Trees

- A tree can be defined as finite set of data items (nodes).
- Tree is non-linear type of data structure in which data items are arranged or stored in a sorted sequence.
- Tree represent the hierarchical relationship between various elements.

Asymptotic Bounds and Algorithms

- In all of the examples so far, we have assumed we knew the exact running time of the algorithm.
- In general, it may be very difficult to determine the exact running time.
- Thus, we will try to determine a bounds without computing the exact running time.

  - **Example:** What is the complexity of the following algorithm?

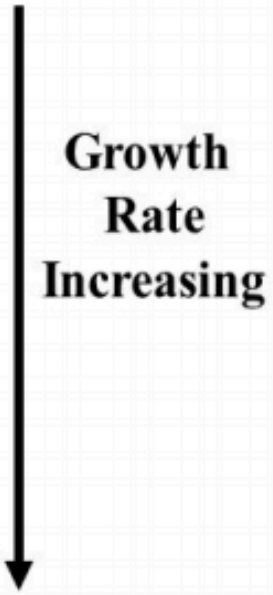
```
for (i = 0; i < n; i ++)
for (j = 0; j < n; j ++)
a[i][j] = b[i][j] * x;
```

Answer: $O(n^2)$

  - We will see more examples later.

Rates of Growth

constant	$\theta(n^0) = \theta(1)$
logarithmic	$\theta(\lg n)$
linear	$\theta(n)$
<"en log en">	$\theta(n \lg n)$
quadratic	$\theta(n^2)$
cubic	$\theta(n^3)$
polynomial	$\theta(n^k), k \geq 1$
exponential	$\theta(a^n), a > 1$



Basic Terminology: Elementary Data Organization

- **Data:** Data are simply values or sets of values.
- **Data items:** Data items refers to a single unit of values.
  - Data items that are divided into sub-items are called **Group items**.
    - **Ex:** An Employee Namemay be divided into three subitems-first name, middle name, and last name.
  - Data items that are not able to divide into sub-items are called **Elementary items**.
    - **Ex:** SSN

Basic Terminology: Elementary Data Organization

- **Entity:** An entity is something that has certain attributes or properties which may be assigned values. The values may be either numeric or non-numeric.
  - **Ex:**

Attributes	Names	Age	Sex	SSN
Values	Rohit	25	M	123-88

  - Entities with similar attributes form an **entity set**. Each attribute of an entity set has a range of values, the set of all possible values that could be assigned to the particular attribute. The term "information" is sometimes used for data with given attributes, of, in other words meaningful or processed data.

Basic Terminology: Elementary Data Organization

- **Field** is a single elementary unit of information representing an attribute of an entity.
- **Record** is the collection of field values of a given entity.
- **File** is the collection of records of the entities in a given entity set.

Basic Terminology: Elementary Data Organization

- Data types
- Variable
- Program
- Objects

Definition

- Data structure is representation of the logical relationship existing between individual elements of data.

- In other words, a data structure is a way of organizing all data items that considers not only the elements stored but also their relationship to each other.

Introduction

- Data structure affects the design of both structural & functional aspects of a program.

Program = algorithm + Data Structure

- You know that a algorithm is a step by step procedure to solve a particular function.

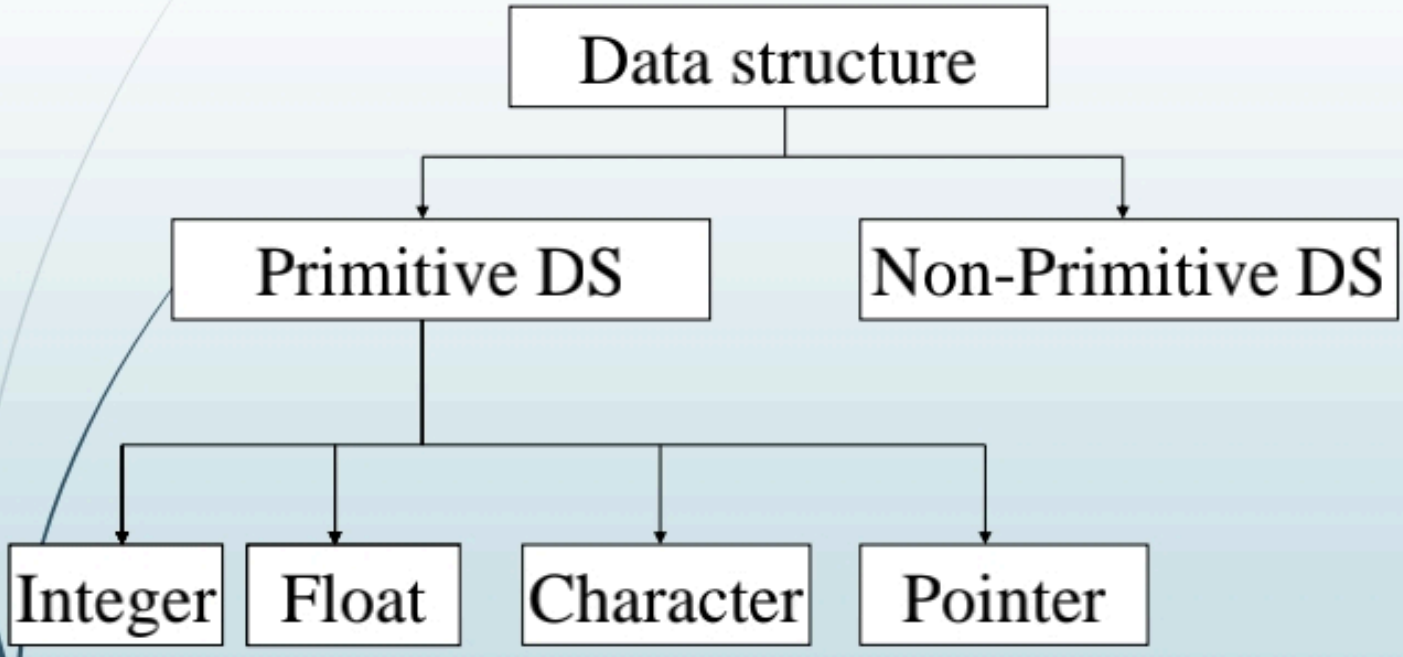
Introduction

- That means, algorithm is a set of instruction written to carry out certain tasks & the data structure is the way of organizing the data with their logical relationship retained.
- To develop a program of an algorithm, we should select an appropriate data structure for that algorithm.
- Therefore algorithm and its associated data structures from a program.

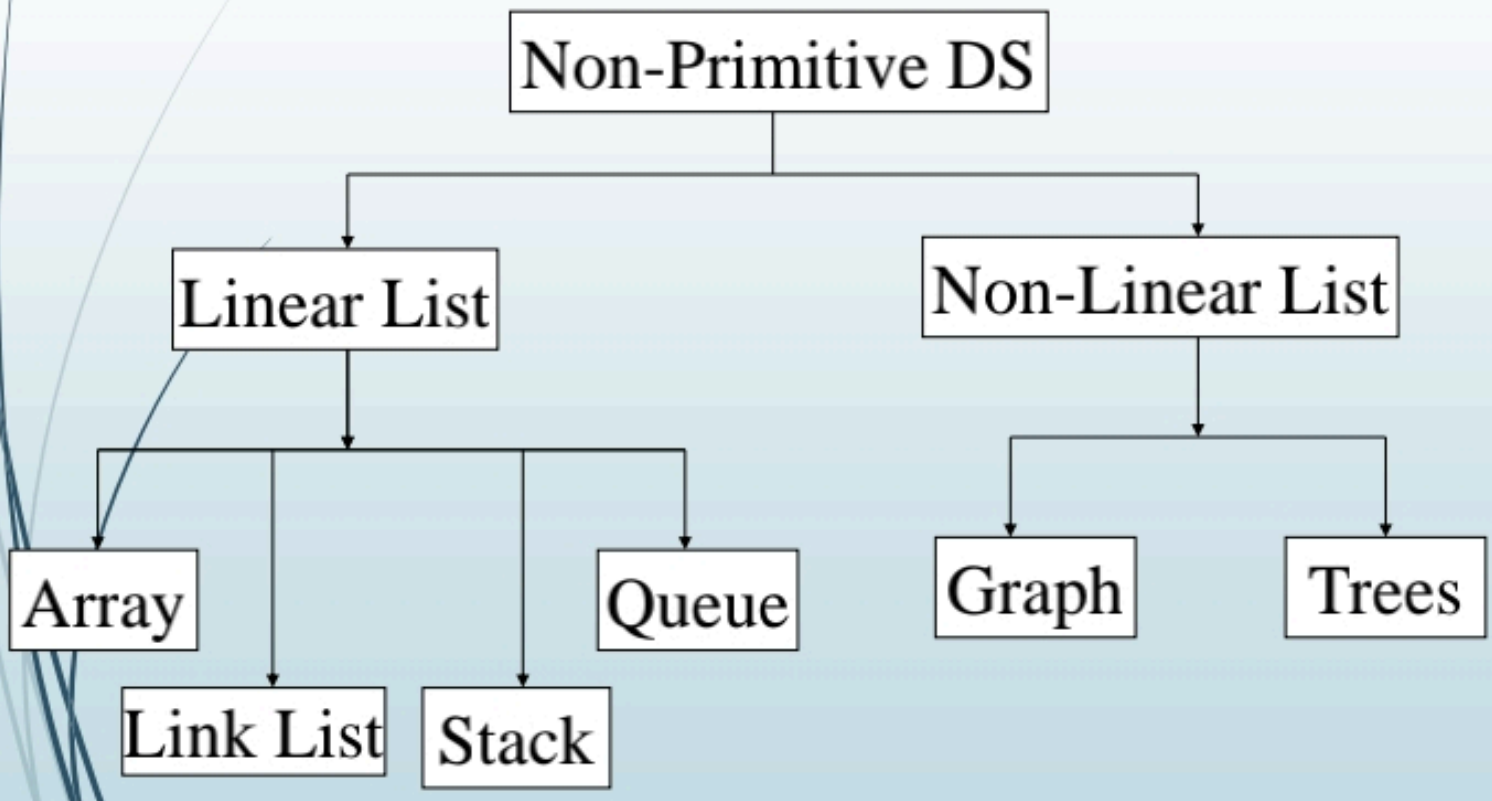
Classification of Data Structure

- Data structure are normally divided into two broad categories:
  - Primitive Data Structure
  - Non-Primitive Data Structure

Classification of Data Structure



Classification of Data Structure



Primitive Data Structure

- There are basic structures and directly operated upon by the machine instructions.
- In general, there are different representation on different computers.
- Integer, Floating-point number, Character constants, string constants, pointers etc, fall in this category.

Non-Primitive Data Structure

- There are more sophisticated data structures.
- These are derived from the primitive data structures.
- The non-primitive data structures emphasize on structuring of a group of homogeneous (same type) or heterogeneous (different type) data items.

Non-Primitive Data Structure

- Lists, Stack, Queue, Tree, Graph are example of non-primitive data structures.
- The design of an efficient data structure must take operations to be performed on the data structure.

Non-Primitive Data Structure

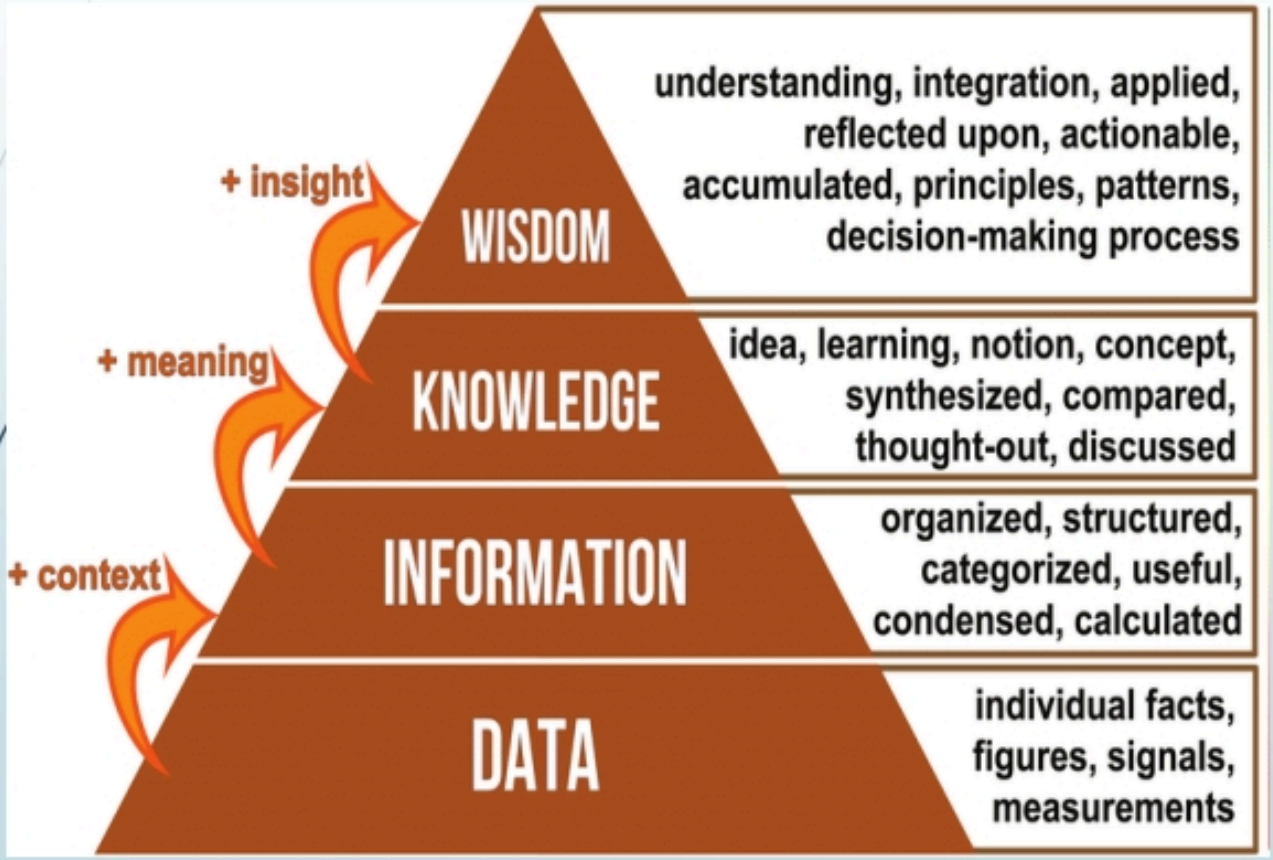
- The most commonly used operation on data structure are broadly categorized into following types:
  - Create
  - Selection
  - Updating
  - Searching
  - Sorting
  - Merging
  - Destroy or Delete

Different between them

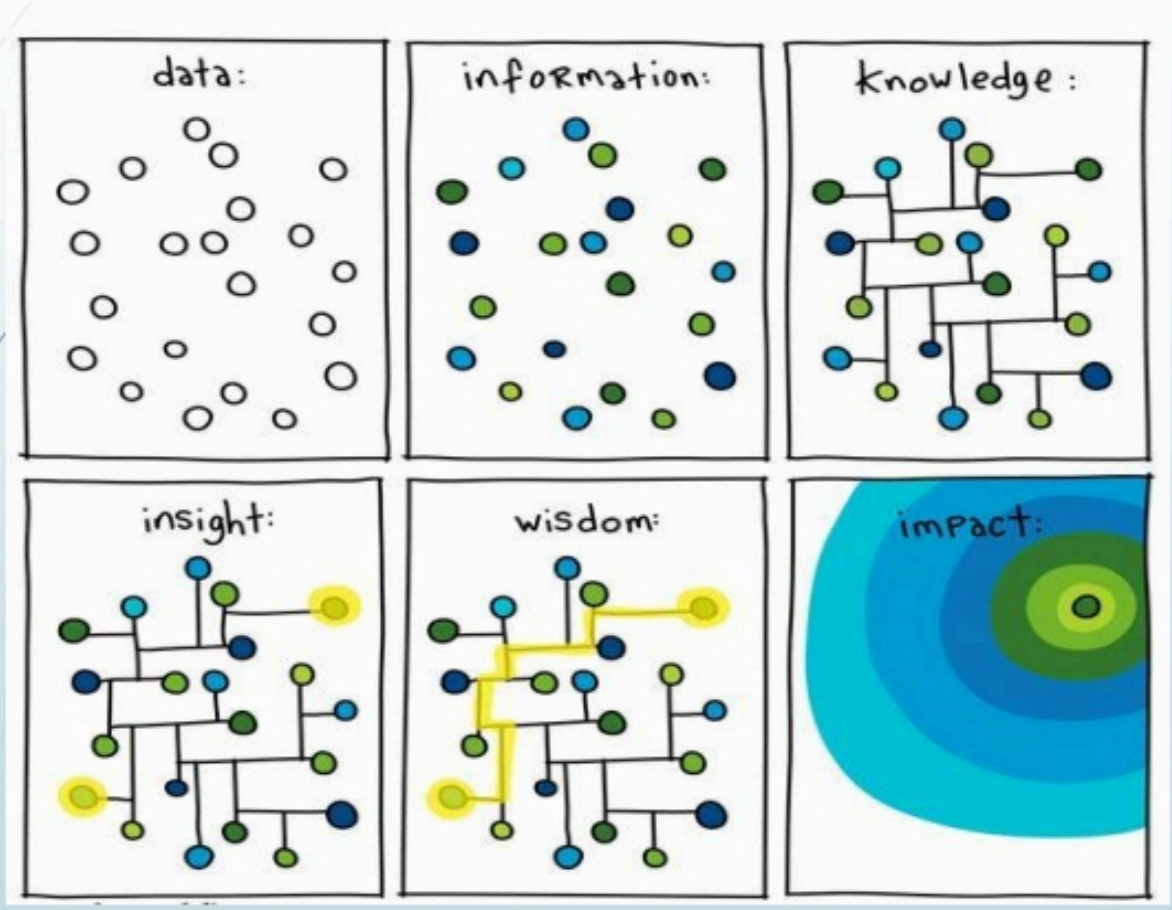
- A primitive data structure is generally a basic structure that is usually built into the language, such as an integer, a float.

- A non-primitive data structure is built out of primitive data structures linked together in meaningful ways, such as a or a linked-list, binary search tree, AVL Tree, graph etc.

Data at Glance



Data at Glance



Description of various Data Structures : Arrays

- An array is defined as a set of finite number of homogeneous elements or same data items.

- It means an array can contain one type of data only, either all integer, all float-point number or all character.

Arrays

- Simply, declaration of array is as follows:

```
int arr[10]
```

- Where int specifies the data type or type of elements arrays stores.

- “arr” is the name of array & the number specified inside the square brackets is the number of elements an array can store, this is also called sized or length of array.

Arrays

- Following are some of the concepts to be remembered about arrays:
  - The individual element of an array can be accessed by specifying name of the array, following by index or subscript inside square brackets.
  - The first element of the array has index zero[0]. It means the first element and last element will be specified as: arr[0] & arr[9]

    Respectively.

Arrays

- The elements of array will always be stored in the consecutive (continues) memory location.

- The number of elements that can be stored in an array, that is the size of array or its length is given by the following equation:

(Upperbound-lowerbound)+1

Arrays

- For the above array it would be

  (9-0)+1=10, where 0 is the lower bound of array and 9 is the upper bound of array.

- Array can always be read or written through loop. If we read a one-dimensional array it require one loop for reading and other for writing the array.

Arrays

- For example: Reading an array

```
For(i=0;i<=9;i++)
    scanf("%d",&arr[i]);
```

- For example: Writing an array

```
For(i=0;i<=9;i++)
    printf("%d",arr[i]);
```

Arrays

- If we are reading or writing two-dimensional array it would require two loops. And similarly the array of a N dimension would required N loops.

- Some common operation performed on array are:
  - Creation of an array
  - Traversing an array

Arrays

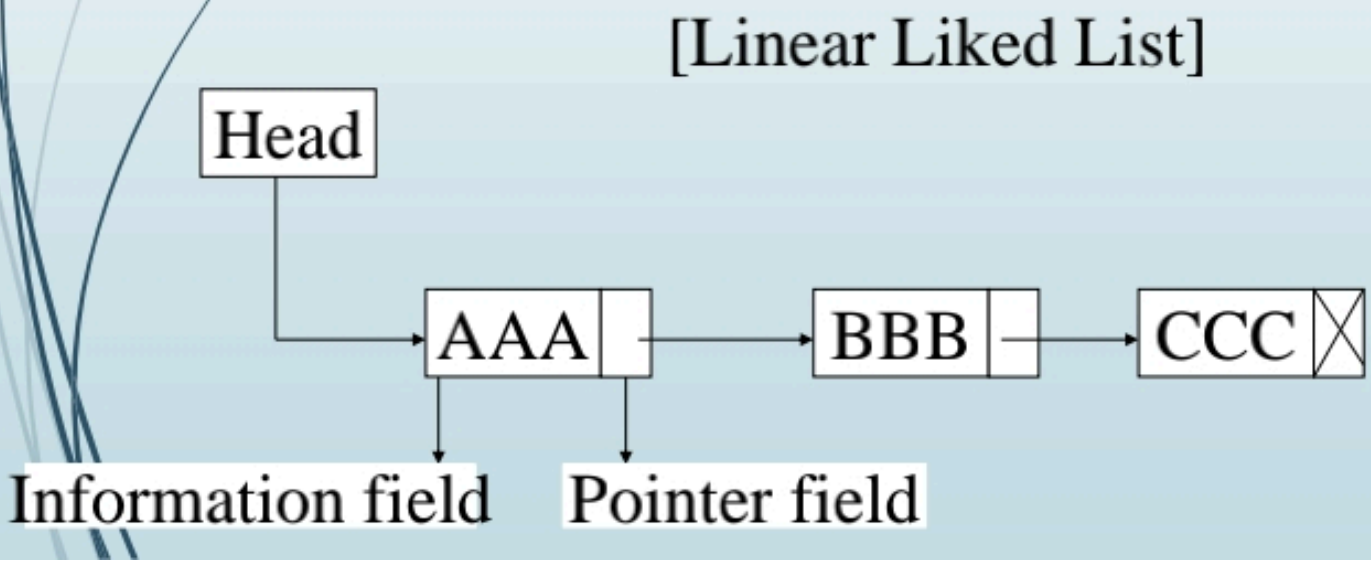
- Insertion of new element
- Deletion of required element
- Modification of an element
- Merging of arrays

Lists

- A lists (Linear linked list) can be defined as a collection of variable number of data items.
- Lists are the most commonly used non-primitive data structures.
- An element of list must contain at least two fields, one for storing data or information and other for storing address of next element.
- As you know for storing address we have a special data structure of list the address must be pointer type.

Lists

- Technically each such element is referred to as a node, therefore a list can be defined as a collection of nodes as show bellow:



Lists

- Types of linked lists:
  - Single linked list
  - Doubly linked list
  - Single circular linked list
  - Doubly circular linked list

Stack

- A stack is also an ordered collection of elements like arrays, but it has a special feature that deletion and insertion of elements can be done only from one end called the top of the stack (TOP)

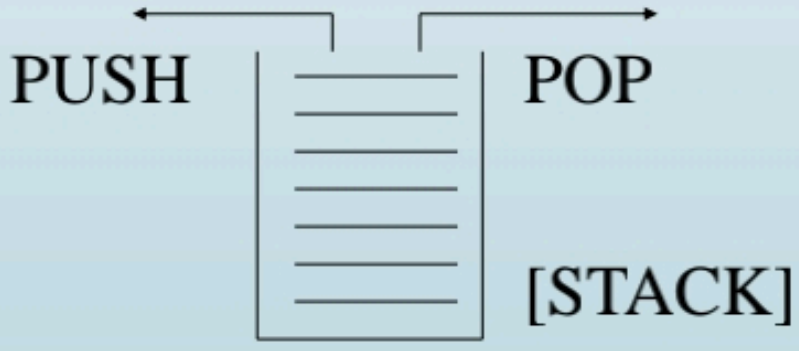
- Due to this property it is also called as last in first out type of data structure (LIFO).

Stack

- It could be through of just like a stack of plates placed on table in a party, a guest always takes off a fresh plate from the top and the new plates are placed on to the stack at the top.
- It is a non-primitive data structure.
- When an element is inserted into a stack or removed from the stack, its base remains fixed where the top of stack changes.

Stack

- Insertion of element into stack is called PUSH and deletion of element from stack is called POP.
- The bellow show figure how the operations take place on a stack:



Stack

- The stack can be implemented into two ways:
  - Using arrays (Static implementation)
  - Using pointer (Dynamic implementation)

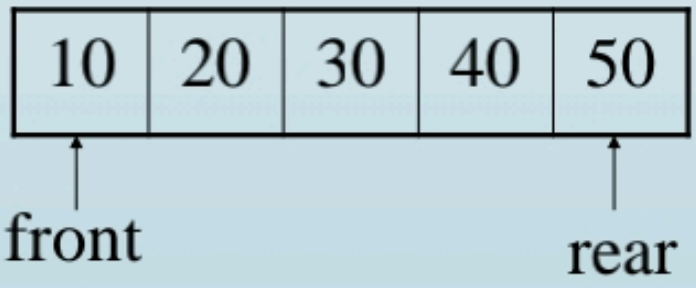
Queue

- Queue are first in first out type of data structure (i.e. FIFO)
- In a queue new elements are added to the queue from one end called REAR end and the element are always removed from other end called the FRONT end.
- The people standing in a railway reservation row are an example of queue.

Queue

- Each new person comes and stands at the end of the row and person getting their reservation confirmed get out of the row from the front end.

- The bellow show figure how the operations take place on a stack:

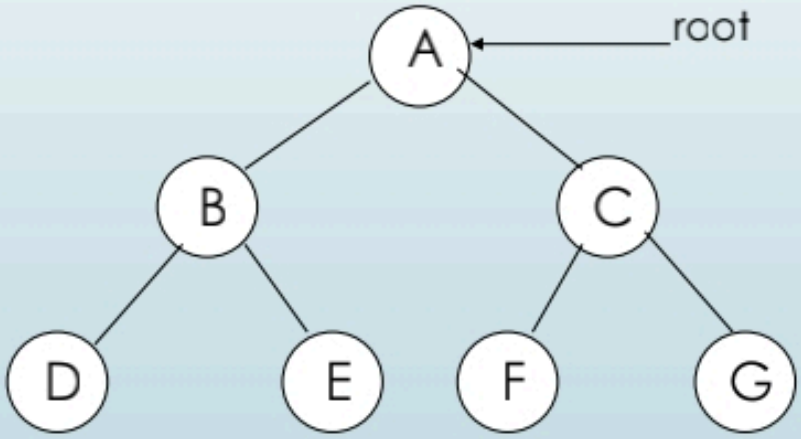


Queue

- The queue can be implemented into two ways:
  - Using arrays (Static implementation)
  - Using pointer (Dynamic implementation)

Trees

- The tree structure organizes the data into branches, which related the information.



Graph

- Graph is a mathematical non-linear data structure capable of representing many kind of physical structures.
- It has found application in Geography, Chemistry and Engineering sciences.
- Definition: A graph G(V,E) is a set of vertices V and a set of edges E.

Trees

- A tree can be defined as finite set of data items (nodes).
- Tree is non-linear type of data structure in which data items are arranged or stored in a sorted sequence.
- Tree represent the hierarchical relationship between various elements.

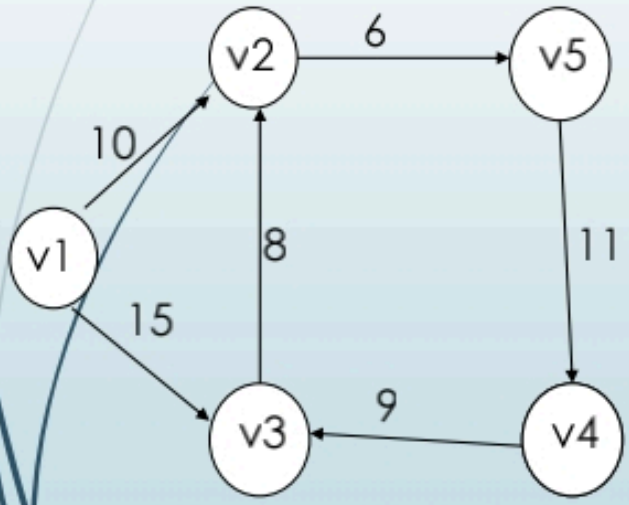
Graph

- An edge connects a pair of vertices and many have weight such as length, cost and another measuring instrument for according the graph.

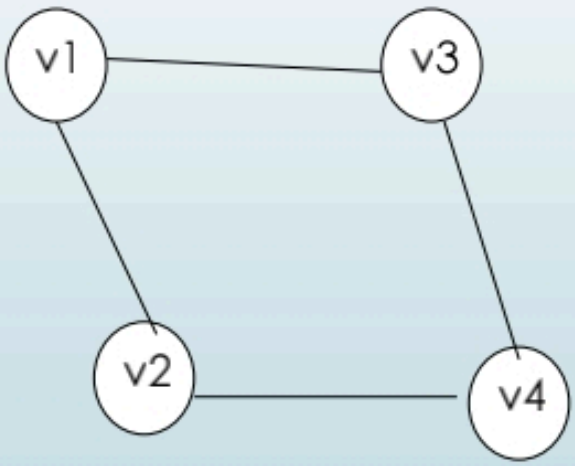
- Vertices on the graph are shown as point or circles and edges are drawn as arcs or line segment.

Graph

- Example of graph:



[a] Directed & Weighted Graph



[b] Undirected Graph

Graph

- Types of Graphs:
  - Directed graph
  - Undirected graph
  - Simple graph
  - Weighted graph
  - Connected graph
  - Non-connected graph

Growth of Functions and Asymptotic Notation

- When we study algorithms, we are interested in characterizing them according to their efficiency.
- We are usually interesting in the order of growth of the running time of an algorithm, not in the exact running time. This is also referred to as the *asymptotic running time*.
- We need to develop a way to talk about rate of growth of functions so that we can compare algorithms.
- *Asymptotic notation* gives us a method for classifying functions according to their rate of growth.

Big-O Notation

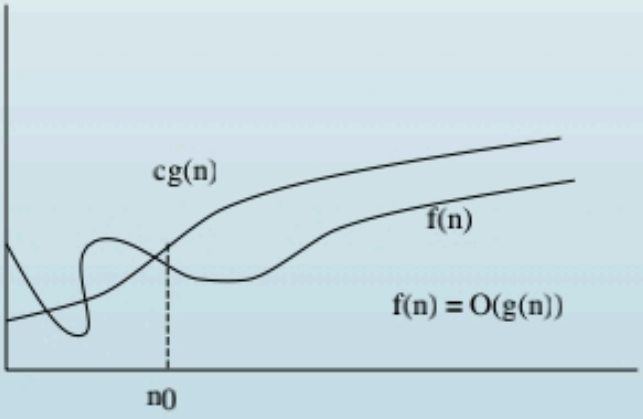
- **Definition:** $f(n) = O(g(n))$ if there are two positive constants c and n_0 such that

$|f(n)| \leq c\,|g(n)|$ for all $n \geq n_0$

- If $f(n)$ is nonnegative, we can simplify the last condition to

$0 \leq f(n) \leq c\,g(n)$ for all $n \geq n_0$

- We say that "$f(n)$ is big-O of $g(n)$."
- As n increases, $f(n)$ grows no faster than $g(n)$. In other words, $g(n)$ is an *asymptotic upper bound* on $f(n)$.



Big-Ω notation

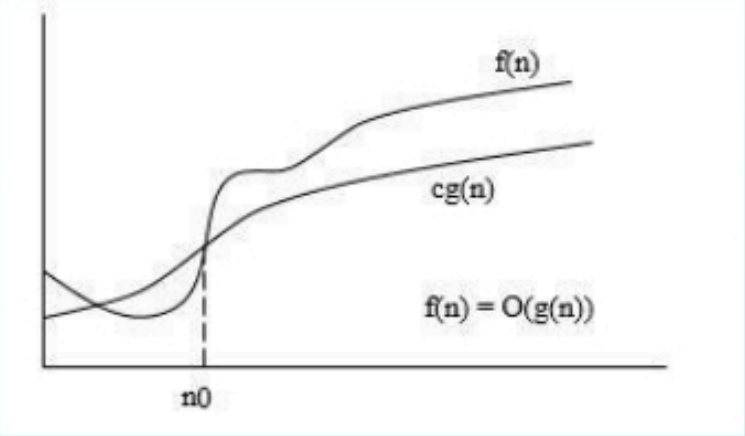
- **Definition:** $f(n) = \Omega(g(n))$ iff there are two positive constants c and n_0 such that

$|f(n)| \geq c\,|g(n)|$ for all $n \geq n_0$

- If $f(n)$ is nonnegative, we can simplify the last condition to

$0 \leq c\,g(n) \leq f(n)$ for all $n \geq n_0$

- We say that "$f(n)$ is omega of $g(n)$."
- As n increases, $f(n)$ grows no slower than $g(n)$. In other words, $g(n)$ is an *asymptotic lower bound* on $f(n)$.



Big-Θ notation

- **Definition:** $f(n) = \Theta(g(n))$ iff there are three positive constants c_1, c_2 and n_0 such that

$c_1|g(n)| \leq |f(n)| \leq c_2|g(n)|$ for all $n \geq n_0$

- If $f(n)$ is nonnegative, we can simplify the last condition to

$0 \leq c_1\, g(n) \leq f(n) \leq c_2\, g(n)$ for all $n \geq n_0$

- We say that "$f(n)$ is theta of $g(n)$."
- As n increases, $f(n)$ grows at the same rate as $g(n)$. In other words, $g(n)$ is an *asymptotically tight bound* on $f(n)$.