Analysis of Algorithms

Module 1 - Introduction to Algorithms

* Algorithm as a Technology

→ An algorithm is simply a set of steps used to complete a specific tasks.

→ If computer memory would be infinitely fast and free, studying different solutions to a problem wouldnt be a necessity.

→ But memory is cheap and not free so finding different solutions of a problem and choosing the cost-efficient method is a must.

* Algorithm Design.

→ The important aspects of algorithm design include.

① creating an efficient algorithm.
② solving the problem using efficient time and space.

→ Both time and space cannot be minimized simultaneously, if one is more other is less.

* Problem Development

→ some common steps toward problem development

① Analyzing the Problem Definition.
② Development of a model.
③ Specification of Algorithm
④ Designing an Algorithm
⑤ Checking correctness of the algorithm.
⑥ Analysis of the Algorithm.
⑦ Implementation of the Algorithm.
⑧ Program Testing.

* Characteristics of Algorithms

① Algorithms must have a unique name

② Algorithms should have explicitly defined set of inputs and outputs.

③ Algorithms are well-ordered unambiguous operations.

④ Algorithms halt after a finite amount of time.

* Pseudocode.

→ A pseudocode gives high-level description of of an algorithm without plain text and without need of actual syntax.

* Difference in Algorithm, Pseudocode and Program.

① Start from the leftmost element of arr[] and one by one compare each element of arr[]

② If x matches the index, return the index.

③ If x doesnt match with any of the elements return -1

④

→ Pseudocode.

```
function LinearSearch( list, searchItem)
    for index from 0 ⟶ length(list)
        If list[index] == SearchItem
            return index.
        Else
            pass;
    Return -1.
End function.
```

→ Program

```
int LinearSearch (int arr[], int index)
{
    for (int i = 0; i<size; i++)
    {
        if (arr[i] == arr[index])
        {
            return i;
        }
        else
        {
            pass;
        } return -1;
    }
```

* Rate of Growth of Functions.

→ The rate at which running time increases as a function of input is called Rate of Growth.

→ Example - Total cost = cost of Car + cost of Candy

∴ Total cost = cost of Car (By approximation)

* Common Running times

	Time complexity	Name	Example
①	1	Constant	Adding element in LL
②	logn	Logarithmic	finding element in a sorted array
③	n	Linear	Linear search
④	nlogn	Linear Log.	Quick sort / Merge sort
⑤	n^2	Quadratic	Shortest path b/w two nodes.
⑥	n^3	Cubic	Matrix Multiplication
⑦	2^n	Exponential	Towers of Hanoi

---

* Asymptotic Notations and Running Time

→ The order of growth of running time is called Asymptotic Running Time.

Some Notations are:-

Page No.
Date

* Calculating Time complexity of any Algorithm

① Loops. → Example.

```
for (int i=0 ; i<=n; i++) // Running n times
{
    x = y+2; { Suppose this step takes
}                'c' time }
```

$\therefore T_n = c * n$

$\therefore \boxed{T_n = O(n)}$

② Nested Loop.

```
for (int i=0; i<n ; i++)  // Running n times
  { for(int j=0; j<n; j++) // Running n times.
    {
        x = y+2; { Takes 'c' time }
    }

}
```

$\therefore T_n = c * n * n$

$T_n = c * n^2$

$\therefore T_n = O(n^2)$

(3) Sequential Statement

i) a = a+b → O(1) time {constant}

ii)
```
for(int i=0 ; i<=n ; i++)
{
    x = y+2;
}
```

(4) If - else statements.

```
if (condition)
{
    ..... O(n)
}
else                                  → O(n²)
{
    ..... O(n²)
}
```

Priority Order:

According to time taken

Q. $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(e^n) < O(n!)$

Algorithm.	Best	Worst	Avg.
① Selection Sort	$\Omega(n^2)$	$O(n^2)$	$\Theta(n^2)$
② Bubble Sort	$\Omega(n)$	$O(n^2)$	$\Theta(n^2)$
③ Insertion Sort	$\Omega(n)$	$O(n^2)$	$\Theta(n^2)$
④ Heap Sort	$\Omega(n \log n)$	$O(n \log n)$	$\Theta(n \log n)$
⑤ Quick Sort	$\Omega(n \log n)$	$O(n \log n)$	$\Theta(n^2)$
⑥ Merge Sort	$\Omega(n \log n)$	$O(n \log n)$	$\Theta(n \log n)$
⑦ Bucket sort	$\Omega(n+k)$	$O(n+k)$	$\Theta(n^2)$
⑧ Radix sort	$\Omega(nk)$	$O(nk)$	$\Theta(nk)$