

CS & IT ENGINEERING

Algorithms

Analysis of Algorithm

Lecture No. - 03

By- Dr. Khaleel Khan
Sir

Recap of Previous Lecture



Topic

Need for Analysis

Topic

Methodology of Analysis

Topic

Aposteriori Analysis

Topic

Apriori Analysis

Topic

Resource Consumption
Perfor. Comparison

Aposteriori
Apriori Analysis

Topics to be Covered



Topics

Types of Analysis

Asymptotic Notations





Topic: Analysis of Algorithms



Step-Count Method

Algorithm Test
{

1

1. $x \leftarrow y + z; \rightarrow 2$

n

2. for $i \leftarrow 1$ to $n \rightarrow 1 + (n+1) + n$
 {
 $a \leftarrow b + c;$
 }

n^2

3. for $i \leftarrow 1$ to $n \rightarrow 1 + (n+1) + n$
 for $j \leftarrow 1$ to $n \rightarrow n + n(n+1) + n \cdot n$

$k \leftarrow k * j; \rightarrow \underline{n \cdot n} + \underline{n \cdot n}$

Time: $\underline{\underline{1 + n + n^2}}$

$$\text{Time} = 2 + (4n + 2) + (4n^2 + 4n + 2)$$

$$\boxed{\text{Time} = (4n^2 + 8n + 6)}$$

1) $a = b + c + d + e + f : 1$



Topic: Analysis of Algorithms

To determine the Time of Algo under Apriori Analysis,

↓
(Order of Magnitude)

→ Order of Magnitude of a Statement/Step of the Algo. refers to the Frequency/Count of the fundamental operation in the Stmt./Step;



Topic: Analysis of Algorithms

Algorithm Sum(a, b, c)

{
 integer a, b, c;

1. Read(a, b); 1

Time = C
= O(1)

2. if (a < b) 1
 {
 c = a + b; 1
 }
 else
 {
 c = a * b; 1
 }

3. print(c); 1

1+1

$$\text{Time } 1 + 2 + 1 = 4 = C = O(1)$$

Time =
(n+1)
O(n)

Algorithm Do-it(A, n)

{
 integer n, A[n];

integer i, Sum = 0; X

Sum = 0;
1. for i ← 1 to n : n
 {

 Sum = Sum + A[i];

 }
2. print(Sum); 1
}



Topic: Analysis of Algorithms

The basic objective of Apriori Analysis is to represent (obtain) the Time Complexity (Running Time) of the Algorithm by means of a Mathematical function w.r.to input size (say 'n');

$$1) T(n) = 1 + n + n^2 \quad (4n^2 + 8n + 6)$$

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

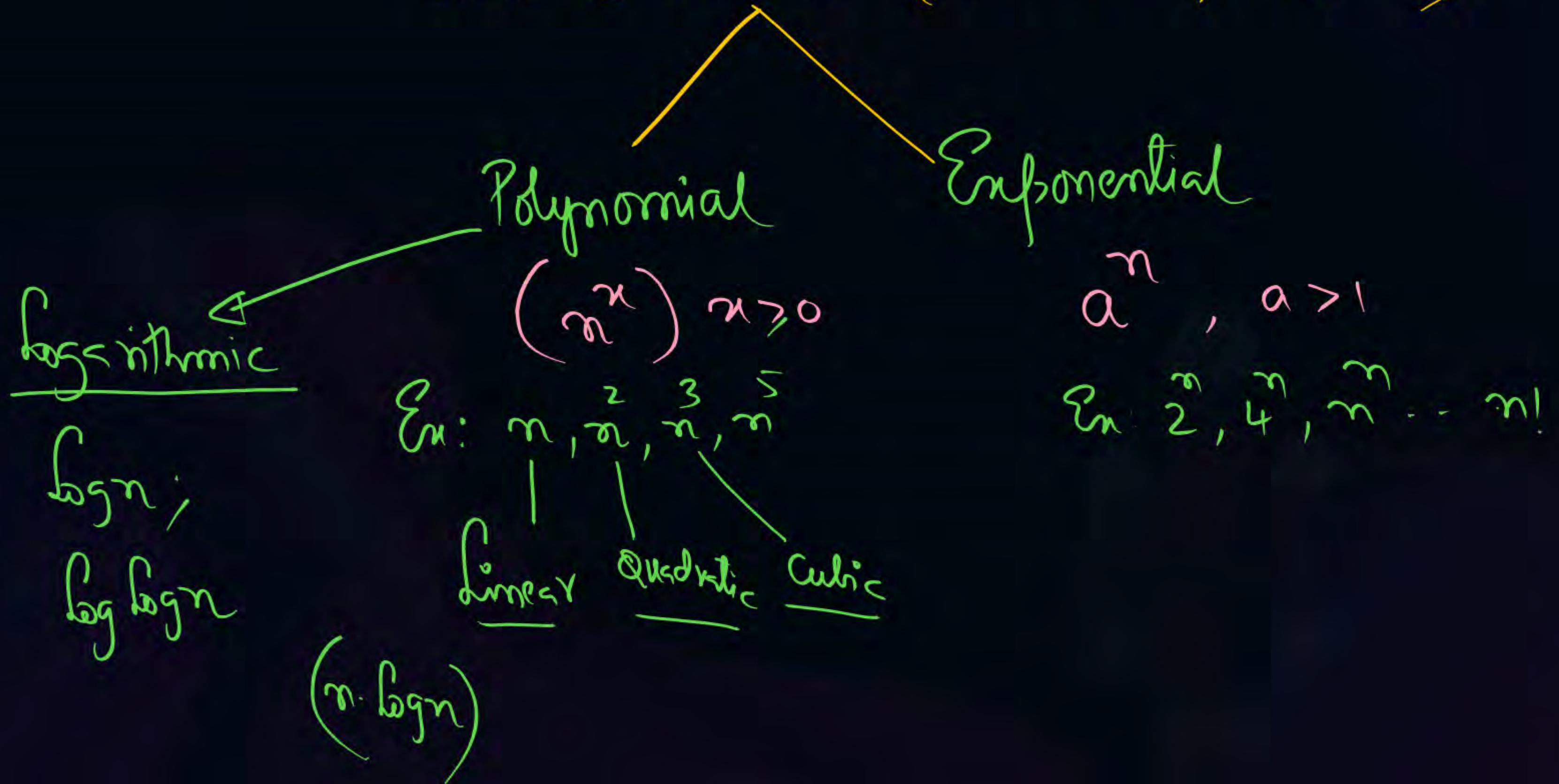
$$3) T(n) = 4 (c)$$

(Rate of Growth of Time)



Topic: Analysis of Algorithms

Time \propto function (w.r.to Input Size 'n')





Topic: Analysis of Algorithms



n	$T_1 A_1$ n^2	$T_2 A_2$ 2^n
2	4	4
3	9	8
4	16	16
5	(25)	(32) units
6	(36)	(64)
7	(49)	(128)

→ Exponential functions have higher rates of growth (Takes more time)

→ Polynomial Functions have lesser rate of growth (Take less time)



Topic: Analysis of Algorithms



The objective is always to develop Algo's for Problems,
having Polynomial Time Complexity [Efficient]
(Complexity - Theory)



Topic: Analysis of Algorithms



Apriori Analysis

To determine the running time w.r.to increasing input size n

To observe the behaviour of the Algorithm for a fixed input of size 'n';

(Behaviour | Types of Analysis)



Topic: Analysis of Algorithms



I_1 I_2 I_3 \dots I_k
Inc dec random \dots

Algorithm $LS(A, n, x)$
integer $n, A[n]$

Time & Comparison

1) Best Case:

$$1 + 1 + 1 = C \\ O(1)$$

2) Worst Case

$$: n \\ O(n)$$

```
int i;
for i ← 1 to n
{
    if ( $x = A[i]$ )
    {
        print(i);
        exit;
    }
}
```

print("Elem. not found");

Time Complexity

~ Worst Case

✓ 1) Worst Case: The IP class for which Algo. takes max Time, is W.C input & Corresp. Time is W.C. Time

✓ 2) Best Case: The IP class for which the algo takes Min. Time is B.C input & B.C Time,

3) Average Case: is derived in 3-Steps

(i) Enumerate all IP classes
(I_1, I_2, \dots, I_k)

(ii) determine the Time for (t_1, t_2, \dots, t_k)
each IP-class

(iii) Assoc. with each input class the Prob. fn
(P_i)

$$A(n) = \sum_{i=1}^k P_i * t_i$$

1) Linear Search $A(n)$

✓ (i) Best Case : 1 : $O(1)$ ✓

✓ (ii) Worst Case : n : $O(n)$ ✓

(iii) Average Case : (for a successful linear search)

x

No. of Comp's :

$$(1 + 2 + 3 + \dots + n)$$

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

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

- 1) Best Case Time: $B(n)$
- 2) Worst Case Time: $w(n)$
- 3) Avg. Case Time: $A(n)$

$$B(n) < A(n) < w(n)$$

H/w

$$B(n) \leq A(n) \leq w(n) \quad - \textcircled{1}$$

- 1) $B(n) = A(n) = w(n)$: Mergesort; Sel-sort; H-S
- 2) $B(n) < (A(n) = w(n))$: Linear Search; B.S
- 3) $(B(n) = A(n)) < w(n)$: Quicksort



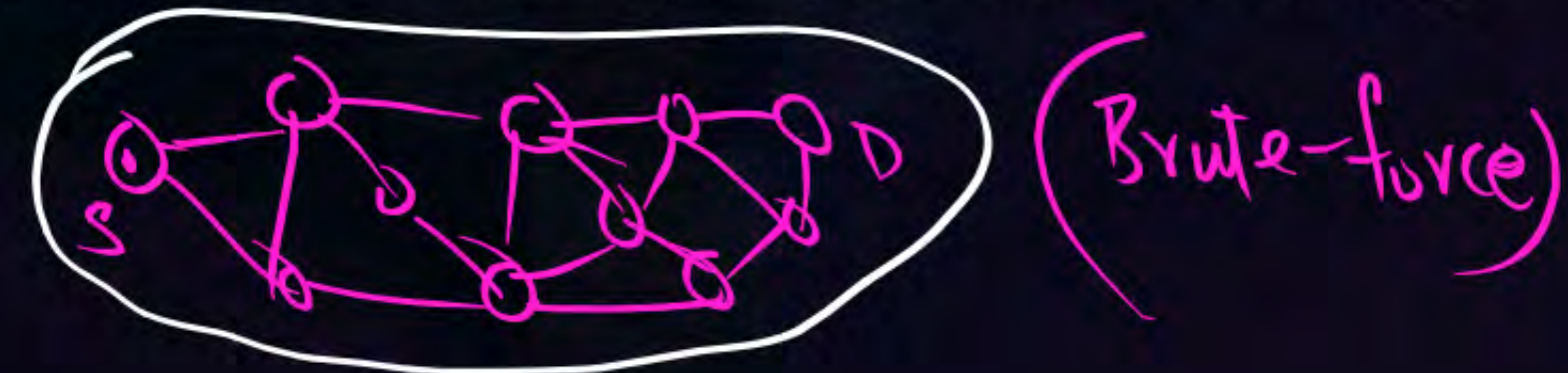
Topic: Analysis of Algorithms

Analyzing algorithms involves thinking about how their resource requirements-the amount of time and space they use-will scale with increasing input size.

Proposed Definition of Efficiency (1): An algorithm is efficient if, when implemented, it runs quickly on real input instances.

Proposed Definition of Efficiency (2): An algorithm is efficient if it achieves qualitatively better worst-case performance, at an analytical level, than brute-force search.

✓ **Proposed Definition of Efficiency (3):** An algorithm is efficient if it has a polynomial running time





Topic: Analysis of Algorithms

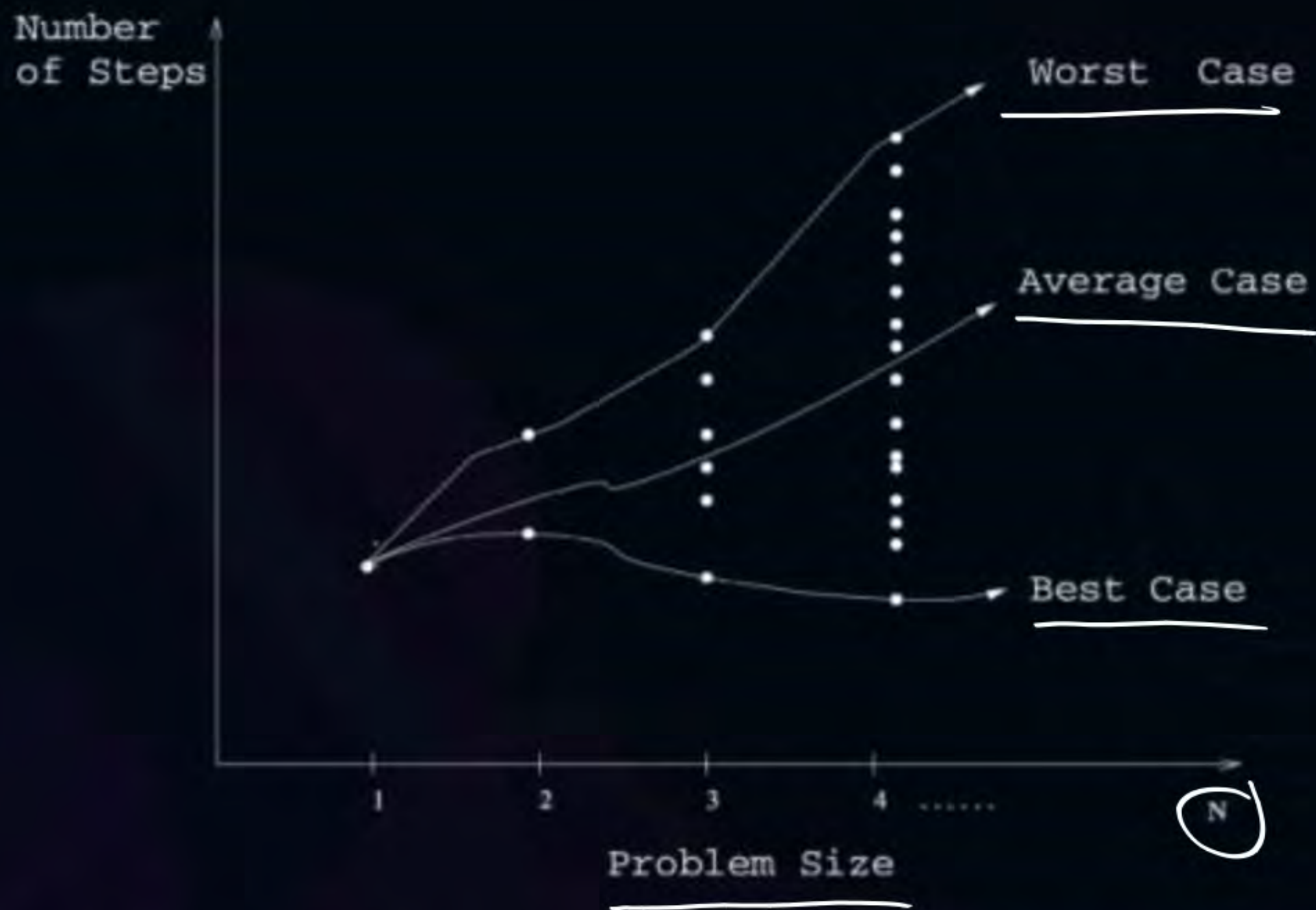


Figure 2.1: Best, worst, and average-case complexity



Topic: Analysis of Algorithms

- **The Worst-case Complexity of the Algorithm** is the function defined by the maximum number of steps taken in any instance of size n . This represents the curve passing through the highest point in each column.
- **The Best-case Complexity of the Algorithm** is the function defined by the minimum number of steps taken in any instance of size n . This represents the curve passing through the lowest point of each column.
- **The Average-case complexity of the Algorithm**, which is the function defined by the average number of steps over all instances of size n .

THANK - YOU