CS & IT ENGINEERING

Algorithms

Analysis of Algorithms

Lecture No.- 02

By- Aditya sir

Recap of Previous Lecture

Topic Intro Schedule

Topic Intro to Algo

Topics to be Covered

Topic

Topic

Topic

Analysis
- → Why ?
- → What ?
- → How ?

About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 12,000+ students & working professions in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.

@ADITYASIR_PW

Telegram Link for Aditya Jain sir: https://t.me/AdityaSir_PW

Topic : Lecture Schedule

Algorithm:-

- An Algorithm is a collection of finite number of instruction to solve a given problem.
- These instruction are fundamental and should follow a proper sequence.
- It Should be unambiguous in nature.
- An Algorithm should be terminated after finite time.
- It Should produce at least one output.
- It is independent of programming language .

Topic : Lecture Schedule

Input (i/p) → Algorithm → Output (o/p)

[An algorithm may take 0 or more inputs]

[Must always produce at least one output]

Topic : Lecture Schedule

1 Algo AJSir ( ) → 0 inputs

```
{
    printf ("Hello students!");   →   One output
}
```

2 Algo AJsir2( );

```
{
    return 100;
}
```

Topic : Lecture Schedule

Algorithm → Pseudo code- For i: $1 \to n$

(Set of sequential rules /statement /instructions)

Program→ Algorithm implemented using some programing language

Python:- for i in range (0, n+1)

C++:- for (i =0; i ≤ n; i + +)

Topic : Lecture Schedule

Analysis of Algorithm:-

1. Why to analysis ?→ Compare two Algo.
2. What to analysis?→ on the basis of time/space {Reduce consumption}
3. How to analysis?

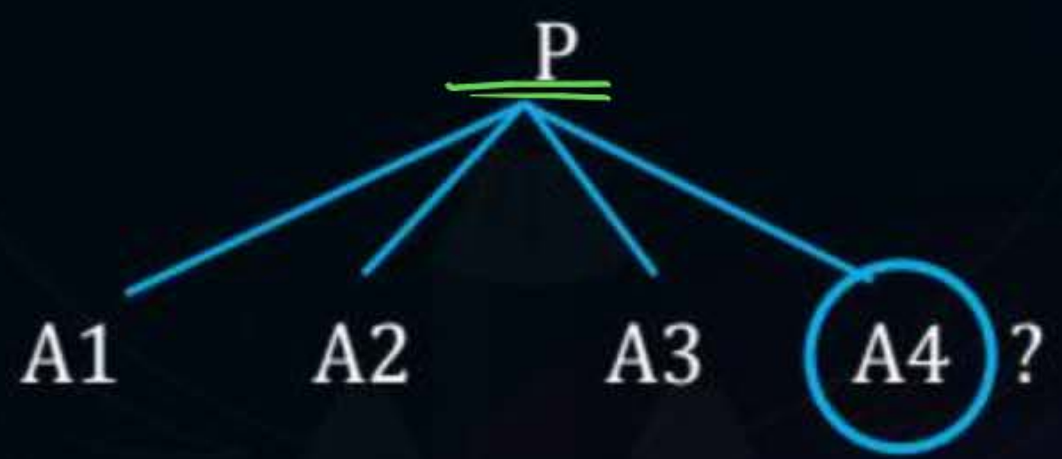
Topic : Lecture Schedule

#Q. Why to Analysis ?

To compare and decide which solution is the best among different available solution (Algo.)

Problem :- Nagpur → Delhi

1. Walk → Soln. (A1)
2. Bike /Car→ (A2)
3. Train→ (A3)
4. Flight→ (A4)



Time ↓ cost ↑ (Money)

Topic : Lecture Schedule

Maggie (P)

Self Cook	Chef	Zomato	Visit Restaurant
(A1)	(A2)	(A3)	(A4)

Topic : Lecture Schedule

#Q. What to Analysis ?

Analys~~is~~e the consumption of Resources

Resources:–

Time, space /memory,

Money, Intern~~al~~et Bandwidth,

Number of programmers, etc..

Topic : Lecture Schedule

#Q. How to Analysis ?

How to Analysis ?

1. Aposteriori Analysis

Analyzing After Implementation of Algo.

2. Apriori Analysis

Analyzing before implementation of Algo.

Topic : Lecture Schedule

1. Aposteriori Analysis

Advantages

- Gives the exact measurement of time units.

  High level lang → User/Program friendly

  Low level lang → Machine friendly

Disadvantage

- Platform dependent

  Software → OS (Linux, windows, MAC)

  Hardware → (Process /CPU , memory (RAM))

  Programming language dependent

Topic : Analysis of Algorithms

2. Apriori Analysis

Carried out before the actual implementation of Algorithm.

Advantages	Disadvantages
1. Platform independent	1. It gives approximate values of time and space complexity
2. Can be carried out without actual implementation	
3. Helps us to compare the relative performance of 2 or more algorithms.	

Topic : Analysis of Algorithms

Eg.1.

Maggie

A1	A2
x units	x^2 units

$x < x^2$

Better for large values of x.

Comparison can be made
With approximate values

Topic : Analysis of Algorithms

- Steps to carry out Apriori Analysis:
  - Pseudo code of the algorithm is required
  - Need a matrix to compare two or more algorithm running time.
  - Need some notation/ symbol to represent the running time.

Asymptotic

Topic : Analysis of Algorithms

- Apriori Analysis can be carried out in 2 ways.

1. Step- count method → gives exact no. of operations (not much used)
2. Order of magnitude method → gives approximate no. of operations.

Assumption:- Every fundamental operation takes 1 unit of time.

Topic : Analysis of Algorithms

1. Step-count method:-

```
Algo. AJSir(n) {
  1.    p = q + r                          → 2units
  2.    x = y * z                          → 2units

  3.    for (i = 1; i ≤ n ; i++) {         → (4n + 2) units
                a = b + c;
        }
  4.    for (j = 1; j ≤ n; j++) {          → (4n^2 + 4n + 2) units

            for (k = 1;  k ≤ n; k ++) {
                x =  y + z;
            }
        }
}
```

Topic : Analysis of Algorithms

Eg.2. for (i =1; i <=n; i++)→ (2n+2)

```
{
        a = b +c    → (2*n)
}
```

Total operation = (2n + 2) + 2*n

= (4n + 2) units

Loop → Initialization   i = 1 ⟹ 1 times

→ Condition $i \le n$ ⟹ n+1 times

→ Updation   i ++ ⟹ n times

Eg.

for (i=1; $i \le 3$; i++)

i = 1 ⟹ 1

$i \le 3$ ⟹ 4

i++ ⟹ 3

Topic : Analysis of Algorithms

$$\left.\begin{array}{l} \text{For } (i=1,\ i<=n,\ i++) \\ 1 \rightarrow \text{initialization} \\ (n+1) \rightarrow \text{comparison} \\ n \rightarrow \quad \text{updation} \end{array}\right\} \quad 1 + (n+1) + n = (2n+2)$$

Topic : Analysis of Algorithms

runs n time → for (j =1; j<=n; j++) → (2n+2)

{

runs n time → for (k =1; k<=n; k++) → (2n+2)

{

x = y + z ; → 2 (n^2)

}

}

Total operation

$=(2n+2) + n^*(2n+2) + n^2 * 2$

$= 2n^2 + 2n + 2n + 2 + 2n^2$

$= (4n^2 + 4n + 2)$

Topic : Analysis of Algorithms

Algo AJSir(n)

{

....

....

....

}

Total units of time taken by Algo AJSir(n):

$$= 2 + 2 + (4n+2) + (4n^2 + 4n + 2)$$
$$= 4n^2 + 8n + 8 \text{ units of time}$$

Topic : Analysis of Algorithms

Approach 2 - Order of Magnitude

→ we just need to consider the order of the no of times fundamental statement executes.

Constant → 1

Linear → n

Quadratic → n^2

Topic : Analysis of Algorithms

1. Step-count method:-

```
Algo. AJSir(n) {
  1.    p = q + r;                          → 1 or 2      → constant →1
  2.    x = y * z;                          → 1 or 2      → constant →1
  3.    for (i = 1; i <= n ; i++) {                 → n units
                a = b + c;
        }
  4.    for (j = 1; j <= n; j++) {                    → n^2 units
        for (k = 1;  k  <= n; k ++) {
                x =  y + z;
            }
        }
}
```

Topic : Analysis of Algorithms

Total units as per order of Magnitude approach

$\Rightarrow \quad 1 + n + n^2$

$\Rightarrow \quad (n^2 + n + 1)$

for same algorithm the number of units by step- count method

$\Rightarrow \quad (4n^2 + 8n + 8)$

Through Notation Both Method take : $\boxed{O(n^2) \text{ time}}$

Note.

Aim of Apriori analysis is to get/represent the running time of an algo. As a mathematical function of input size 'n'.

E.g.3.

$$\left.\begin{aligned} T(n) &= 2n^2 + 2 \\ T(n) &= 4n^2 + 8n + 8 \\ T(n) &= 5n^2 + 2 \end{aligned}\right\}$$

Order of magnitude refers to the rate of growth of time w.r.t. 'n'

Topic : Analysis of Algorithms

Rate of growth of time as a mathematical function

Logarithmic

$\Downarrow$

Form: Log(n)

E.g. $\log_5(n)$, $\log_2(n)$
log(log(n))

Polynomial

$\Downarrow$

Constant n^x

E.g. n^0, $n^{0.5}$, n^1, $n^2 \rightarrow$ Quadratic

$n^0 \rightarrow$ Constant, $n^{0.5} \rightarrow$ Sq root, $n^1 \rightarrow$ Linear

$\boxed{n^* \log(n)}$

Exponential

$\Downarrow$

a^n

E.g.

3^n, 2^n, 7^n...
n^n, n!

Note:-

Rate of growth comparison (In general)

$\frac{1}{2} > \frac{1}{3} > \frac{1}{4} > \frac{1}{5}$ ....

Usually

Decreasing < Constant < logarithmic < Polynomial < Exponential Function

$\frac{1}{n} < 10 < \log n < \sqrt{n} < n < n^2 < n^3$ .... $< 2^n < 3^n < n! < n^n$

Topic : Analysis of Algorithms

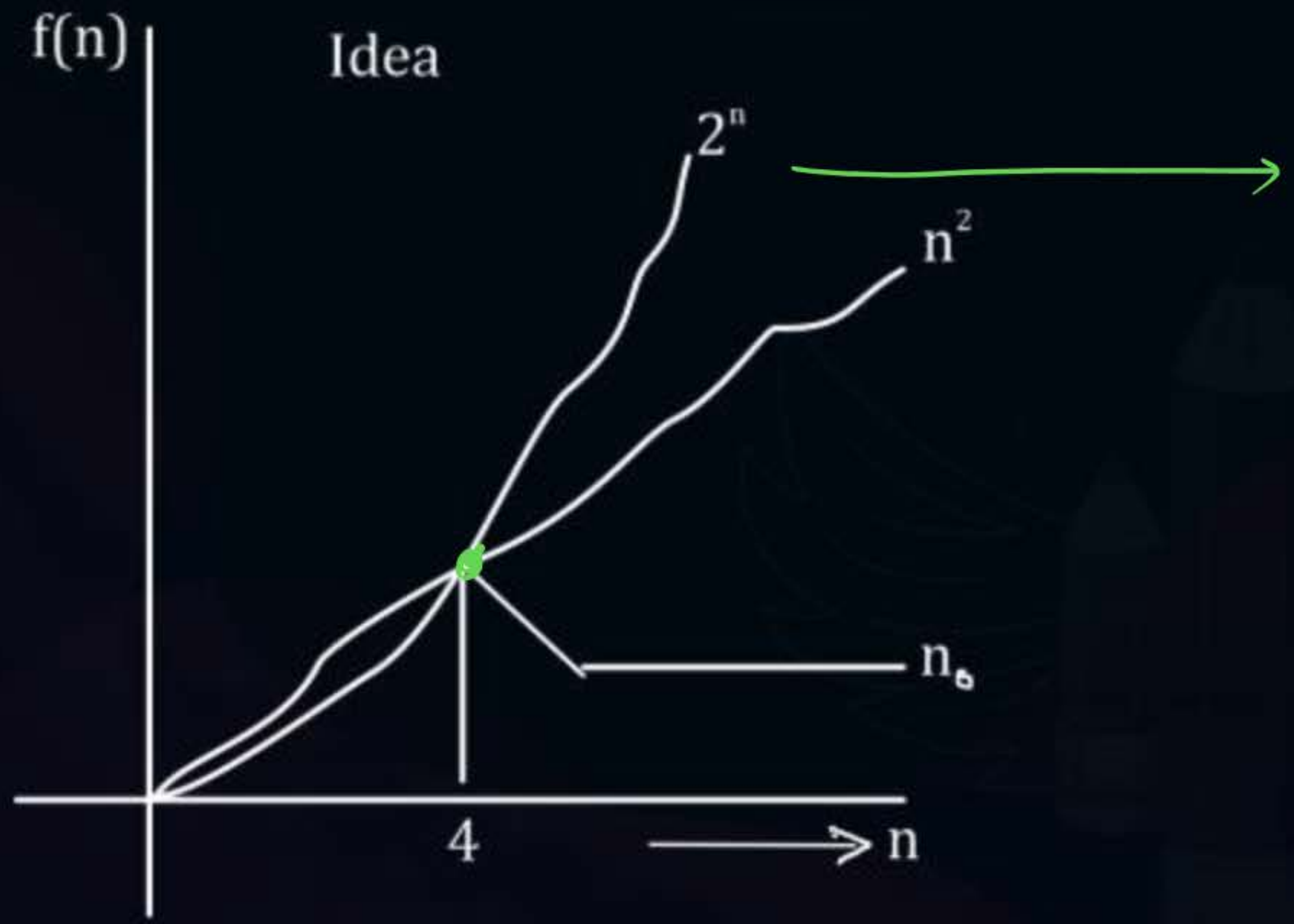
How to compare the rate of growth of two functions?

(Input) (n)	(expo) 2^n	(quadratic) n^2
1	2	1
2	4	4
3	8	9
4	16	16
5	32	25
6	64	36
7	128	49

($2^n > n^2$ for larger values of n)

Topic : Analysis of Algorithms

(Poly vs Expo)



Topic : Analysis of Algorithms

Algo→ n^2 → Lower rate of growth → takes less time → better

Algo→ 2^n → High rate of growth → takes more time

Note:-

[Algorithm that takes polynomial unit of time all are efficient that those with exponential time.]

Topic : Analysis of Algorithms

- Apriori Analysis

$n^2 \to n = 2$, $n = 5$

1. Time complexity/ Running time as a function of 'n' → (input size n grows)

2. To understand to change in nature or running time for a given algo and a fixed input size 'n' but for different input classes. → (test cases)

$n = 5$

Best case (BC)

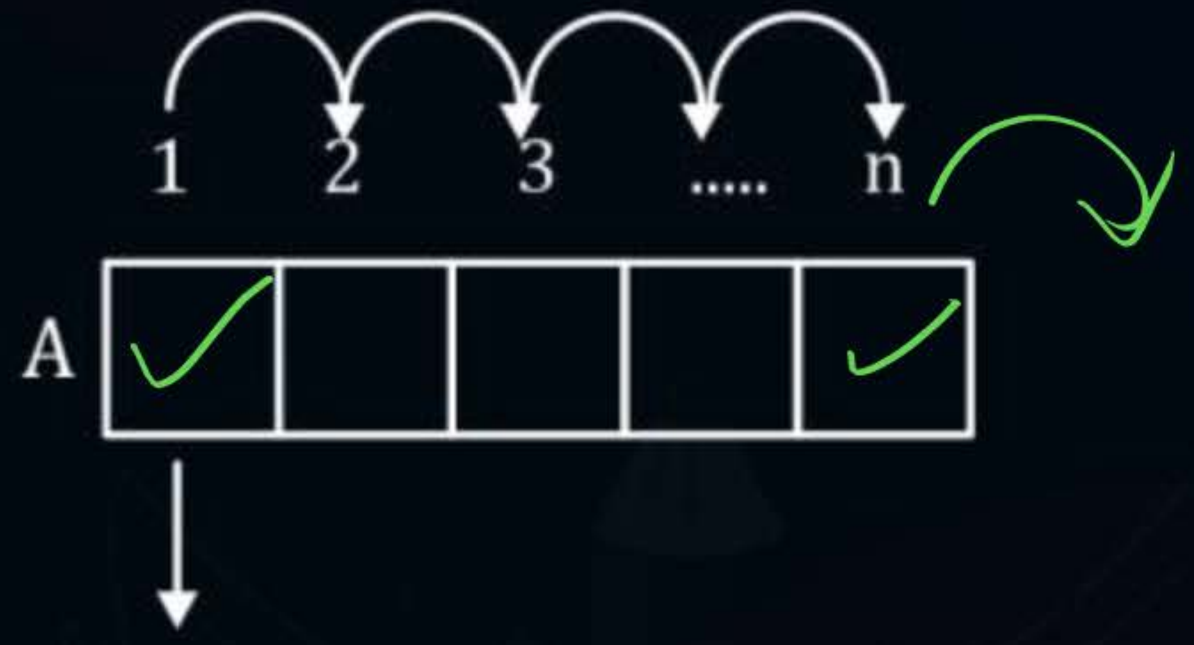
Average case (AC)

Worst case (WC)

Topic : Analysis of Algorithms

Example to understand point (2):-

```
Algorithm linear search (A, n, key)
{
    for (i = 1, i ≤ n, i++)
        {
            if (A[i] == key)
                return i;
        }
    Printf("element not found");
}
```



$\boxed{n=5}$

Topic : Analysis of Algorithms

Eg.5. n = 5 (fixed)

Case1:- input case-1

A :

2	5	10	7	9

key = 2

Loop runs how many times?

$\Rightarrow$ 1 time $\rightarrow$ Constant

Best case TC $\Rightarrow$ TC $\Rightarrow$ O(1)

Case2:-

Key is not present or is present at last position

Input case-2

A :

3	9	15	7	5

key = 4 5

$\rightarrow$ Loop runs how many times?

$\Rightarrow$ 5 time $\rightarrow$(n times)

Linear Rate of growth

Worst case TC of $\Rightarrow$ TC= O(n) linear search

Topic : Analysis of Algorithms

- Different cases for time complexity for a fixed input size.

1. Best case → The input for which the Algo. runs min no. of times (does min work.)

   The time complexity for such input ⇒ Best case TC

2. Worst case→ The input which of the Algorithm run max no. of times

   The time complexity for such input ⇒ worst case TC

3. Average case→ Probability dependent.

Ex. Linear search

$\boxed{\text{BC} \quad \text{TC} \rightarrow O(1)}$ $\boxed{\text{WC} \quad \text{TC} \rightarrow O(n)}$ $\boxed{\text{AC} \quad \text{TC} \rightarrow O(n)}$

Topic : Analysis of Algorithms

V. imp

Best case TC $\rightarrow$ B(n)
Worst case TC $\rightarrow$ W(n)
Avg. case TC $\rightarrow$ A(n)

- For any given algo, in general:

Always: $\boxed{B(n) \leq A(n) \leq W(n)}$

Examples:

1. $B(n) < [A(n) = W(n)] \Rightarrow$ Linear search, Binary search
2. $[B(n) = A(n)] < W(n) \Rightarrow$ Quick sort
3. $[B(n) = A(n) = W(n)] \Rightarrow$ Merge sort, Heap sort, Selection sort.

Topic : Asymptotic Notations

Asymptotic Notations:

- The bounds/range of the function that are represented using these notations.

Topic : Asymptotic Notations

Types of Bounds:

(1) Upper Bound

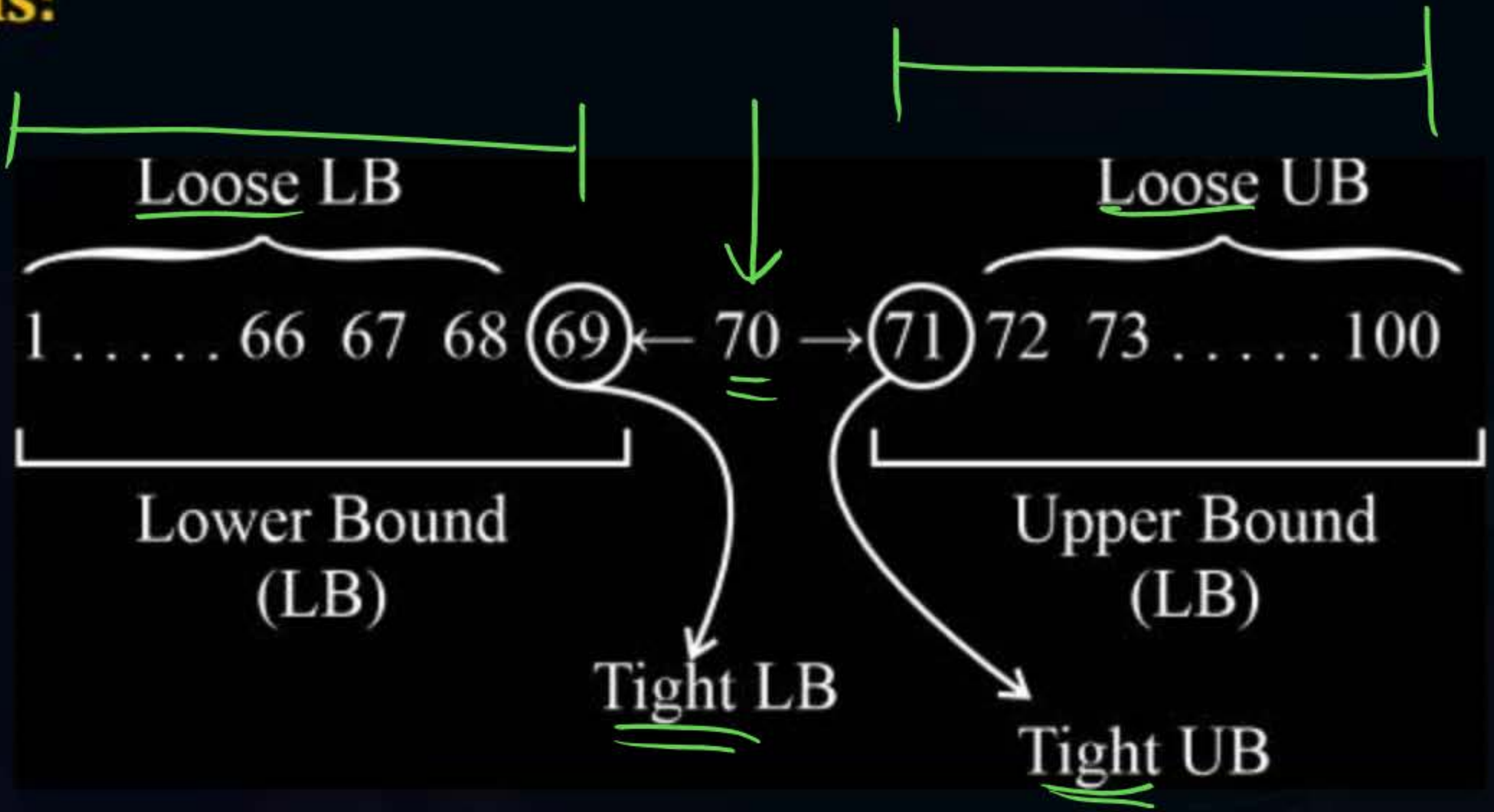
(2) Lower Bound

(i) Lose Bound

(ii) Tight Bound

Topic : Asymptotic Notations

Types of Bounds:



Topic : Asymptotic Notations

Types of Asymptotic Notations: (V. imp)

Asymptotic Notations

(T+L) (1) Big Notations

(i) UB : Big Oh → O

(ii) LB : Big Omega → Ω

(2) Small Notations (L)

(iii) UB : Small oh → o

(iv) LB : Small omega → ω

(v) Theta : θ → Tight Bound

(T)

$\boxed{O, o, \Omega, \omega, \theta}$

Topic : Asymptotic Notations

Let 'f' and 'g' be functions from the set g integers/real to real number;

1. **Big-Oh(O):** Upper Bound (UB)

- $f(n) = O(g(n))$ if there exists some constant $c > 0$ and $n_0 \geq 0$ such that $\boxed{f(n) \leq c^*(g(n))}$, whenever $n \geq n_0$.

Topic : Asymptotic Notations

Example:

(1) Order of Magnitude

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

$1 + n \leq n^2 + n^2$

$n = 2 \quad 1 + 2 \leq 2^2 + 2^2$

$n = 3 \quad 1 + 3 \leq 3^2 + 3^2$

$1 \leq n^2$

$1 + n \leq n^2 + n^2$

$1 + n + n^2 \leq n^2 + n^2 + n^2$

$\boxed{1 + n + n^2 \leq 3n^2}$

$f(n) \leq c * g(n)$

Hence,

$f(n) = O(g(n))$, $c > 3$, $n \geq n_0$, $n_0 \geq 1$

$1 + n + n^2 = O(n^2)$

- Whenever we determine the upper bound and lower bound, we should find that function 'g' which is closest to the given function.

Example:

$$\text{Nagpur} \xrightarrow[\text{(Non-stop)}]{\text{Flight}} \text{Delhi}$$

$$\left.\begin{array}{ll} \text{Loose Upper bound} & \left[\begin{array}{l} P1 \rightarrow < 1 \text{ year} \\ P2 \rightarrow < 1 \text{ week} \end{array}\right. \\ \text{Tight UB} & \left[\ P3 \rightarrow < 5 \text{ hr}\right. \end{array}\right] \text{UB}$$

Topic : Asymptotic Notations

Example2: Purchasing a Car

{ 5 seater
AC
Automatic }

F1 : > 50 Rs
F2 : > 5000 Rs
→ Loose LB

Tight LB ← F3 : > 5L Rs

Topic : Asymptotic Notations

Shortcut:

Dominating term → Highest term with rate of growth with increasing value of n

Example:

(i) $f(n) = n^2 + n + 1$

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

(ii) $f(n) = 5n^3 + 8n + 7$

$f(n) = O(5n^3) = O(n^3)$

Topic : Asymptotic Notations

(1) Step-count method

$$T(n) = 4n^2 + 8n + 8$$

$$T(n) = O(4n^2)$$

$$T(n) = O(n^2)$$

(2) Order of magnitude

$$T(n) = n^2 + n + 1$$

$$T(n) = \underline{O(n^2)}$$

$\rightarrow$ 2028 $\longrightarrow$ 20K

2030 $\longrightarrow$ 70K

2035 $\longrightarrow$ 3L

2040 $\longrightarrow$ $\boxed{10\ Cr}$

Topic : Asymptotic Notations

H.W

2. **Big-Omega(Ω):** Lower Bound (LB)

- $f(n) = \Omega(g(n))$ if there exists some constant 'c' and 'n_0' such that $f(n) \geq c^*(g(n))$, whenever $c > 0$, $n \geq n_0$, $n_0 \geq 0$

Topic : Asymptotic Notations

H.W

Example: $8n^2 + 3n + 5$

$f(n) = 1 + n + n^2 \geq 1 * n^2 \rightarrow \Omega(n^2)$

$f(n) = 1 + n + n^2 \geq 1 * n \rightarrow \Omega(n)$

$f(n) = n + n + n^2 \geq 1 * \sqrt{n} \rightarrow \Omega(\sqrt{n})$

$f(n) = n + n + n^2 \geq 1 * 1 \rightarrow \Omega(1)$

$1 + n + n^2 \geq 1 * n^2$

$f(n) \geq c^*(g(n))$

$f(n) = \Omega(g(n)) = \Omega(n^2)$

Hence,

$1 + n + n^2 = \Omega(n^2)$

2 min Summary

Topic

Topic

Asymtotic Notations

Topic

Topic

THANK - YOU