Foundations of Software Development

Priti Srinivas Sajja

Professor

Department of Computer Science
Sardar Patel University

Visit pritisajja.info for details

Unit 1: Data Structure

- **Introduction**
- Need of DT
- Types of DT
- Array
- Stack
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

- **Name: Dr. Priti Srinivas Sajja**
- **Communication:**
  - **Email :** priti@pritisajja.info
  - **Mobile :** +91 9824926020
  - URL :http://pritisajja.info
- ***Academic qualifications*** : Ph. D in Computer Science
- ***Thesis title***: Knowledge-Based Systems for Socio-
- Economic Rural Development (2000)
- ***Subject area of specialization*** : Artificial Intelligence
- ***Publications*** : **216** in Books, Book Chapters, Journals and in Proceedings of International and National Conferences

Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- Array
- Stack
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

Unit 1: Basics of Data Structures

- Introduction to Data Structures, Applications, Operations
- Primitive and Non-primitive Data Structures
- Linear and Non-linear Structures
- Introduction to Array, Stack, Queue, Linked List, Trees and Graphs

Unit 1: Data Structure

- **Introduction**
- Need of DT
- Types of DT
- Array
- Stack
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

▪ **Data:** Row observations and values

▪ **Structure:** Way of organizing the values, so that it is easier to use

▪ The data structure is defined as a **way of organizing data** in such a way so that data can be used **efficiently (Space and Time)**.

Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- Array
- Stack
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

Need For Data Structure

- **Searching Large amounts of Data**: To retrieve required data efficiently from the large amount of data generated and stored
- **Speed of Processing:** Searching and retrieving data from the well organized bunch takes less time and less effort.
- **Concurrent Requests:** Many and simultaneous requests can be easily handled.

Unit 1: Data Structure

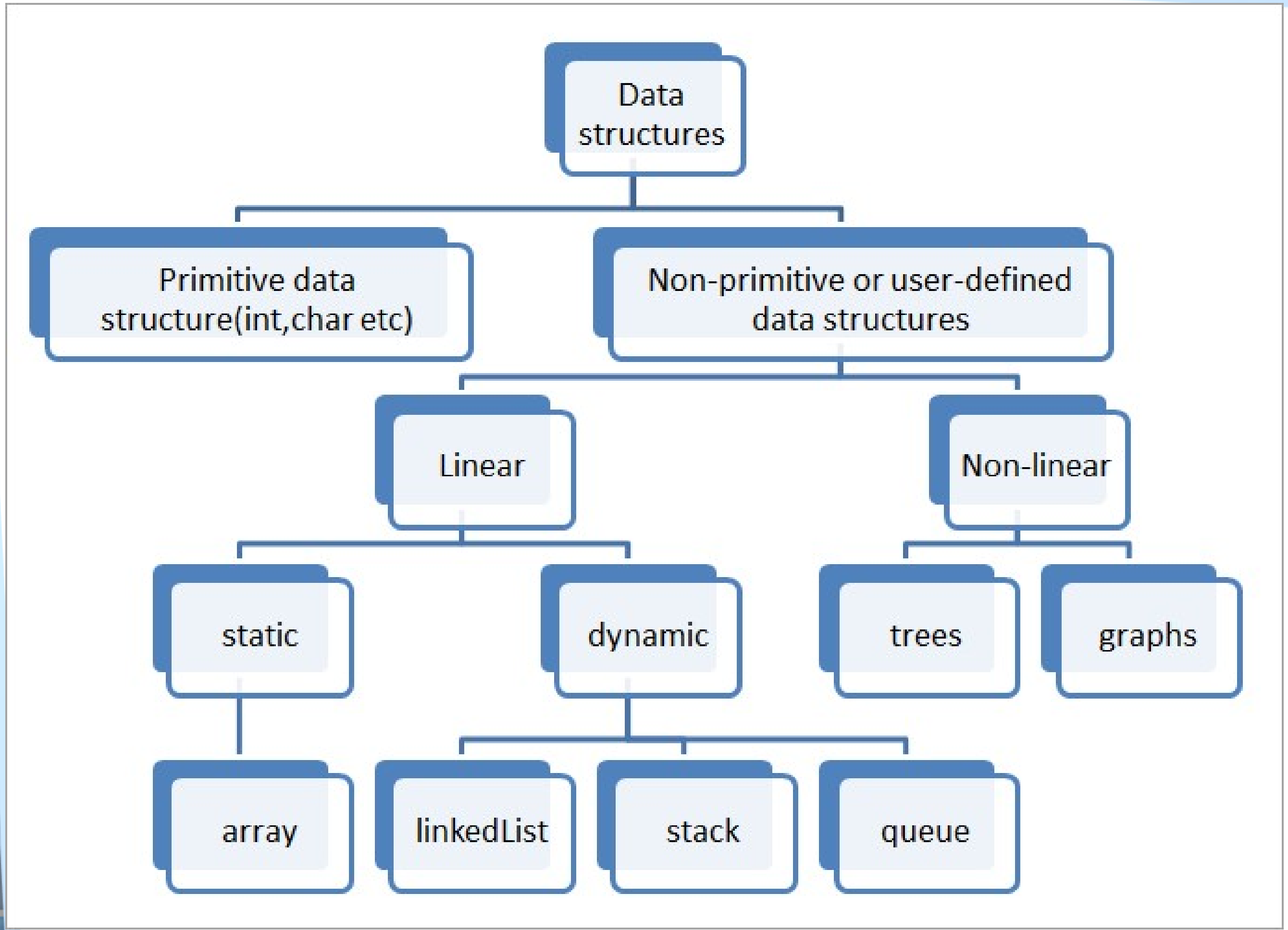
Introduction
Need of DT
Types of DT
Array
Stack
Queue
Linked List
Tree & Graphs
Acknowledgement

Characteristics of a Data Structure

- **Correctness** – correctly implemented.
- **Time Complexity** – Running time or the execution time of operations of data structure must be as small as possible.
- **Space Complexity** – Memory usage of a data structure operation should be as little as possible.

Unit 1: Data Structure

- Introduction
- Need of DT
- **Type of DT**
- Array
- Stack
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement



Unit 1: Data Structure

Operations on Data Structure

- Define or create structure
- Add an element
- Delete an element
- Traverse / Display
- Sort the list of elements
- Search for a data element
- Merging and spiting
- Delete an element or delete complete structure

Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- Array
- Stack
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

Primitive Data Structures

- Primitive data structures are **basic structures** and are **directly operated** upon by machine instructions.
- Primitive data structures have different representations on different computers.
- **Integers, floats, character and pointers** are examples of primitive data structures.
- These data types are available in most programming languages as **built in type**.

Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- Array
- Stack
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

Non Primitive Data Structures

- These are **more sophisticated** data structures.
- These are **derived from primitive data** structures.
- A Non-primitive data type is further divided into **Linear and Non-Linear** data structure
- **Linear→ Array, Linked list, Stack and Queue**
- **Non linear→ Tree and Graph**

Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- Array
- Stack
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

Linear Data Structures

- A data structure is said to be Linear, if its **elements are connected in linear fashion** by means of logically or in sequence memory locations.
- There are two ways to represent a linear data structure in memory,
  - **Static memory allocation**
  - **Dynamic memory allocation**

Unit 1: Data Structure

Difference between Linear and Non Linear Data Structure

	Linear Data Structure	Non-Linear Data Structure
1	Every item is related to its previous and next time.	Every item is attached with many other items.
2	Data is arranged in linear sequence.	Data is not arranged in sequence.
3	Data items can be traversed in a single run.	Data cannot be traversed in a single run.
4	Eg. Array, Stacks, linked list, queue.	Eg. tree, graph.
5	Implementation is easy.	Implementation is difficult.

Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- **Array**
- Stack
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

Array

- Group of data with **same type and same size stored adjacent** to each other
- Arrays are always stored in **consecutive** memory locations.
- It is a **linear data structure** with known number of elements.
- Each memory location stores one fixed-length data item
- Used in all programming languages
- Can be used to **create other data structures** such as stacks and queues.
- It can be one dimensional or many dimensional.

One Dimensional Array of Integers

A[0]	A[1]	A[2]	A[3]	A[4]
10	20	30	40	50

- Here A is the name of an array.
- The value in bracket are called index.
- To refer 3rd item in array A, A[2] is used.
- Total number of elements are 5.
- All are integers.
- If you know the starting address of A and size of the data, you can know address of any element in the array by knowing the index value.

Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- **Array**
- Stack
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

Two Dimensional Array of Integers

- A[Row, Col]

Columns→ Row	0	1	2	3	4
0	A[0,0]	A[0, 1]	A[0, 2]	A[0, 3]	A[0, 4]
1	A[1,0]	A[1, 1]	A[1, 2]	A[1, 3]	A[1, 4]
2	A[2,0]	A[2, 1]	A[2, 2]	A[2, 3]	A[2, 4]
3	A[3,0]	A[3, 1]	A[3, 2]	A[3, 3]	A[3, 4]
4	A[4,0]	A[4, 1]	A[4, 2]	A[4, 3]	A[4, 4]

Two Dimensional Array of Integers

- A[Row, Col]

Columns→ Row	0	1	2	3	4
0	11	12	13	14	15
1	21	22	23	24	25
2	31	32	33	34	35
3	41	42	43	44	45
4	51	52	53	54	55

Two Dimensional Array of Integers

- Marks[Row, Col]

Columns→ Row	Sub 1	Sub 2	Sub 3	Sub 4	Sub 5
1st student	67	45	68	79	56
2nd student	66	77	85	43	78
3rd student	56	78	98	34	55
4th student	23	45	56	67	77
5th student	55	44	66	77	65

Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- **Array**
- Stack
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

Examples of one dimensional array

An array of five characters called C

- C[0]= ‘A’
- C[1]=‘B’
- C[2]=‘C’
- C[3]=‘D’
- C[4]=‘E’

C[0]	C[1]	C[2]	C[3]	C[4]
‘A’	‘B’	‘C’	‘D’	‘E’

Unit 1: Data Structure

Examples of one dimensional array

An array of five integer numbers called NUM

NUM[0]	NUM[1]	NUM[2]	NUM[3]	NUM[4]
44	55	66	77	88

- NUM[0]= 44 NUM[1]=55
- NUM[2]=66 NUM[3]=77
- NUM[4]=88

Examples of one dimensional array

An array of five integer numbers called NUM

NUM[0]	NUM[1]	NUM[2]	NUM[3]	NUM[4]
44	55	66	77	88

- **Find out NUM[0] + NUM[1]**

  = 44+55 =99
- **Find out NUM[4]- Num[3]**

  = 88-77=11
- **Find out yourself…**
- NUM[0] + NUM[1]+ NUM[2] +NUM[3] +NUM[4]

Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- **Array**
- Stack
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

Examples of one dimensional array

NUM[0]	NUM[1]	NUM[2]	NUM[3]	NUM[4]
44	55	66	77	88

- **Calculate total of all five integers from array NUM.**

  = NUM[0] + NUM[1] + NUM[2] + NUM[3] + NUM[4]

  = 44+55+66+77+88 = 330

- **Calculate average of all five integers from array NUM.**

  = (NUM[0] + NUM[1] + NUM[2] + NUM[3] + NUM[4])/5

  = (44+55+66+77+88) /5 = 330/5 = 66

Unit 1: Data Structure

Examples of one dimensional array

An array of five real numbers called NUM2

NUM2[0]	NUM2[1]	NUM2[2]	NUM2[3]	NUM2[4]
44.0	55.2	66.3	77.6	88.9

- NUM2[0]= 44.0
- NUM2[1]=55.2
- NUM2[2]=66.3
- NUM2[3]=77.6
- NUM2[4]=88.9

- You can calculate total, average, maximum number, and minimum number fro the array.

Two Dimensional Array of Characters called Names

	0	1	2	3	4
0	‘K’	‘i’	‘r’	‘t’	‘i’
1	‘P’	‘r’	‘i’	‘t’	‘i’
2	‘D’	‘i’	‘p’	‘t’	‘i’
3	‘S’	‘w’	‘a’	‘t’	‘i’
4	‘S’	‘t’	‘u’	‘t’	‘i’

- Names[0, 0] = ‘K’
- Names [1, 0]= ‘P’

Unit 1: Data Structure

Introduction
Need of DT
Types of DT
Array
Stack
Queue
Linked List
Tree & Graphs
Acknowledgement

- Defining An Array in Various Programming Languages

```
Array Declaration In Different Languages:

JAVA                                  long arr [] = new long [5];

 C                                    long arr[5];

Python                                arr = [None] * 5

JavaScript                            var arr = [];
```

Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- Array
- **Stack**
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

Stack

- Named stack as it behaves like a real-world stack, for example – a deck of cards or a pile of plates, etc.

- **First in Last out** Structure
- Only **top element** can be accessed.
- For example, we can place or remove a card or plate from the top of the stack only.

Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- Array
- **Stack**
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

- Stack A stack can be implemented by means of Array, Structure, Pointer, and Linked List.
- Stack can either be a fixed size one or it may have a sense of dynamic resizing.

Basic Operations

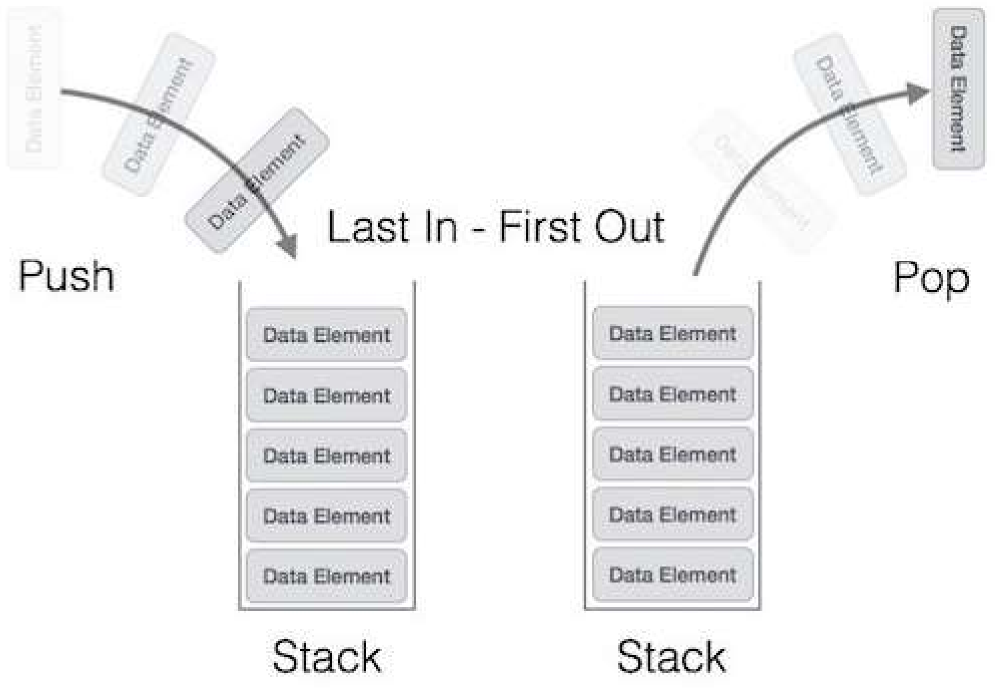
- **push()** – Pushing (storing) an element on the stack.
- **pop()** – Removing (accessing) an element from the stack.

Other Operations

- **peek()** – get the top data element of the stack, without removing it.
- **isFull()** – check if stack is full.
- **isEmpty()** – check if stack is empty.

Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- Array
- **Stack**
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

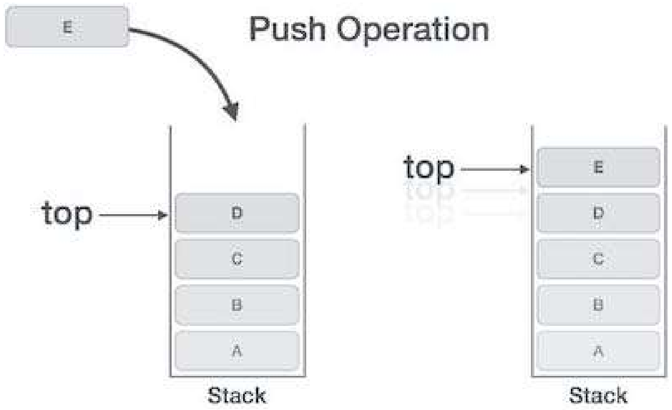


Unit 1: Data Structure

Push Operation

(Putting a new data element into the stack)

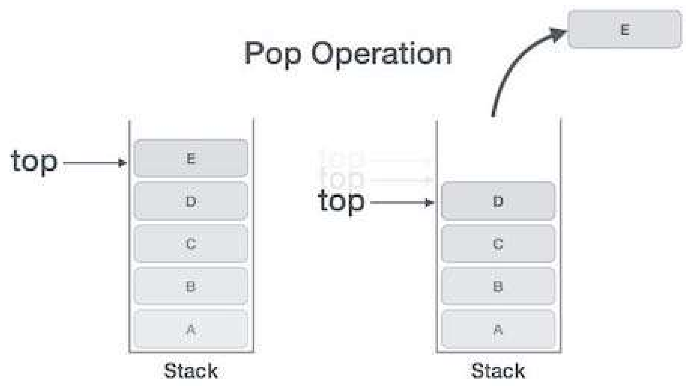
- **Step 1** – Checks if the stack is full.
- **Step 2** – If the stack is full, produces an error and exit.
- **Step 3** – If the stack is not full, increments **top** to point next empty space.
- **Step 4** – Adds data element to the stack location, where top is pointing.
- **Step 5** – Returns success.



Pop Operation

(Taking off the top data element from the stack)

- **Step 1** – Checks if the stack is empty.
- **Step 2** – If the stack is empty, produces an error and exit.
- **Step 3** – If the stack is not empty, accesses the data element at which **top** is pointing.
- **Step 4** – Decreases the value of top by 1.
- **Step 5** – Returns success.



Introduction
Need of DT
Types of DT
Array
Stack
Queue
Linked List
Tree & Graphs
Acknowledgement

Uses of Stack

- Parsing expression (infix, prefix and postfix conversion)
- Recession
- Flow of control and function call
- Back tracking procedures and games

Queue

- First in first out

Queue

- A queue can be implemented by means of Array, Structure, Pointer, and Linked List.
- Stack can either be a fixed size one or it may have a sense of dynamic resizing.

Basic Operations

- **Insert ()** – always at the end
- **Delete ()** – always from the front

Other Operations

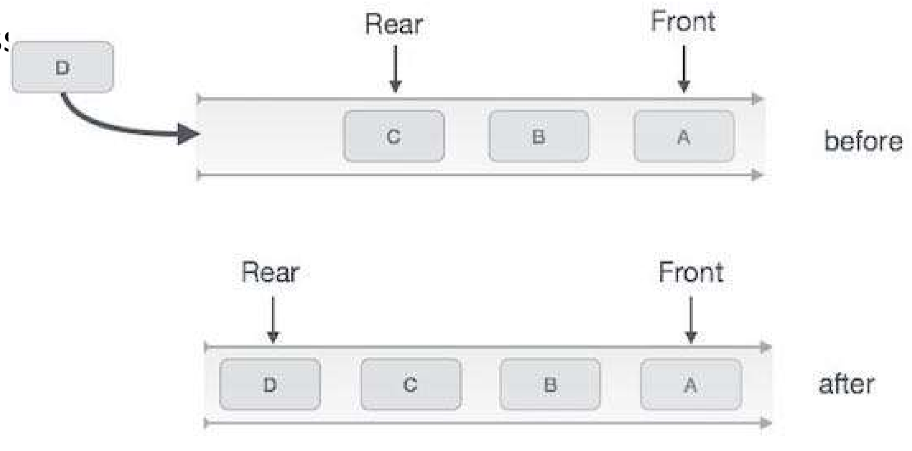
- **peek()** – get the data element from the queue, without removing it.
- **isFull()** – check if stack is full.
- **isEmpty()** – check if stack is empty.

Unit 1: Data Structure

Insert Operation

(inserting data in the queue at the end /rear position)

- **Step 1** – Check if the queue is full.
- **Step 2** – If the queue is full, produce overflow error and exit.
- **Step 3** – If the queue is not full, increment **rear** pointer to point the next empty space.
- **Step 4** – Add data element to the queue location, where the rear is pointing.
- **Step 5** – return succes



Queue Enqueue

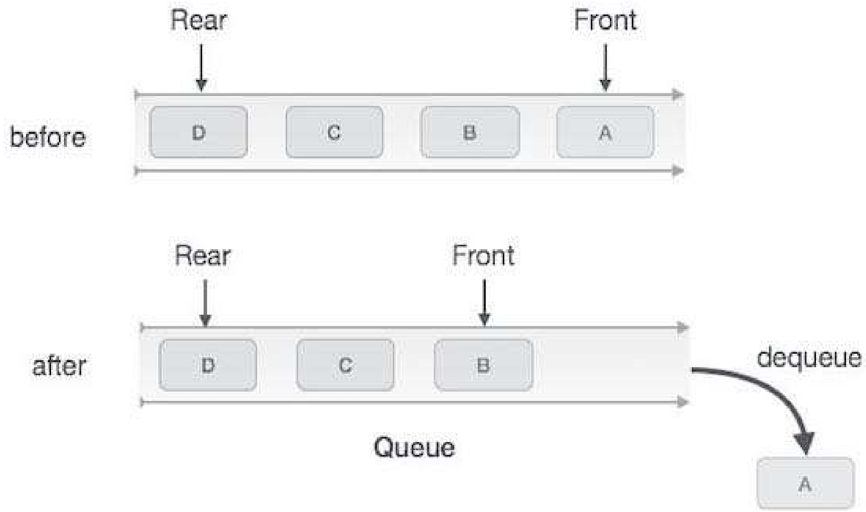
Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- Array
- Stack
- **Queue**
- Linked List
- Tree & Graphs
- Acknowledgement

Delete Operation

(Deleting data at the beginning /front position)

- **Step 1** – Check if the queue is empty.
- **Step 2** – If the queue is empty, produce underflow error and exit.
- **Step 3** – If the queue is not empty, access the data where **front** is pointing.
- **Step 4** – Increment **front** pointer to point to the next available data element.
- **Step 5** – Return success.



Queue Dequeue

Unit 1: Data Structure

Variations on Queue

- Priority queue
- Circular queue

Uses of Queue

- Operating systems and Resource management such as CPU scheduling, memory scheduling, printer queue, etc.
- Call centre phone systems
- Scheduling jobs
- Searching

Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- Array
- Stack
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

Linked List

- **A linked list is a sequence of data structures, which are connected together via links.**

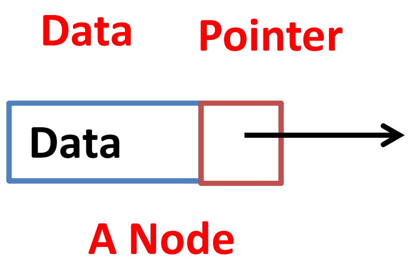
Basic Operations

- **Insertion** – Adds an element at the beginning of the list.
- **Deletion** – Deletes an element at the beginning of the list.
- **Display** – Displays the complete list.
- **Search** – Searches an element using the given key.
- **Delete** – Deletes an element using the given key.

Introduction
Need of DT
Types of DT
Array
Stack
Queue
Linked List
Tree & Graphs
Acknowledgement

Linked List

- Linked list consists of linked nodes.
- Each node is having a data and a pointer part.
- The data part stores data in it.
- The pointer is an address pointing towards the next element of the list.



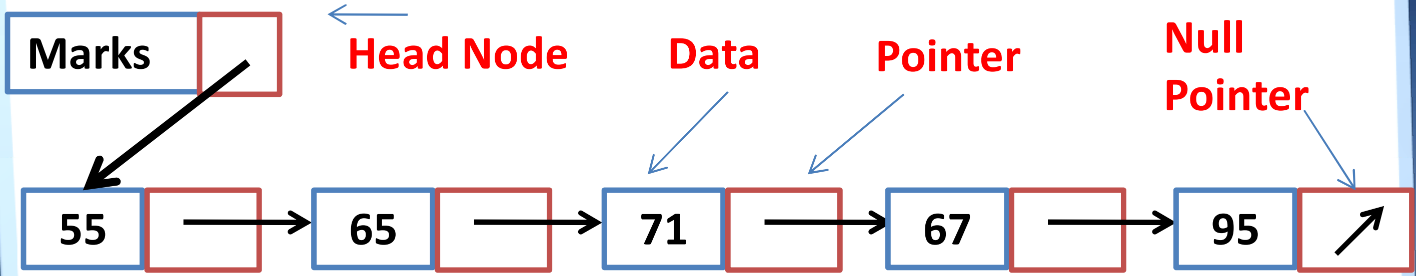
- Each list is having a head node.
- In a head node data part contains name of list and pointer contains address of the first (next) node of the list.

Unit 1: Data Structure

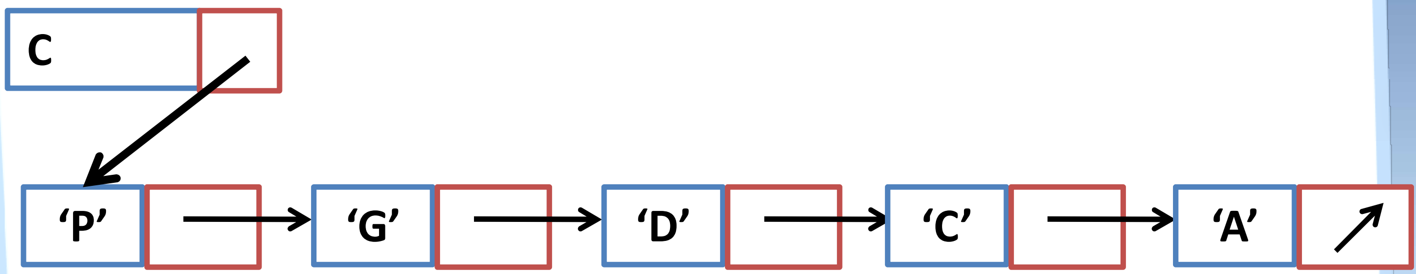
- Introduction
- Need of DT
- Types of DT
- Array
- Stack
- Queue
- **Linked List**
- Tree & Graphs
- Acknowledgement

Linked List Example

- Create a link list of 5 subjects marks of a student called Marks.

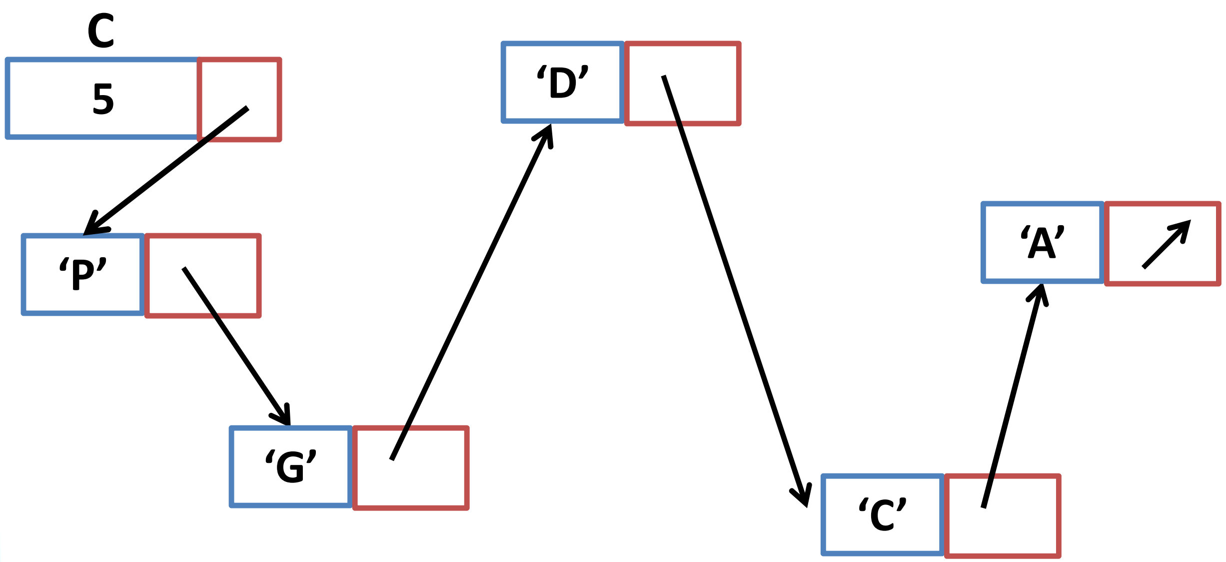


- Create a link list of 5 characters called C.



Linked List Example

- Create a link list of 5 characters called C.

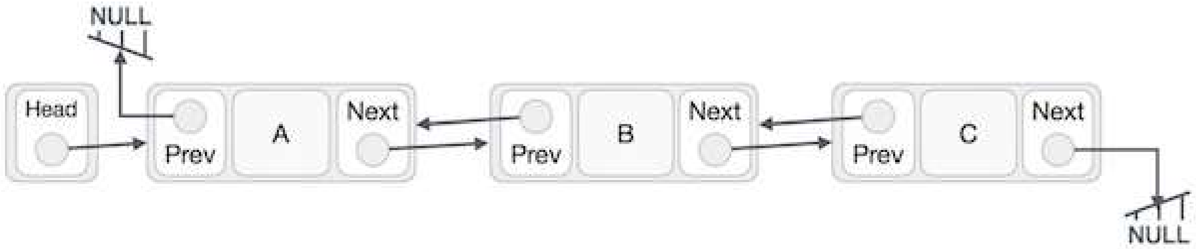


Unit 1: Data Structure

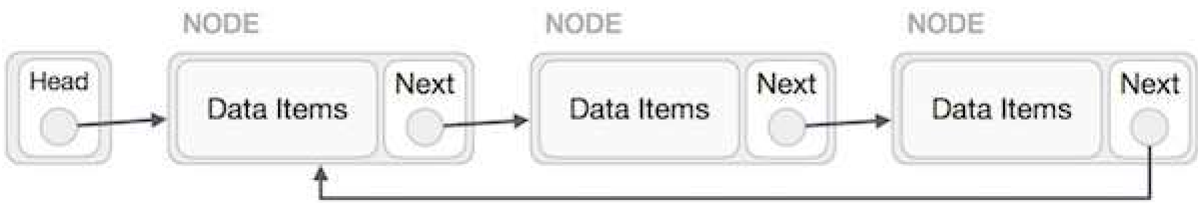
- Introduction
- Need of DT
- Types of DT
- Array
- Stack
- Queue
- **Linked List**
- Tree & Graphs
- Acknowledgement

Types of Linked List

- **Simple Linked List** – Item navigation is forward only.
- **Doubly Linked List** – Items can be navigated forward and backward.

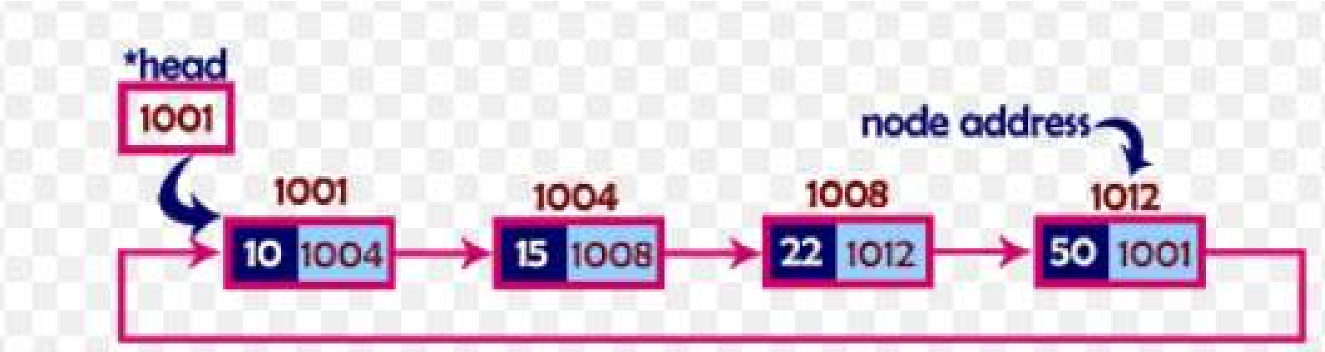


- **Circular Linked List** – Last item contains link of the first element as next and the first element has a link to the last element as previous.



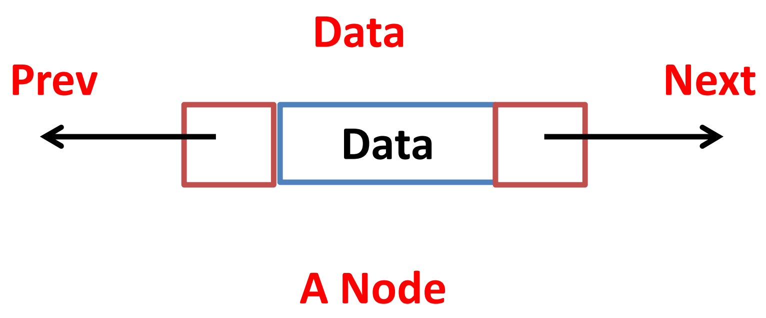
Circular Linked List

- In single linked list, every node points to its next node in the sequence and the last node points NULL. But in circular linked list, every node points to its next node in the sequence but the last node points to the first node in the list.

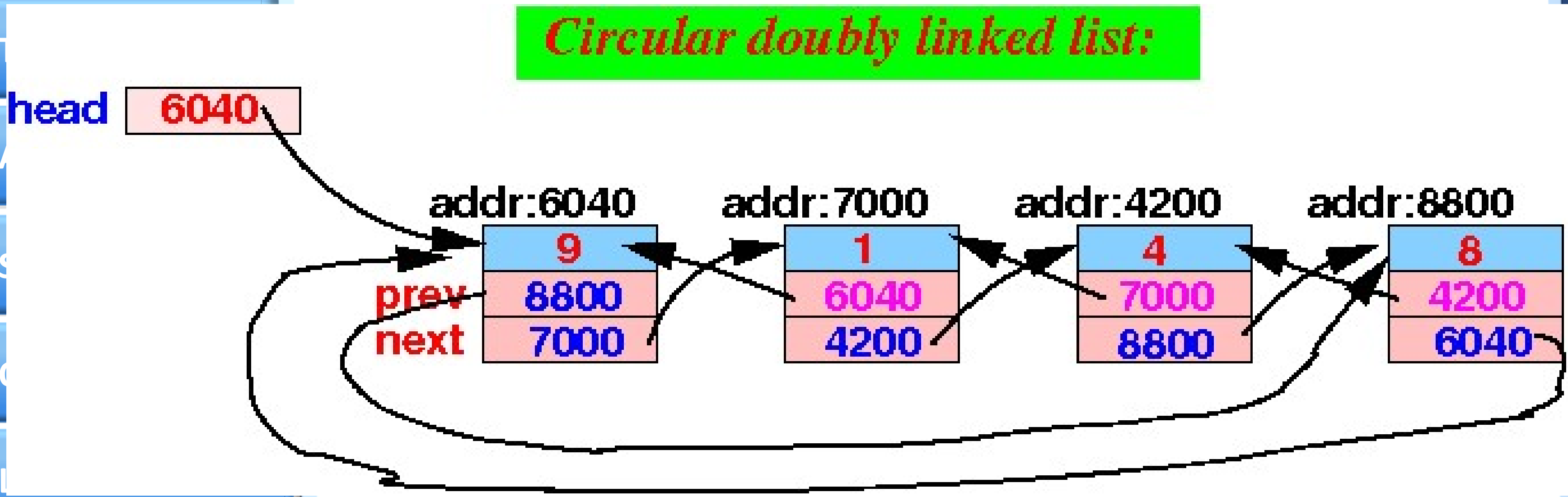


Java2novice.com

Doubly Linked List Node



Doubly Linked List Example

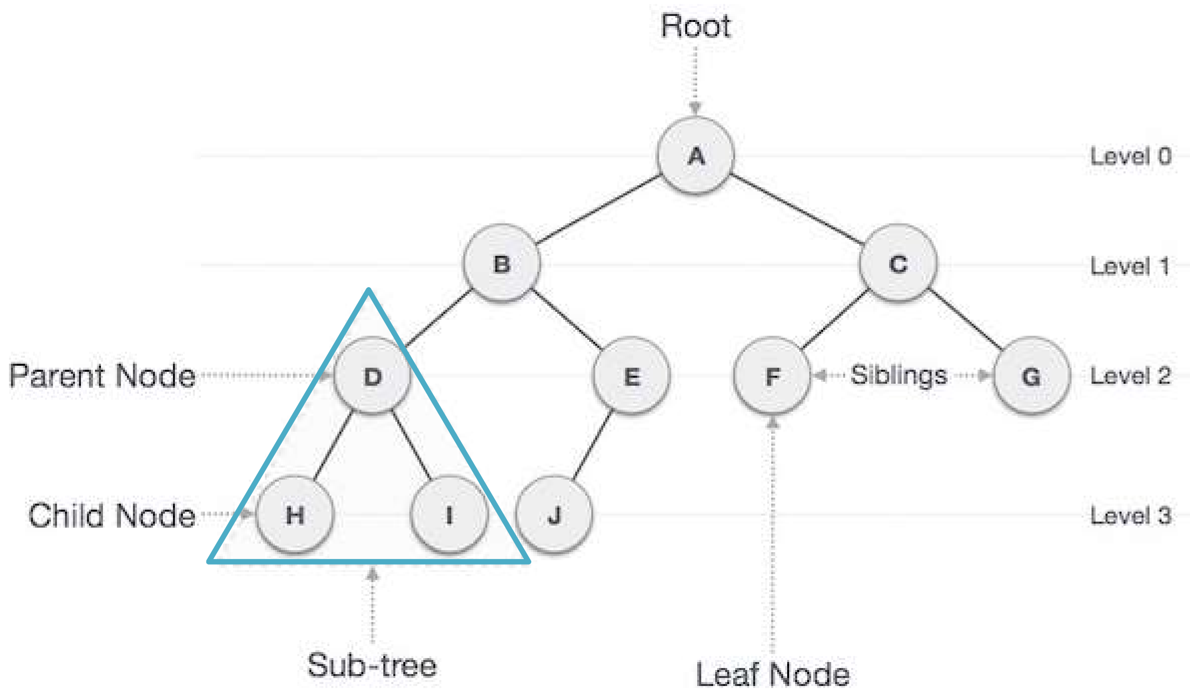


Mathcs.emory.edu

Unit 1: Data Structure

Tree

- **Hierarchical data structure**
- **Tree represents the nodes connected by edges.**
- **If each node is having maximum 2 connected nodes, then it is a binary tree.**

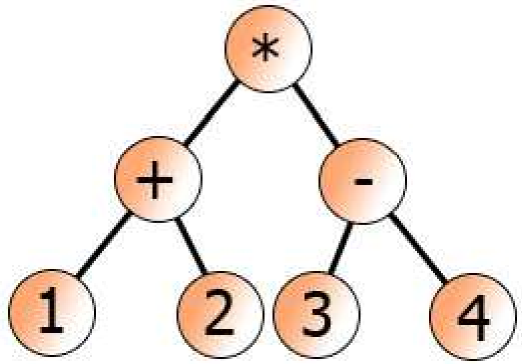


Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- Array
- Stack
- Queue
- Linked List
- **Tree**
- Acknowledgement

Basic Operations on Tree

- Insertion
- Deletion
- Traversal



((1+2)*(3-4))

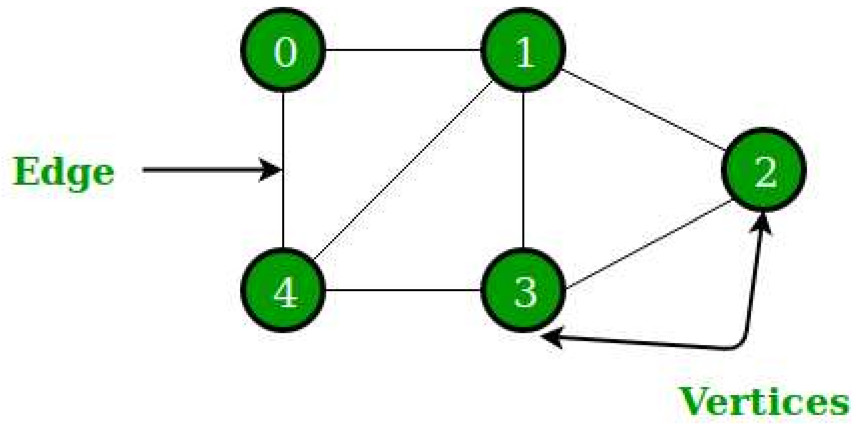
Uses of Tree

- Expression handling
- Compilers
- Searching
- Gaming

Unit 1: Data Structure

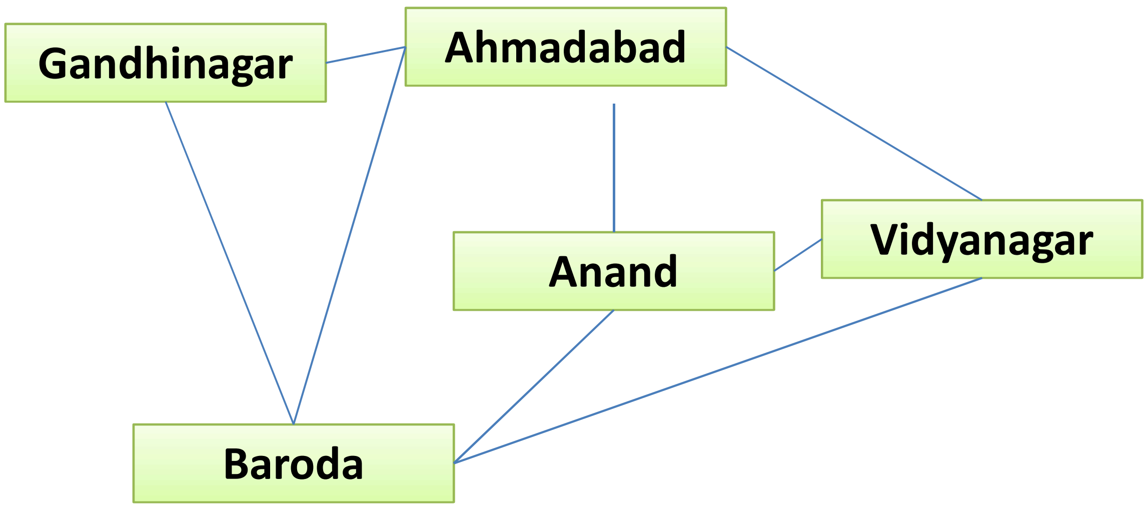
Graph

- A **graph** is represented as a set of vertices (nodes or points) connected by edges (arcs or line)
- It is a non linear data structure



Unit 1: Data Structure

Graph Example



Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- Array
- Stack
- Queue
- Linked List
- **Graph**
- Acknowledgement

Graph

- Graphs are used to represent networks of **cities or telephone network or circuit** network.
- Graphs are also used in **social networks** like linkedIn, Facebook. For example, in Facebook, each person is represented with a vertex(or node). Each node is a structure and contains information like person id, name, gender, locale etc.
- **Basic operations** such as insertion, deletion, traversal, finding path etc are possible on the Graph data structure.

Unit 1: Data Structure

- Introduction
- Need of DT
- Types of DT
- Array
- Stack
- Queue
- Linked List
- Tree & Graphs
- Acknowledgement

Main References

- Tremblay J. & Sorenson P. G. : An Introduction to Data Structures with Applications, 2nd Edition, McGraw-Hill International Edition, 1987.

Other References

- Code project.com, Tutorialspoint.com, Geeksforgeeks.com , etc.