Unit 4

Saturday, April 19, 2025 3:33 PM

THEORY OF COMPUTATION

UNIT-IV

Time Complexity -:

- No. of moves which are required to perform computation
- No. of machine cycles

PROBLEM ⟶ ALGORITHM (Sol^n) → TIME COMPLEXITY

- BIG-OH — Max no. of cycles
- BIG Θ (Theta) — Min no. of cycles
- Big Ω (Omega) — Avg. no. of cycles

TRACTABLE PROBLEM	INTRACTABLE PROBLEM
(i) A problem which has an efficient Algorithm to solve it.	(i) There is no efficient algorithm to solve it
(ii) which can be solved in Polynomial time	(ii) A problem that cannot be solved in Polynomial time
eg $O(n)$, $O(n^2)$, $O(n \log n)$	eg Tower of Hanoi $O(2^n)$. There Hamiltonian cycle

Scanned with OKEN Scanner

P and NP Class -:

Polynomial time class	Non-Deterministic Polynomial time class
1) This class includes those languages which are recognized by DTM in P Time	1) This class includes those languages recognized by NDTM in P time
2) You can solve it quickly in Polynomial time	2) You cannot solve it necessarily, but can verify in Polynomial Time — Yes
eg Sorting, Matrix Multiplication	eg SAT (Boolean satisfiability), Hamiltonian cycle
(P class)	NP Class

CO-NP Class -:

NP-class → NO

- P Class
- NP Class
- CO NP Class

LOCKERS 1000

(i) Is key Present in Locker No. 237? Yes/NO

(ii) Is key Present in any locker? NO

237

Scanned with OKEN Scanner

SATISFIABILITY PROBLEM → Check for the answer as True

$x_i = \{x_1, x_2, x_3\}$

$CNF = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$ } find for all such that x_i is true

CNF — Conjunctive Normal form

$2^3 = 2^n$

$O(2^n)$ exp

x_1	x_2	x_3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

8 cases $= 2^3 = 2^n$

$O(2^n) \rightarrow$ exp

REDUCTION → Transformation of one problem to another problem

Sat α THanoi (POLYNOMIAL TIME)

I_1 — NP Hard, I_2 — NP Hard

Transitivity

Sat α L

L α X

⇒ Sat α X

$I_1 \alpha I_2 \rightarrow$ sol^n

Reduction

Ch-3

A —Reduce→ B (solve, easy)

S/S Difficult

Because

All x α

Th

$L \alpha X$, $X \alpha Y$ } $L \alpha Y$

Ref transi

$(a,b) \in R$

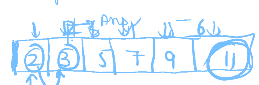
$(b,c) \in R$

$(a,c) \in R$

Polynomial

Time Complexity

Algorithm



- Avg. Case: Find 3 → 2 cycles / Moves Ω( )
- Worst Case: Find 11 → 6 cycles / Moves O( )
- Best Case: Find 2 → 1 cycle / Moves Θ( )

Polynomial time complexity

n size array / cycle

Worst case → n times → $O(n)$, Polynomial expression $O(n^2)$ $O(n^3)$ $O(n \log n)$

Polynomial ✗

eg Tower of Hanoi → exp time $O(2^n)$

NP Class → Yes/NO

Time C → efficiency

$O(n) < O(n^2) < (2^n)$

P.T.C (eff) < NPTC (eff) } Generally

exp — Polynomial time

P Class:
- Linear Search $O(n)$
- Binary Search $O(n \log n)$
- Merge Sorting

NP Class: exp → Sol^n Polynomial time

P Class, NP Class, CoNP Class, NP Hard, NP Complete

P Class : Lang / Problems — DTM

NP Class — NDTM

DTM: Moves ≠ surity ✓

NDTM: Moves ≠ surity ✗

(a, a, R) — surity

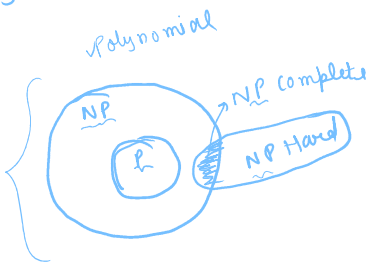
(a, a, R), (a, b, R) } Ambiguity

2 SAT → 4 → 2^2

1 SAT → 2 → 2^1

Quick Notes Page 1

All x α { L α X, X α Y } ⇒ L α Y



NP Hard NP Complete

(a) P class → Polynomial time ✓
NP class → Polynomial time ✓

NP Hard → A problem is said to be belonging to NP hard class iff all problems of NP class can be reduced to it.

NP Complete → ✓ NP Hard
✓ P ∈ NP

// Cook Levin Theorem / Cook's Theorem

✓ Statement :- SAT Problem is NP complete.

Exp :- ✓ SAT ∈ NP
✓ SAT is NP Hard → Any NP Problem can be reduced to SAT in Polynomial time

Proof :-

- A Logic Problem/formula ϕ is called SAT if there is some value/variable that makes it true
- It is generally in CNF (Conjunctive Normal form)
  eg. P ∧ Q → P = 1, Q = 1

Step 1 → SAT is NP

SAT is NP because a Non deterministic Turing machine can guess an assignment of True value (Truth value) of n variables (ie 3 in case of 3 SAT Problem). ∴ ~~SAT is NP, SAT ∈ NP~~

Step 2 → The Non deterministic algorithm also determines the value of expression for corresponding assignment and accepts if true. The algorithm is composed of an input tape with the tape having finite no. of cells, thus verifying also in Polynomial time
∴ SAT is NP, SAT ∈ NP

Step - 3 To Prove NP Hard

Experimentally, verified and even using reduction we can verify that all problems in NP class can be reduced to SAT Problem.

$\chi_{NP} \alpha$ SAT (exp) — ①
$\chi_{NP} \alpha$ SAT (exp) — ②

Using ① & ② we can say

$\forall \chi_{NP} \alpha$ SAT [where $\chi_{NP} \in NP$]

∴ SAT is NP Hard

Y/S → S/S

Step - 4 :- We have verified SAT is NP Hard as well as SAT ∈ NP. Hence, we can say that SAT is NP Complete

(b) Hamiltonian Cycle Proof → NP Complete
① Hamiltonian cycle is NP
② Hamiltonian α SAT (exp/conversing (Cook theorem)
✓ SAT α Hamiltonian [Transitivity]
χ_{NP} α Hamiltonian → NP Hard

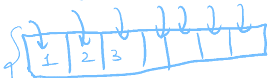
③ Hamiltonian ∝ SAT } ~~scip~~ / conversing (cook theorem)

SAT ∝ Hamiltonian [Transitivity]

x NP ∝ Hamitonial → NP Hard

(Storage req. for comp)

** SPACE COMPLEXITY (space)



Turing Machine

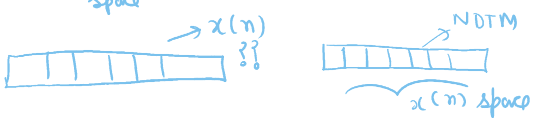
- P space → Set of Lang decided by DTM in P time
- NP space → Set of Lang decided by NDTM in P time

D Space ($x(n)$)

L | L is decided by a DTM in $x(n)$ space

N Space ($x(n)$)

L | L is decided by a NDTM in $x(n)$ space



SAVITCH THEOREM

SAVITCH THEOREM

Statement → This theorem states that any NTM can be simulated by a DTM with atmost a quadratic increase in the amount of space required.

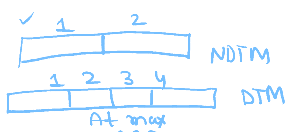
$$NSPACE(s(n)) \subseteq DSPACE(s^2(n))$$

NDTM   DTM

NDTM → space $x(n)$

DTM → $x^2(n)$

$x(n) \times x(n)$



Proof

Let us consider a Turing machine which is non-deterministic solves a problem in $s(n)$ (Polynomial space)

NTM : $C_1 \to C_2$: $s(n)$ —— ①
(state) (state)

DTM : $C_1 \to C_2$ —— ②
(state) (state)

We have to find if case ① can be yielded from ②

DTM → NDTM } Assume

We have to limit t

$t = 2^{s(n)}$ configuration

SAT 2^n

Recursive depth / time t

$t = 2^{sn}$ Reduce

{ As we can clearly see t reduces to Can yield problem where we have

can yield (C_1, C_2, t)

Recursive

can yield (C_1, C_2, t) { Base case;
can yield (C_1, m, t);
can yield (m, C_2, t); }

Delhi —t— Faridabad —t→ C_2

$C_1 \xrightarrow{t} C_2$

$C_1 \xrightarrow{t/2} m \xrightarrow{t/2} C_2$

$C_1 \xrightarrow{t/4} m_1 \xrightarrow{t/4} m \xrightarrow{t/4} m_2 \xrightarrow{t/4} C_2$

Clearly, as we reduce this further

$t \to \frac{t}{2} \to \frac{t}{4} \to \ldots \quad \frac{t}{2^i}$

$\frac{t}{2^i} = 1$

$\frac{t}{2^i} = 1$

$\Rightarrow t = 2^i$ } exp

$t = 2^i$

$\log t = \log(2^i)$

$\log_2 t = i$ } Recursion depth

$\log t = i$ → Recursion depth

$t = 2^{s(n)}$

$\log t = \log 2^{s(n)} = s(n)$ — we need $s(n)$ multiple steps

$\log t = i$

$i = \log 2^{s(n)}$

$i = s(n)$

$s(n) \times s(n) = s^2(n)$

NDTM → DTM

$s(n) \times i$

$s(n) \times s(n)$

Hence we can see that we can execute it in quadratic duplication of space.

Probibalistic Space

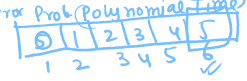
NDTM → DTM

Probabilistic Space

BPP → Randomized way

Bounded error Prob (Polynomial Time)

Error ≤ 1/3



1st step

time complexity efficient

$S(n) \times i$

$S(n) \times S(n)$

$S^2(n)$

✓ Bounded Error Probabilistic Polynomial time class (BPP class)

- ✓ Randomized way
- ✓ Polynomial time
- ✓ Error ≤ 1/3
- ✓ Probabilistic Turing

Error = 1/3

Correct = 2/3

Algo ✓✓

Formal definition → A language L is in BPP if it can be accepted by a Probabilistic T.M, such that

- If $x \in L$, then Pr [M(x) = accept] ≥ 2/3
  $x \notin L$, then Pr [M(x) = reject] ≥ 2/3
  } success (2/3)
- If $x \in L$, then Pr [M(x) = reject] ≤ 1/3
  $x \notin L$, then Pr [M(x) = accept] ≤ 1/3
  } error (1/3)

Probabilistic Turing Machine → Randomized way Turing machine with a high Prob. of right answer.

⊛ solves algorithm in P time

BPL → BPP + Restriction [space required should be Log]

space comp → log (n)

ZPP → BPP + [Error = 0]

INTERACTIVE PROOF SYSTEMS

Used to prove using → Prover (P) → Trust ⊗

→ Verifier (V) → Probabilistic Turing Machine

Prover → Verifier (Yes/No)

Prover → Pass on statements to verifier ✓✓

Verifier → uses randomness, to check the truth values

① Class IP → Class of lang having Interactive Proof System

$L \in IP$, if there exist a polynomial time verifier

- ✓ Error chances are there
- ✓ More the no. of verification, less is the error

Imp vs Prof

IP = PSPACE { Anything you can verify using IP theorem can be computed using Polynomial space }

② ORACLES → Black Box → Instantly answers queries

Problem X Instant solution Pspace, NSpace, DSpace (Instantly)

Proof of IP theorem -: (PYQ) — 7.5 Marks.

IP = PSPACE } → Prove

Statement :- Any Problem that can be verified using IP can be solved in polynomial time.

Proof -: For a language to exist in IP class is for $L \in IP$

solved in polynomial time.

Proof -: For a language to exist in IP class it for $L \in IP$ there should be a protocol such that — 1.5

- ✓ P(success) > 2/3
- ✓ P(failure) < 1/3

To prove

PSPACE ⊆ IP ✓ — (1)

IP ⊆ PSPACE ✓ — (2)

⇒ PSPACE = IP → final statement } (2)

V, P

Let us assume we have a problem that belongs to IP class
QBF
QBF?

IP ∝ QBF

⇒ IP ⊆ PSPACE

(1)

→ Polynomial time, Polynomial, Prover & verifier

P space ∝ 1

→ PSPACE ⊆ IP — (2)

From (1) & (2)

IP = PSPACE

COMPLETED ✓✓

Imp
- ✓ P class, NP class, CONP, NP, NP Hard
- ✓ SAT problem (5-75) (ask the exam + Proof)
- ✓ SAVITCH (statement)