CS & IT ENGINEERING

Algorithms

Introduction to Algorithms and Analysis

Lecture No.- 03

By- Dr. Khaleel Khan Sir

Recap of Previous Lecture

Topic Need for Analysis

- Resource consumption
- Perfor. Comparison

Topic Methodology of Analysis

- Aposteriori
- Apriori Analysis

Topic Aposteriori Analysis

Topic Apriori Analysis

Topic

Topics to be Covered

Topics

- Types of Analysis
- Asymptotic Notations

Step-Count Method

Algorithm Test
{

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

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

n^2 3. for $i \leftarrow 1$ to n $\longrightarrow 1+(n+1)+n$
for $j \leftarrow 1$ to n $\longrightarrow n+n(n+1)+n \cdot n$
$K \leftarrow K * z;$ $\longrightarrow n \cdot n + n \cdot n$

Time: $(1+n+n^2)$

}

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

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

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

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;

```
Algorithm Sum(a, b, c)
{
    integer a, b, c;
  1. Read(a, b);          1
  2. if (a < b)           1
     {
        c = a+b;          1
     }
     else
     {
        c = a*b;          1
     }
  3. print(c);            1
}
```

Time = C
= O(1)

1 + 1

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
}
```

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

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

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

(Rate of Growth of Time)

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

Polynomial

(n^x) $x \geq 0$

Ex: n, n^2, n^3, n^5

Linear, Quadratic, Cubic

Logarithmic

$\log n$;

$\log \log n$

$(n \cdot \log n)$

Exponential

a^n, $a > 1$

Ex: $2^n, 4^n, n^n \ldots n!$

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)

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

(Complexity - Theory)

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

$\langle a_1, a_2 \ldots a_n \rangle$

I_1 I_2 I_3 ... I_k

Inc dec random ...

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) <u>Worst Case</u> : The IP Class for which Algo. takes max Time is W.C input & Corresp. Time is W.C. Time

2) <u>Best Case</u> : The Ip Class for which the algo takes Min. Time is B.C input & B.C Time.

3) <u>Average Case</u> : is derived in 3-Steps

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

(ii) determine the Time for each IP-Class $(t_1, t_2 \cdots t_k)$

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

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

1) Linear Search $A(n)$

(i) <u>Best Case</u> : 1 : $O(1)$ ✓

(ii) <u>Worst Case</u> : n : $O(n)$ ✓

(iii) <u>Average Case</u> : (for a successful linear search)

×

<u>No. of Comp's</u> :

$$\frac{(1+2+3+\cdots+n)}{n} : \frac{\left(\frac{n(n+1)}{2}\right)}{n} = \frac{(1+n)}{2} = O(n)$$

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

$$\boxed{B(n) < A(n) < W(n)}$$ H/W ✓

$$\boxed{B(n) \leq A(n) \leq W(n)} \quad \text{--- (1)}$$

1) $\boxed{B(n) = A(n) = W(n)}$: Mergesort ; Sel. Sort ; H.S

2) $\boxed{B(n) < (A(n) = W(n))}$: Linear Search ; B.S

3) $\boxed{(B(n) = A(n)) < W(n)}$: Quicksort

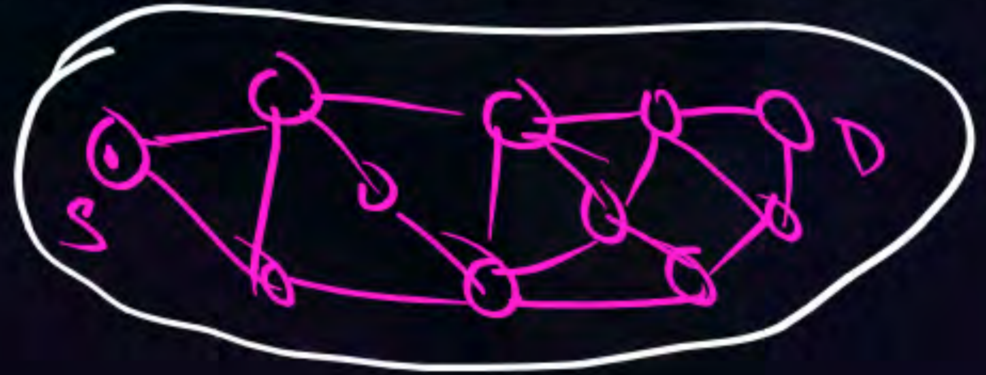
Topic: Analysis of Algorithms

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



(Brute-force)

Topic: Analysis of Algorithms

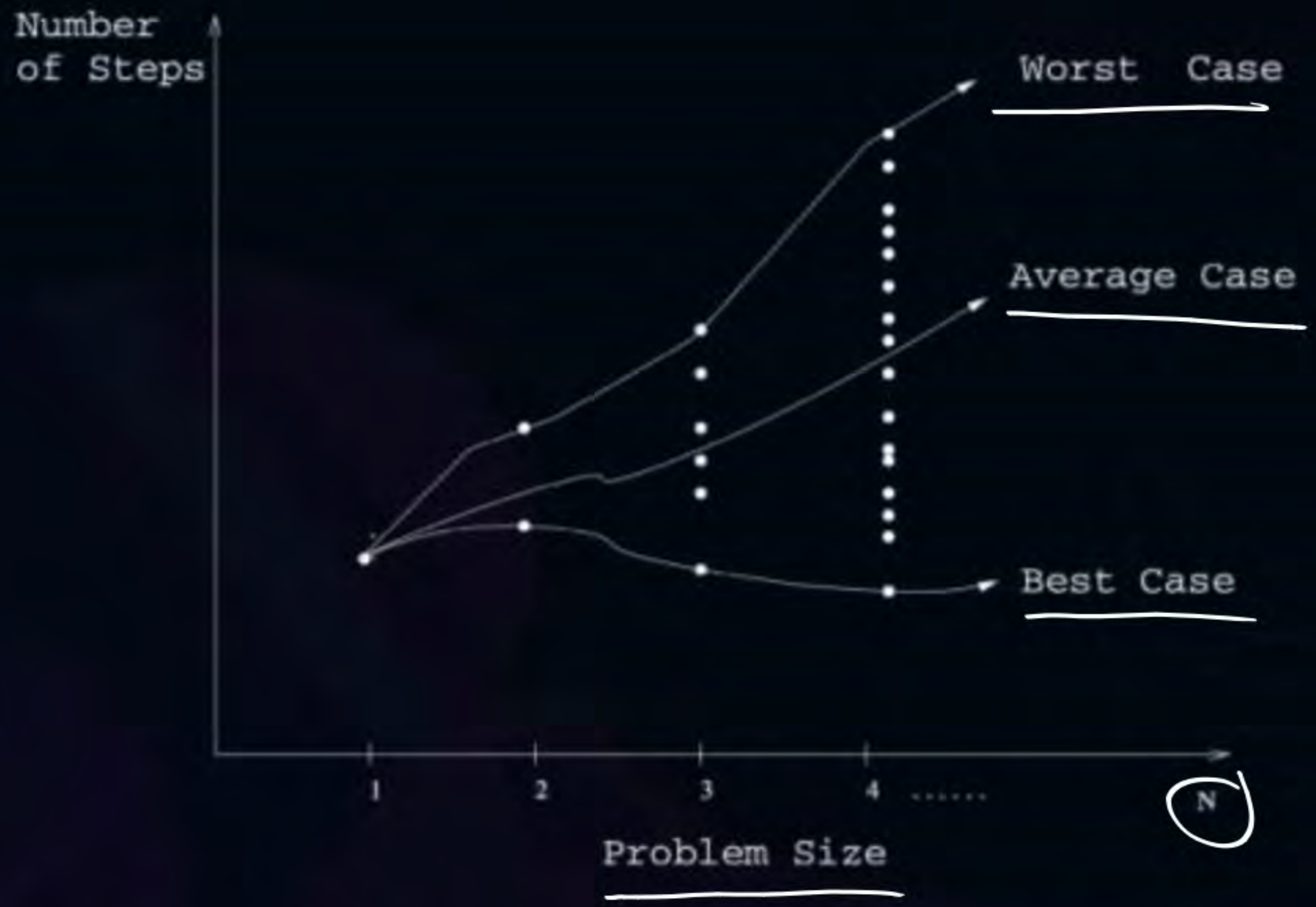


Figure 2.1: Best, worst, and average-case complexity

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