DATA STRUCTURES & ALGORITHMS

Introduction

Instructor: Engr. Laraib Siddiqui

Course Introduction

Objectives

- *To **introduce** the fundamental concepts of data structures*
- *To **deliver** an understanding of how to systematically organize data in a computer system.*
- *To **familiarize** students to analyze and compare data structures in terms of their time and space complexities.*

Learning Outcomes

- *Implement various data structures and their algorithms, and **apply** them in implementing simple applications.*
- ***Analyze** simple algorithms and determine their complexities.*
- ***Apply** the knowledge of data structures to other application domains.*
- ***Design** new data structures and algorithms to solve problems.*

Why Learn this course?

To address these issues:

- ✓Data Search
- ✓Processor speed
- ✓Multiple requests

Introduction

Data can be referred as:

- *Collection of facts*
- *Value or Set of Values*
- *Raw information*

Structure refers to:

- *Way of organizing data*
- *Set of rules to hold data*

Data Structure

- Organizing & storing data
- Mathematical & logical model of organization of data
- Set of procedures to define, store, access & manipulate data

OR

Data Structure is group of data elements which provides an efficient way of storing and organizing data in the memory.

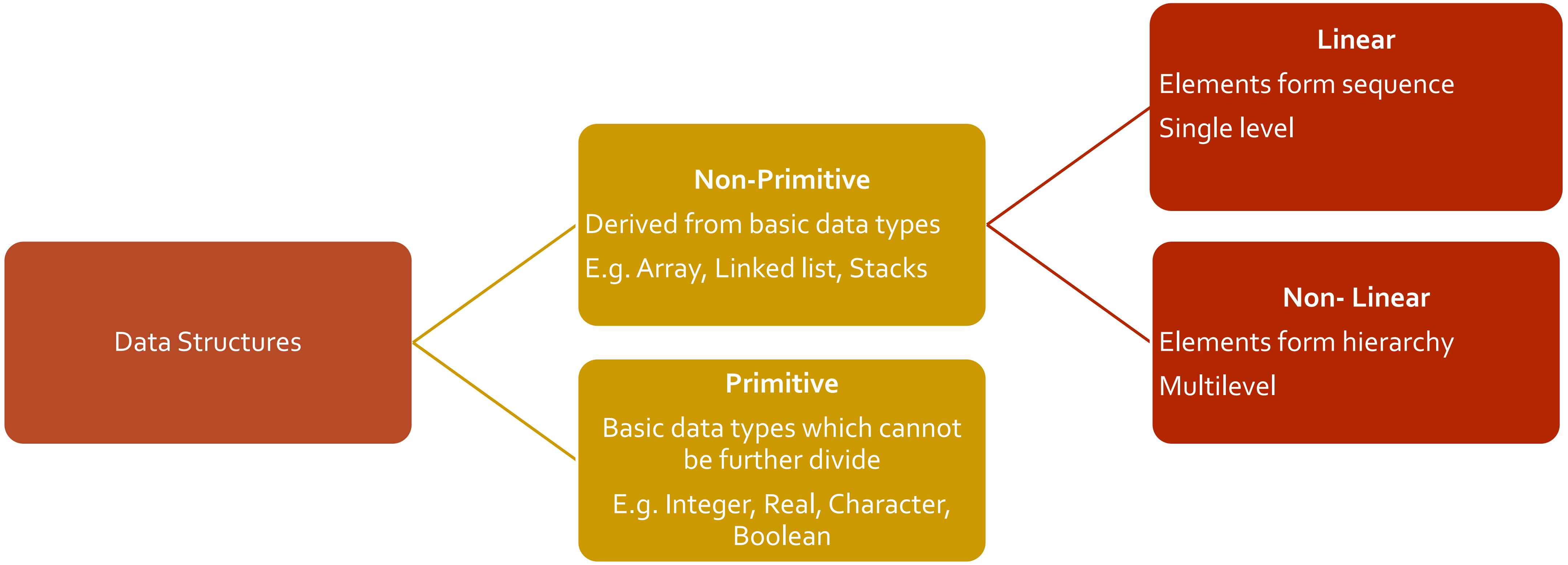
How we define the efficient way of storing & organizing data?

- Efficiency denotes to solve the problem within its resource constraints.
  - ✓Space
  - ✓Time

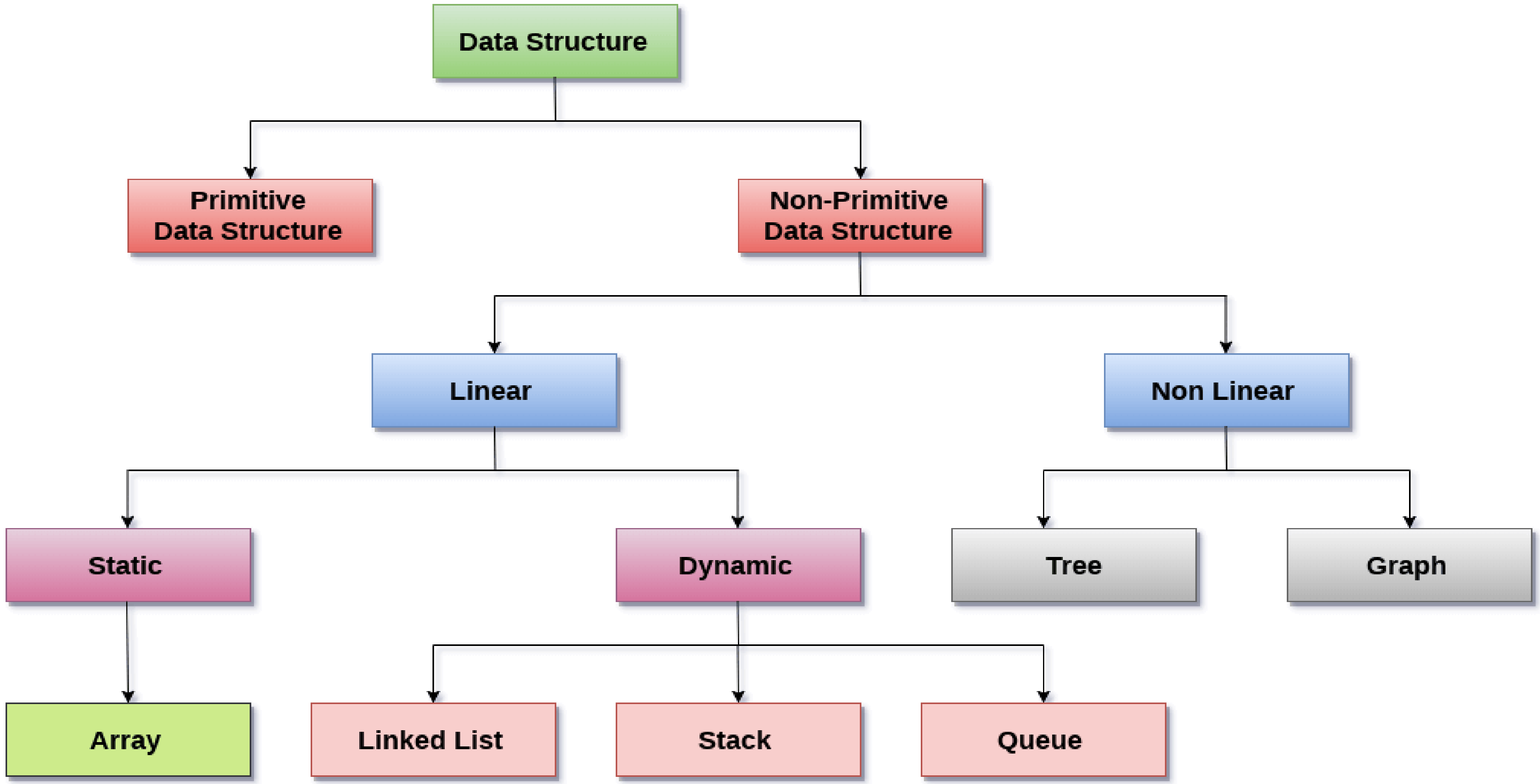
- Total resources that the solution consumes effects the solution’s cost.

DESIGN ISSUE???

Types of Data Structures


Data Structures
Non-Primitive
Derived from basic data types
E.g. Array, Linked list, Stacks
Primitive
Basic data types which cannot be further divide
E.g. Integer, Real, Character, Boolean
Linear
Elements form sequence
Single level
Non- Linear
Elements form hierarchy
Multilevel

Types of Data Structures



Linear Data Structures

Data Structure	Description
Array	Collection of items of same data type which stores in contiguous block of memory
Linked List	Sequence of links which contains items. Each link contains a connection to another link.
Stack	Linear list in which insertion and deletions are allowed only at one end, called top.
Queue	Opened at both end. It follows First-In-First-Out (FIFO) methodology for storing the data items.

Non – Linear Data Structures

Data Structure	Description
Tree	Multilevel data structures with a hierarchical relationship among its elements known as nodes.
Graph	Pictorial representation of the set of elements (represented by vertices) connected by the links known as edges.

Selecting a Data Structure

1. Analyze the problem to determine the resource constraints a solution must meet.
2. Determine the basic operations that must be supported. Quantify the resource constraints for each operation.
3. Select the data structure that best meets these requirements.

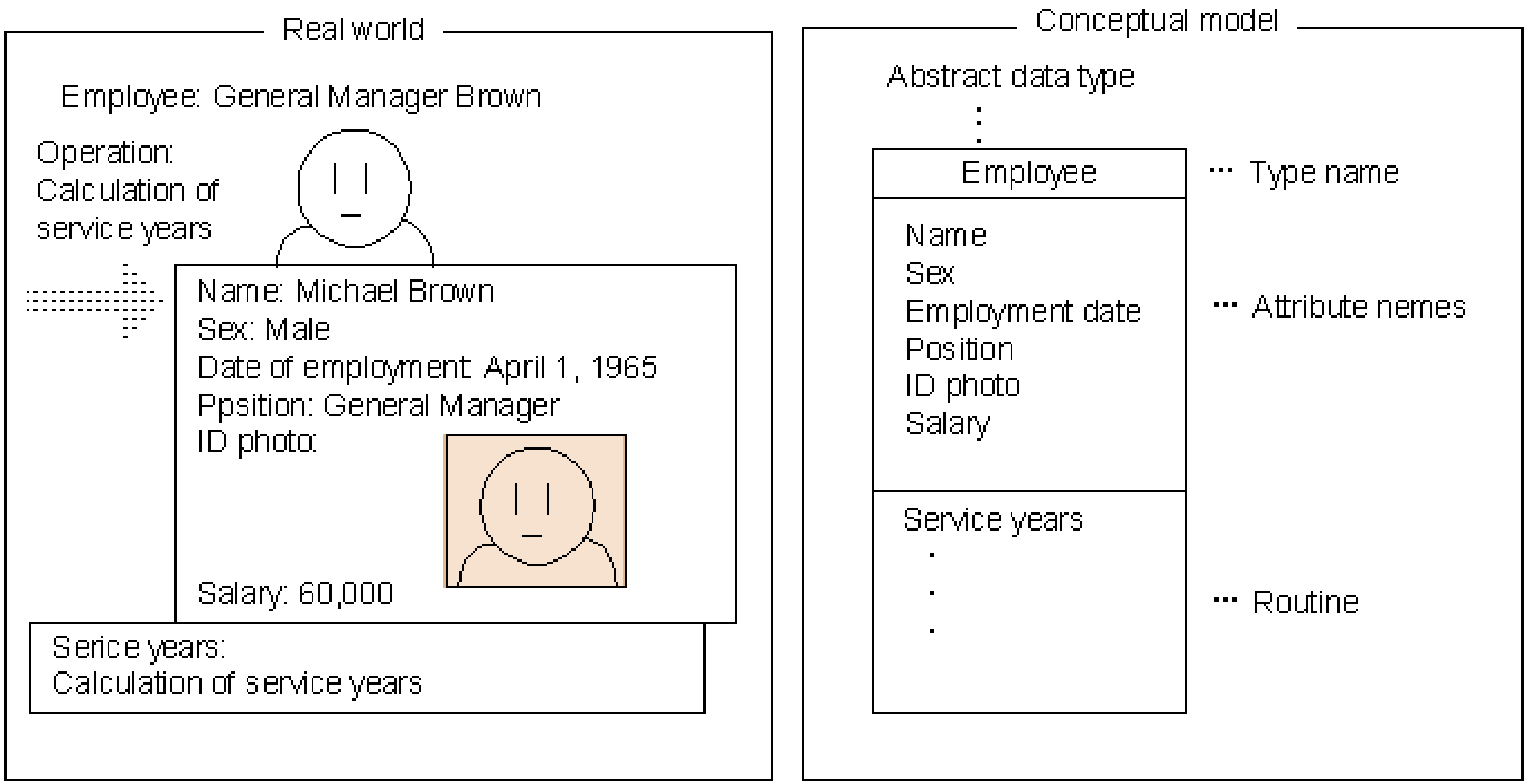
Operations

- Traversing
  - Accessing each data element exactly once so that certain items in the data may be processed
- Searching
  - Finding the location of the data element (key) in the structure
- Inserting
  - Adding a new data element to the structure
- Deleting
  - Removing a data element from the structure
- Sorting
  - Arrange the data elements in a logical order
- Merging
  - Combining data elements from two or more data structures into one

Abstract Data Types

- Abstract data type refers to a set of data values and associated operations that are specifically accurately, independent of any particular implementation.
- It is called "abstract" because it gives an implementation-independent view. The process of providing only the essentials and hiding the details is known as abstraction.

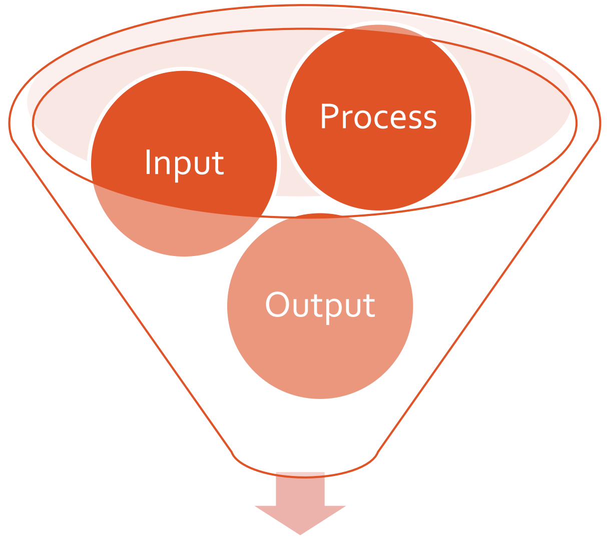
Think of ADT as a black box which hides the inner structure and design of the data type.


Real world
Employee: General Manager Brown
Operation:
Calculation of
service years
Name: Michael Brown
Sex: Male
Date of employment: April 1, 1965
Ppsition: General Manager
ID photo:
Salary: 60,000
Serice years:
Calculation of service years
Conceptual model
Abstract data type
Employee
Name
Sex
Employment date
Position
ID photo
Salary
Service years
··· Type name
··· Attribute nemes
··· Routine

Algorithm

- Set of instructions to accomplish a particular task.
- Method/process to solve a problem.

Program is an instance of an algorithm, written in some specific programming language.



Algorithm

Why we need algorithms?

- Internet
  - Web search, packet routing, distributed file sharing, ...
- Computers.
  - Circuit layout, file system, compilers, ...
- Computer graphics.
  - Movies, video games, virtual reality, ...
- Security.
  - Cell phones, e-commerce, voting machines, ...
- Multimedia.
  - MP3, JPG, DivX, HDTV, face recognition, ...
- Social networks.
  - Recommendations, news feeds, advertisements, ...

Basic Algorithm

Problem???
Find MAX

- Input = a, b, c
- Output = max
- Process
  - Let max = a
  - If b > max then

    max = b
  - If c > max then

    max = c
  - Display max

NOTE: Order is very important!

Steps for Algorithm development

- Devising
  - ✓ Method to solve a problem
- Validating
  - ✓ Proof the correctness of algorithm
- Expressing
  - ✓ Implement in desired programming language

Efficiency of an Algorithm

Measured by the amount or resources it uses (time/space)

Algorithm should:

- Independent of the programming language
- Should be applicable to all input sizes
- Understandable to programmer

Algorithm’s Complexity

- Function *f(n)* which measures the time and space used by an algorithm in terms on input size *n*.
- Way to classify how efficient an algorithm is compared to alternative ones.
- Computational complexity and efficient implementation depends on suitable data structures.

Space Complexity

- Amount of memory consumed by the algorithm (apart from input and output, if required by specification) until it completes its execution.
- The space occupied by the program is generally by the following:
  - ✓ A fixed amount of memory occupied by the space for the program code and space occupied by the variables used in the program.
  - ✓ A variable amount of memory occupied by the component variable whose size is dependent on the problem being solved. This space increases or decreases depending upon whether the program uses iterative or recursive procedures.

Space Complexity

- Memory taken by the instructions is not in the control of the programmer as its totally dependent upon the compiler to assign this memory.
- Memory space taken by the variables is in the control of a programmer. More the number of variables used, more will be the space taken by them in the memory.
- Three different spaces considered for determining the amount of memory used by the algorithm.
- Instruction space
  - ✓ memory occupied by compiled version of program.
- Data space
  - ✓ hold variables, data structures, allocated memory and other data elements.
- Environmental space
  - ✓ memory uses on the run time stack for each function call.

Time Complexity of an Algorithm

Amount taken by an algorithm to run.

- Compilation time
  - ✓ Time taken to compile (checking syntax & semantic errors, linking to libraries) an algorithm.
- Run Time:
  - ✓ Time to execute the compiled program.
  - ✓ Depend upon the number of instructions present in the algorithm.

NOTE: Run time is calculated only for executable statements and not for declaration statements.

Time Complexity of an Algorithm

Time complexity of an algorithm is generally classified as three types.

1. Worst case
   - ✓ Longest time that an algorithm will use to produce desired result.
   - ✓ Maximum value of f(n) for any possible input.

2. Average Case
   - ✓ Average time (average space) that an algorithm will use to produce desired result
   - ✓ Expected value of f(n)

3. Best Case
   - ✓ Lowest time that an algorithm will use to produce desired result