Billoo

Date :

Page no.

Complexity:

When a problem / language is decidable, it simply means that the problem is computationally solvable in principle.

Growth Rate of Functions:

When we have 2 algorithms for the same problem, we may require a comparison between the running time of these 2 algorithms..

Definition: Let $f, g : N \rightarrow R^+$ (R^+ being the set of all ~~possible~~ +ve real nos.) we say that $f(n) = O(g(n))$ if there exist +ve integers c and N_0 such that

$$f(n) \leq c\,g(n) \text{ for all } n \geq N_0$$

Here, f is 'big Oh' of g.

** $f(n) = O(g(n))$ is not an eq^n, it only ~~denotes~~ expresses a relation btn. 2 fns. f and g.

Q: Let $f(n) = 4n^3 + 5n^2 + 7n + 3$.
P.T. $f(n) = O(n^3)$

Sol^n: Let us take $\boxed{C = 5}$ ← ~~Not~~ compulsory for this prob. and $N_0 = 10$

$\therefore$ $f(n) = 4n^3 + 5n^2 + 7n + 3$ for $n \geq 10$
$\Downarrow \qquad \leq cn^3$

take $n = 10$, $f(n) = 573 < 10^3$ $\therefore f(n) = O(n^3)$

$n > 10$ $\qquad < 11^3$

e.g. $n = 11$ $\qquad < n^3$

Scanned by TapScanner

Date :
Page no.

The classes P and NP

P-class

→ A TM is said to be of time complexity T(n) if the following holds: Given an input w of length n, M halts after making at most T(n) moves.

→ A language L is in class P if there exists some polynomial T(n) such that L = T(M) for some deterministic TM M of time complexity T(n).

~~construct the time complexity T(n) for the TM, [illegible]~~

Q: Find the running time for the Euclidean Algo. for evaluating gcd(a,b) where a & b are +ve integers expressed in binary representation.

Solⁿ ⇐ Polynomial algorithm

The Euclidean algo. is based on the below facts

1. If we subtract smaller number from larger no., GCD doesn't change. so, if we keep subtracting repeatatedly the larger of 2, we end up with GCD.

2. Instead of subtraction, if we dive smaller no., the algorithm stops when we find remainder '0'.

Scanned by TapScanner

Date :
Page no.

e.g. 36 , 60

$36 = 2 \times 2 \times 3 \times 3$
$60 = 2 \times 2 \times 3 \times 5$

GCD = Multiplication of common factors
$= 2 \times 2 \times 3$
$= 12$

In other words.
⇒ The algo has the following steps :

1. The i/p is (a, b)
2. Repeat unit b = 0
3. Assign $a \leftarrow a \bmod b$
4. Exchange a & b
5. Output a .

step 3 replaces a by a mod b.
If $a/2 \geq b$, then $a \bmod b < b \leq a/2$.

If $a/2 < b$,
then $a < 2b$

Write $a = b + r$ for some $r < b$.
Then $a \% b = r < b < a/2$

∴ $a \% b \leq a/2$
So, a is reduced by at least half in size on the application of step 3.

∴ One iteration of step 3 & 4 ~~red~~ reduces a & b by at least half in size.

Date :
Page no.

$\therefore$ The max^m no. of execution of step 3 & are $(\log_2 a)$ and $(\log_2 b)$
The no. of iterations of step 3 & 4 is $O(n)$.

$\Rightarrow$ we have to perform step 2 at most $\min\{[\log_2 a], [\log_2 b]\}$ times or $\underline{n}$ times,
$\therefore T(n) = n\,O(n) = O(n^2)$

NP Class :

$\rightarrow$ A language L is in class NP if there is a non-deterministic TM M and a polynomial time complexity $T(n)$ such that $L = T(M)$ and M executes at most $T(n)$ moves for every input w of length n.

Polynomial time Reduction & NP completeness:

Theorem :

1. If there is a polynomial time reduction from P_1 to P_2 and if P_2 in P, then P_1 is also in P.

Date :

Page no.

2. Let L be a language or problem in NP. Then L is NP-complete if

i) L is in NP
ii) For every language L' in NP there exists a polynomial-time reduction of L' to L.
iii) The class of NP complete language is a subclass of NP.

3. If P_1 is NP-complete and there is a polynomial time reduction of P_1 and P_2, then P_2 is NP-complete.

4. If some NP-complete problem is in P, the P = NP.

Scanned by TapScanner