Introduction to Data Structures

Instructor: Maninder Kaur Tatla
Email: maninder.kaur2@sheridancollege.ca
Course: PROG20799

INTRODUCTION

- It is important to understand the concept of Information and how it is organized or how it can be utilized.
- What is Information?
  - If we arrange some data in an appropriate sequence, then it forms a Structure and gives us a meaning. This meaning is called Information
  - The basic unit of Information in Computer Science is a bit, Binary Digit

- So, we found two things in Information:
  - One is Data and the other is Structure .

Data Structures

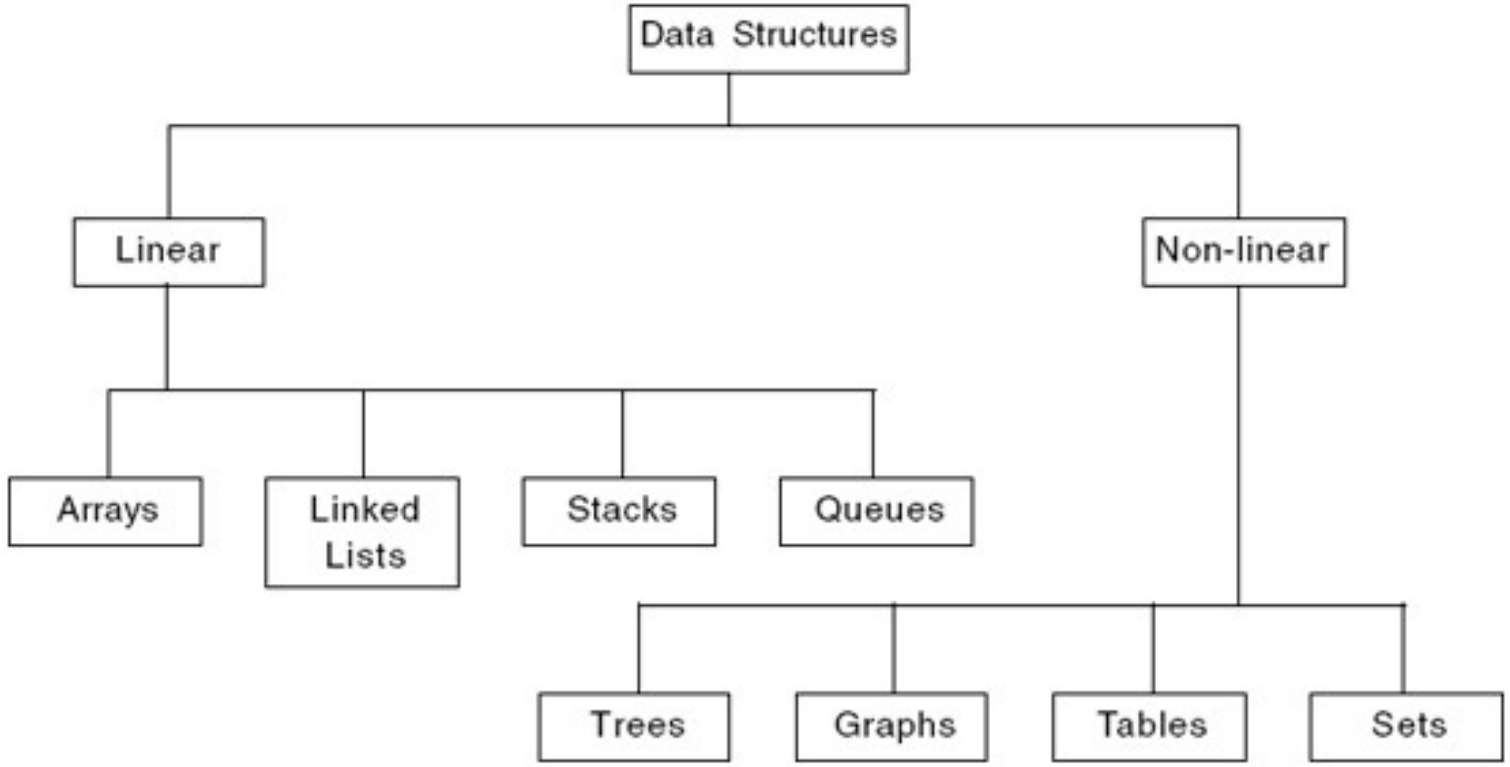
- Data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently.
- The way in which the data is organized affects the performance of a program for different tasks.
- Computer programmers decide which data structures to use based on:
  - the nature of the data and
  - the processes that need to be performed on that data.

Types of Data Structures

- **Linear:**
  - In linear data structures, values are arranged in linear fashion.
  - Arrays, linked lists, stacks and queues are examples of linear data structures in which values are stored in a sequence.

- **Non-Linear:**
  - The data values in this structure are not arranged in order or which are not in sequence
  - Tree, graph, table and sets are examples of non-linear data structures.

Types of Data Structures



Arrays

- Array means collection.
- An array is used to store elements of the same type.
- It stores data elements in contiguous memory locations.
- Array is a *linear* and *homogenous* data structure.
  - Homogenous means that the same types of elements are stored in it.

What is an Array?

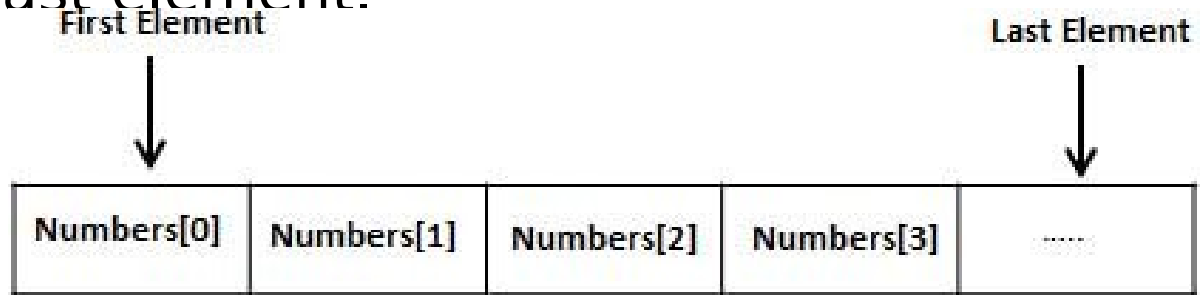
- ***"A linear array is a collection of related data items with similar data type and having a common name".***
- This means that array can store either all integers, all floating points, all characters or any other complex data type, but all of the same data type.
- Each element of an array is referenced by a subscripted variable called **Index**.

Types of Array

- **One-dimensional Array:**
  - If **single subscript** is required to reference an element, then the array is known as one-dimensional array.
  - One-dimensional arrays are also known as ***linear arrays***.
- **Two-dimensional Array:**
  - If **two subscripts** are required to reference an element, then the array is known as two-dimensional array**.**
- **Multi-dimensional Array:**
  - The arrays whose elements are referenced by **two or more subscripts** are called multidimensional arrays.

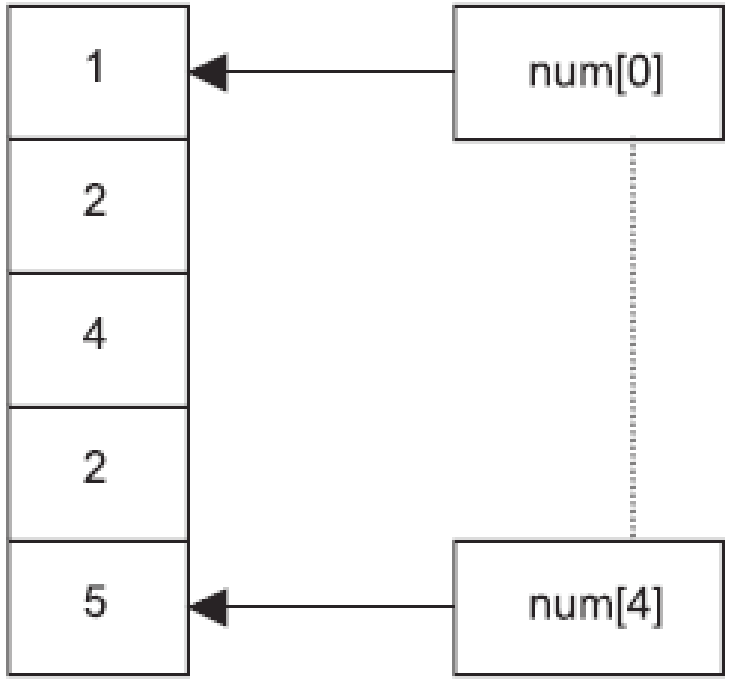
One-Dimensional Array

- A specific element in an array is accessed by an ***index***.
- All arrays consist of contiguous memory locations.
- The lowest address corresponds to the first element and the highest address to the last element.



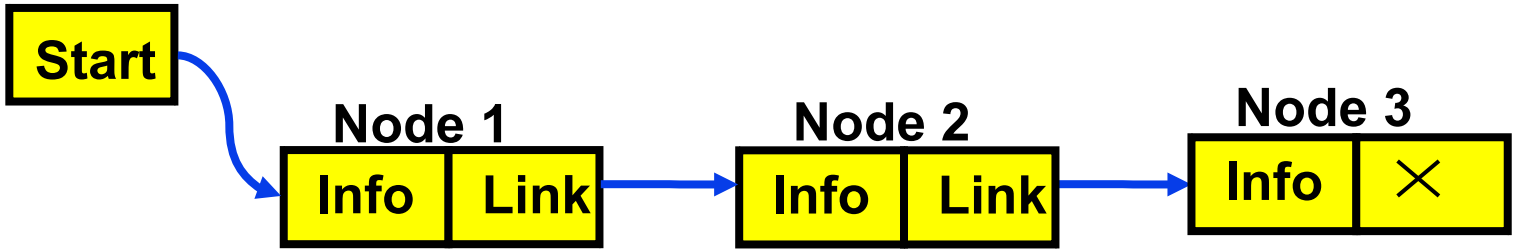
- In **C, C++** and **Java**, if array size is `n`, then the index set consists of `0, 1, 2, 3,……, n-1`.
- The elements of an array `A` are denoted by bracket notation like `A[0], A[1], A[2], A[3], A[4],……, A[N-1]`.

Arrays



Introduction

- A linked list is a linear collection of data elements, called ***nodes***, where the linear order is given by means of pointers.
- Each node is divided into two parts:
  - The first part contains the information (Info) of the element, and
  - The second part, called the link field or next pointer field, contains the address of the next node in the list.
- The pointer of the last node contains a special value, called the null pointer.
- A special pointer variable – called START contains the address of the first node.
- A special case is the list that has no nodes, such a list is called the null list or empty list and is denoted by the null pointer in the variable START.



Linked list with 3 nodes

Advantages of Linked Lists

- The items do ***not*** have to be stored in consecutive memory locations: the successor can be anywhere physically.
  - So, can be inserted and deleted items without shifting data.
  - Can increase the size of the data structure easily.
- Linked lists can grow ***dynamically*** (i.e. at run time):
  - The amount of memory space allocated can grow and shrink as needed.
- Insertion and deletion of nodes is quicker with linked lists.

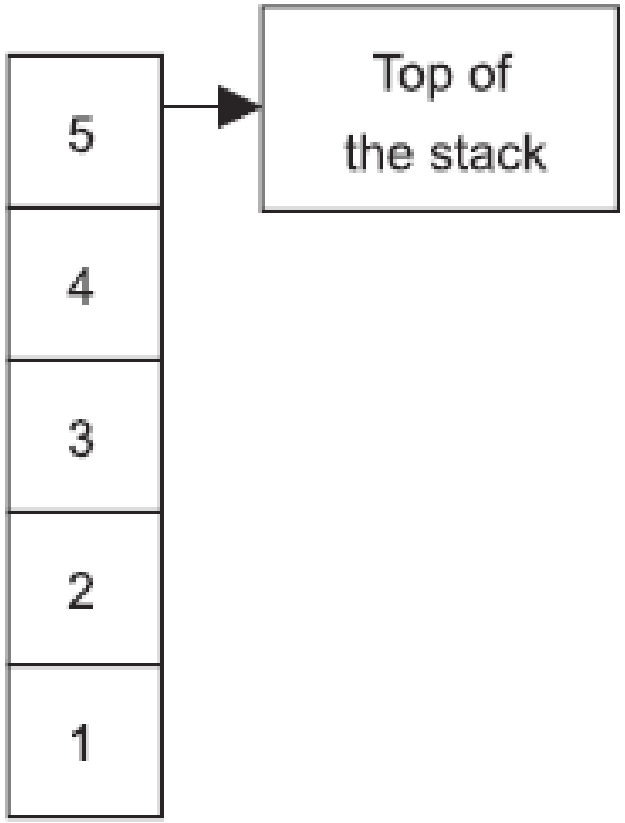
Types of Linked Lists

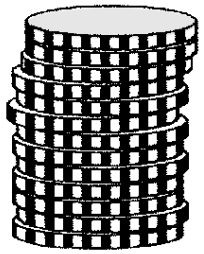
- **Singly Linked List (One-Way Linked List):**
  - Begins with a pointer to the first node
  - Terminates with a null pointer
  - Only traversed in one direction
- **Circular, Singly Linked:**
  - Pointer in the last node points back to the first node
- **Doubly Linked List (Two-Way Linked List):**
  - Two “start pointers” – first element and last element
  - Each node has a forward pointer and a backward pointer
  - Allows traversals both forwards and backwards
- **Circular, Doubly Linked List:**
  - Forward pointer of the last node points to the first node and backward pointer of the first node points to the last node

Stacks

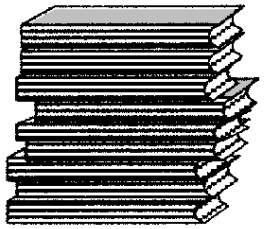
- A stack is an arranged collection of elements into which
  - new elements can be inserted
  - or from which existing elements can be deleted at one end.
- Stack is a set of elements in a *last-in-first-out* (LIFO) technique.
- The last item pushed onto the stack is always the first to be removed from the stack.
- The insertion of element onto the stack is called as ***push*** and deletion operation is called ***pop***.
- The end of the stack from where the insertion or deletion operation is carried out is called ***top***.

Stacks

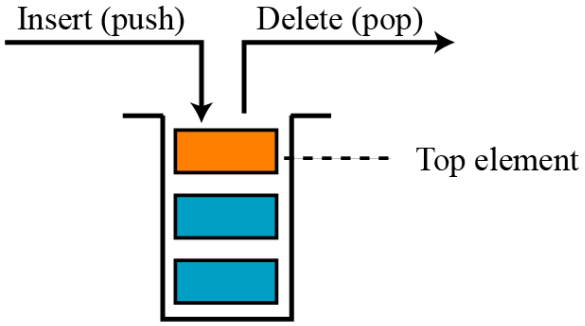




Stack of coins



Stack of books

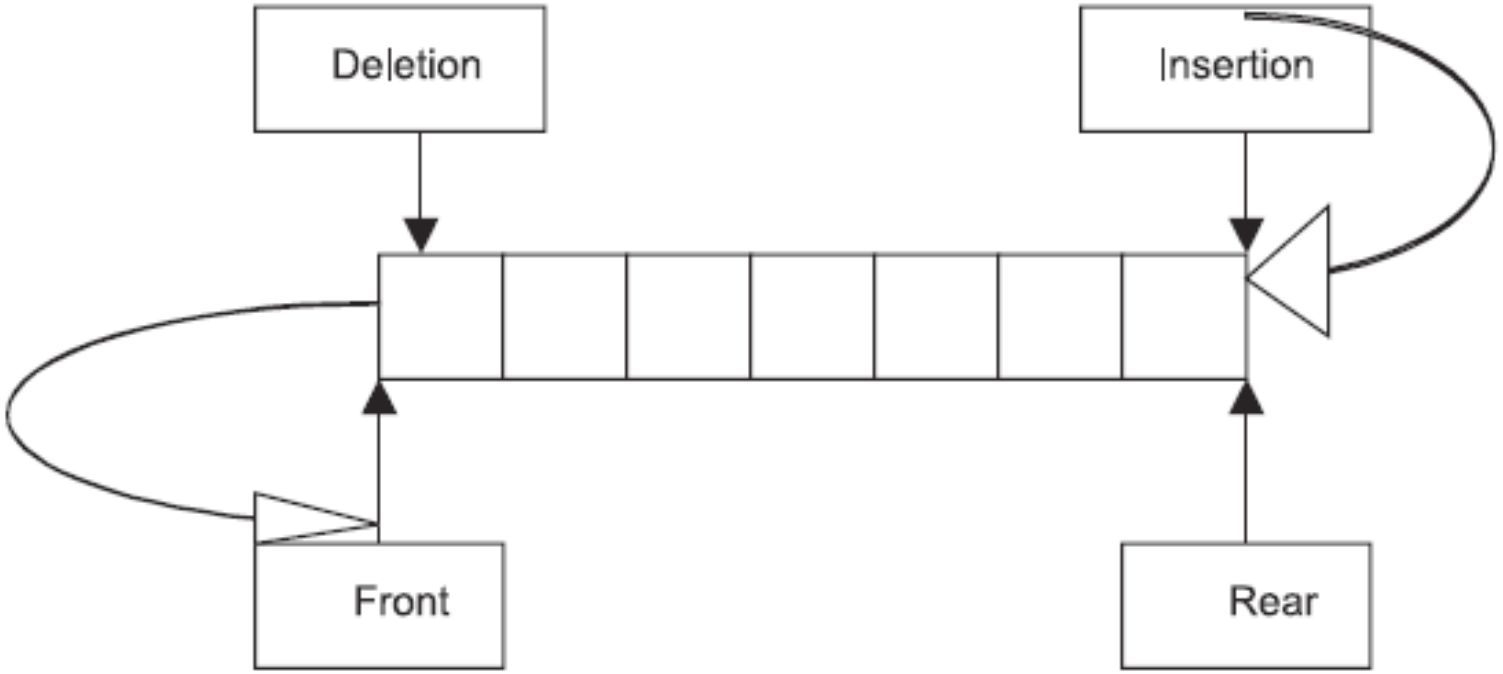


Computer stack

Queues

- Queue is a collection of elements in which
  - elements are inserted at one end called ***rear*** end
  - and elements are deleted at other end called ***front*** end.
- The first entry in a queue to which the service is offered is to the element that is on front.
- After servicing, it is removed from the queue.
- The information is manipulated in the same sequence as it was collected.
- Queue follows the rule ***first-in-first-out*** (FIFO).

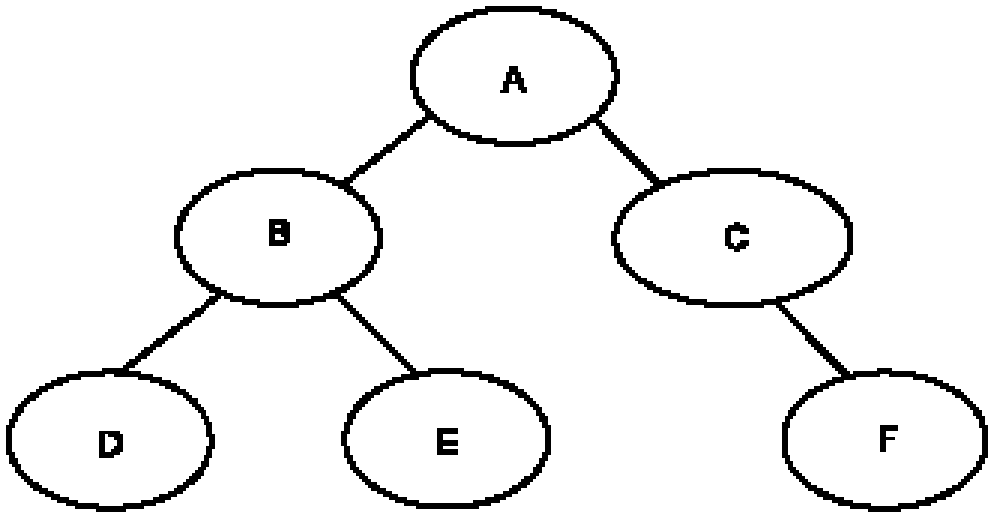
Queues



Trees

- Data frequently contains a hierarchical relationship between various elements.
- The data structure which reflects this relationship is called ***tree***.
- In tree, elements are arranged in non-linear fashion.
- Tree has several practical applications:
  - It is immensely useful in manipulating data and to protect hierarchical relationship among data.
  - The fundamental operations such as insertion, deletion etc. is easy and efficient in tree data structure than in linear data structures.

Trees

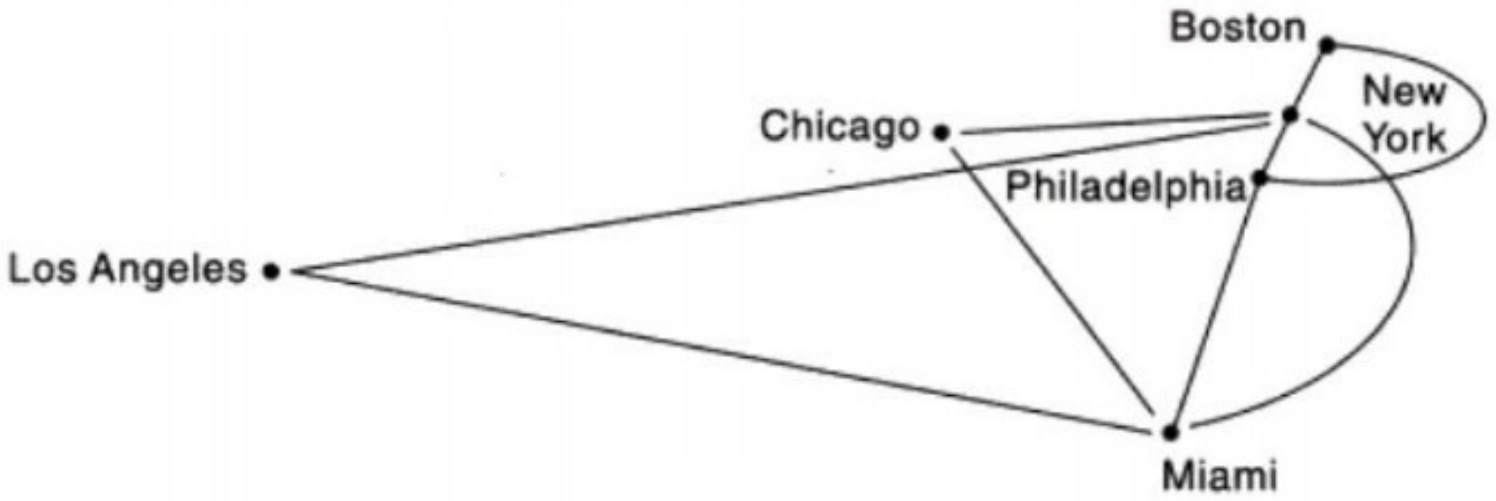


Graphs

- Sometimes, data contains a relationship between pairs of elements which is not necessarily hierarchical in nature.
- The data structure which reflects this type of relationship is called a ***graph***.

Graphs

- For example:
  - Suppose an airline flies only between the cities connected by lines.



Operations on Data Structures

- **Traversing:** Access and process every data in data structure at least once.
- **Searching:** Search for a location of data.
- **Insertion:** Insert item in the list of data.
- **Deletion:** Delete item from a set of data.
- **Sorting:** Sort data in certain order.
- **Merging:** Merge multiple group of data.

Algorithm

- **Algorithm**: a step-by-step procedure for performing a task within a finite period of time.
- **Algorithms** often operate on a collection of data, which is stored in a structured way in the computer memory (*Data Structure*).
- It is problem solving using logic.

Data Structures and Algorithms

- A *data structure* is a systematic way of organizing and accessing data.
- 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 and vice-versa.

Any questions please?