CSE 1201

Data Structure

Lecture 01

Instructor: Md. Shahid Uz Zaman

Abstract Data Type

ADT: It is a logical description of how we view the data and operations that are allowed without regard to how they will be implemented.

- The interface of the ADT is defined in terms of type and a set of operation on that type.
- ADT does not specify how data type is implemented.
- Internal details is hidden from the user.

Example: An ADT for a list of integers might specify the following operation

1. Insert a new integer at a particular position in the list.
2. Return true if the list is empty.
3. Return the number of integers
4. Return the maximum integer
5. Update the list
6. Sort the list

All are logical concept but to implement if physically we need data structure.

Data Structures

A data structure is a scheme for organizing data in the memory of a computer in order to use it efficiently.

Some of the more commonly used data structures include lists, arrays, stacks, queues, heaps, trees, and graphs.

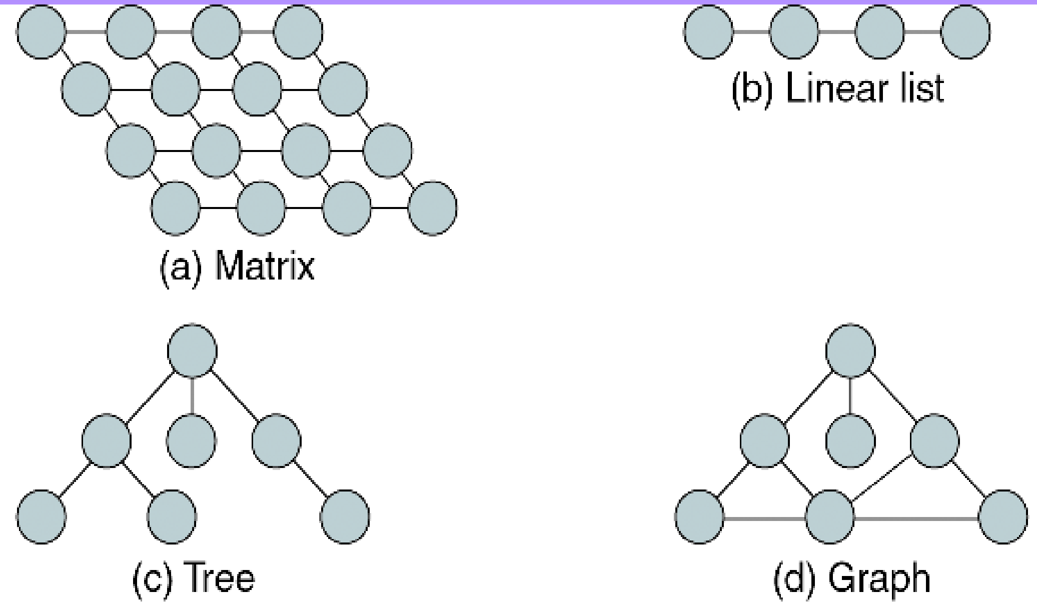
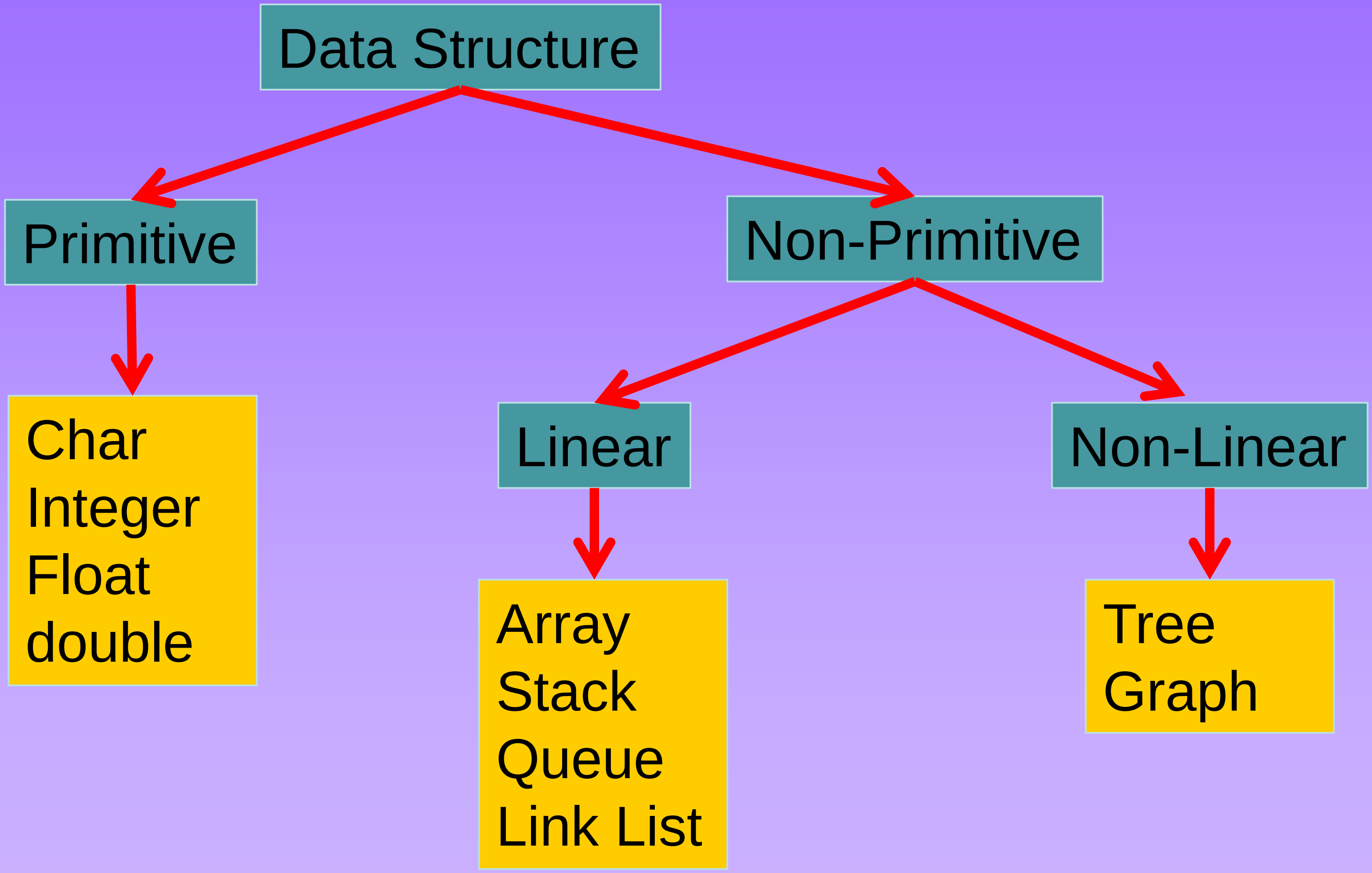


FIGURE 1-1 Some Data Structures

Data Structures: Preliminaries

- **Type**: A type is a collection of values. Example Boolean, integer, character, floats etc.
- **Data Item**: A data item is a piece of information whose value is drawn from a type.
- **Record**: Set of data item.
- **File**: Set of Records

Types Data Structures



Linear and non Linear Data Structure

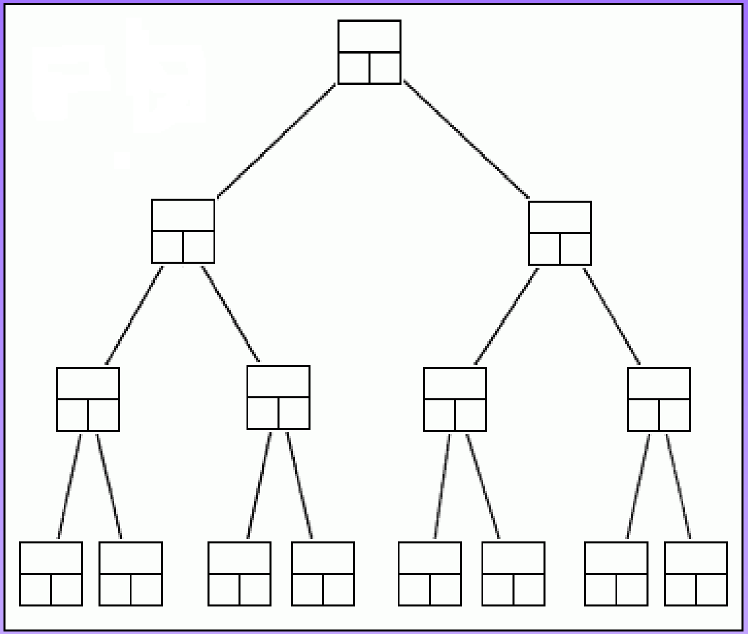
Linear data structure: A linear data structure traverses the data elements sequentially, in which only one data element can directly be reached. **Ex:** Arrays, Linked Lists

Non-Linear data structure: Every data item is attached to several other data items in a way that is specific for reflecting relationships. The data items are not arranged in a sequential structure. **Ex:** Trees, Graphs

Data Structures

The way in which the data is organized affects the performance of a program for different tasks.

Computer programmers decide which data structures to use based on the nature of the data and the processes that need to be performed on that data.



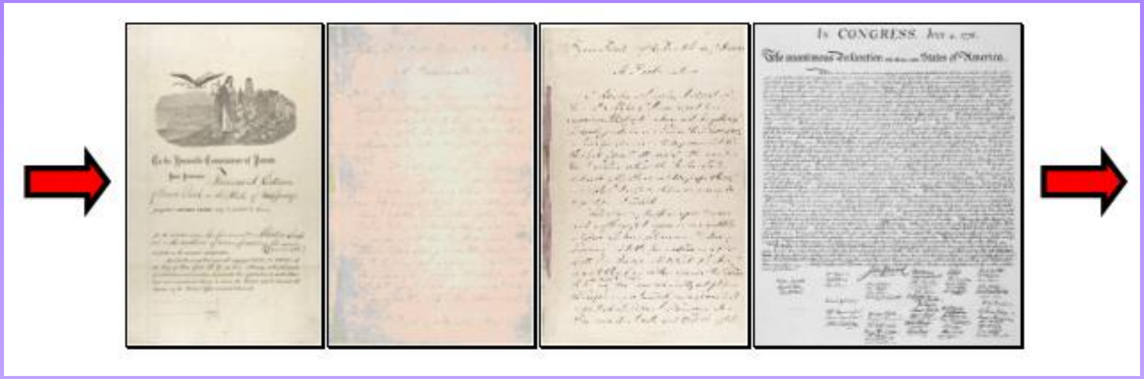
Binary Tree

Algorithms + Data Structures = Programs

Algorithms $\leftrightarrow$ Data Structures

Example: A Queue

A *queue* is an example of commonly used simple data structure. A queue has beginning and end, called the *front* and *back* of the queue.

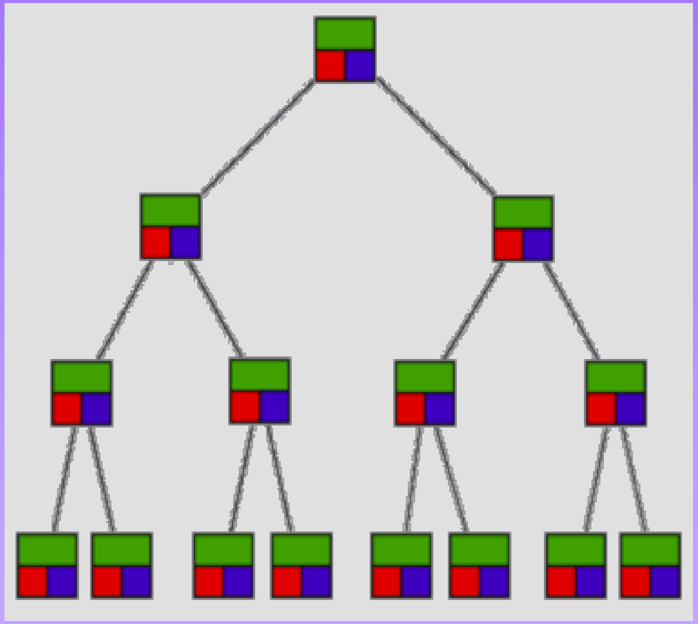


Data enters the queue at one end and leaves at the other. Because of this, data exits the queue in the same order in which it enters the queue, like people in a checkout line at a supermarket.

Example: A Binary Tree

A *binary tree* is another commonly used data structure. It is organized like an upside down tree.

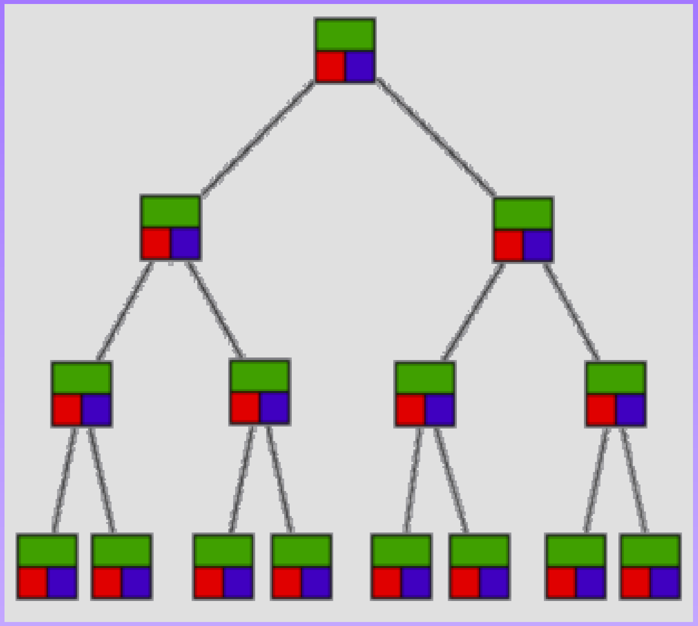
Each spot on the tree, called a *node*, holds an item of data along with a left pointer and a right pointer.



Binary Tree

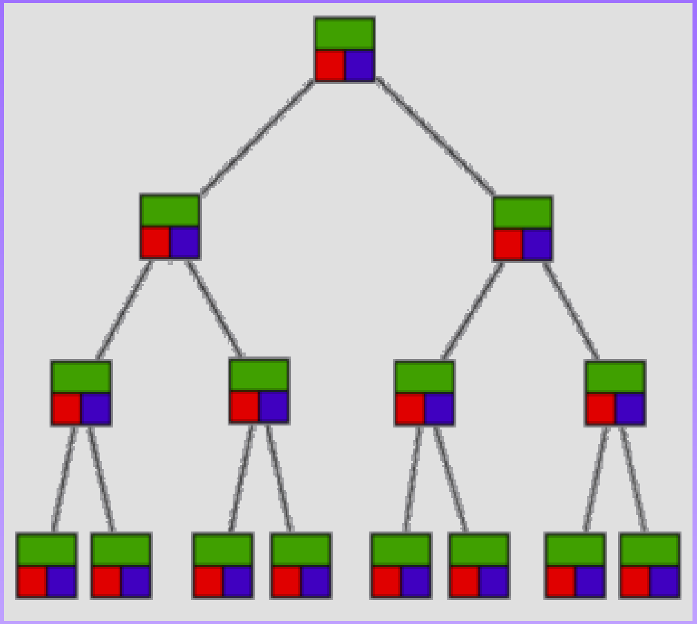
Example: A Binary Tree

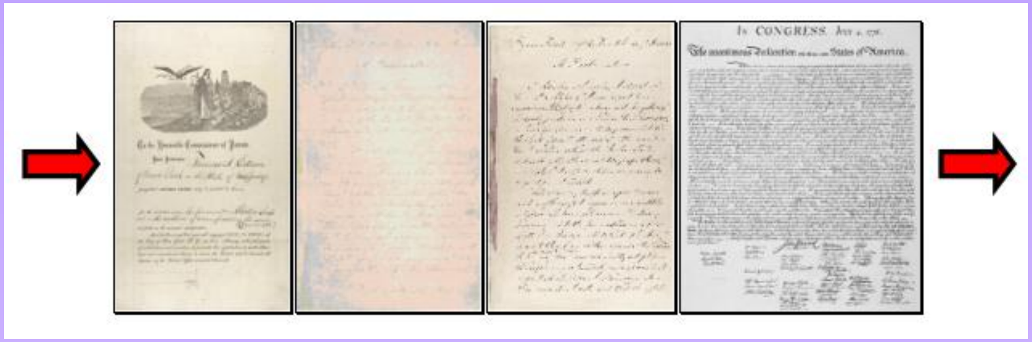
The pointers are lined up so that the structure forms the upside down tree, with a single node at the top, called the root node, and branches increasing on the left and right as you go down the tree.


Binary Tree

Choosing Data Structures

By comparing the queue with the binary tree, you can see how the structure of the data affects what can be done efficiently with the data.

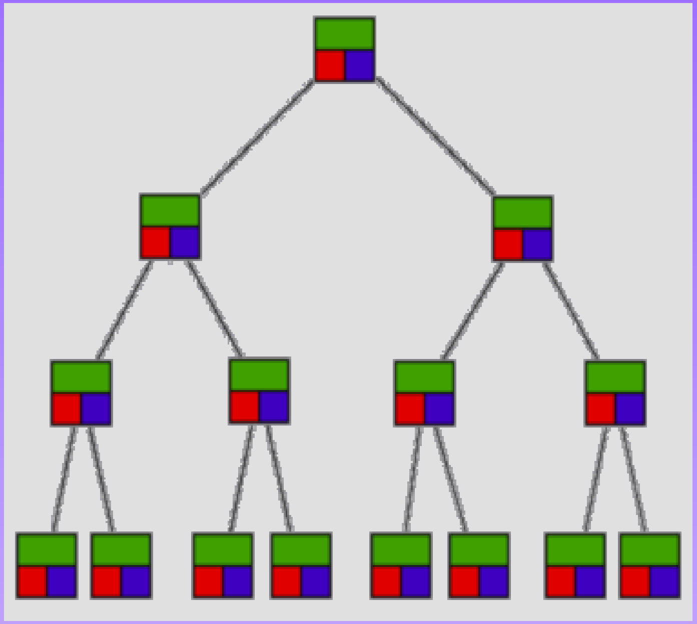


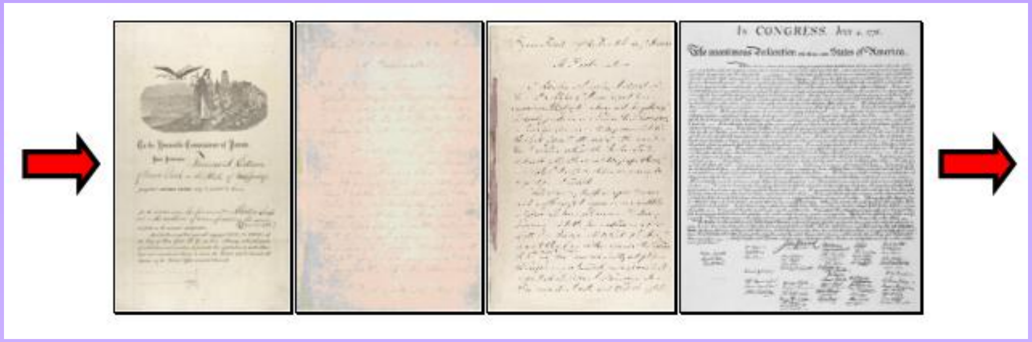


Choosing Data Structures

A queue is a good data structure to use for storing things that need to be kept in order, such as a set of documents waiting to be printed on a network printer.

.



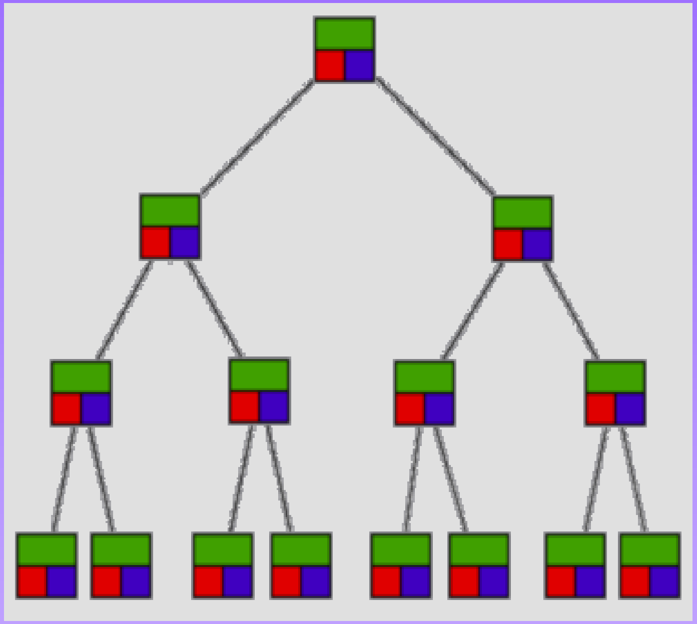


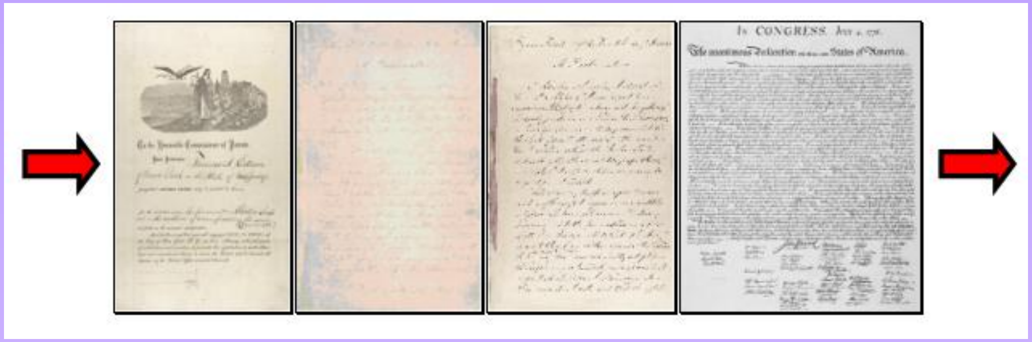
Choosing Data Structures

The jobs will be printed in the order in which they are received.

Most network print servers maintain such a *print queue*.

.

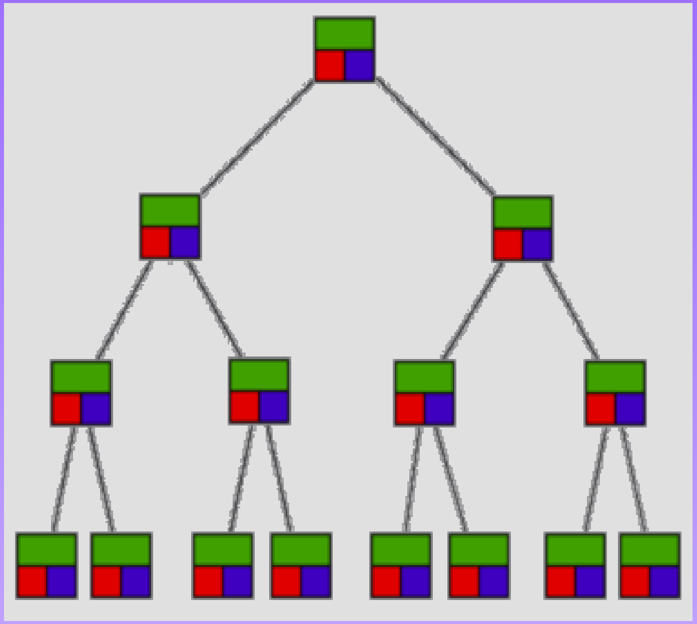


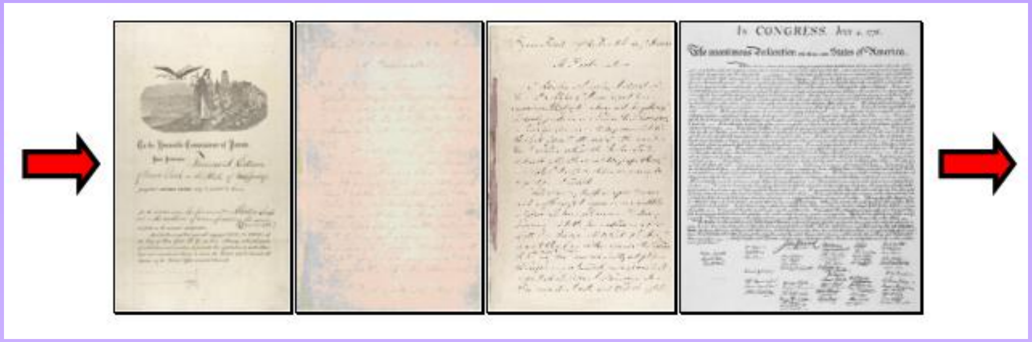


Choosing Data Structures

A binary tree is a good data structure to use for searching sorted data.

The middle item from the list is stored in the root node, with lesser items to the left and greater items to the right.



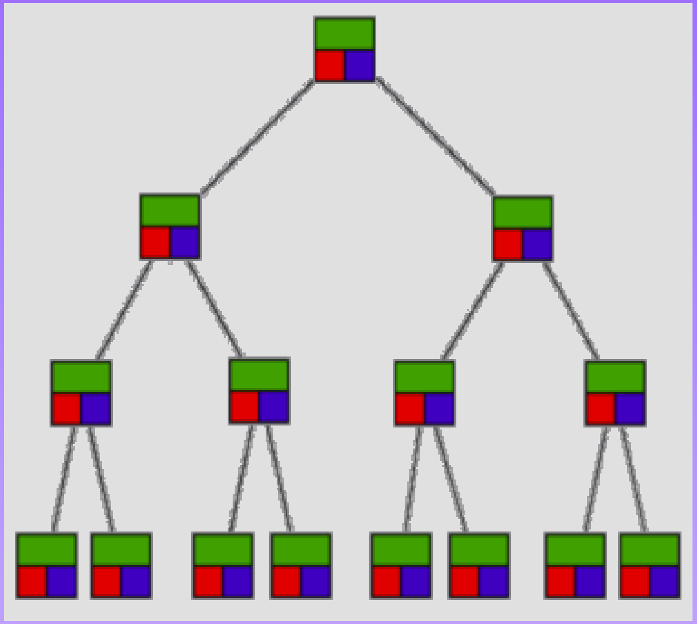


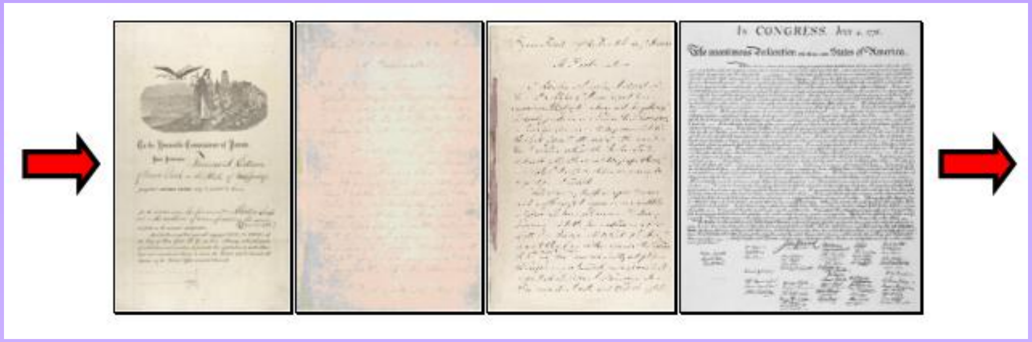
Choosing Data Structures

For some applications, a queue is the best data structure to use.

For others, a binary tree is better.

Programmers choose from among many data structures based on how the data will be used by the program.

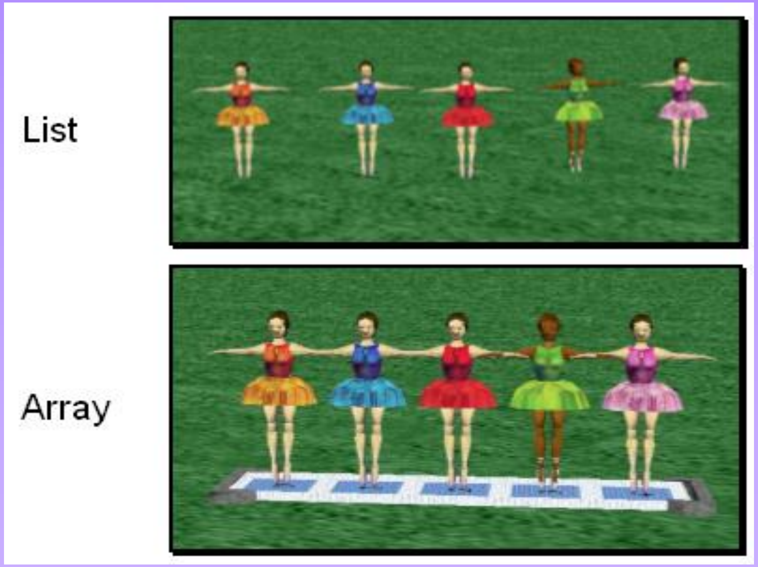




List & Arrays

List and arrays are two built-in data structures that can be used to organize data, or to create other data structures:

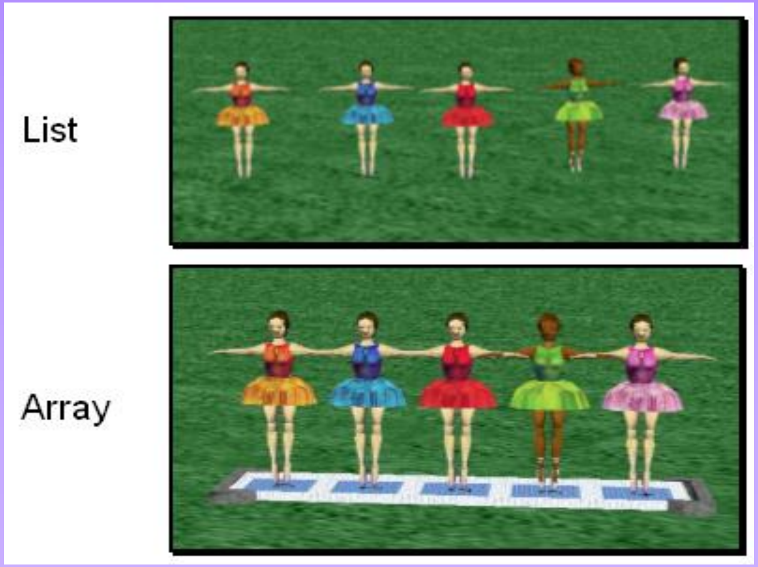
- Lists
- Arrays



List & Arrays

A list is an ordered set of data. It is often used to store objects that are to be processed sequentially.

A list can be used
 to create a queue.



List & Arrays

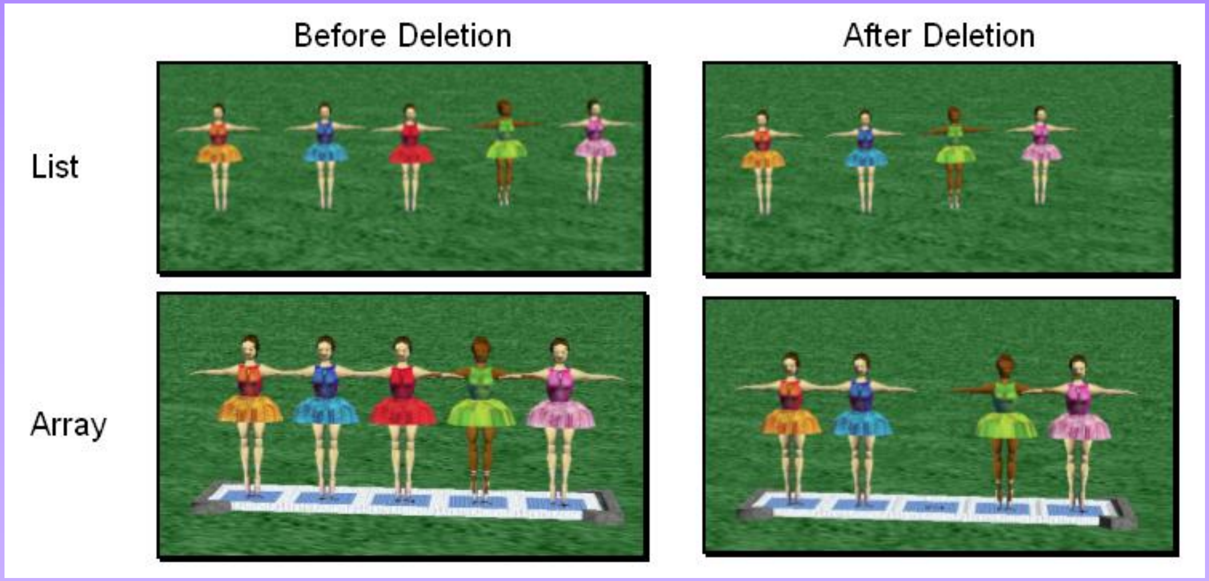
An array is an indexed set of variables, such as dancer$_{[1]}$, dancer$_{[2]}$, dancer$_{[3]}$,… It is like a set of boxes that hold things.

A list is a set of items.

An array is a set of variables that each store an item.

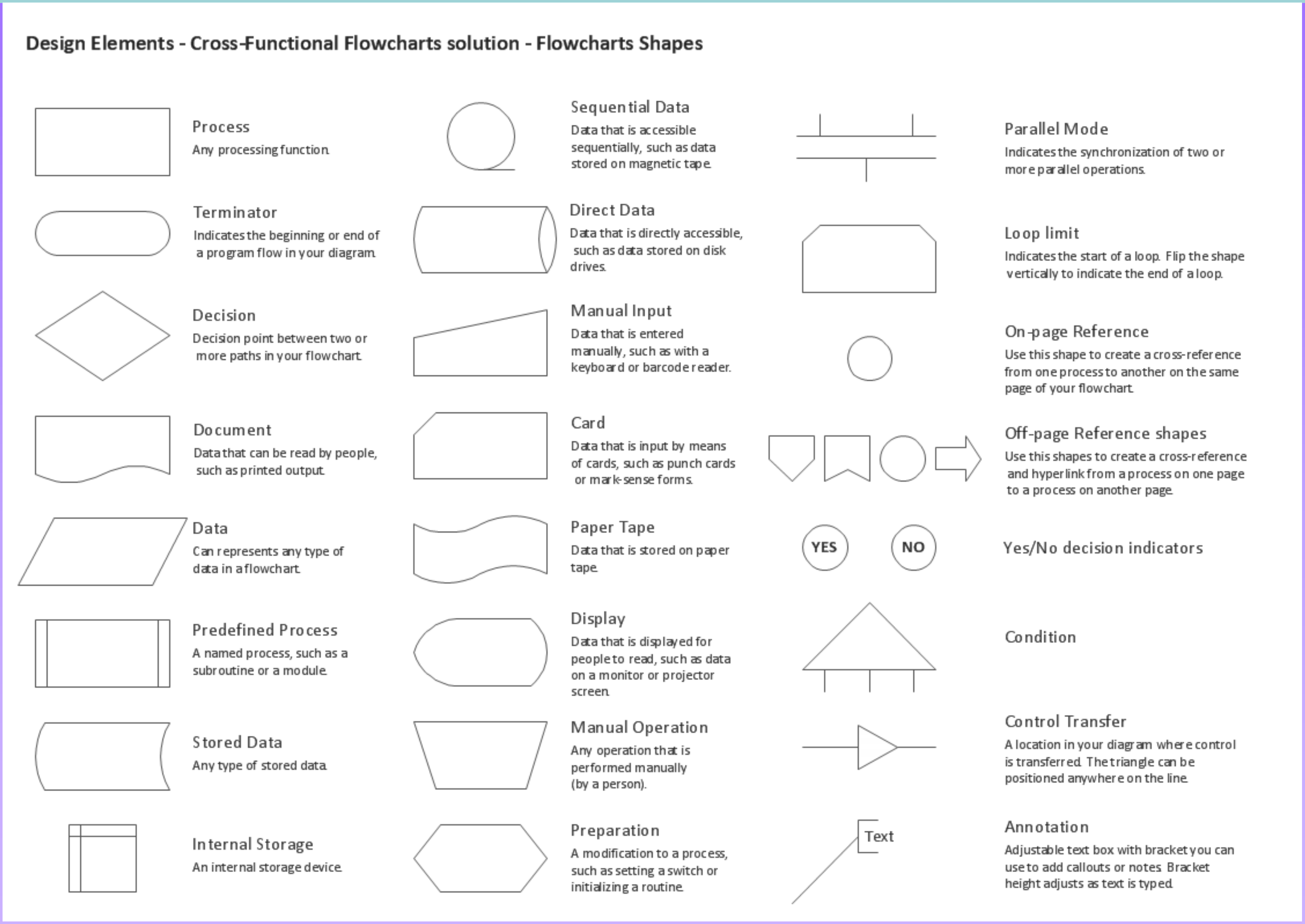
Arrays and Lists

You can see the difference between arrays and lists when you delete items.



Flow Chart

Flowchart is a pictorial presentation of an algorithm



Design Elements - Cross-Functional Flowcharts solution - Flowcharts Shapes

Process
Any processing function.

Terminator
Indicates the beginning or end of a program flow in your diagram.

Decision
Decision point between two or more paths in your flowchart.

Document
Data that can be read by people, such as printed output.

Data
Can represents any type of data in a flowchart.

Predefined Process
A named process, such as a subroutine or a module.

Stored Data
Any type of stored data.

Internal Storage
An internal storage device.

Sequential Data
Data that is accessible sequentially, such as data stored on magnetic tape.

Direct Data
Data that is directly accessible, such as data stored on disk drives.

Manual Input
Data that is entered manually, such as with a keyboard or barcode reader.

Card
Data that is input by means of cards, such as punch cards or mark-sense forms.

Paper Tape
Data that is stored on paper tape.

Display
Data that is displayed for people to read, such as data on a monitor or projector screen.

Manual Operation
Any operation that is performed manually (by a person).

Preparation
A modification to a process, such as setting a switch or initializing a routine.

Parallel Mode
Indicates the synchronization of two or more parallel operations.

Loop limit
Indicates the start of a loop. Flip the shape vertically to indicate the end of a loop.

On-page Reference
Use this shape to create a cross-reference from one process to another on the same page of your flowchart.

Off-page Reference shapes
Use this shapes to create a cross-reference and hyperlink from a process on one page to a process on another page.

Yes/No decision indicators

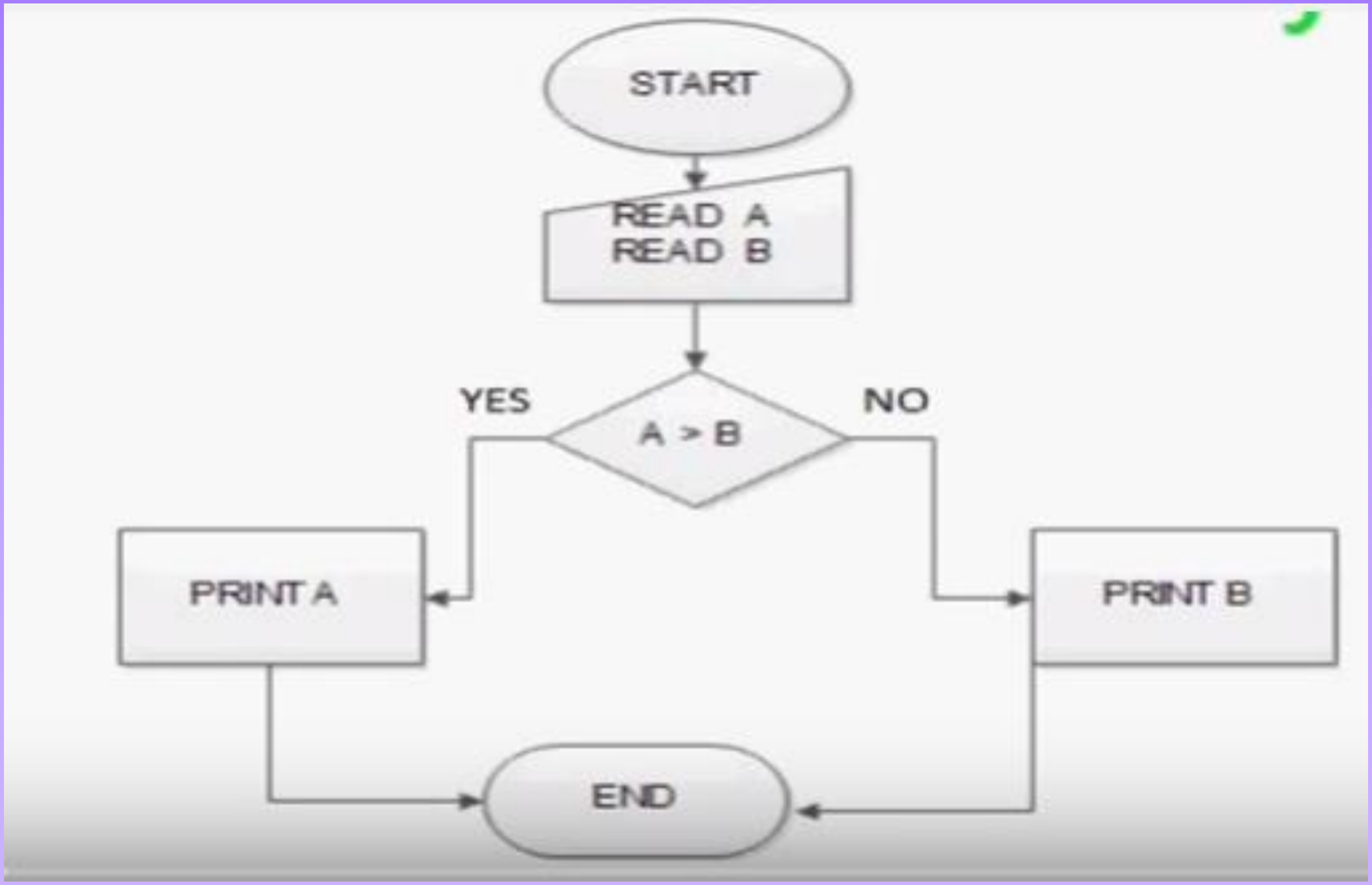
Condition

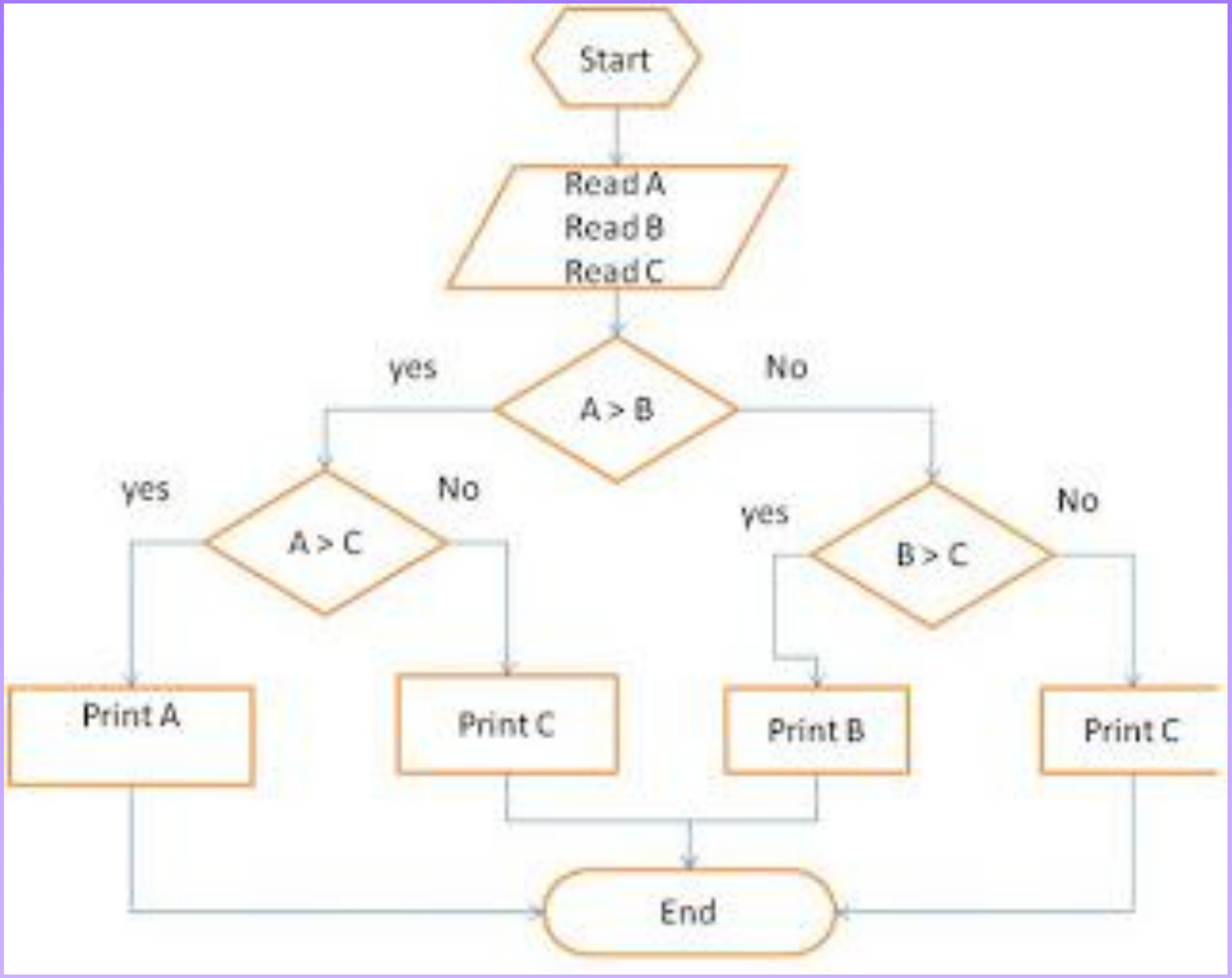
Control Transfer
A location in your diagram where control is transferred. The triangle can be positioned anywhere on the line.

Annotation
Adjustable text box with bracket you can use to add callouts or notes. Bracket height adjusts as text is typed.

Examples

Prob: Find the bigger between two numbers.




Start
Read A
Read B
Read C
A > B
yes
No
A > C
yes
No
B > C
yes
No
Print A
Print C
Print B
Print C
End