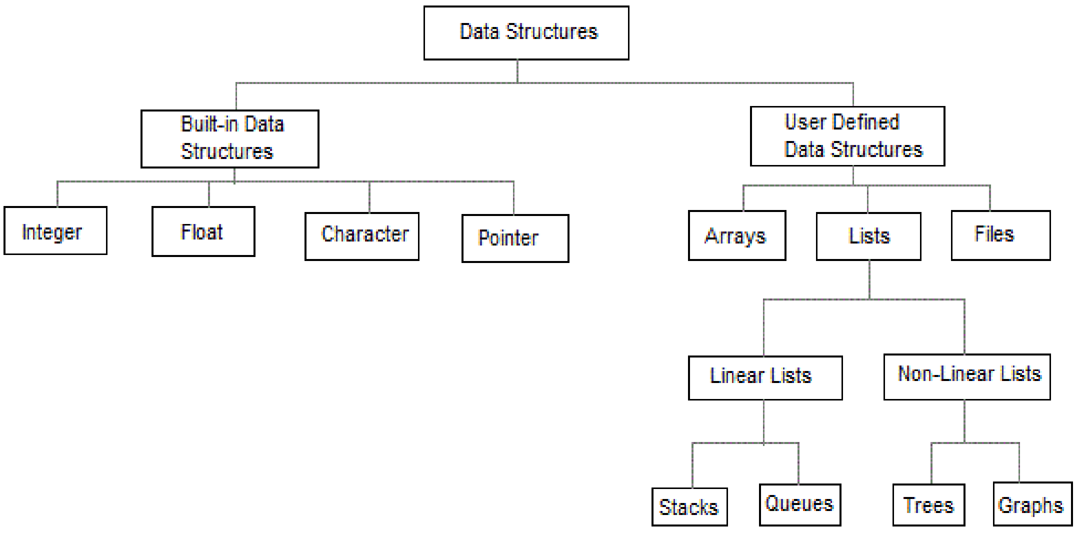
IBPS IT Officer Exam Capsule Part 2

Data Structures

Data Structure is a way of collecting and organising data in such a way that we can perform operations on these data in an effective way. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage. For example, we have data player's name "Virat" and age 26. Here "Virat" is of String data type and 26 is of integer data type.

Basic types of Data Structures

As we discussed above, anything that can store data can be called as a data strucure, hence Integer, Float, Boolean, Char etc, all are data structures. They are known as Primitive Data Structures.



INTRODUCTION TO DATA STRUCTURES

Introduction to Sorting

Sorting is nothing but storage of data in sorted order, it can be in ascending or descending order. The term Sorting comes into picture with the term Searching. There are so many things in our real life that we need to search, like a particular record in database, roll numbers in merit list, a particular telephone number, any particular page in a book etc.

Types of Sorting Techniques

There are many types of Sorting techniques, differentiated by their efficiency and space requirements.

Insertion Sort:0 It is a simple Sorting algorithm which sorts the array by shifting elements one by one. Following are some of the important characteristics of Insertion Sort.

1- It has one of the simplest implementation

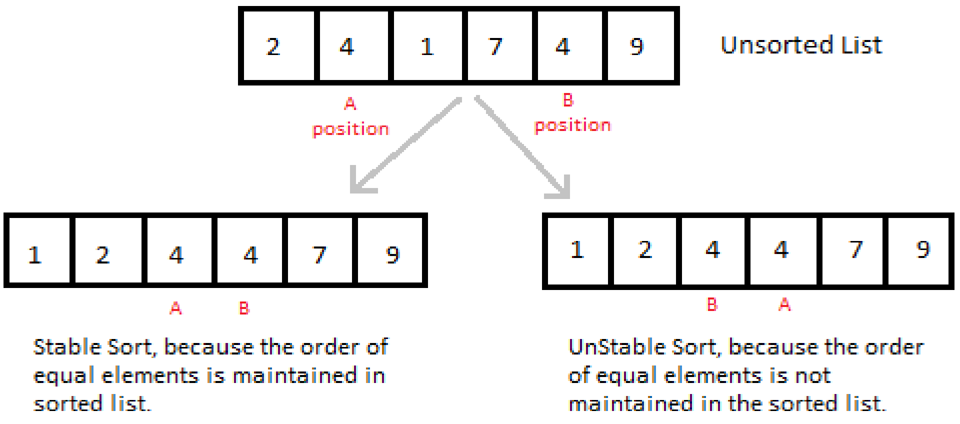
2- It is efficient for smaller data sets, but very inefficient for larger lists.

3- Insertion Sort is adaptive, that means it reduces its total number of steps if given a partially sorted list, hence it increases its efficiency.

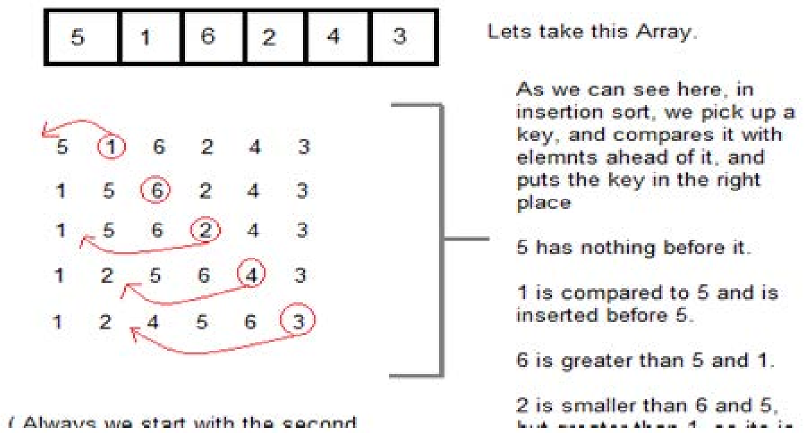
4- It is better than Selection Sort and Bubble Sort algorithms.

5- Its space complexity is less, like Bubble Sorting, inerstion sort also requires a single additional memory space.

6- It is Stable, as it does not change the relative order of elements with equal keys

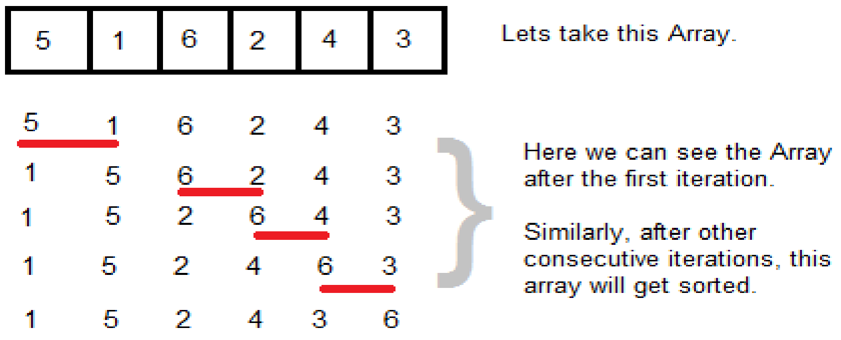

2 4 1 7 4 9
Unsorted List
A position
B position
1 2 4 4 7 9
A B
Stable Sort, because the order of equal elements is maintained in sorted list.
1 2 4 4 7 9
B A
UnStable Sort, because the order of equal elements is not maintained in the sorted list.

How Insertion Sorting Works


5 1 6 2 4 3
Lets take this Array.
As we can see here, in insertion sort, we pick up a key, and compares it with elemnts ahead of it, and puts the key in the right place
5 has nothing before it.
1 is compared to 5 and is inserted before 5.
6 is greater than 5 and 1.
2 is smaller than 6 and 5,

Best Case Time Complexity : O(n)

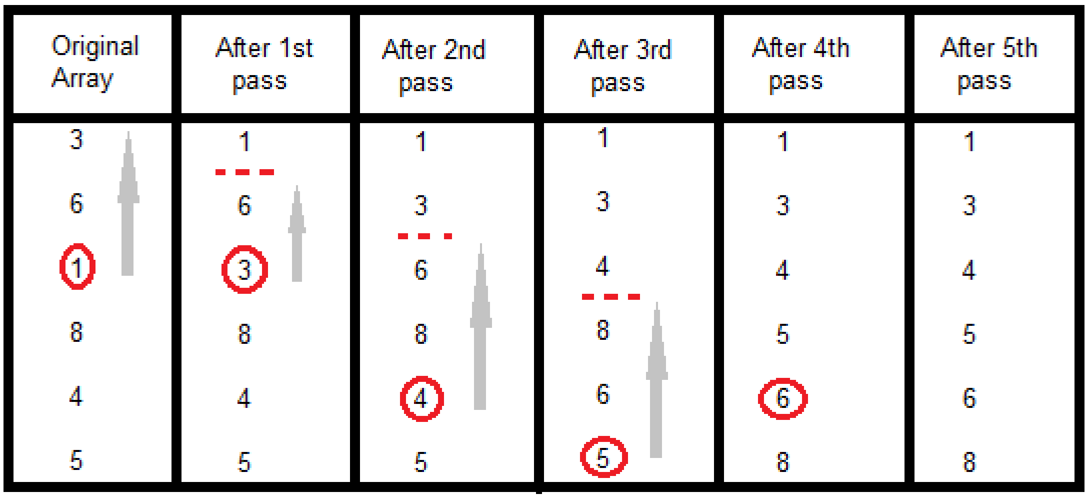
Bubble Sort:- Bubble Sort is an algorithm which is used to sort N elements that are given in a memory for eg: an Array with N number of elements. Bubble Sort compares all the element one by one and sort them based on their values.
It is called Bubble sort, because with each iteration the smaller element in the list bubbles up towards the first place, just like a water bubble rises up to the water surface.
Sorting takes place by stepping through all the data items one-by-one in pairs and comparing adjacent data items and swapping each pair that is out of order.


5 1 6 2 4 3
Lets take this Array.
Here we can see the Array after the first iteration.
Similarly, after other consecutive iterations, this array will get sorted.

Time Complexity will be O(n)

Selection Sort:- Selection sorting is conceptually the most simplest sorting algorithm. This algorithm first finds the smallest element in the array and exchanges it with the element in the first position, then find the second smallest element and exchange it with the element in the second position, and continues in this way until the entire array is sorted.

How Selection Sorting Works



Original Array	After 1st pass	After 2nd pass	After 3rd pass	After 4th pass	After 5th pass
3	1	1	1	1	1
6	6	3	3	3	3
1	3	6	4	4	4
8	8	8	8	5	5
4	4	4	6	6	6
5	5	5	5	8	8

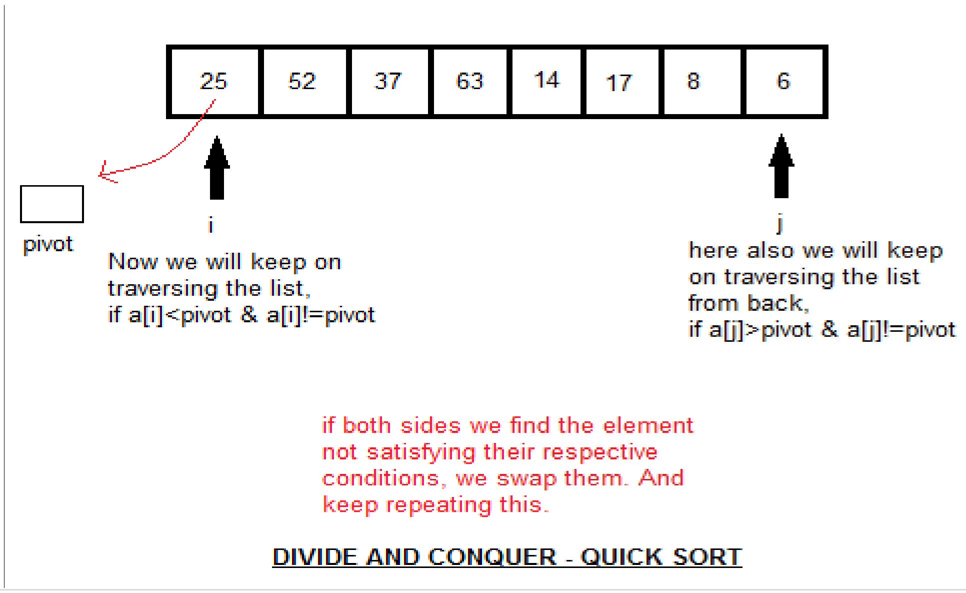
In the first pass, the smallest element found is 1, so it is placed at the first position, then leaving first element, smallest element is searched from the rest of the elements, 3 is the smallest, so it is then placed at the second position. Then we leave 1 nad 3, from the rest of the elements, we search for the smallest and put it at third position and keep doing this, until array is sorted.

Best Case Time Complexity : O(n2)

Quick Sort :- Quick Sort, as the name suggests, sorts any list very quickly. Quick sort is not stable search, but it is very fast and requires very less aditional space. It is based on the rule of Divide and Conquer(also called partition-exchange sort). This algorithm divides the list into three main parts :

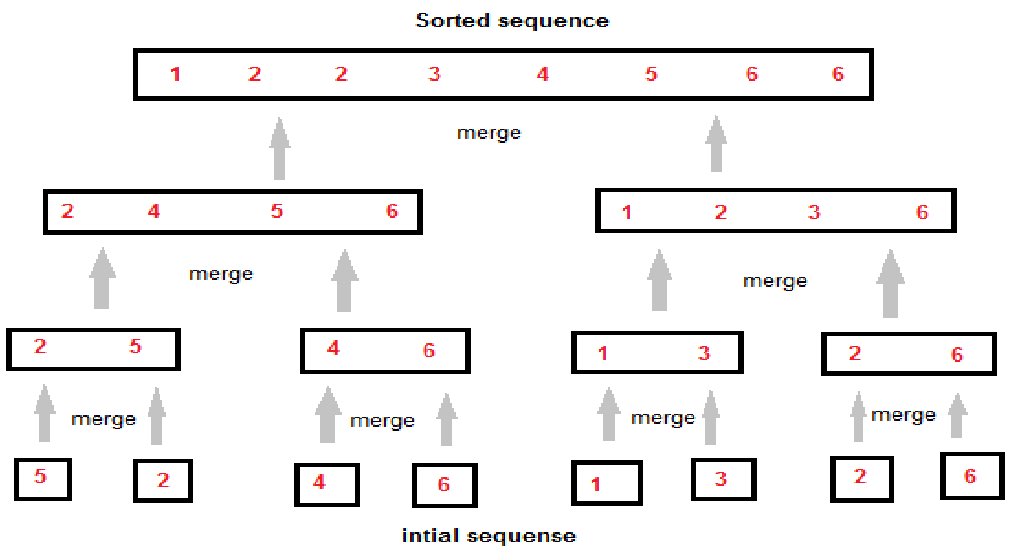
1- Elements less than the Pivot element
2- Pivot element
3- Elements greater than the pivot element

How Quick Sorting Works


25
52
37
63
14
17
8
6
pivot
i
Now we will keep on traversing the list, if a[i]<pivot & a[i]!=pivot
j
here also we will keep on traversing the list from back, if a[j]>pivot & a[j]!=pivot
if both sides we find the element not satisfying their respective conditions, we swap them. And keep repeating this.
DIVIDE AND CONQUER - QUICK SORT

Best Case Time Complexity : O(n log n)

Merge Sort:- Merge Sort follows the rule of Divide and Conquer. But it doesn't divides the list into two halves. In merge sort the unsorted list is divided into N sublists, each having one element, because a list of one element is considered sorted. Then, it repeatedly merge these sublists, to produce new sorted sublists, and at lasts one sorted list is produced.



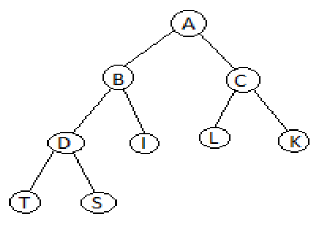
Best Case Time Complexity : O(n log n)

Heap Sort :- Heap Sort is one of the best sorting methods being in-place and with no quadratic worst-case scenarios. Heap sort algorithm is divided into two basic parts :

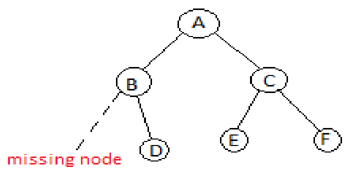
1- Creating a Heap of the unsorted list.
2- Then a sorted array is created by repeatedly removing the largest/smallest element from the heap, and inserting it into the array. The heap is reconstructed after each removal**.**

What is a Heap ?
Heap is a special tree-based data structure, that satisfies the following special heap properties :

1- **Shape Property :** Heap data structure is always a Complete Binary Tree, which means all levels of the tree are fully filled.

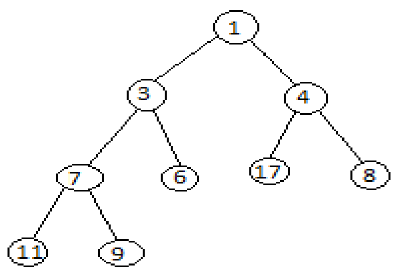


Complete Binary Tree



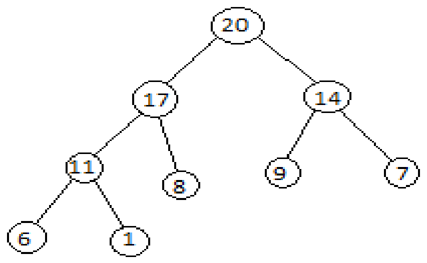
In-Complete Binary Tree

2- **Heap Property :** All nodes are either [greater than or equal to] or [less than or equal to] each of its children. If the parent nodes are greater than their children, heap is called a Max-Heap, and if the parent nodes are smalled than their child nodes, heap is called Min-Heap



Min-Heap

In min-heap, first element is the smallest. So when we want to sort a list in ascending order, we create a Min-heap from that list, and picks the first element, as it is the smallest, then we repeat the process with remaining elements.



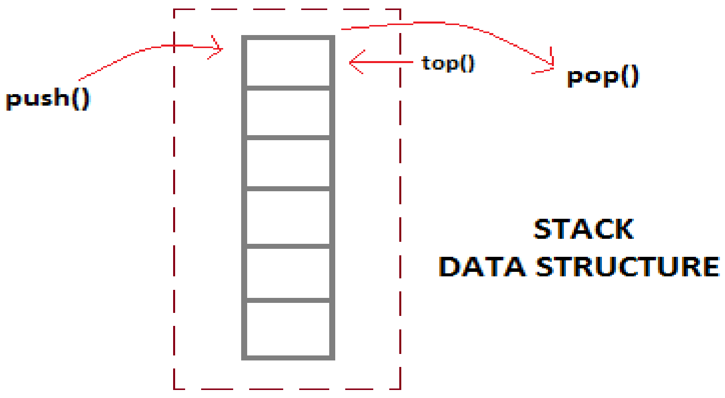
Max-Heap

In max-heap, the first element is the largest, hence it is used when we need to sort a list in descending order.

Best Case Time Complexity : O(n log n)

Stacks

Stack is an abstract data type with a bounded(predefined) capacity. It is a simple data structure that allows adding and removing elements in a particular order. Every time an element is added, it goes on the top of the stack, the only element that can be removed is the element that was at the top of the stack, just like a pile of objects.



Basic features of Stack

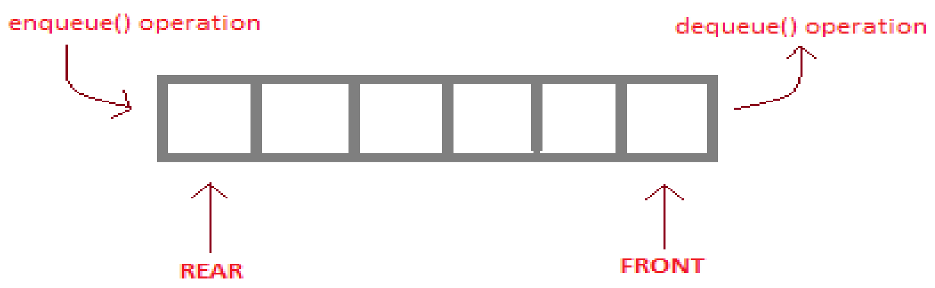
1- Stack is an ordered list of similar data type.
2- Stack is a LIFO structure. **(Last in First out).**
3- **push() function** is used to insert new elements into the Stack and pop() is used to delete an element from the stack. Both insertion and deletion are allowed at only one end of Stack called Top.
4- Stack is said to be in Overflow state when it is completely full and is said to be in Underflow state if it is completely empty.

Applications of Stack

1- The simplest application of a stack is to reverse a word. You push a given word to stack - letter by letter - and then pop letters from the stack.
2- There are other uses also like : Parsing, Expression Conversion(Infix to Postfix, Postfix to Prefix etc) and many more.

Queue Data Structures

Queue is also an abstract data type or a linear data structure, in which the first element is inserted from one end called REAR(also called tail), and the deletion of exisiting element takes place from the other end called as FRONT(also called head). This makes queue as FIFO data structure, which means that element inserted first will also be removed first.
The process to add an element into queue is called Enqueue and the process of removal of an element from queue is called Dequeue.



enqueue( ) is the operation for adding an element into Queue.
dequeue( ) is the operation for removing an element from Queue .

QUEUE DATA STRUCTURE

Basic features of Queue

1- Like Stack, Queue is also an ordered list of elements of similar data types.
2- Queue is a FIFO(First in First Out) structure.
3- Once a new element is inserted into the Queue, all the elements inserted before the new element in the queue must be removed, to remove the new element.
4- peek( ) function is oftenly used to return the value of first element without dequeuing it.

Applications of Queue

1- Queue, as the name suggests is used whenever we need to have any group of objects in an order in which the first one coming in, also gets out first while the others wait for there turn, like in the following scenarios :

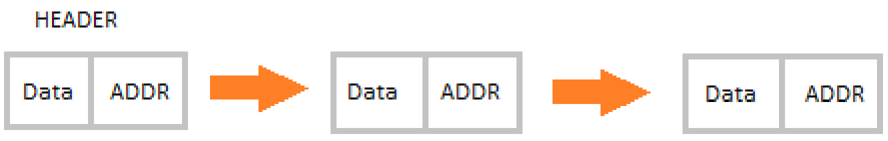
Serving requests on a single shared resource, like a printer, CPU task scheduling etc.

2- In real life, Call Center phone systems will use Queues, to hold people calling them in an order, until a service representative is free.
3- Handling of interrupts in real-time systems. The interrupts are handled in the same order as they arrive, First come first served.

A standard queue:- The main property of a queue is that we have access to the item at the front of the queue. The queue data structure can be efficiently implemented using a singly linked list
Priority Queue:- Items in a priority queue being ordered by priority it remains the same as a normal queue, you can only access the item at the front of the queue.
Double Ended Queue:- Double ended queue allows you to access the items at both the front, and back of the queue. A double ended queue is commonly known as a deque.

Introduction to Linked Lists

Linked List is a linear data structure and it is very common data structure which consists of group of nodes in a sequence which is divided in two parts. Each node consists of its own data and the address of the next node and forms a chain. Linked Lists are used to create trees and graphs.

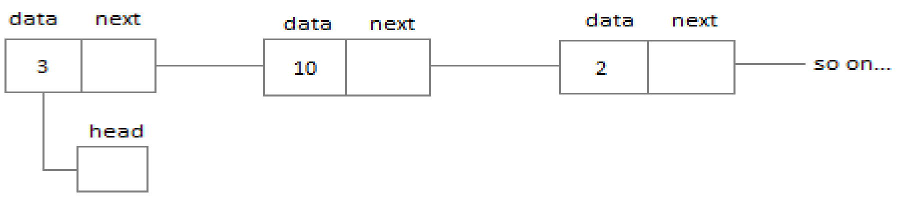


Applications of Linked Lists

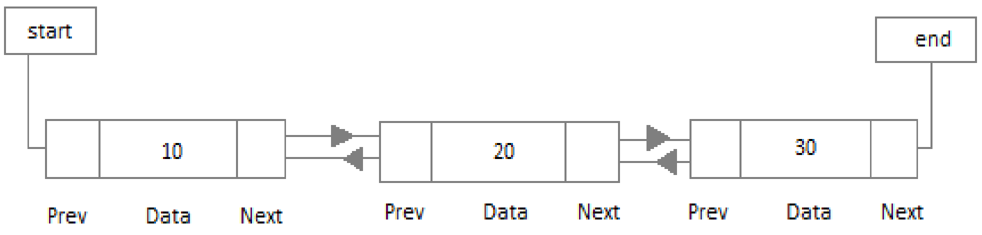
1- Linked lists are used to implement stacks, queues, graphs, etc.
2- Linked lists let you insert elements at the beginning and end of the list.
3- In Linked Lists we don't need to know the size in advance.

Types of Linked Lists

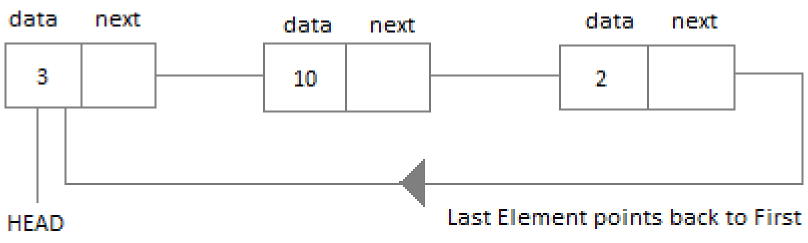
Singly Linked List :- Singly linked lists contain nodes which have a data part as well as an address part i.e. next, which points to the next node in sequence of nodes. The operations we can perform on singly linked lists are insertion, deletion and traversal.



Doubly Linked List :- In a doubly linked list, each node contains two links the first link points to the previous node and the next link points to the next node in the sequence.



Circular Linked List :- In the circular linked list the last node of the list contains the address of the first node and forms a circular chain.



Linear Linked List:- The element can be inserted in linked list in 2 ways :

1- Insertion at beginning of the list.
2- Insertion at the end of the list.

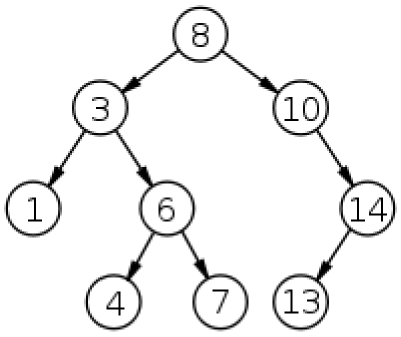
We will also be adding some more useful methods like :

1- Checking whether Linked List is empty or not.
2- Searching any element in the Linked List
3- Deleting a particular Node from the List

Binary Search Tree

Binary Search Tree, is a node-based binary tree data structure which has the following properties:

1- The left subtree of a node contains only nodes with keys less than the node's key.
2- The right subtree of a node contains only nodes with keys greater than the node's key.
3- The left and right subtree each must also be a binary search tree.
4- There must be no duplicate nodes

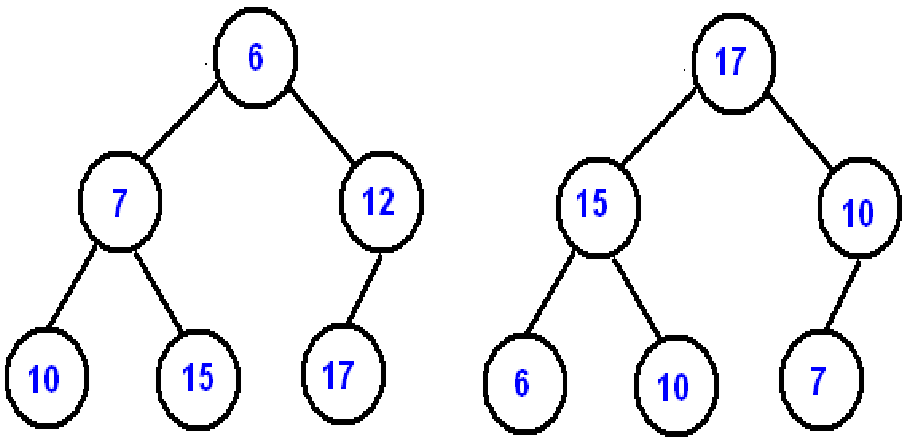


Time Complexity:O(n)

Binary Heaps

A binary heap is a complete binary tree which satisfies the heap ordering property. The ordering can be one of two types:

1- the min-heap property: the value of each node is greater than or equal to the value of its parent, with the minimum-value element at the root.
2- the max-heap property: the value of each node is less than or equal to the value of its parent, with the maximum-value element at the root.

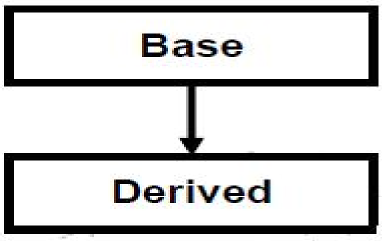


Time Complexity: O(n log n)

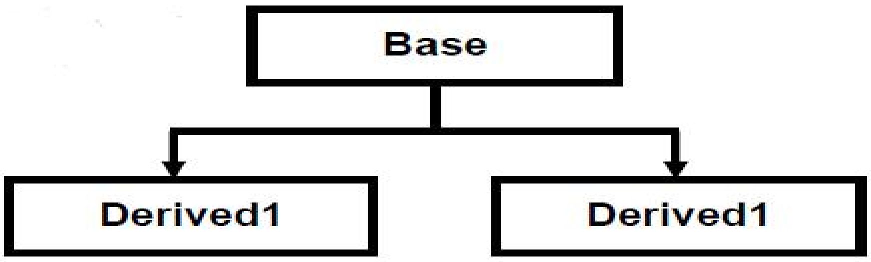
BASIC CONCEPT OF OBJECT ORIENTED PROGRAMMING LANGUAGE

Object:- Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.
Class:- Collection of objects is called class. It is a logical entity.
Inheritance:- When one object acquires all the properties and behaviors of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.
Inheritance can be classified to 5 types

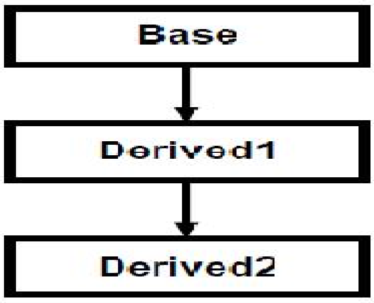
1. **Single Inheritance**:- when a single derived class is created from a single base class then the inheritance is called as single inheritance.



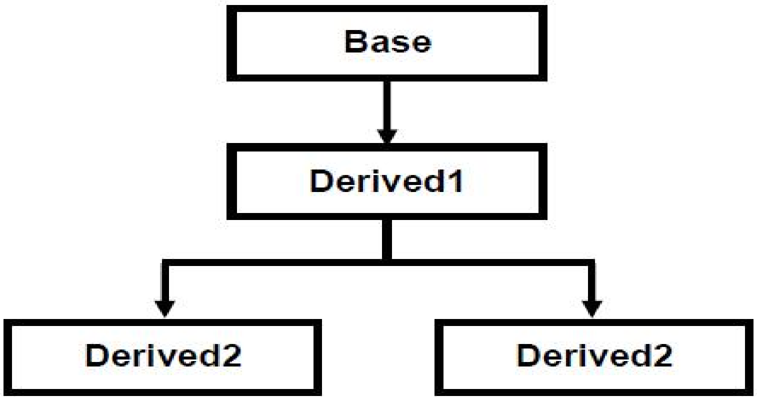
2. Hierarchical Inheritance:- when more than one derived class are created from a single base class, then that inheritance is called as hierarchical inheritance.



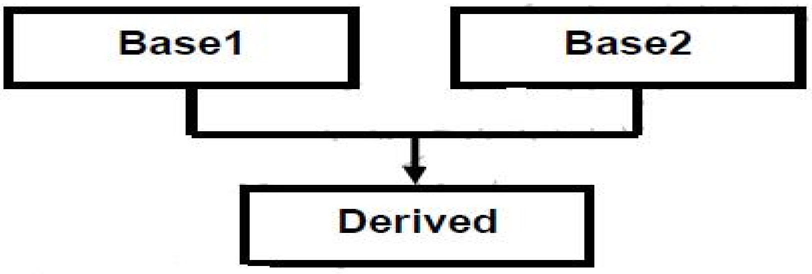
2. **Multi Level Inheritance:-** when a derived class is created from another derived class, then that inheritance is called as multi level inheritance.



3. Hybrid Inheritance:-Any combination of single, hierarchical and multi level inheritances is called as hybrid inheritance.



5. Multiple Inheritance:- when a derived class is created from more than one base class then that inheritance is called as multiple inheritance. But multiple inheritance is not supported by .net using classes and can be done using interfaces.



Polymorphism:- When one task is performed by different ways i.e. known as polymorphism. For example: to convense the customer differently, to draw something e.g. shape or rectangle etc.
In java, we use method overloading and method overriding to achieve polymorphism.
Another example can be to speak something e.g. cat speaks meow, dog barks woof etc.
Abstraction:- Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing.
Encapsulation:- Binding (or wrapping) code and data together into a single unit is known as encapsulation
Advantage of OOPs over Procedure-oriented programming language
1) OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.
2) OOPs provide data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere.
3) OOPs provides ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.

SOFTWARE ENGINEERING

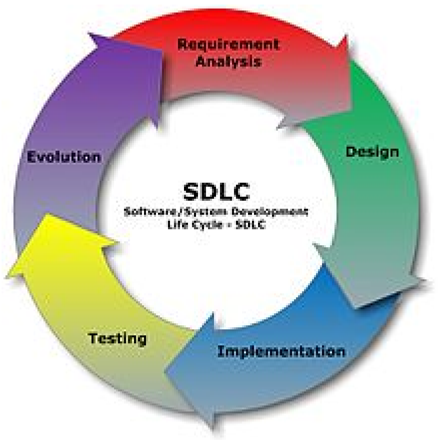
SDLC, Software Development Life Cycle is a process used by software industry to design, develop and test high quality softwares. The SDLC aims to produce a high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

1- SDLC is the acronym of Software Development Life Cycle.

2- It is also called as Software development process.

3- The software development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process.

4- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.
A typical Software Development life cycle consists of the following stages:



Stage 1: Planning and Requirement Analysis: Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry.
Stage 2: Defining Requirements: Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts
Stage 3: Designing the product architecture: SRS is the reference for product architects to come out with the best architecture for the product to be developed
Stage 4: Building or Developing the Product : In this stage of SDLC the actual development starts and the product is built
Stage 5: Testing the Product : This stage is usually a subset of all the stages as in the modern SDLC models
Stage 6: Deployment in the Market and Maintenance : Once the product is tested and ready to be deployed it is released formally in the appropriate market
Popular SDLC models followed in the industry:
1- **Waterfall Model**
2- **Iterative Model**
3- **Spiral Model**
4- **V-Model**
5- **Big Bang Model**

Black Box Testing:- Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester. Generally, independent Software Testers
Mainly applicable to **higher levels of testing:**
Acceptance Testing
System Testing
White Box Testing:- White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester. Generally, Software Developers.

Mainly applicable to **lower levels of testing:**
Unit Testing
Integration Testing

Software engineering is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product.
The process of developing a software product using software engineering principles and methods is referred to as Software Evolution

Characteristics of good software

A software product can be judged by what it offers and how well it can be used. This software must satisfy on the following grounds:

A- Operational
B- Transitional
C- Maintenance

Well-engineered and crafted software is expected to have the following characteristics:

Operational:- This tells us how well the software works in operations. It can be measured on:

- Budget
- Usability
- Efficiency
- Correctness
- Functionality
- Dependability
- Security
- Safety

Transitional:- This aspect is important when the software is moved from one platform to another:

A- Portability
B- Interoperability
C- Reusability
D- Adaptability

Maintenance:- This aspect briefs about how well the software has the capabilities to maintain itself in the ever-changing environment:

A- Modularity
B- Maintainability
C- Flexibility
D- Scalability

AffairsCloud Important Page Links

Current Affairs Sections

- Current Affairs
- **Current Affairs 2016**
- Current Affairs Today
- Current Affairs PDF
- Current Affairs Quiz
- Current Affairs Mock Test
- Study Materials PDF

Subject-Wise Question Sections

- Reasoning Questions
- Quantitative Aptitude Questions
- English Questions
- Computer Questions
- Banking Awareness Questions

Study Guide Sections

- General Aptitude
- Reasoning
- Letter Writing
- Math Tricks
- Computer Awareness

GOOD LUCK WITH YOUR EXAMS!!!

Suggestions are welcomed; Contact us any time at **Affairscloud@gmail.com & AffairsCloud1@gmail.com**