)

→ $P \wedge \neg P$ (not satisfiable)

3SAT

A language $3SAT = \{ \phi \mid \phi$ is satisfiable 3-CNF formula$\}$.

$(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee z)$ ← It is in CNF (Conjuctive Normal Form) and every clause has exactly 3 literals

↓ Clauses ↓ OR

Proof :—

Theorem: SAT is NP-complete

→ Proof consists of 2 steps

1. Convert the execution of a polynomial time Non Deterministic TM to a bunch of well formed formulae such that

formula satisfies iff the machine accept input.

2. Show the sum of lengths of formulae is polynomial in the size of problem.

- NP – Hard → Can polynomially reduce any NP problem to L.
- NP – Complete → L ∈ NP
- L ∈ NP → NDTM (Non Deterministic TM) for L that runs in polynomial time.
- An NDTM is the only model we have for NP problem.
- SAT ∈ NP
- Therefore, if we can polynomially reduce and an arbitary polynomial NDTM to SAT. Its mean we have proven SAT is NP – complete.

Bittoo TRADE MARK REGD.

Date :

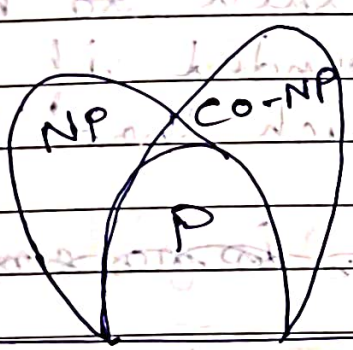
Page no.

Co-NP — It is the collection of ~~completeness~~ complements of languages in NP, and P is closed under complementation, Co-NP can also be ~~characterised~~ characterised as the collection of languages of the form:

$$L = \{x \mid \forall y \mid y < p(|x|) \rightarrow R'(x, y)\}$$

NP — The collection of languages with succinct (means briefly or clearly explained) certificates of membership.

Co-NP — The collection of languages with succinct certificates of disqualification.



$P = NP = Co\text{-}NP$

$\therefore P = NP \cap Co\text{-}NP \neq NP \neq Co\text{-}NP$

$P \neq NP \cap Co\text{-}NP = NP = Co\text{-}NP.$

$P \neq NP \cap Co\text{-}NP \neq NP \neq Co\text{-}NP$

Co-NP: Complete

VAL — The collection of Boolean Expressions that are valid is Co-NP complete.

Any language L that is the complement of an NP-complete language is Co-NP-complete.

Any reduction of a language L1 to L2 is also a reduction of L1- the complement of L1 - to - L2 - the complement of L2.

There is an easy reduction from the complement of SAT to VAL, namely the map that takes an expression to its negation.

VAL ∈ P ⇒ P = NP = co-NP
VAL ∈ NP ⇒ NP = co-NP

** Computability :-

Theorem 1 : A TM is said to be polynomially bounded if there is a polynomial p(n) such that the following is true:

~~For any i/p, there is no configuration such that~~

For any i/p x, the machine always halts after a most p(n) steps, where n is the length of the i/p.

* A language is called polynomially decidable if there is a polynomially bounded TM that decides it.

The class of all polynomially decidable languages is denoted P.

Theorem 2: P is closed u
complement.

Proof. If a lang. L is
by a polynomially
M, then its complement is
the version of M that inve

Obviously, the polynomial
unaffected.

e.g. E = {"M", "w" : M accep
after at m

Based on halting prob.lem

Theorem 3 : E ∉ P

Proof. E1 = {"M" : M accepts "
at most 2|"M"

Undecidability of halting proble

L, NL, NL - Completeness

L ⊆ NL ⊆ P ⊆ NP ⊆ PSpace ⊆ E

Any reduction of a language L_1 to L_2 is also a reduction of L_1 - the complement of L_1 - to - L_2 - the complement of L_2.

There is an easy reduction from the complement of SAT to VAL, namely the map that takes an expression to its negation.

$$VAL \in P \Rightarrow P = NP = co\text{-}NP$$
$$VAL \in NP \Rightarrow NP = co\text{-}NP$$

** Computability :-

Theorem 1 : A TM is said to be polynomially bounded if there is a polynomial $p(n)$ such that the following is true :

~~For any i/p, there is no configuration such that~~

For any i/p x, the machine always halts after a most $p(n)$ steps, where n is the length of the i/p.

* A language is called polynomially decidable if there is a polynomially bounded TM that decides it.
The class of all polynomially decidable languages is denoted P.

Theorem 2: P is closed under complement.

Proof. If a lang. 'L' is decidable by a polynomially bounded TM M, then its complement is decided by the version of M that inverts 'y' and 'n'.

Obviously, the polynomial bound is unaffected.

e.g. $E = \{\text{"M"} \text{ "w"} : M \text{ accepts i/p } w \text{ after at most } 2^{|w|} \text{ steps}\}$.

→ Based on halting prob.lem

Theorem 3 : $E \notin P$

Proof. $E_1 = \{\text{"M"} : M \text{ accepts "M" after at most } 2^{|\text{"M"}|} \text{ steps}\}$;

Undecidability of halting problem.

L, NL, NL- Completeness

$L \subseteq NL \subseteq P \subseteq NP \subseteq PSpace \subseteq Exp \subseteq NExp$