Data Structures and Algorithm

Department of Computer Science
COMSATS University, Islamabad

Today Covered

After completing this lecture you will be able to know

- **What this course is about?**
- **What is Data Type?**
- **What is Abstract Data Type(ADT)?**
- **What is Data Structure?**
  - What good will it do me to know about them?
  - Why can't I just use arrays and for loops to handle my data?
- **Why do you need to study DS?**
- **Classification of Data Structure**
- **When does it make sense to apply what I learn here?**
- **Conclusion**

Let us Start our Journey

Lecture No 1

Introduction

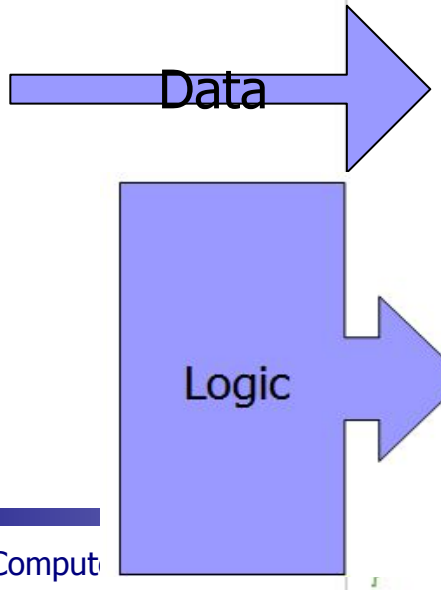
(What, Why and Where Data Structure . . .)

WHAT

- **What is primary job of a software engineer?**
  - To develop software
- **What is software?**
  - Programs that run on a device.
- **What is program?**
  - Data + logic

A simple Program with built in data type

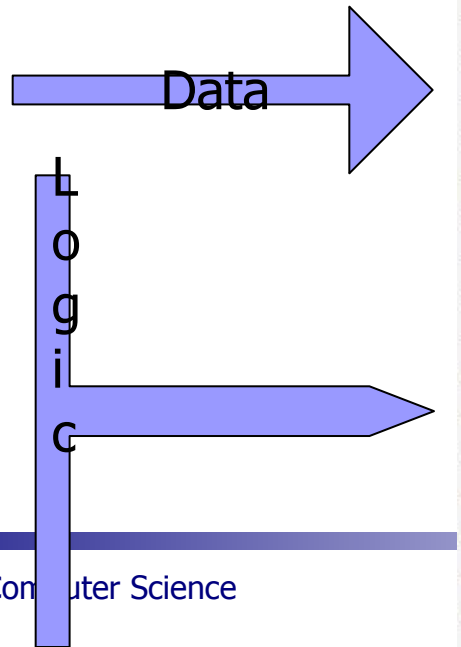
- What is Program?
  - Program = Data + logic
- java program finds largest of three numbers and then prints it.




```
import java.util.Scanner;

class LargestOfThreeNumbers
{
   public static void main(String args[])
   {
      int x, y, z;
      System.out.println("Enter three integers ");
      Scanner in = new Scanner(System.in);

      x = in.nextInt();
      y = in.nextInt();
      z = in.nextInt();

      if ( x > y && x > z )
         System.out.println("First number is largest.");
      else if ( y > x && y > z )
         System.out.println("Second number is largest.");
      else if ( z > x && z > y )
         System.out.println("Third number is largest.");
      else
         System.out.println("Entered numbers are not distinct.")
```

Operation on data set

- What about if we have 100 elements and we want to find the largest element from them?



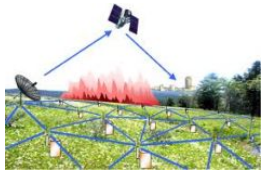

```java
import java.util.Scanner;
public class Largest_Number
{
    public static void main(String[] args)
    {
        int n, max;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter number of elements in the array:");
        n = s.nextInt();
        int a[] = new int[n];
        System.out.println("Enter elements of array:");
        for(int i = 0; i < n; i++)
        {
            a[i] = s.nextInt();
        }
        max = a[0];
        for(int i = 0; i < n; i++)
        {
            if(max < a[i])
            {
                max = a[i];
            }
        }
        System.out.println("Maximum value:"+max);
    }
}
```

- **Computer:**
  - Machine that manipulates **information/data.**
- **Study of computer Science:** includes the study of
  - How information is **organized** in a computer?
  - How it can be **manipulated**?
  - How it can be **utilized**?
- More powerful computers ⇒ more complex applications.
- More complex applications demand more calculations.

Introduction

- If computer science is fundamentally the study of information/data, the first question that arises is, **what is data/information?**
- Data is derived from a Latin word "Datum" which means collection.
- So data can be defined as **collection of facts and figures** collected from any **specific environment** for a **specific purpose.**
- **SOURCE:.**

Social Media

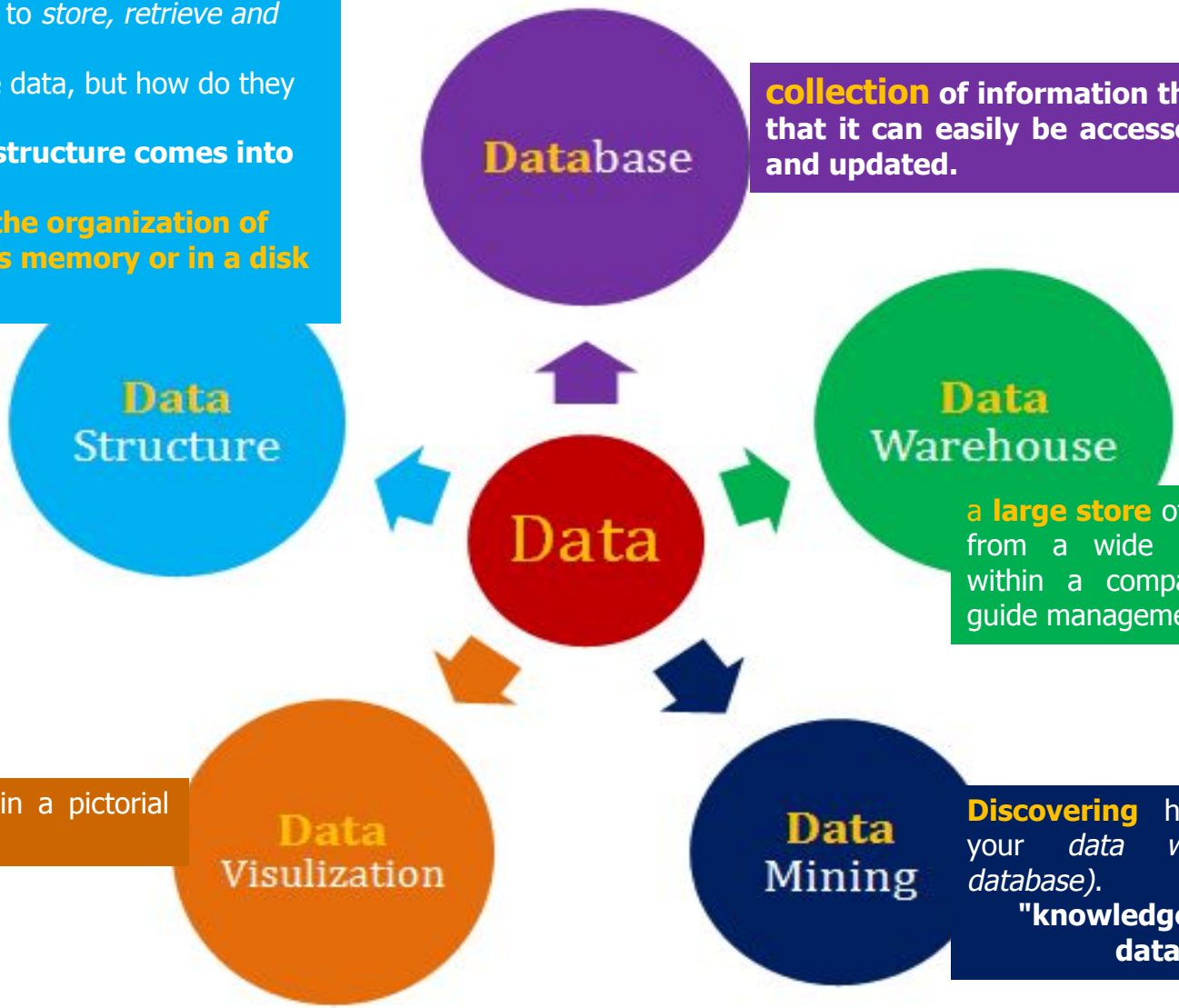


Sensor technology and networks
(measuring all kinds of data)

Mobile devices
(tracking all objects all the time)

Subjects Related to DATA

Database can be used to *store, retrieve and filter* out data.
So Database can store data, but how do they store data internally?
This is where Data structure comes into play.
A data structure is the organization of data in a computer's memory or in a disk file.



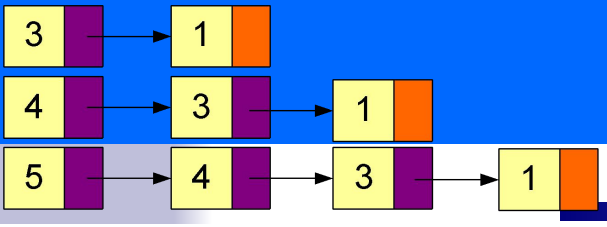
collection of information that is store so that it can easily be accessed, managed, and updated.

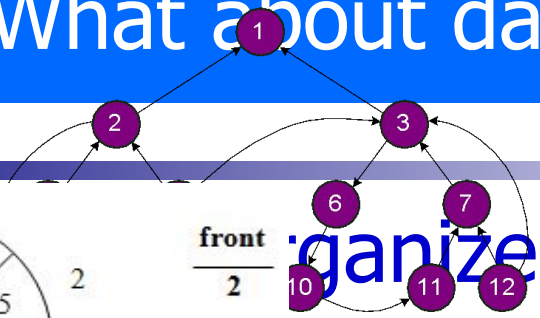
a **large store** of data accumulated from a wide range of sources within a company and used to guide management decisions.

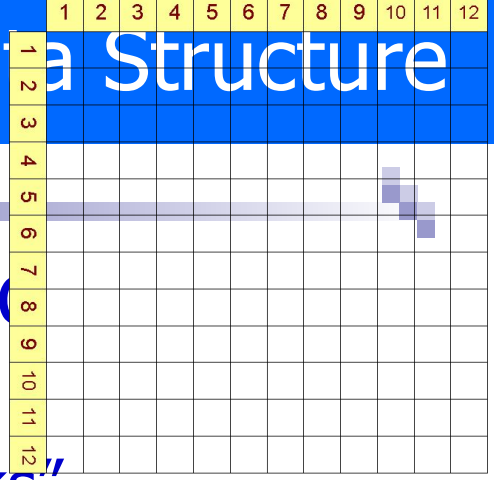
presentation of *data* in a pictorial or graphical format

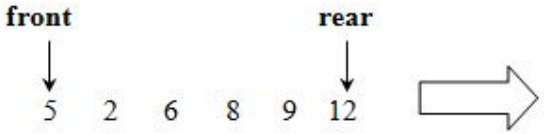
Discovering hidden pattern in your *data warehouse* (large *database*).
"knowledge discovery in databases"

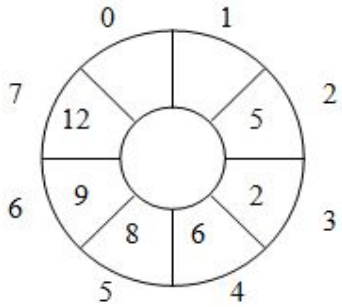
What about data Structure











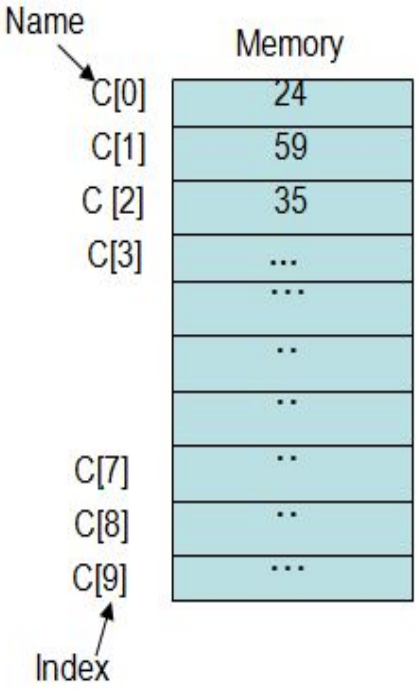
front 2

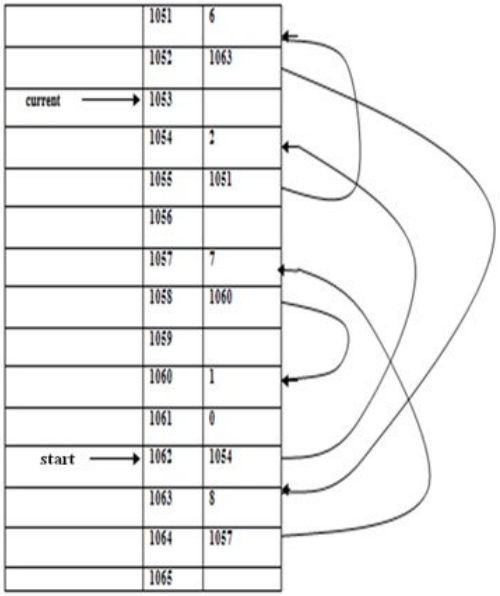
rear 7

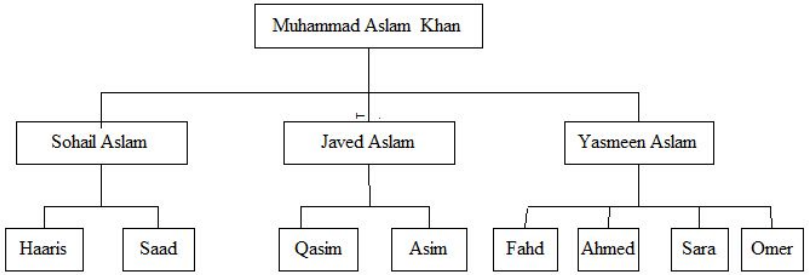
ganize

with "links"

- Place data with **"formula"**(some specific order)







What about data Structure

- Up Shot:
- A data structure is a systematic way of organizing and accessing data

What about algorithms

- Algorithms: how to access data for a result
  - Scan data sequentially
  - Scan data according to the sequence of a structure
  - Scan data with "formula"
- Algorithms: how to provide a smart solution
  - An **algorithm is a step-by-step procedure** for solving a problem in a finite amount of time.
- Algorithms and Data Structures go hand-in hand:
  - Certain Algorithms require certain data structures to run efficiently
  - Certain data structures require certain Algorithms to run efficiently

What about algorithms

- Up Shot:
- An algorithm is a **procedure** for carrying out a particular task

Algorithm

- The order of the actions in an algorithm is important
  - Example: algorithm to go to work every morning
    - Get out of bed;
    - Take a bath;
    - Get dressed;
    - Eat breakfast;
    - Take the bus to work.
- If I carry out the same actions in a different order
    - Get out of bed;
    - Get dressed;
    - Take a bath;
    - Eat breakfast;
    - Take the bus to work.
- Then, I will still get to work, however soaking wet

So order is important

Designing Data Structures

- Each data structure has costs and benefits.
  - Rarely is one data structure better than another in all situations.
- A data structure requires:
  - space for each data item it stores,
  - time to perform each basic operation,
  - programming effort.
- Each problem has constraints on available time and space.
- Only after a careful analysis of problem characteristics and solution requirements can we know the best data structure for the task.

Selecting Data Structures

- Select a data structure as follows
  1. Analyze the problem to determine the resource constraints a solution must meet.
  2. Determine operations that must be supported. Quantify resource constraints for each operation.
  3. Select the data structure that best meets these requirements.
- This three-step approach to selecting a data structure operationalizes a data centered view of the design process.
  - The first concern is for the data and the operations to be performed on them,
  - the next concern is the representation for those data, and
  - the final concern is the implementation of that representation.

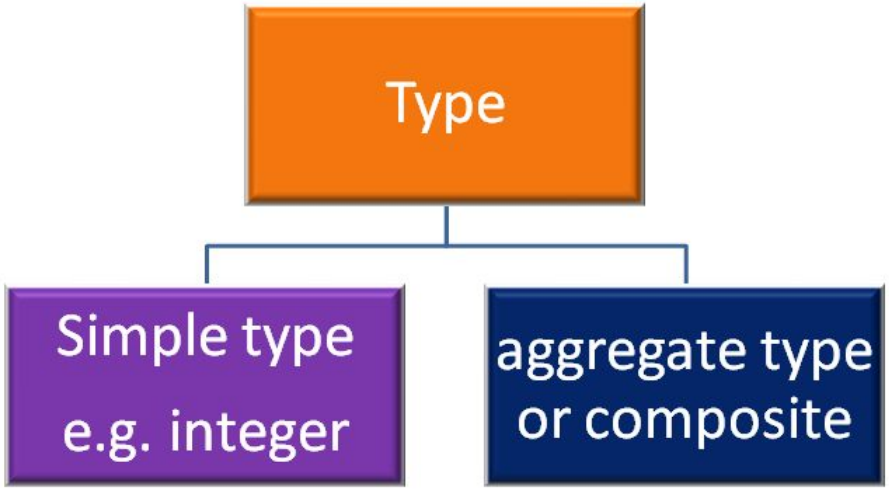
Selecting Data Structures

- Select a data structure as follows
  1. Analyze the problem to determine the resource constraints a solution must meet.
  2. Determine basic operations that must be supported. Quantify resource constraints for each operation.
  3. Select the data structure that best meets these requirements.
- Some questions to ask:
  - Are all the data inserted into the structure at the beginning or are insertions interspersed with other operations?
  - Can data be deleted?
  - Are the data processed in some well-defined order, or is random access allowed?

Review of some basic Terminology/ concept

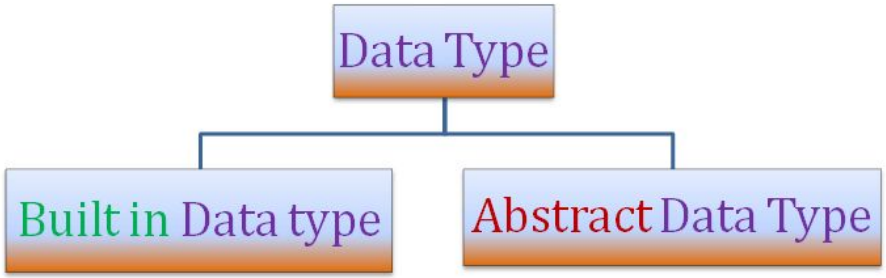
Data Type

- Before going to study the actual subject, it will be a advantage if we aware ourselves about the various terms involved in the subject.
- **What is Type?**
  - A type is a collection of values.
  - Example, the Boolean type consists of the values **true and false.**
  - The integers also form a type.
  - A bank account record will typically contain several pieces of information such as name, address, account number, And account balance.



Data Type

- **What is Data item?**
  - a piece of information or a record whose value is drawn from a type. A data item is said to be a member of a type.
- **What is Data Type?**
  - A data type is a **type** together with a **collection of operations** to manipulate the type.
  - DT = (**T**, **O**)
  - For example, an **integer variable** is a member of the integer **data type**.
  - **Addition** is an example of an **operation** on the integer data type.
- Types of Data Type



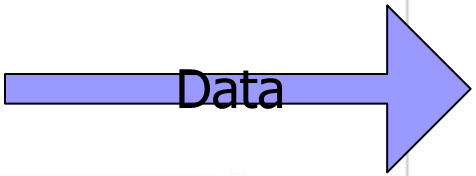
Built in Data Type

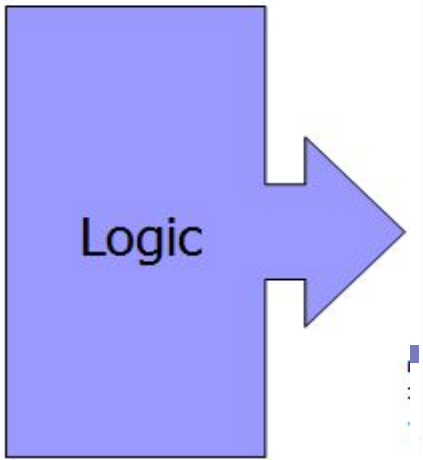
- With every programming language, there is a set of data types called built in data types.
- In Java, our programs are all built from just a few basic types of data:

TYPE NAME	KIND OF VALUE	MEMORY USED	SIZE RANGE
boolean	true or false	1 byte	not applicable
char	single character (Unicode)	2 bytes	all Unicode characters
byte	integer	1 byte	−128 to 127
short	integer	2 bytes	−32768 to 32767
int	integer	4 bytes	−2147483648 to 2147483647
long	integer	8 bytes	−9223372036854775808 to 9223372036854775807
float	floating-point number	4 bytes	$-3.40282347 \times 10^{+38}$ to $-1.40239846 \times 10^{-45}$
double	floating-point number	8 bytes	$\pm 1.76769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$

A simple Program with built in data type

- What is Program?
  - Program = Data + logic
- java program finds largest of three numbers and then prints it.

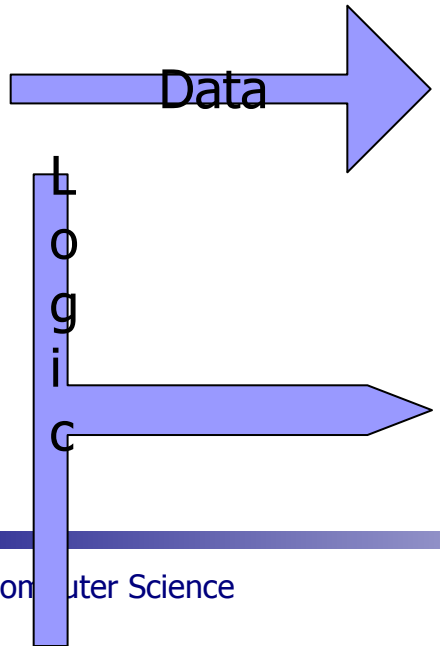





```java
import java.util.Scanner;

class LargestOfThreeNumbers
{
   public static void main(String args[])
   {
      int x, y, z;
      System.out.println("Enter three integers ");
      Scanner in = new Scanner(System.in);

      x = in.nextInt();
      y = in.nextInt();
      z = in.nextInt();

      if ( x > y && x > z )
         System.out.println("First number is largest.");
      else if ( y > x && y > z )
         System.out.println("Second number is largest.");
      else if ( z > x && z > y )
         System.out.println("Third number is largest.");
      else
         System.out.println("Entered numbers are not distinct.")
   }
}
```

Operation on data set

- What about if we have 100 elements and we want to find the largest element from them?




```java
import java.util.Scanner;
public class Largest_Number
{
    public static void main(String[] args)
    {
        int n, max;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter number of elements in the array:");
        n = s.nextInt();
        int a[] = new int[n];
        System.out.println("Enter elements of array:");
        for(int i = 0; i < n; i++)
        {
            a[i] = s.nextInt();
        }
        max = a[0];
        for(int i = 0; i < n; i++)
        {
            if(max < a[i])
            {
                max = a[i];
            }
        }
        System.out.println("Maximum value:"+max);
    }
}
```

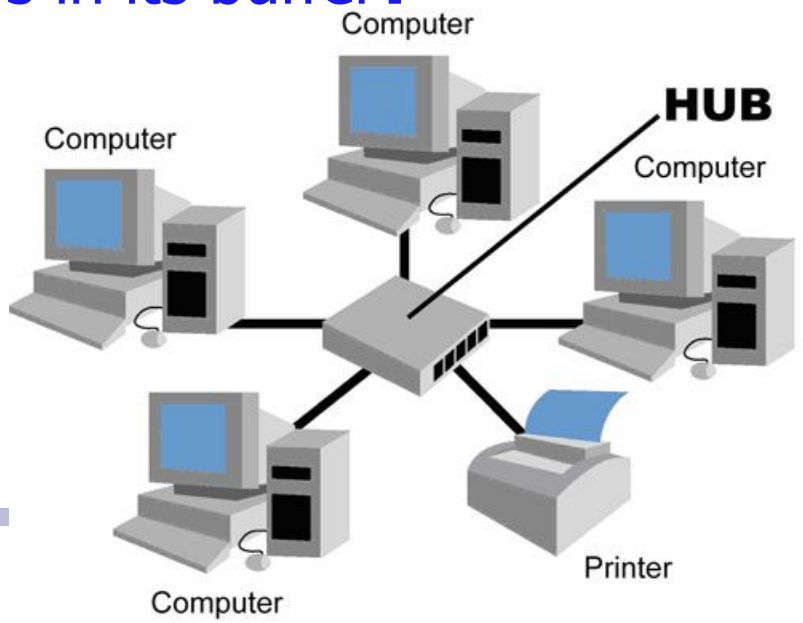
Need for Data Structures

Why not just ARRAY?

- Can I just use **arrays** and **for loops** to handle my data?
- No(Why)
- **Example 1: How is undo and redo functionality typically implemented?**
  - Suppose you perform following in word application
  - First you typed a word
  - Secondly, you **bold** it
  - Thirdly, You underlined it
  - Fourth, you changed its color?
  - Fifth you change its font size?
- In which **order** action will redo?

Why not just ARRAY?

- Example 2: Suppose there are four computers and one printer is connected. Computer B send print at time 9:30 am which takes 15 minutes for printing, Computer D send print command at time 9:35 which takes 7 minutes, computer A send print command at 9: 36 which takes 2 minutes and computer C send print command at 9:37 which takes 30 seconds. How printers organized these jobs in its buffer**?**
- **In which order job will prints?**



Need for Data Structures

- Suppose your are given a set of keys(e.g. words). You want to store these keys in lexicographical order(dictionary Order). How best to arrange this set of key to minimize the average search time if we know that some keys are looked up more than others.

Need for Data Structures

- How many cities with more than 250,000 people lie within 500 miles of Islamabad?
- How many people in my company make over Rs1000,000 per year?
- Can we connect all of our telephone customers with less than 1,000 miles of cable?
- To answer questions like these, it is not enough to have the necessary information.
- We must organize that information in a way that allows us to find the answers in time to satisfy our needs.

Need for Data Structures

- When an application requires a special kind of data which is not available as built in data type then it is the programmer's burden to implement his own kind of data.
- Here, the programmer has to give more effort regarding:
  - How to store values for that data,
  - What are the operations that meaningfully manipulate variables of that kind of data, amount of memory requires to store for a variable.
- The programmer has to decide all these things and accordingly implement it.

Need for Data Structures

- Programmer's own data type is termed as abstract data type.
- Abstract data type(ADT) is also alternatively termed as user-defined data type.
- An ADT, in fact, can be built with the help of built in data type.
- Some programming languages allow facility to build ADT easily.
- For example, using struct/class in C/C++, and using record in Pascal, programmers can defined their own data type.

Need for Data Structures

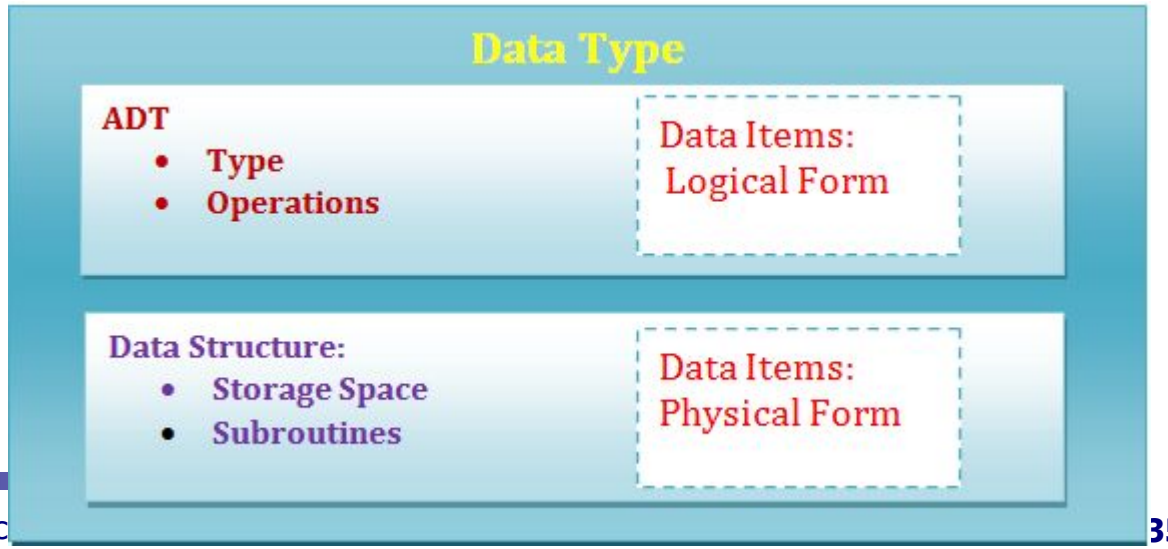
- An abstract data type (ADT) is the realization of a data type as a software component.
  - The interface of the ADT is defined in terms of a type and a set of operations on that type.
  - The behavior of each operation is determined by its inputs and outputs.
  - An ADT does not specify how the data type is implemented.
  - These implementation details are hidden from the user of the ADT and protected from outside access, a concept referred to as encapsulation.

Data Structures vs ADT

- For a given kind of user data, its structure implies the following:
- Domain(**D**): This is the range of values that data may have. This domain is also termed as data objects.
- Function(**F**): This is the set of operations which may be legally be applied to elements of data objects.
- Axiom(**A**): This is the set of rules with which the different operations belongs to F can be implemented(Algorithm).

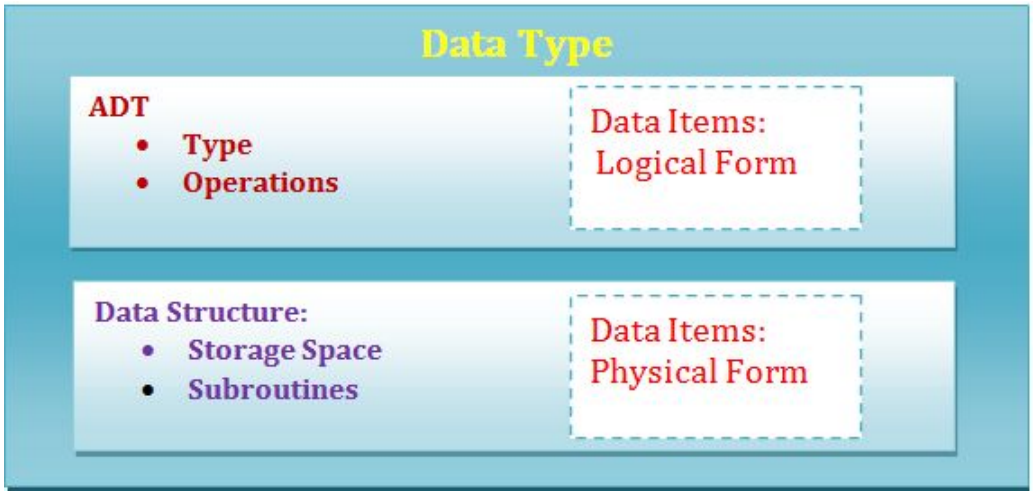
Need for Data Structures

- Now we can defined the term "Data Structure"
- A Data Structure D is a triplet, that is,
- D = (**D**,**F**,**A**), where **D** is a set of objects, F is set of operations and A is a set of rules to implement the functions.
- A data structure is the implementation for an ADT.



Need for Data Structures

- A data structure is the implementation for an ADT.
- The relationship between data items, abstract data types, and data structures.
  - The ADT defines the logical form of the data type.
  - The data structure implements the physical form of the data type.



Need for Data Structures

- In an object-oriented language such as Java, an ADT and its implementation together make up a class.
- Each operation associated with the ADT is implemented by a member function or method.
- Program = Data + logic
- Now we can redefined our definition of Program as:
- **Program = Data Structure + Algorithm**

MOTIVATION

WHY

Motivating Discussion

- Structures and algorithms are required for efficient work in all walks of daily life. Examples?
- Hostel-room analogy:
  - A "cluttered" room: quick to store but takes time to find items
  - An "organized" room: quick to find items but takes time to store
- Points:
  - For each "application" a different structure may be required
  - Depending on the structure, the process of using it efficiently will be different

Need for Data Structures

- We study data structures so that we can learn to write more efficient programs.
- A solution/program is said to be efficient if it solves the problem within the required resource constraints. Examples of resource constraints include:
  - the total space available to store the data—possibly divided into separate main memory and disk space constraints
  - the time allowed to perform each subtask.
- Data structures organize data $\Rightarrow$ more efficient programs.

Need for Data Structures

- Each data structure and each algorithm has costs(amount of resources that the solution consumes) and benefits.
- Practitioners need a thorough understanding of how to assess costs(amount of resources that the solution consumes) and benefits to be able to adapt to new design challenges.
- This requires an understanding of the principles of algorithm analysis.

Need for Data Structures

- Data structures organize data ⇒ more efficient programs.
- More powerful computers ⇒ more complex applications.
- More complex applications demand more calculations.

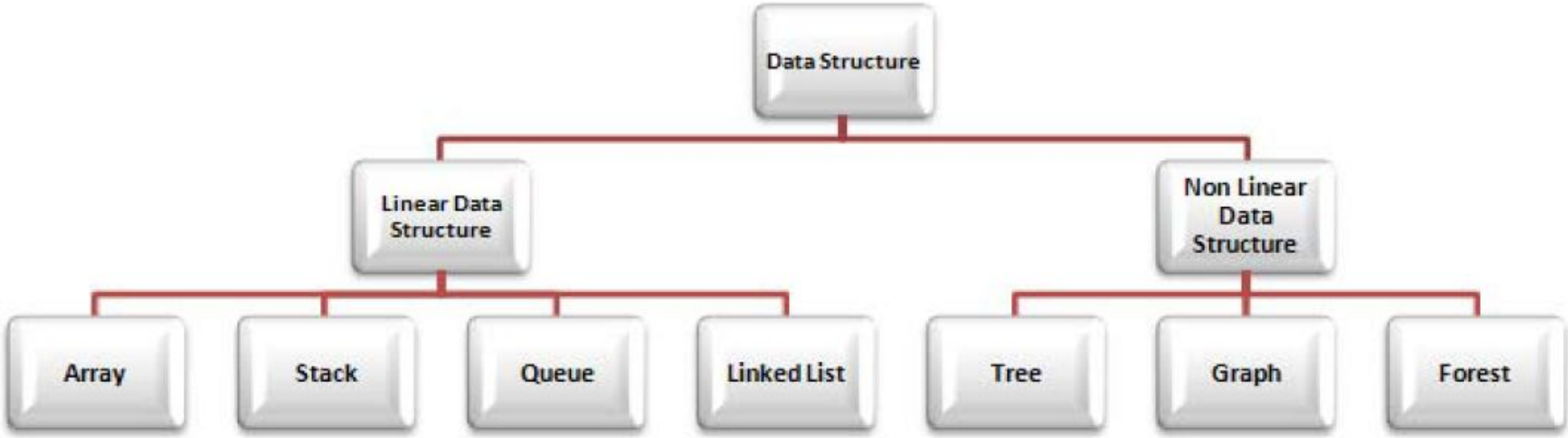
Classification

Classification of Data Structure

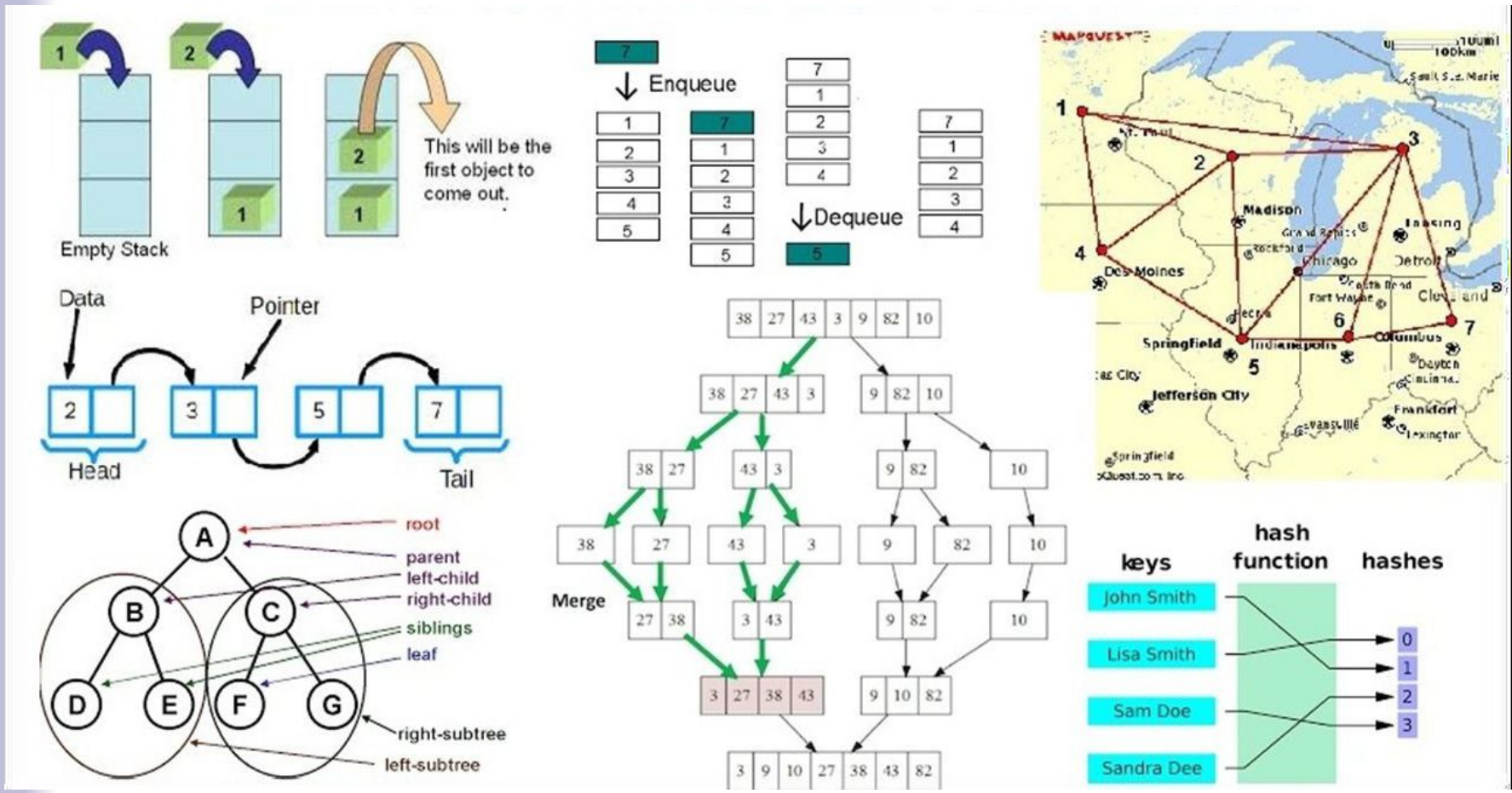
- Data Structures are classified into two main groups:
  - Linear Data structure
  - Non Linear Data structure
- Linear Data structure
  - The data structures whose elements are arranged in a sequence are called linear data structure.
  - There are two ways of representing linear data structures in memory.
    - One way is to have the linear relationship between the elements by means of sequential memory locations.
    - The other way is to have the relationship between the elements represented by means of pointers or links.
    - **Example**: Arrays, Linked Lists, Stacks, Queues etc.

Classification of Data Structure

- Non Linear Data structure
  - The data structures whose elements are arranged in non-linear or non-sequence form are called non linear data structures
  - **Example:** Trees, Graphs.
- **Up Shot**



- **Upshot**


1
2
2
1
1
This will be the first object to come out.
Empty Stack
Data
Pointer
2
3
5
7
Head
Tail
A
B
C
D
E
F
G
root
parent
left-child
right-child
siblings
leaf
right-subtree
left-subtree
7
Enqueue
Dequeue
38 27 43 3 9 82 10
Merge
3 27 38 43
3 9 10 27 38 43 82
MAPQUEST
Madison
Chicago
Des Moines
Springfield
Indianapolis
Columbus
Jefferson City
keys
hash function
hashes
John Smith
Lisa Smith
Sam Doe
Sandra Dee
0
1
2
3

Operations

Operations

- The data in a data structure is processed using certain operations.
- Some commonly used operations performed on data structures are:
  - Insertion
  - Deletion
  - Traversing
  - Search
  - Sorting
  - Merging

Operations

- Insertion
  - **Adding** new data items into a data structure is called inserting
- Deletion
  - **Removing** data items from a data structure is called deletion
- Traversing
  - **Accessing** each record or item in a data structure exactly once for processing is called traversing. It is also called visiting
- Search
  - **Finding** specific data items in a data structure is called searching
- Sorting
  - **Arranging** data items in a data structure into a specific order is called sorting
- Merging
  - **Combing** two list of data items into a single data list is called merging

CONCLUSION

Points to remember

Data Structure	**A data structure is the organization of data in a computer's memory or in a disk file.**
Classification	**Linear Data structure, Non Linear Data structure**
Program Efficiency	**The correct choice of data structure allows major improvements in program efficiency.**
Operations	**Insertion, Deletion, Traversing, Searching, Sorting, Merging**
Algorithm	**An algorithm is a procedure for carrying out a particular task**