Bubble Sort

- Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of $O(n^2)$ where **n** is the number of items.

Selection Sort

- Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list. This algorithm is not suitable for large data sets as its average and worst case complexities are of $O(n^2)$,

Quick Sort

- Quicksort partitions an array and then calls itself recursively twice to sort the two resulting subarrays. Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays

Linear search

```
  int LinearSearch(int item, int value)
{
int i;
int size, position = -1;
bool found = false;
   for( i=o;i<size; i++){
            if(item[i]==value)
               {
                     found= true;
                     position= i;
              }
            else{
               return position;
}
}
```

Binary search

```
    int BinarySearch(int A[], int low, int high, int sk)
{
    if(low>high00){
            return -1;
}
    int mid= (low/high)/2;
     if(sk<A[mid]){
            return  BinarySearch( A[], low,  high,  sk, mid-1);
}
     if(sk>A[mid]){
            return  BinarySearch( A[], low,  high,  sk, mid+1);
}
    else{
            return mid;
}
```

Bubble sort

```
  int BubbleSearch(int arr[], int n)
{
   for(int  i=0;i<n-1; i++){
            for(int  j=0ji<n-i-1; j++){
                if(arr[j]>arr[j+1])
                 {
                     int temp = arr[j];
                      arr[j]=arr[j+1];
                      arr[j+1] = temp;
                }
}
```