Procedure Even Odd (N)
Begin
1. $R \leftarrow N \% 2$
2. If $(R == 0)$ Then
3. Print 'Number is even'
4. Else
5. Print 'Number is odd'
6 EndIf
End.

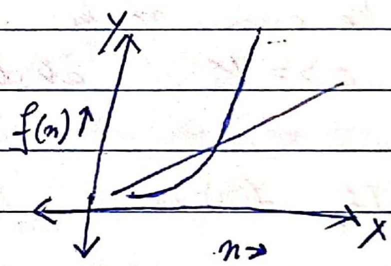
Date 15/09/21

lecture - 3

Earlier class → freq. $= 2n + 3$
→ n :→ size of input
→ size of output

$f(n) = 3n^4 + 10n^3$
$f(n) = 2^n$

order of growth →



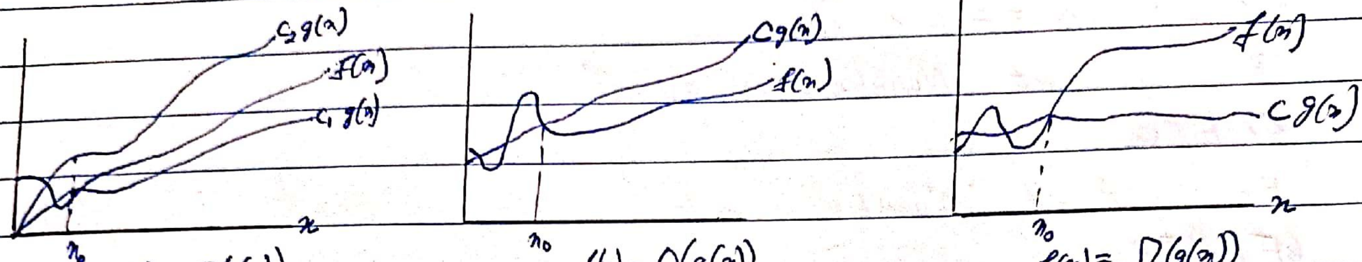
- Time complexity
- Space complexity

- Meaning of Asymptotic Analysis

It means that the analysis is valid when the value of n [size of input or size of output] is very large.

O-Notation (Big O Notation):

$O(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 <= f(n) <= c \cdot g(n)$ for all $n >= n_0\}$



$f(n) = \Theta(g(n))$ (a) $f(n) = O(g(n))$ (b) $f(n) = \Omega(g(n))$ (c)

Let $f(n) = \frac{1}{2} n^2 - 3n$ ← We want to check whether $f(n) = O(n^2)$ → $g(n) = n^2$

We need to find constant c such that $f(n) <= c \cdot n^2$

So, $\frac{1}{2} n^2 - 3n <= c \cdot n^2$

Dividing both sides by n^2,

$\frac{1}{2} - \frac{3}{n} <= c$

We can make the inequality hold by taking as const. $c >= \frac{1}{2}$ and $n >= 1$

If $f(n)$ is a polynomial of order k, then $f(n) = O(n^k)$

Example: $f(n) = 4n^3 + 3n^2 + 10$, $f(n) = O(n^3)$

⟹ Big O notation is not asymptotically tight.

Let, $f(n) = 3n^2$

Then, $f(n) = O(n^2)$

Also, $f(n) = O(n^3)$

+ $O(1)$ means constant time, that is the time does not depend on size of input or size of output.

Aj - ⑤ Aj ⑥

Date 18/09/21

Lecture - 4

Θ Notation :

$\Theta(g(n)) = \{ f(n)$: there exist positive constants c_1, c_2 and n_0 such that $0 <= c_1 g(n) <= f(n) <= c_2 g(n) \}$

Let $f(n) = \frac{1}{2} n^2 - 3n$

We want to check whether $f(n) = \Theta(n^2)$

We need to find constant c_1, c_2 such that $0 <= c_1 n^2 <= f(n) <= c_2 n^2$

So, $\frac{1}{2} n^2 - 3n <= c_2 n^2$ $[g(n) = n^2]$

$=> \frac{1}{2} - \frac{3}{n} <= c_2$ [Dividing both sides by n^2]

We can make this equality hold by taking a constant $c_2 >= \frac{1}{2}$ and $n >= 1$.

Now,

$c_1 n^2 <= \frac{1}{2} n^2 - 3n$ $[c_1 g(n) <= f(n)]$

$=> c_1 \leq \frac{1}{2} - \frac{3}{n}$ [Dividing both sides by n^2]

This inequality can be made to hold by taking $n \geq 7$ and $c \leq 1/4$.

So, the given $f(n)$ is $\Theta(n^2)$

Ω Notation (Big Omega Notation) :

$\Omega(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 <= c \cdot g(n) <= f(n)$ for all $n >= n_0$.

3 important topics :

① Best case time complexity analysis

② Worst case time complexity analysis

③ Average case time complexity analysis

Linear Search in Array

0	1	2	3	4	5	6	7
20	30	10	5	8	60	90	15

[Search 60]

~~for(i=0; i<7; i++)~~

for 'n' elements $\rightarrow f(n) = n$

Best case $\rightarrow O(1)$ [if searching for 20]

Worst case $\rightarrow O(n)$ [if element is present at ~~nth~~ last place]

Average case

As equally likely in all position, the probability in any position is $(1/n)$

$\therefore \quad 1 \times 1/n + 2 \times 1/n + \cdots + n \times 1/n$ [Position 1 : $1 \times 1/n$, " 2 : $2 \times 1/n$]

$= 1/n \, (1 + 2 + 3 + \cdots + n)$

$= \frac{1}{n} \times \frac{n(n+1)}{2}$

$= \frac{n+1}{2}$

$\therefore f(n) = O(n)$ [$\because \frac{n+1}{2}$ is a polynomial in n]

An example : Multiplication of matrix $A[n \times n]$ and matrix $B[n \times n]$

- Input : matrices A and B
- Let C be a new matrix of the appropriate size
- for i from 1 to n : $\longrightarrow$ 'n' times
  - for j from 1 to n : $\longrightarrow n^2$
    - Let sum $= 0$
    - for k from 1 to n : $\longrightarrow n^3$
      - Set sum $\leftarrow$ sum $+ A_{ik} \times B_{kj}$
    - Set $C_{ij} \leftarrow$ sum
- Return C.

L-5

A-7

Date
20/09/21

$i = 1$, $j = 1 - - n$ | $j = 1$, $k = 1, - -, n$

$i = 2$, $j = 1, - -, n$ | $j = 2$, $k = 1, - -, n$

$n + n^2 + n^2 + n^2 + n^3 + n^2 + 1$

$f(n) = n^3 + 4n^2 + n + 1$

$\therefore O(n) = n^3$

Data Structure :-

Linear List :-

— An ordered list of elements.

( )

(10)

(10 8)

(10 8 20 30 15)

predecessor

successor.

22/09/21

Stack :- A stack is a linear list where all ~~the~~ insertions and deletions are made at one end of the list. This end is called the 'top' of the stack.

- Last In first Out (LIFO)
- Insert a new element → push
- Delete → pop.

Scanned with CamScanner