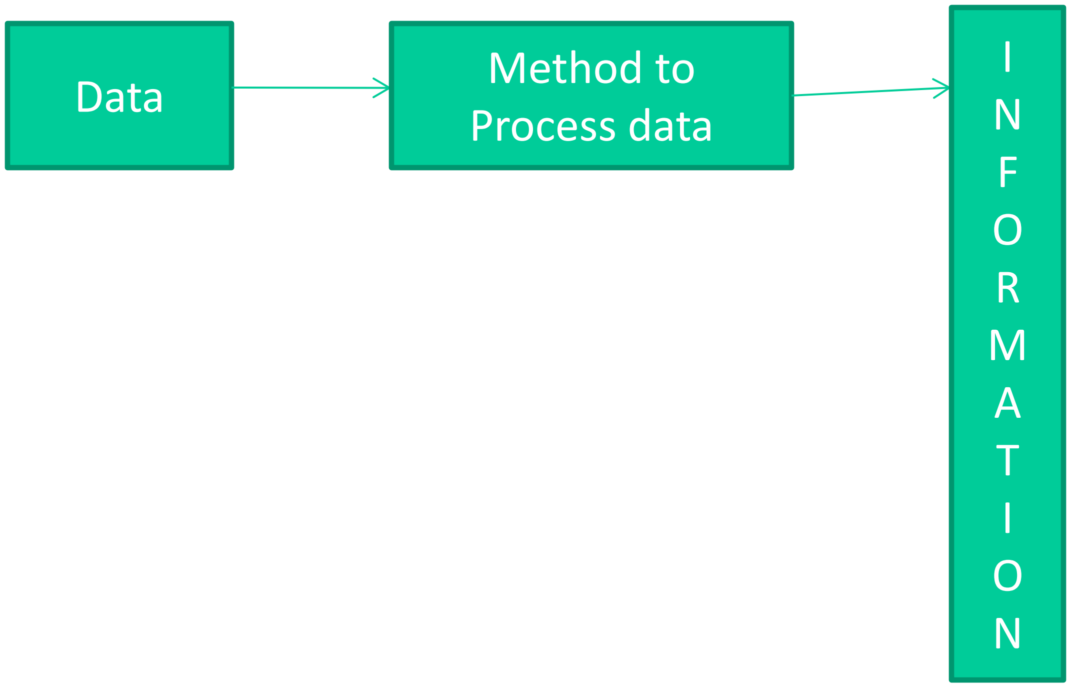
Introduction to Data Structures

Data and information

- Data means value or set of values . Data items refers to a single unit of values.
- Data is represented in the from of text/numbers/figures/ tables/graphs/pictures etc. A computer Language is incomplete without data.
- Information is derived from data, such as:

  per capita income, Average Population of the states , price index ,etc.

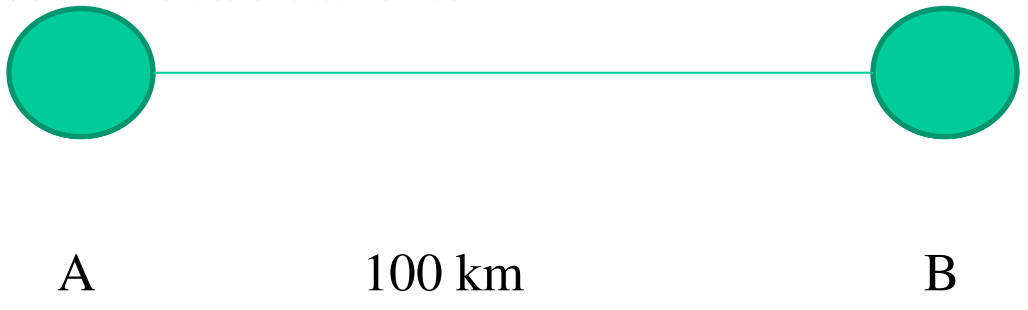


Example of data processed in information

- In a month we everyday get the newspaper then at the end we can calculate the bill from the data of the cost of newspaper we daily receive.
- Data about some persons in a group:

Information Details	Data
Name	Satish
Gender	Male
Age	32
Phone	987680987

- Distance between the two stations:



Examples of data into information

- From the score of an individual cricket player we can draw following information:
    - Total Runs Scored
    - Highest score
    - Average
    - Strike rate
    - No. of Sixes
    - No. of fours
- This information helps the planners/analysts/to make decisions.
- Info. In Computer: A computer is an electronic device that manipulates the information presented in the form of data.
- Data: Data is nothing but a collection of numbers, alphabets and symbols combined to represent information.
- Entity: An entity possesses some attributes and some values can be allocated to it. E.g. student entity in a college.

Data Structures

- Data Structure: **Organization** of data in computers’ memory so it can be accessed easily to solve the problem

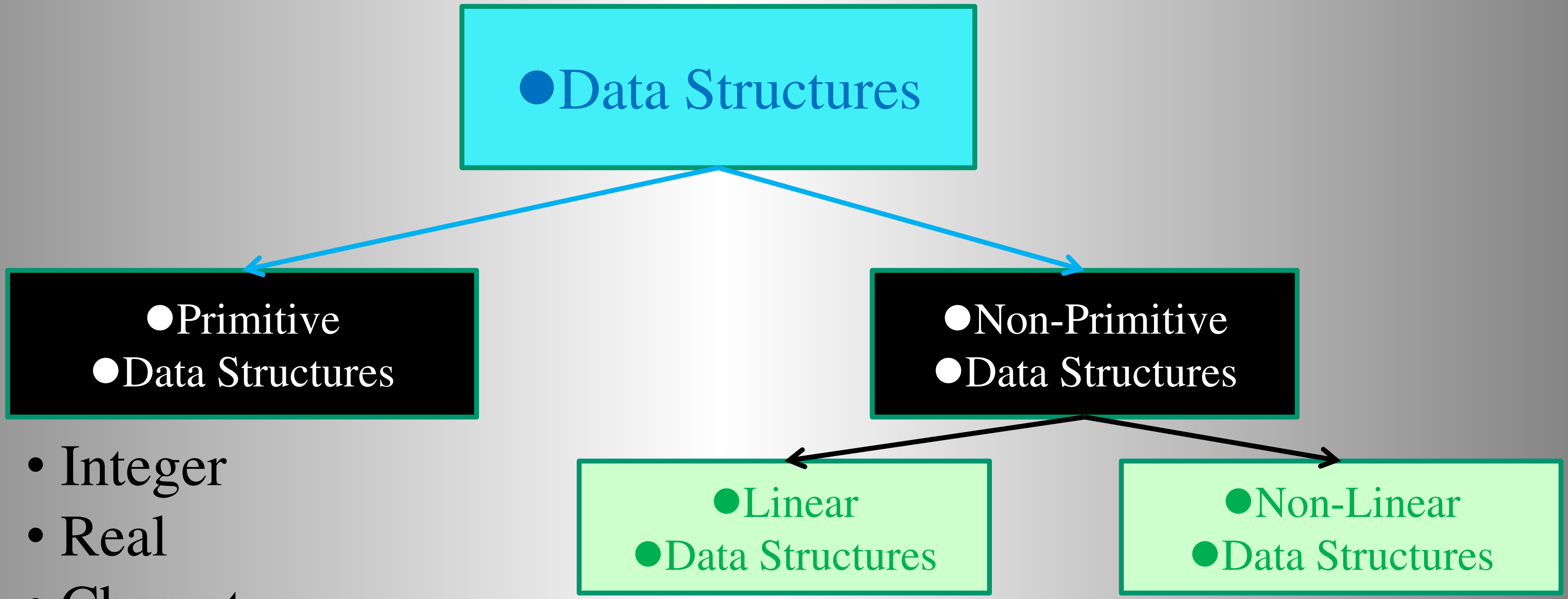
- Algorithm: Outline, the essence of a computational procedure, step-by-step instructions

- Program: an implementation of an algorithm in some programming language

Classification of Data Structures

- **Linear:** The values are arranged in a linear fashion. E.g. Arrays, Linked Lists, Stacks, Queues etc.
- **Non-Linear:** The values are not arranged in an order. E.g. Tree, Graph, Table etc.
- **Homogeneous:** Here all the values stored are of same type e.g. arrays
- **Non- Homogeneous:** Here all the values stored are of different type e.g. structures and classes.
- **Dynamic:**A *dynamic* data structure is one that can grow or shrink as needed to contain the data you want stored. That is, you can allocate new storage when it's needed and discard that storage when you're done with it. E.g. *pointers,* or *references*
- **Static:**They're essentially fixed-size and often use much space E.g. Array

Classification of Data Structures



- Integer
- Real
- Character
- Boolean

- Array
- Stack
- Queue
- Linked List

- Tree
- Graph

Types of Data Structures

- **Arrays:**

1. Linear data Structure
2. All elements stored at contiguous memory locations
3. Every element is identified by an index (Logical Address)
4. At the same time every element is identified by a physical address (Physical Address)
5. We can have 2 D and 3D arrays also but stored in memory in a linear way.

Types of Data Structures

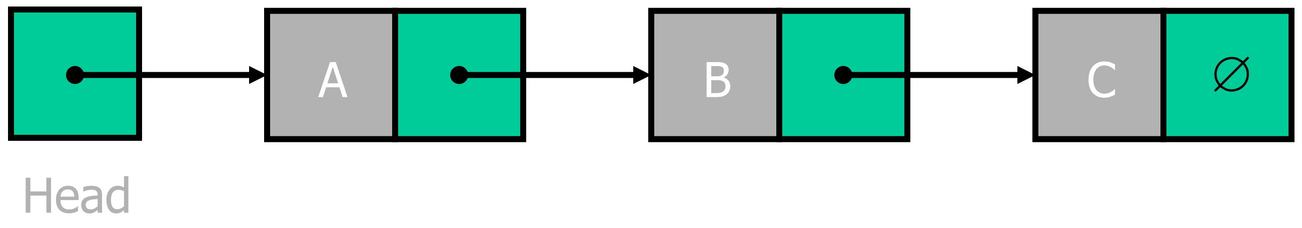
- **Stack:**
  - Only the top item can be accessed
  - Can only extract one item at a time
- A stack is a data structure with the property that only the top element of the stack is accessible
- The stack's storage policy is Last-In, First-Out
- Only the top element of a stack is visible, therefore the number of operations performed by a stack are few
- Need the ability to
  - Inspect the top element
  - Retrieve the top element
  - Push a new element on the stack
  - Test for an empty stack

Types of Data Structures

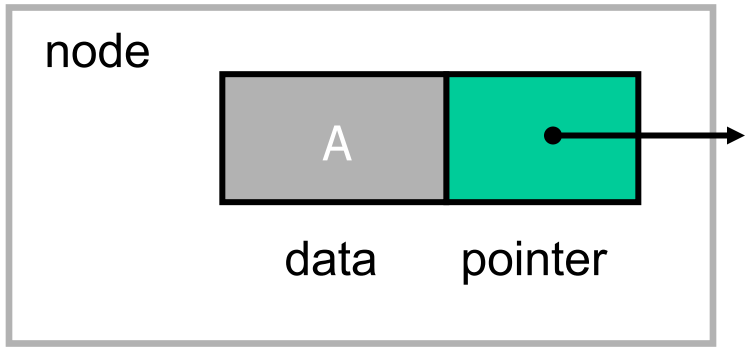
- **Queue**:

- Stores a set of elements in a particular order
- Stack principle: FIRST IN FIRST OUT
- = FIFO
- It means: the first element inserted is the first one to be removed
- Real life examples
  - Waiting in line
  - Waiting on hold for tech support
- Applications related to Computer Science
  - Threads
  - Job scheduling (e.g. Round-Robin algorithm for CPU allocation)

Linked Lists



- A *linked list* is a series of connected *nodes*
- Each node contains at least
  - A piece of data (any type)
  - Pointer to the next node in the list
- *Head*: pointer to the first node
- The last node points to NULL

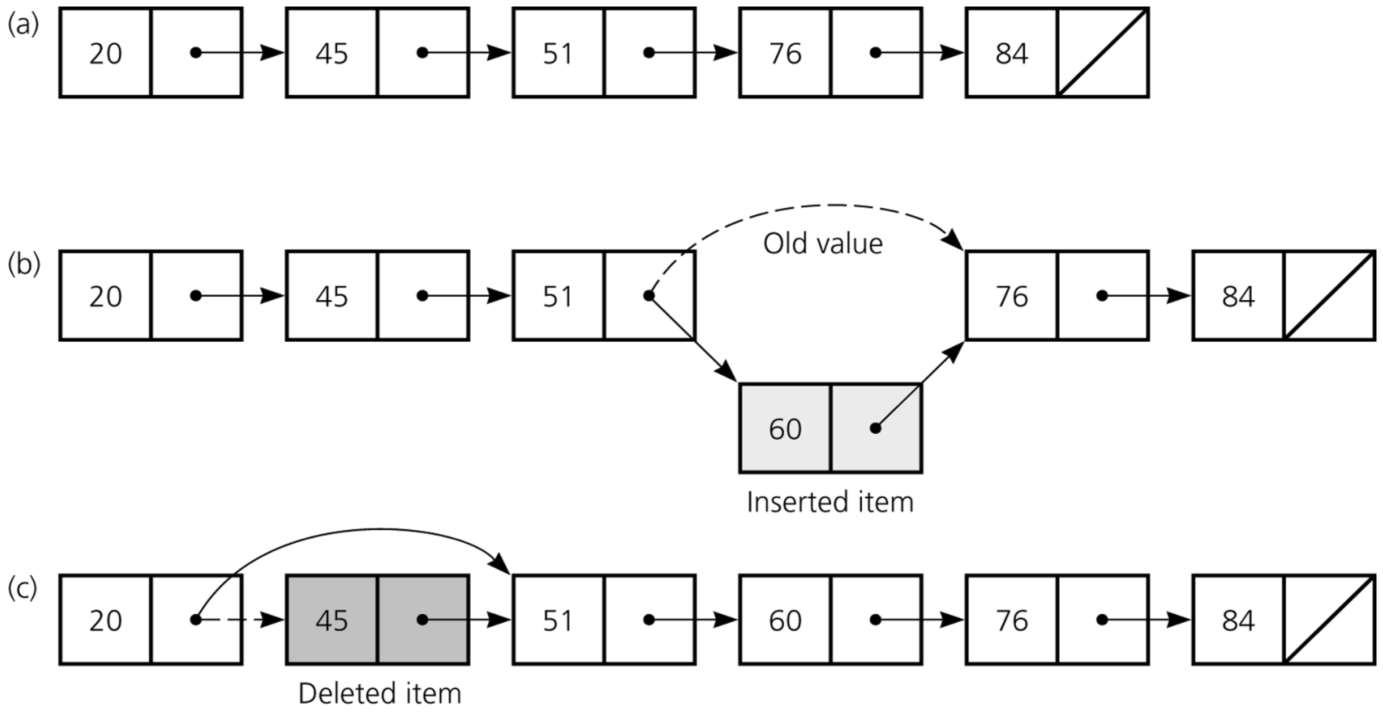


Types of Data Structures

- <u>Linked List:</u>

1. Linear data structure.
2. Primarily used when the elements are to stored at non-contiguous locations.

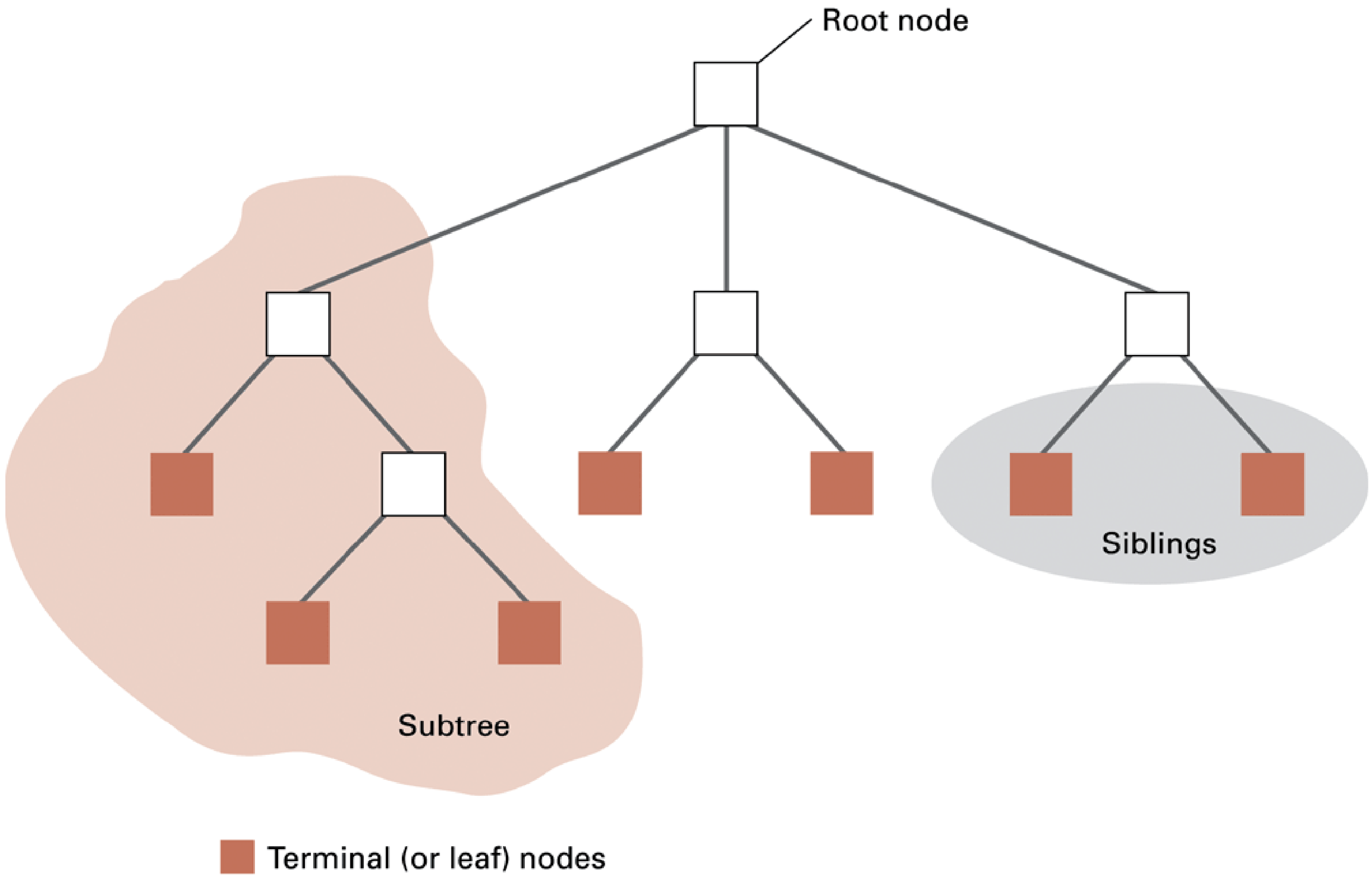
1. Linked list is able to grow in size as needed.
2. Does not require the shifting of items during insertions and deletions.



Terminology for a Tree

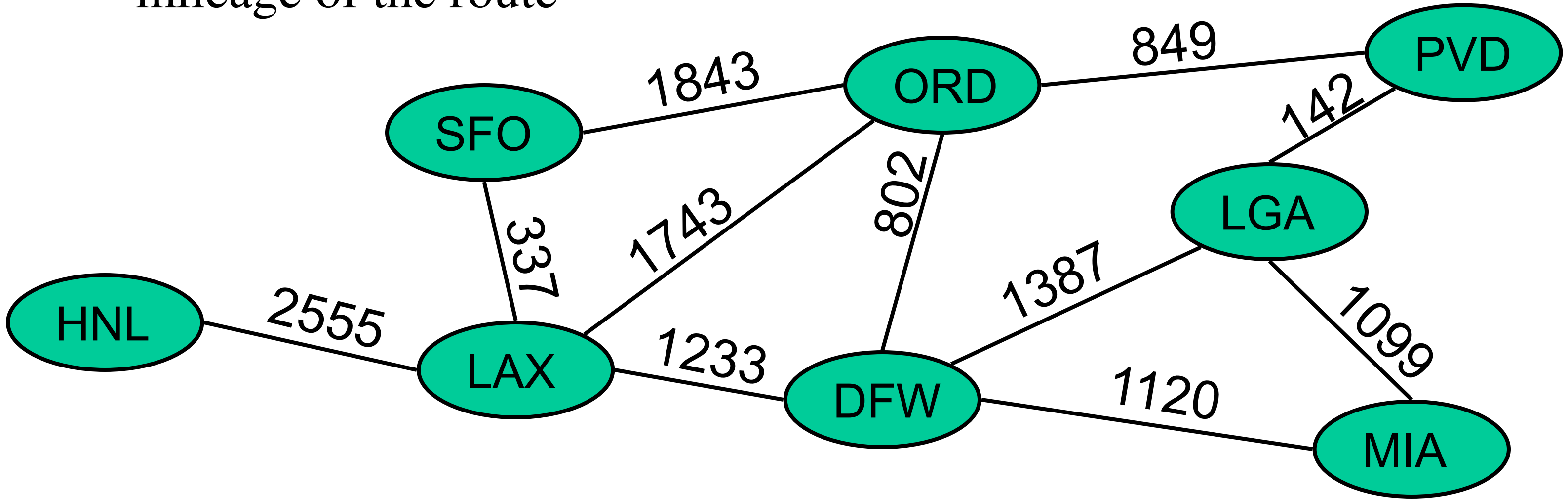
- **Tree:** A collection of data whose entries have a hierarchical organization
- **Node:** An entry in a tree
- **Root node:** The node at the top
- **Terminal** or **leaf node:** A node at the bottom
- **Parent:** The node immediately above a specified node
- **Child:** A node immediately below a specified node
- **Ancestor:** Parent, parent of parent, etc.
- **Descendent:** Child, child of child, etc.
- **Siblings:** Nodes sharing a common parent
- **Binary tree:** A tree in which every node has at most two children
- **Depth:** The number of nodes in longest path from root to leaf

Tree Terminology



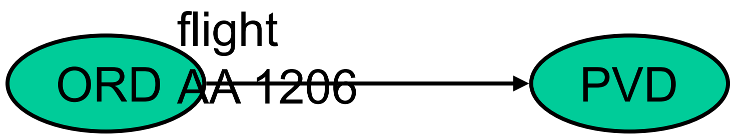
Graph

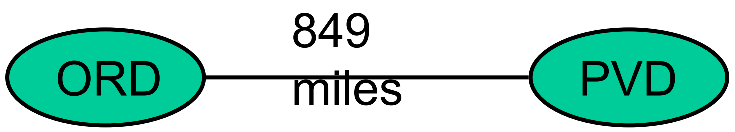
- A graph is a pair (***V, E***), where
  - ***V*** is a set of nodes, called vertices
  - ***E*** is a collection of pairs of vertices, called edges
  - Vertices and edges are positions and store elements
- Example:
  - A vertex represents an airport and stores the three-letter airport code
  - An edge represents a flight route between two airports and stores the mileage of the route



Edge Types

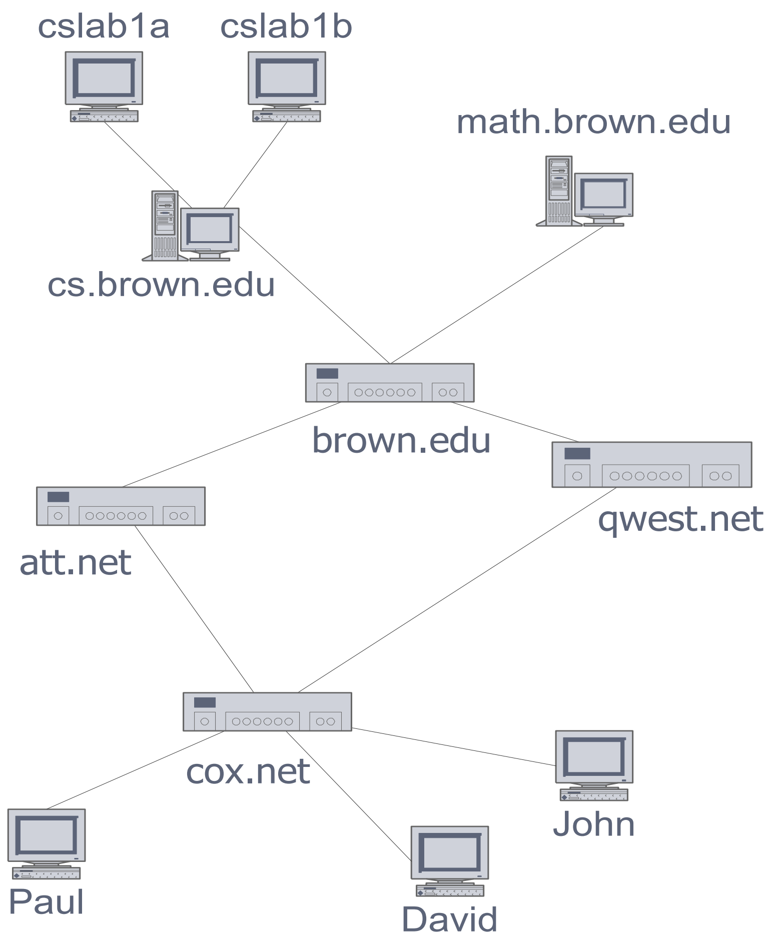
- Directed edge
  - ordered pair of vertices (***u***,***v***)
  - first vertex ***u*** is the origin
  - second vertex ***v*** is the destination
  - e.g., a flight
- Undirected edge
  - unordered pair of vertices (***u***,***v***)
  - e.g., a flight route
- Directed graph
  - all the edges are directed
  - e.g., route network
- Undirected graph
  - all the edges are undirected
  - e.g., flight network





Applications

- Electronic circuits
  - Printed circuit board
  - Integrated circuit
- Transportation networks
  - Highway network
  - Flight network
- Computer networks
  - Local area network
  - Internet
  - Web
- Databases
  - Entity-relationship diagram



Data Structure Operations

Data Structures are processed by using certain operations.

1. Traversing: Accessing each record exactly once so that certain items in the record may be processed.

2. Searching: Finding the location of the record with a given key value, or finding the location of all the records that satisfy one or more conditions.

3. Inserting: Adding a new record to the structure.

4. Deleting: Removing a record from the structure.

Special Data Structure-Operations

- Sorting: Arranging the records in some logical order (Alphabetical or numerical order).

- Merging: Combining the records in two different sorted files into a single sorted file.

Algorithm

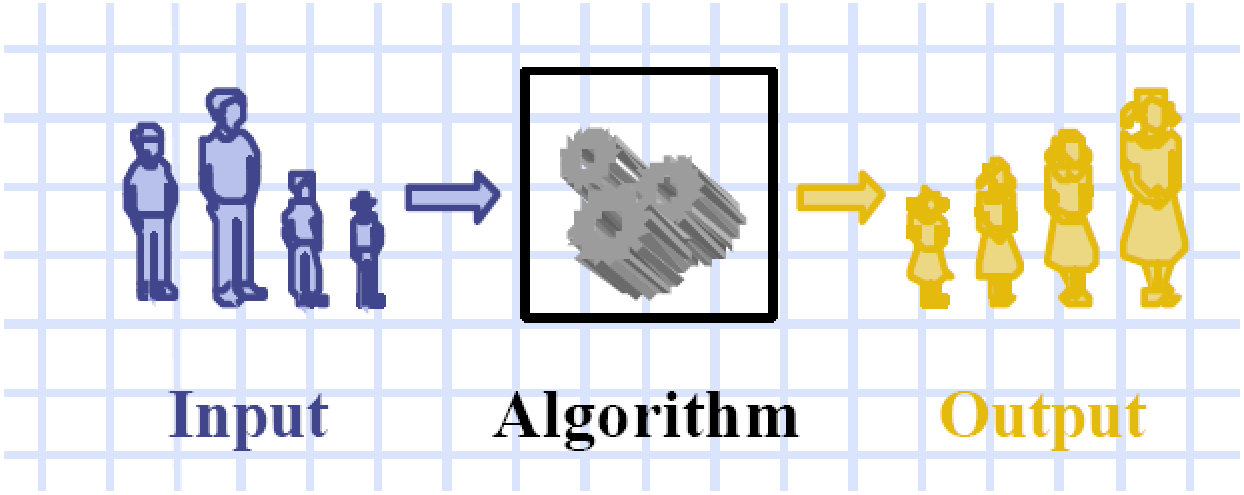
Algorithm

- Finite step by step list of well-defined instructions for solving a particular problem.
- Algorithm is providing a way to access the data from the data structure in most convenient way.
- Algorithm is programming independent.
- It is written in combination of english language with little syntax conditions.

Algorithm

- A clearly specified set of instructions to solve a problem.
- Characteristics:
  - **Input:** Zero or more quantities are externally supplied
  - **Definiteness:** Each instruction is clear and unambiguous
  - **Finiteness:** The algorithm terminates in a finite number of steps.
  - **Effectiveness:** Each instruction must be primitive and feasible
  - **Output:** At least one quantity is produced

Algorithm



Algorithmic Notations:

Formal presentation of the Algorithm consists of two parts:

1. A paragraph that tells the purpose of an algorithm. Identifies the variables which occur in the algorithm and lists the input data.
2. The list of the steps that is to be executed.

Some conventions used in Algorithms:

- ➢ Identifying Number (Algorithm 2:)
- ➢ Steps, Control and Exit (All statements are numbered, Go to and Exit)
- ➢ Comments (The comments are to be given to understand the meaning)
- ➢ Variable Names (capital letters are used for variable names)
- ➢ Assignment Statement (Max:=DATA[1])
- ➢ Input and output (Read: variable names, Write: Messages and/or variable names)
- ➢ Procedure

Basic Notations

Control Structures:

1. **Sequential Logic or Sequential Flow**

2. **Selection Logic or Conditional Flow**

```
•    If  (condition) , then:
      -----
      [Endif]
•    If  (condition) ,   then:
     -----
      Else :
    -------
       [ Endif]
```

- Multiple If (condition), then : -----

```
                    Elseif  :  -----
                    Else :
                          ------
                    [Endif]
```

3. Iteration Logic (Repetitive Flow)

-

```
Repeat step ---- for K=I to N:
        [Module]        [End of Loop]
```

- While loop

```
Repeat tep --- while K<= N:
  (logic)
[end of while loop]
```

C++ Program of sum of two numbers

```
#include<iostream>
Using namespace std;
Main()
{
int a,b,sum;
cout<<“enter a and b”;
cin>> a>>b;
sum=a+b;
cout<<“sum of a and b is “;
cout<<sum;
}
```

Convert it into algorithm

Sum of N numbers

```
#include<iostream>
Using namespace std;
Main()
{
int sum=0;
For(int i=0;i<n;i++)
{
cin>>i;
 sum=sum +i;
cout<<"sum of n numbers is ";
cout<<sum;
}
```

Convert it into algorithm

Practice Algorithms

- Write an Algorithm (WAA) to find largest of three numbers.
- WAA to find the sum of first 10 natural numbers using for loop.
- WAA to find the sum of first 10 natural numbers using while loop.