Faculty of Computers and Information
Mansoura University

Modeling and Simulation

Prof. Dr. Hazem El-Bakry

Information Systems Dept.

elbakry@mans.edu.eg

Modeling and Data Structure

Modeling

- **Modeling can be defined as describing things in the world in terms of their *structure* and *behavior.***
  - ***F=ma* (Force=mass * acceleration) is part of a *model* of the world that describes what happens when one thing hits another.**
  - **Maps *model* physical spaces and their physical relationships**

On a computer

- **we can *execute* these models: Make them work, plug values into equations, move things in space, see what happens.**
  - **That's *simulation*: Executing a model**

What's a data structure?

- Data Structure is a systematic way to organize data in order to use it efficiently. It is a specialized format for organizing and storing data. General data structure types include the array, the file, the record, the table, the tree, and so on.
- Any data structure is designed to organize data to suit a specific purpose so that it can be accessed and worked with in appropriate ways.
- In computer programming, a data structure may be selected or designed to store data for the purpose of working on it with various algorithms.

What's a data structure?

- **In computer science, a data structure is a data organization, management and storage format that enables efficient access and modification. More precisely, a data structure is a collection of data values, the relationships among them, and the functions or operations that can be applied to the data.**

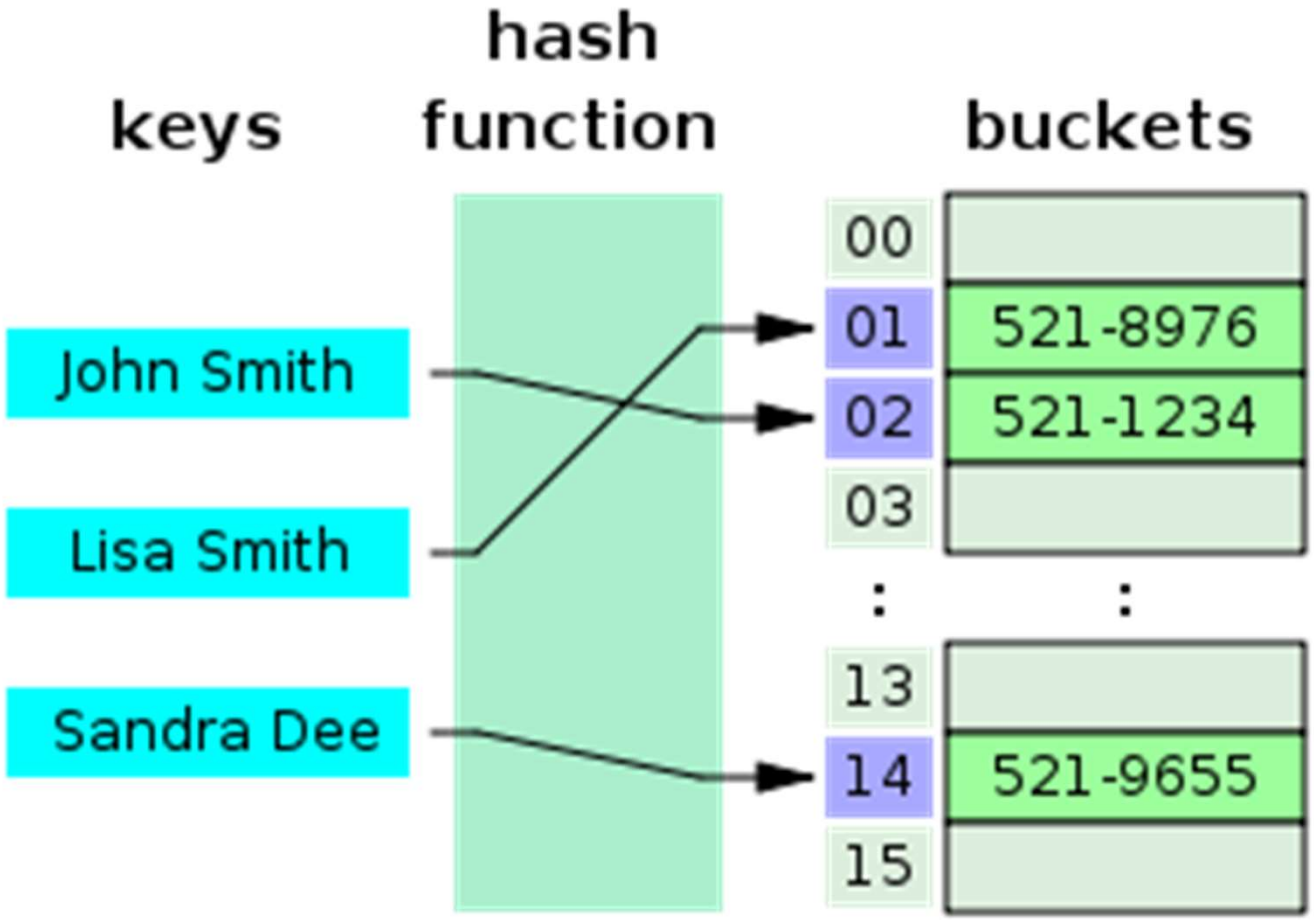
Data structures definition in general

- **In general it is a way of organizing information.**
- **Different *physical* structures organize space differently.**
  - **Skyscrapers vs. ranch homes.**
- **Data structures organize the information we use in our programs in different ways.**

Data structures have different *properties*

- **Arrays and tables keep things organized right next to one another.**
  - **Makes it easy to find something in the array or table**
  - **But if you want to insert something *new*, you have to move everything over.**
- **Linked lists and trees keep track of relationships with *links* (or *edges*)**
  - **Easier to insert new things**

A data structure known as a hash table.



Abstract Data Types (ADT)

- **Data structures can implement one or more particular abstract data types (ADT), which specify the operations that can be performed on a data structure and the computational complexity of those operations. In comparison, a data structure is a concrete implementation of the space provided by an ADT.**

kinds of data structures

- **Different kinds of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks. For example, relational databases commonly use B-tree indexes for data retrieval, while compiler implementations usually use hash tables to look up identifiers.**

kinds of data structures

- Data structures provide a means to manage large amounts of data efficiently for uses such as large databases and internet indexing services. Usually, efficient data structures are key to designing efficient algorithms.
- Some formal design methods and programming languages emphasize data structures, rather than algorithms, as the key organizing factor in software design.
- Data structures can be used to organize the storage and retrieval of information stored in both main memory and secondary memory

Data

Data are values or set of values. Data Item – Data item refers to single unit of values.

- **Data *(pl.):***
  - **Raw facts.**
  - **Distinct pieces of information usually formatted in a special way.**
- ***Note:* the term 'raw data' refers to unprocessed data.**

Information

- **A collection of facts organized in such a way that they have additional value beyond the value of the facts themselves.**

Knowledge base

- **The collection of data, rules, procedures, and relationships that must be followed to achieve value or the proper outcome.**

Process

- **(n) An executing program. The term is used loosely as a synonym of task.**
- **(v) To perform some useful operations on data.**

Data ‘Discussion’

- **Distinct pieces of information usually formatted in a special way. All software is divided into two general categories: data and programs. Programs are collections of instructions for manipulating data.**

Data ‘Discussion’

- **Data can exist in a variety of forms -- as numbers or text on pieces of paper, as bits and bytes stored in electronic memory, or as facts stored in a person's mind.**
- **Strictly speaking, data is the plural of datum, a single piece of information. In practice, however, people use data as both the singular and plural form of the word.**

Data ‘Discussion’

- **The term data is often used to distinguish binary machine-readable information from textual human-readable information. For example, some applications make a distinction between data files (files that contain binary data) and text files (files that contain ASCII data).**

Types of Data

Data	Represented by
Alphanumeric data	Numbers, letters, and other characters
Image data	Graphic images or pictures
Audio data	Sound, noise, tones
Video data	Moving images or pictures

Characteristics of Valuable Data

- Characteristics
  - Accurate
  - Complete
  - Economical
  - Flexible
  - Reliable
  - Relevant
  - Simple
  - Timely
  - Verifiable
  - Accessible
  - Secure

Group Items

- **Data items that are divided into sub items are called as Group Items.**

Elementary Items

- **Data items that cannot be divided are called as Elementary Items.**

Attribute and Entity

- **An entity is that which contains certain attributes or properties, which may be assigned values.**

Entity Set

- **Entities of similar attributes form an entity set.**
- **Field – Field is a single elementary unit of information representing an attribute of an entity.**

File

- **File is a collection of records of the entities in a given entity set.**

Simple variables – are primitive types

- **primitive data types:**
  - **An array**
  - **A linked list**
  - **random access**
  - **A record**
  - **A union**
  - **tagged union (variant)**
  - **Objects**

An array

- is a number of elements in a specific order, typically all of the same type (depending on the language, individual elements may either all be forced to be the same type, or may be of almost any type).
- Elements are accessed using an integer index to specify which element is required.
- Typical implementations allocate contiguous memory words for the elements of arrays (but this is not always a necessity).
- Arrays may be fixed-length or resizable.

linked list

- A linked list (also just called list) is a linear collection of data elements of any type, called nodes, where each node has itself a value, and points to the next node in the linked list.
- The principal advantage of a linked list over an array, is that values can always be efficiently inserted and removed without relocating the rest of the list.
- Certain other operations, such as random access to a certain element, are however slower on lists than on arrays.

record

- **A record (also called tuple or struct) is an aggregate data structure. It is a collection of field values of a given entity. A record is a value that contains other values, typically in fixed number and sequence and typically indexed by names. The elements of records are usually called fields or members.**

A union

- is a data structure that specifies which of a number of permitted primitive types may be stored in its instances, e.g.
- float or long integer. Contrast with a record, which could be defined to contain a float and an integer; whereas in a union, there is only one value at a time.
- Enough space is allocated to contain the widest member datatype.

A tagged union

- **also called variant, variant record, discriminated union, or disjoint union) contains an additional field indicating its current type, for enhanced type safety.**

Objects

- are collection of data items of various types. An object is a data structure that contains data fields, like a record, as well as various methods which operate on the contents of the record.
- In the context of object-oriented programming, records are known as plain old data structures to distinguish them from objects.

Implementation

- **Implementation provides the internal representation of a data structure. Implementation also provides the definition of the algorithms used in the operations of the data structure.**

Characteristics of a Data Structure

- **1. Correctness: Data structure implementation should implement its interface correctly.**
- **2. Time Complexity: Running time or the execution time of operations of data structure must be as small as possible.**
- **3. Space Complexity: Memory usage of a data structure operation should be as little as possible.**

Need for Data Structure

- As applications are getting complex and data rich, there are three common problems that applications face now-a-days.
- 1. Data Search:- Consider an inventory of 1 million(106) items of a store. If the application is to search an item, it has to search an item in 1 million(106) items every time slowing down the search. As data grows, search will become slower.

Need for Data Structure

- **Processor Speed :- Processor speed although being very high, falls limited if the data grows to billion records.**
- **3. Multiple Requests :- As thousands of users can search data simultaneously on a web server, even the fast server fails while searching the data.**

To solve this problems

- **data structures come to rescue. Data can be organized in a data structure in such a way that all items may not be required to be searched, and the required data can be searched almost instantly.**

Execution Time Cases

- There are three cases which are usually used to compare various data structure's execution time in a relative manner.
  - Worst Case
  - Average Case
  - Best Case

Worst Case

- **This is the scenario where a particular data structure operation takes maximum time it can take. If an operation's worst case time is ƒ(n) then this operation will not take more than ƒ(n) time, where ƒ(n) represents function of n.**

Average Case

- **This is the scenario depicting the average execution time of an operation of a data structure. If an operation takes ƒ(n) time in execution, then m operations will take mƒ(n) time.**

Best Case

- **This is the scenario depicting the least possible execution time of an operation of a data structure. If an operation takes ƒ(n) time in execution, then the actual operation may take time as the random number which would be maximum as ƒ(n).**

ROLE OF DATA-STRUCTURES IN COMPUTATION

- **Makes Computations Faster:**
- **• Faster is better. (Another way to make computations faster is to use parallel or distributed computation.)**
- **Computation = Sequence of Computation Steps**

Three Basic Computation Steps:

- **(1) Locate/Access data-values (inputs to a step)**
- **(2) Compute a value (output of a step)**

(3) Store the new value •

Program: Algorithm + Data Structure + Execution

- **Algorithm – The basic method; it determines the data-items computed. – Also, the order in which those data-items are computed (and hence the order of read/write data-access operations).**

Algorithms

- **Algorithms are designed to solve problems.**
  - **A problem can have multiple solutions.**
- **How do we determine which solution is the most efficient?**

Measuring the execution Time

- Construct a program for a given solution.
- Execute the program.
- Time it using a “wall clock”.
- Dependent on:
  - amount of data
  - type of hardware and time of day
  - programming language and compiler

Data structures

Data structures – Supports efficient read/write of data-items used/computed.

-

Total Time

- **Time to access/store data + Time to compute data.**

Efficient Algorithm

- **Efficient Algorithm = Good method + Good data structures + Good Execution**

Questions:

- • What is an efficient program?
- • What determines the speed of an Algorithm?
- • A program must also solve a "problem". Which of the
- three parts algorithm, data-structure, and implementation embodies this?

ALGORITHM OR METHOD vs. DATA STRUCTURE

- **Problem: Compute the average of three numbers.**
- **Two Methods:**
- **aver=(x + y + z)/3.**
- **aver=(x/3) + (y/3) + (z/3).**
- **• Method (1) superior to Method (2); two less div-operations.**
- **• They access data in the same order: ⟨x, y, z, aver⟩.**
- **• Any improvement due to data-structure applies equally well to both methods.**

Data structures:

- (a) Three variables x, y, z.
- (b) An array nums[0..2].
- This is inferior to (a) because accessing an array-item takes more time than accessing a simple variable. (To access nums[i], the executable code has to compute its address addr(nums[i]) = addr(nums[0]) + i*sizeof(int), which involves 1 addition and 1 multiplication.)

Data structures:

- When there are large number of data-items, naming individual data-items is not practical.
- Use of individually named data-items is not suitable when a varying number of data-items are involved (in particular, if they are used as parameters to a function).
- A Poor Implementation of (1): Using 2 additions and 1 division as follows:-
- a=x+y; //uses 2 additional assignments
- b=a+z;
- aver = b/3;

LIMITS OF EFFICIENCY

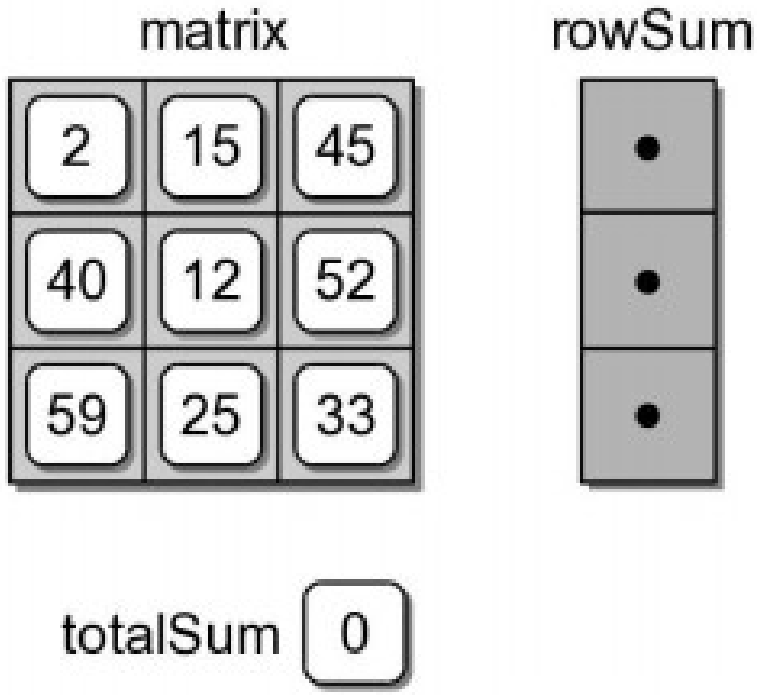
- **Hardware limit:**
- **• Physical limits of time (speed of electrons) and space (layout of circuits). This limit is computation problem independent. From 5 mips (millions of instructions per sec) to 10 mips is an improvement by the factor of 2. One nano-second = 10–9 (one billionth of a second); 10 mips = 100 ns/instruction.**

LIMITS OF EFFICIENCY

- **Limitless in a way, except for the inherent nature of the problem. That is, the limit is problem dependent.**
- **Sorting Algorithm A1: O(n. log n) time**
- **Sorting Algorithm A2: $O(n^2)$ time**
- **(n = number of items sorted) A1 is an improvement over**
- **A2 by the factor n^2 /n. log n = n /log n = → ∞ as n → ∞. • O(n. log n) is the efficiency-limit for sorting Algorithms.**

Example Algorithm

- Given a matrix of size *n x n*, compute the:
  - sum of each row of a matrix.
  - overall sum of the entire matrix.



Example Algorithm (v.1)

- How many additions are performed?

$= 2n(n) = 2n^2$

```
rowSum = Array(n)
totalSum = 0
for i in range( n ) :
    rowSum[i] = 0
    for j in range( n ) :
        rowSum[i] = rowSum[i] + matrix[i,j]
        totalSum = totalSum + matrix[i,j]
```

n

n

2

Example Algorithm (v.2)

- How many additions are performed?

$= (n + 1)\, n = n^2 + n$

```
rowSum = Array(n)                                   1
totalSum = 0                                        1
for i in range( n ) :                               n
    rowSum[i] = 0                                   n
    for j in range( n ) :                           n
        rowSum[i] = rowSum[i] + matrix[i,j]         1
    totalSum = totalSum + matrix[i,j]               1
```

Compare the Results

- Number of additions: v_1: $2n^2$ v_2: $n^2 + n$
- Second version has fewer additions ($n > 1$)
  - Will execute faster than the first.
  - Difference will not be significant.

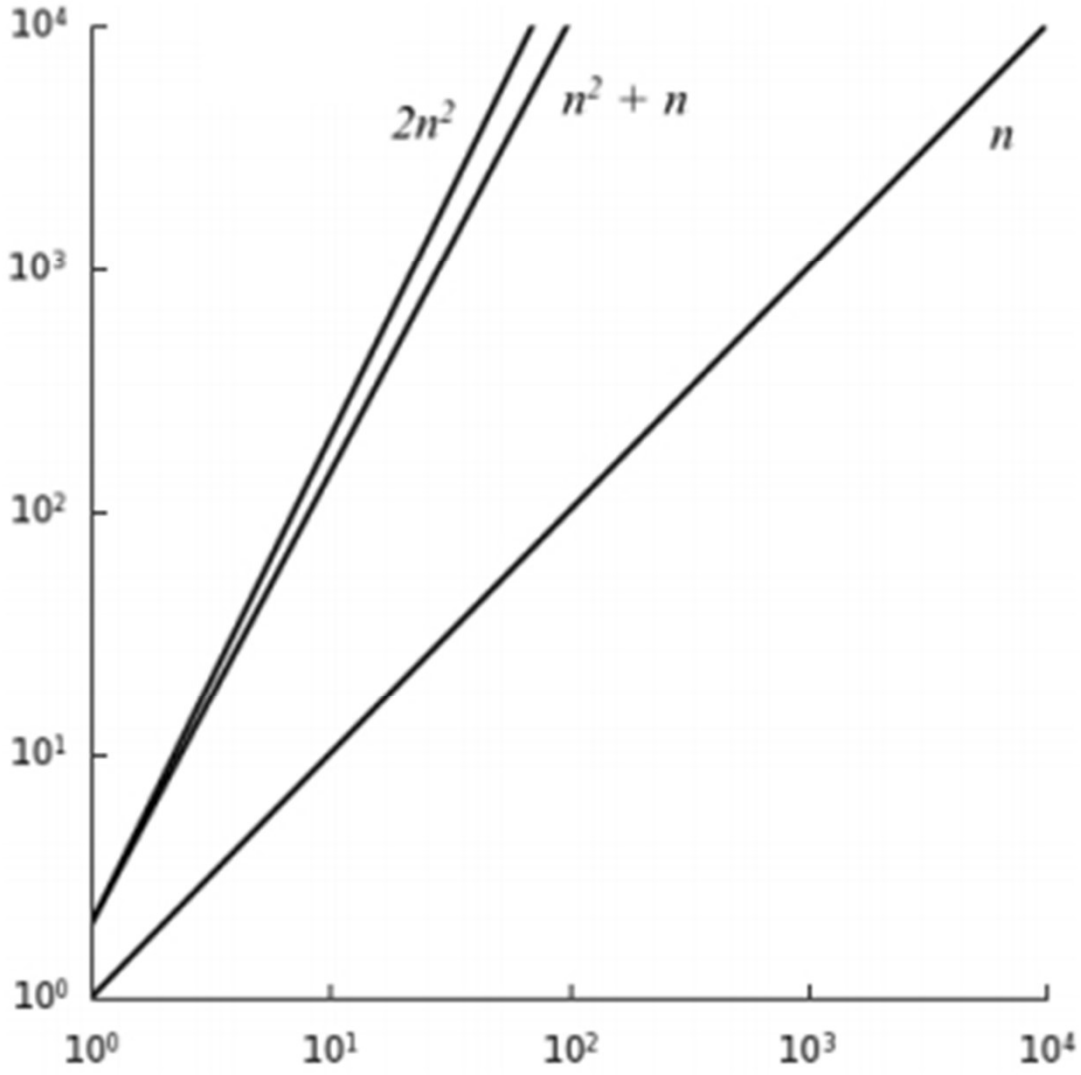
Both algorithms execute on the same order of magnitude, n^2

Growth Rates

- As n increases, both algorithms increase at approx the same rate:

n	$2n^2$	$n^2 + n$
10	200	110
100	20,000	10,100
1000	2,000,000	1,001,000
10,000	200,000,000	100,010,000
100,000	20,000,000,000	10,000,100,000

Growth Rates



MEASURING PERFORMANCE

Analytic Method:

- **Theoretical analysis of the Algorithm's time complexity.**

Empirical Methods:

- **Count the number of times specific operations are performed by computer device and then multiply this number by the execution time of an operation.**
- **Measure directly the actual program-execution time in a run.**

Classification of Data Structure Types

- **Data Structure can be classified into two branches:-**
- **Main Data Structure (ex. Arrays, Record, File, Set)**
- **Advanced Data Structure (ex. Stack, Queue, Double ended Queue, Tree, graph)**

Data type

- consists of two parts, a set of data and the operations that can be performed on the data, i. e.
- 1. A domain of allowed values, and
- 2. A set of operations on those values.

Abstract Data Types

- **An Abstract Data Type (ADT) is a well-specified collection of data and a group of operations that can be performed upon the data.**
- **The ADT's specification describes what data can be stored (the characteristics of the ADT), and how it can be used (the operations), but not how it is implemented or represented in the program.**

ADT

- **ADT is a data type whose properties (domain and operations) are specified independently of any particular implementation**

Domains & operations

Type	values	operations
Char	ASCII	= =,!=,>,<,...
Integer	maxint....maxint -32768 to 32767 0 to 65535 unsig	+, -, *, mod, =, 0, <,...

Domains & operations

- **Objects such as lists, sets, and queues, and their operations, can be viewed as ADT, just as integers, real, and Booleans are data types**
-
- **Example: set ADT has operations as union, intersection, size, and complement**

An abstract data type (ADT) defines

- a state of an object and
- operations that act on the object, possibly changing the state.
- Similar to a Java class.

The ADTs to be studied (and some sample applications) are:

- **Stack**
  - evaluate arithmetic expressions
- **Queue**
  - simulate complex systems, such as traffic
- **General list**
  - AI systems, including the LISP language
- **Tree**
  - simple and fast sorting
- **Table**
  - database applications