Search algorithms

-In computer science, a linear search or **sequential search** is a method for finding an element within a list. It sequentially checks each element of the list until a match is found or the whole list has been searched.

-In computer science, **binary search**, is a search algorithm that finds the position of a target value within a sorted array.

The basic steps to perform Binary Search are:

Begin with the mid element of the whole array as a search key.

If the value of the search key is equal to the item then return an index of the search key.

Or if the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half.

Otherwise, narrow it to the upper half.

Repeatedly check from the second point until the value is found or the interval is empty.

Sort algorithms

-The **Selection** sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning.

In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

-**Bubble** Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order.

-Like Merge Sort, **QuickSort** is a Divide and Conquer algorithm.

It works by selecting a 'pivot' element from the array and partitioning the other elements into two sub-arrays, according to whether they are less than or greater than the pivot.

**- Insertion** sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

The **Merge Sort** algorithm is a sorting algorithm that is considered as an example of the divide and conquer strategy. So, in this algorithm, the array is initially divided into two equal halves and then they are combined in a sorted manner.

A picture containing timeline

Description automatically generated

-**Abstract Data type** (ADT) is a type (or class) for objects whose behavior is defined by a set of values and a set of operations. The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented.

Examples: Array, List, Map, Queue, Set, Stack, Table, Tree, and Vector are ADTs.

**OOP principles**

oop principles-data abstraction, encapsulation, inheritance, polymorphism

**data abstraction** - interfaces and abstract classes, hide unnecesary details

**encapsulation** - the idea of bundling data and methods that work on that data within one unit, like a class in Java. This concept is also often used to hide the internal representation, or state of an object from the outside

**inheritance** - derived classes, a child inherits method/attribute from parent. code reusability(single vs multi vs hierarchical level)

**polymorphism** - ability of an object to action in different ways

1. **Static binding**/Compile-Time binding/Early binding/Method **overloading**.(in same class)

2. **Dynamic binding**/Run-Time binding/Late binding/Method **overriding**.(in different classes)

Overloaded methods MUST change the argument list

Overloaded methods CAN change the return type

Overloaded methods CAN change the access modifier

Overloaded methods CAN declare new or broader checked exceptions

overload- compile time, change the number of parameters, type of parameters, return type of function, for example in constructors with different parameters

override- run time, change the definition of a function in a subclass, same return type and parameters, for example animal class with walk method

A **class** defines a set of properties and methods that are common to all objects of one type. (blueprint for creating objects)

An **object** is an instance of a class. An object will be created with a set of attributes.

**Association** - I have a relationship with an object. Foo uses Bar

**Composition** - I own an object and I am responsible for its lifetime. When Foo dies, so does Bar

**Aggregation** - I have an object which I've borrowed from someone else. When Foo dies, Bar may live on.

**Constructor** is used to initialize an object of the class and assign values to data members corresponding to the class. While **destructor** is used to deallocate the memory of an object of a class.

**Access modifiers** (or access specifiers) are keywords in object-oriented languages that set the accessibility of classes, methods, and other members.

Table

Description automatically generated

-A **virtual** function is a member function which is declared within a base class and is re-defined(Overriden) by a derived class

-A **pure virtual**(abstract) function or pure virtual method is a virtual function that is required to be implemented by a derived class if the derived class is not abstract. Classes containing pure virtual methods are termed "abstract" and they cannot be instantiated directly.

Type of methods: Interface can have only abstract methods. An abstract class can have abstract and non-abstract methods. From Java 8, it can have default and static methods also.

Final Variables: Variables declared in a Java interface are by default final. An abstract class may contain non-final variables.

Type of variables: Abstract class can have final, non-final, static and non-static variables. The interface has only static and final variables.

Implementation: Abstract class can provide the implementation of the interface. Interface can’t provide the implementation of an abstract class.

Inheritance vs Abstraction: A Java interface can be implemented using the keyword “implements” and an abstract class can be extended using the keyword “extends”.

Multiple implementations: An interface can extend another Java interface only, an abstract class can extend another Java class and implement multiple Java interfaces.

Accessibility of Data Members: Members of a Java interface are public by default. A Java abstract class can have class members like private, protected, etc.