Sunbeam Institute of Information Technology

Pune and Karad

PG-DESD

Module – Data Structures

Trainer - Devendra Dhande

Email – devendra.dhande@sunbeaminfo.com

Data Structures - Introduction

What is Data structure?

- Organising data into memory
- Processing the data efficiently

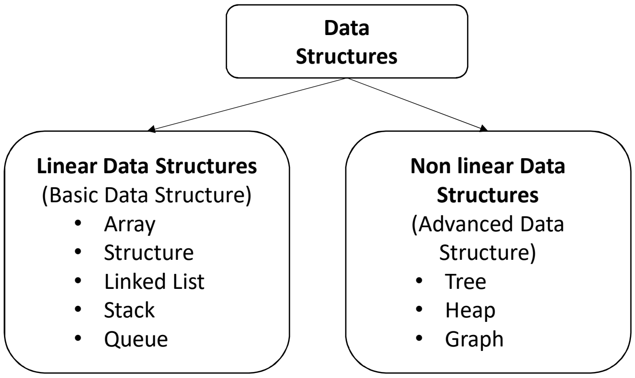
Why we need Data Structure?

To achieve

1. Efficiency
2. Reusability

Programming Language

- DS and Algorithms are language independent
- We will use **C programming** to implement Data structures



Data Structures

Linear Data Structures (Basic Data Structure)

- Array
- Structure
- Linked List
- Stack
- Queue

Non linear Data Structures (Advanced Data Structure)

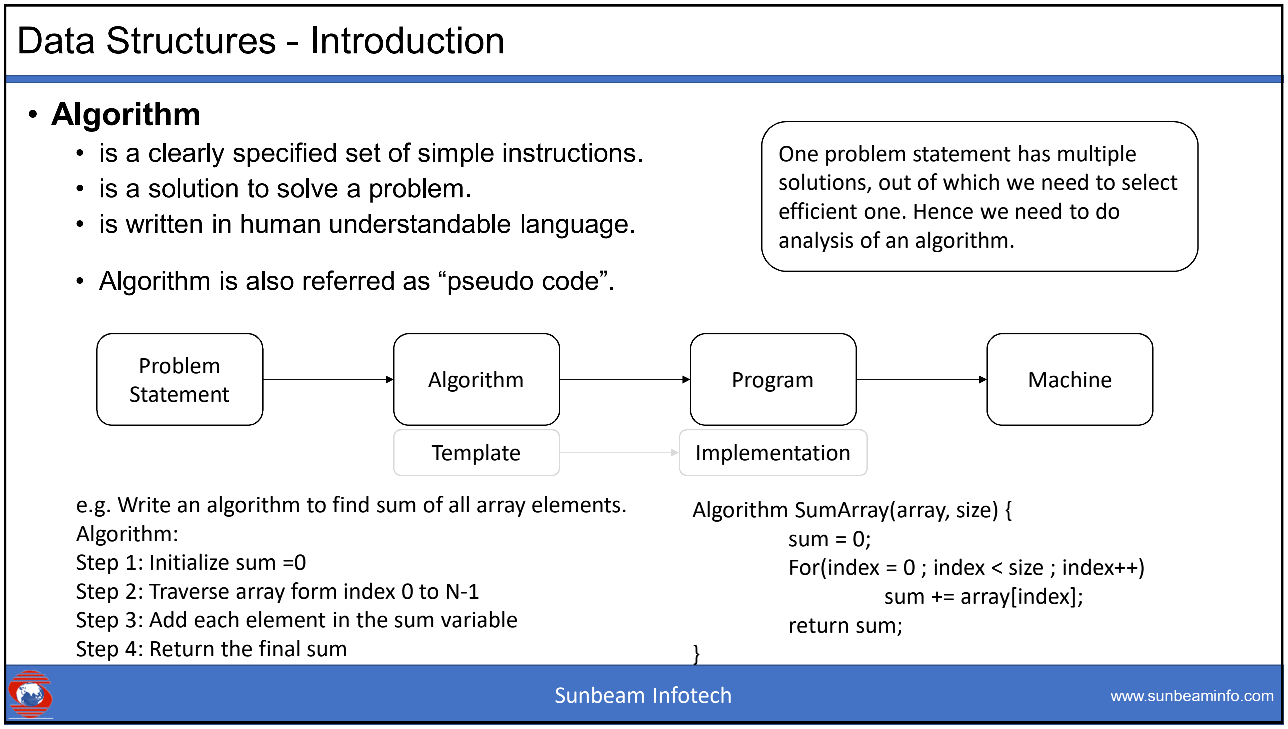
- Tree
- Heap
- Graph

Linear Data Structures

- Data elements are arranged linearly (sequentially) into the memory.
- Data elements can be accessed linearly / Sequentially.

Non linear Data Structures

- Data elements are arranged in non linear manner (hierarchical) into the memory.
- Data elements can be accessed non linearly.



Data Structures - Introduction

- **Algorithm**
  - is a clearly specified set of simple instructions.
  - is a solution to solve a problem.
  - is written in human understandable language.
  - Algorithm is also referred as “pseudo code”.

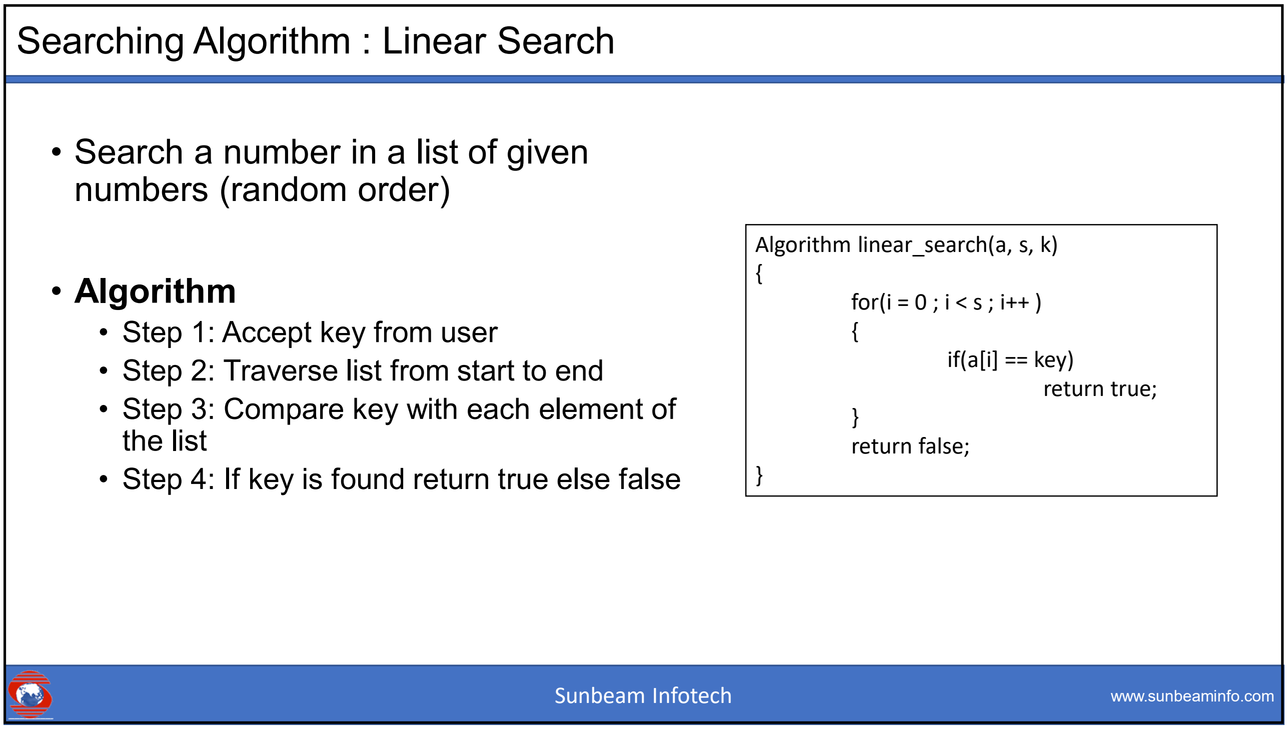
One problem statement has multiple solutions, out of which we need to select efficient one. Hence we need to do analysis of an algorithm.

Problem Statement
Algorithm
Program
Machine
Template
Implementation

e.g. Write an algorithm to find sum of all array elements.

```
Algorithm:
Step 1: Initialize sum =0
Step 2: Traverse array form index 0 to N-1
Step 3: Add each element in the sum variable
Step 4: Return the final sum
```

```
Algorithm SumArray(array, size) {
        sum = 0;
        For(index = 0 ; index < size ; index++)
                sum += array[index];
        return sum;
}
```



Searching Algorithm : Linear Search

- Search a number in a list of given numbers (random order)

- **Algorithm**
  - Step 1: Accept key from user
  - Step 2: Traverse list from start to end
  - Step 3: Compare key with each element of the list
  - Step 4: If key is found return true else false

```
Algorithm linear_search(a, s, k)
{
        for(i = 0 ; i < s ; i++ )
        {
                if(a[i] == key)
                        return true;
        }
        return false;
}
```

Searching Algorithm : Binary Search

- Given an integer x and integers A0, A1, …An-1, which are pre-sorted and already in memory, find i such that Ai = x or return i = -1 if x is not in the input

- **Algorithm**
  - Step 1: Accept key from user
  - Step 2: Check if x is the middle element. If so x is found at mid
  - Step 3: If x is smaller than the middle element, apply same strategy to the sorted subarray to the left of middle element.
  - Step 4: If x is larger than the middle element, apply same strategy to the sorted subarray to the right of middle element

Sorting Algorithm : Selection Sort

Algorithm:

- Find the minimum element in an array A[i -> n-1] and place it at beginning
  - where n – size of array and i – 0, 1, 2, …n-2
- Repeat the above procedure n – 1 times where n is size of array

- Select ith element (i = 0 -> n-1)
  - Compare with all elements other than ith
    - if(A[i] > A[other])
      - Swap both elements

arr	44	11	55	22	66	33
	0	1	2	3	4	5

Algorithm Analysis

- Analysis is done to determine how much resources it require.
- Resources such as time or space
- There are two measures of doing analysis of any algorithm
  - Space Complexity
    - Unit space to store the data into the memory (Input space) and additional space to process the data (Auxiliary space)

    e.g. Algorithm to find sum of all array elements.

    int arr[n] – n units of input space
    sum, index, size – 3 units of auxiliary space
    Total space required = input space + auxiliary space = n + 3 = n units

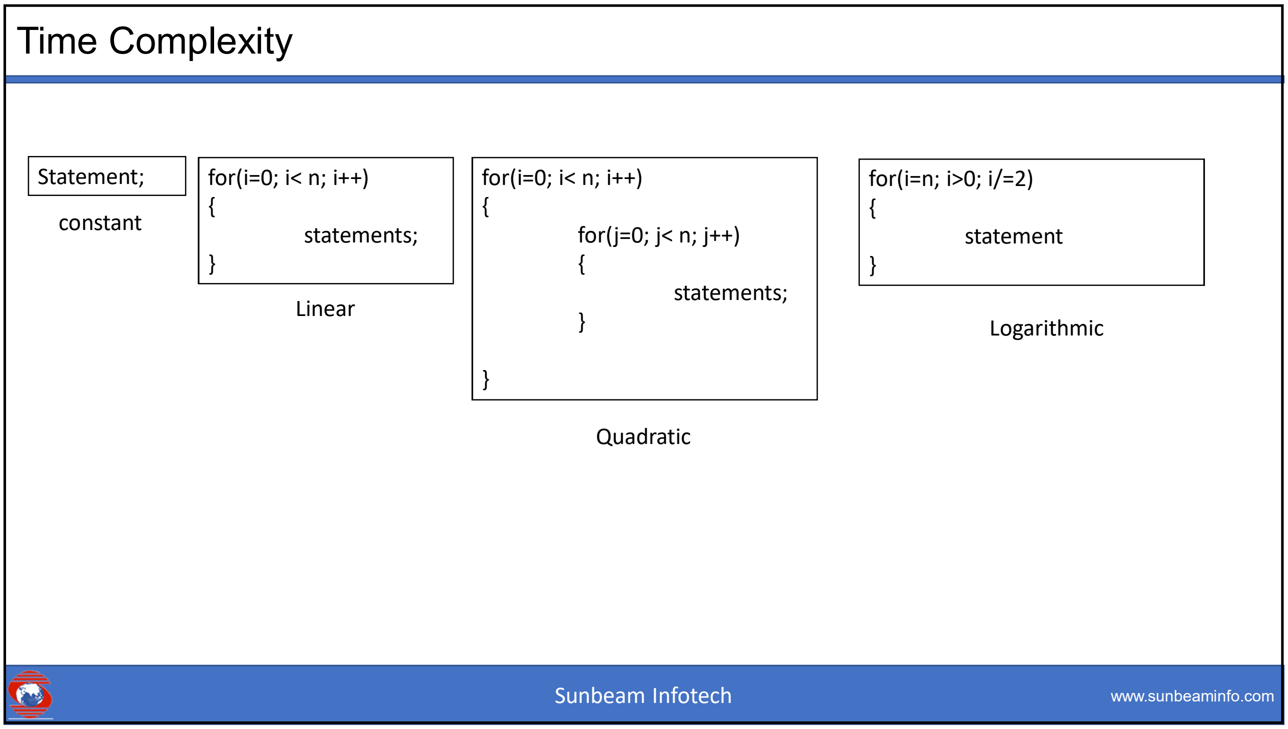
  - Time Complexity
    - Unit time required to complete any algorithm
    - Approximate measure of time required to complete algorithm
    - Depends on loops in the algorithm
    - Also depends on some external factors like type of machine, no of processed running on machine.
    - That's why we can not find exact time complexity.
  - Method used to calculate complexities, is "**Asymptotic Analysis**"

Asymptotic Analysis

- It is a mathematical way to calculate complexities of an algorithm.
- It is a study of change in performance of the algorithm, with the change in the order of inputs.
- It is not exact analysis
- Few mathematical notations are used to denote complexities.
- These notations are called as "Asymptotic notations" and are
  - Omega notation (Ω)
    - Represents lower bound of the running algorithm
    - It is used to indicate the best case complexity of an algorithm

  - Big – Oh notation (O)
    - Represents upper bound of the running algorithm
    - It is used to indicate the worst case complexity of an algorithm

  - Theta notation (ϴ)
    - Represents upper and lower bound of the running time of an algorithm (tight bound)
    - It is used to indicate the average case complexity of an algorithm



Time Complexity

```
Statement;
```

constant

```
for(i=0; i< n; i++)
{
            statements;
}
```

Linear

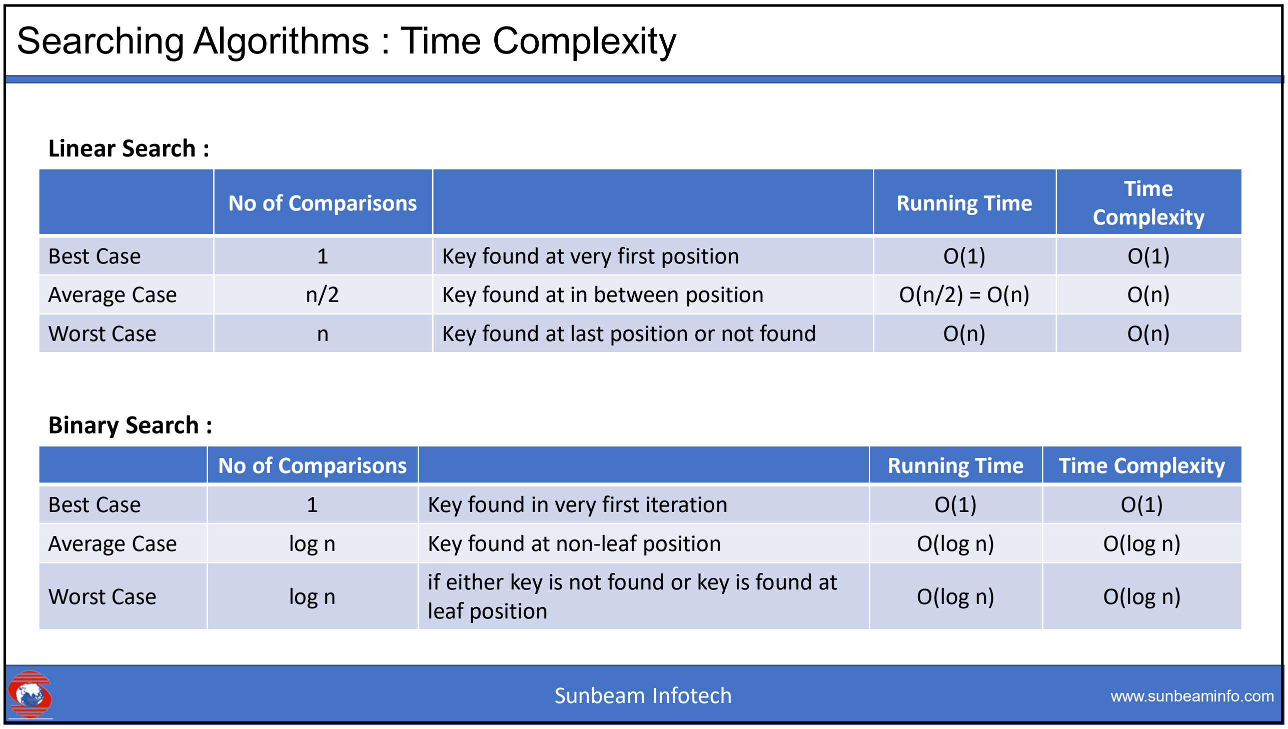
```
for(i=0; i< n; i++)
{
            for(j=0; j< n; j++)
            {
                        statements;
            }

}
```

Quadratic

```
for(i=n; i>0; i/=2)
{
            statement
}
```

Logarithmic



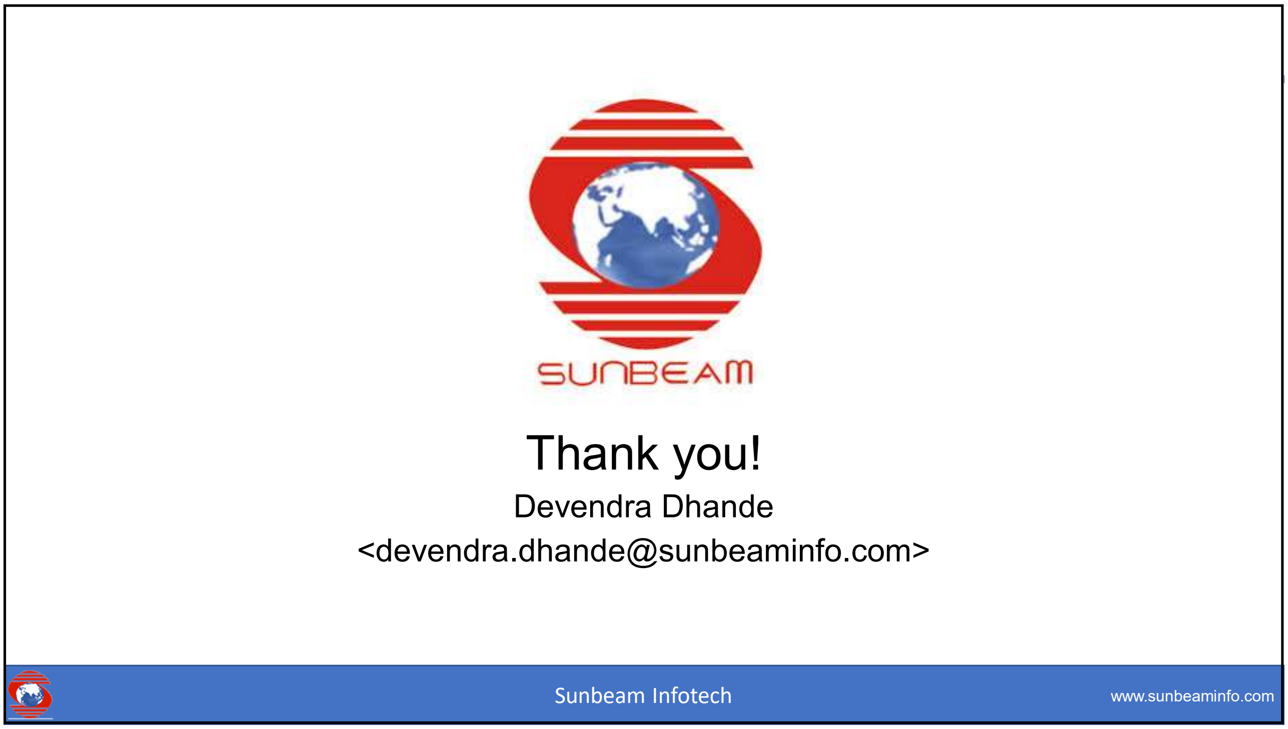
Searching Algorithms : Time Complexity

Linear Search :

	No of Comparisons		Running Time	Time Complexity
Best Case	1	Key found at very first position	O(1)	O(1)
Average Case	n/2	Key found at in between position	O(n/2) = O(n)	O(n)
Worst Case	n	Key found at last position or not found	O(n)	O(n)

Binary Search :

	No of Comparisons		Running Time	Time Complexity
Best Case	1	Key found in very first iteration	O(1)	O(1)
Average Case	log n	Key found at non-leaf position	O(log n)	O(log n)
Worst Case	log n	if either key is not found or key is found at leaf position	O(log n)	O(log n)



SUNBEAM

Thank you!

Devendra Dhande
<devendra.dhande@sunbeaminfo.com>