Bubble sort

1. Traverse from left to right everytime and place the higher ones at right side.

2. Time complexity - O(n*n)

3. Auxiliary space - O(1)

In [13]:

```
class Solution:
    def bubbleSort(self,arr):
        l=len(arr)
        for i in range(l):
            flag=0
            for j in range(l-i-1):
                if (arr[j+1]<arr[j]):
                    arr[j+1],arr[j]=arr[j],arr[j+1]
                    flag=1
            if(flag==0):
                return arr
        return arr
obj=Solution()
obj.bubbleSort([5,2,7,9,1,0,3,4,8,6])
```

Out[13]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

insertion sort

insertion sort traverse from 1 to len(Arr) and shifts right elements to left position

In [11]:

```
class Solution:
    def insertionSort(self,arr):
        for i in range(1,len(arr)):
            val=arr[i]
            gap=i
            while gap>0 and arr[gap-1]>val:
                arr[gap]=arr[gap-1]
                gap-=1
            arr[gap]=val
        return arr
obj=Solution()
obj.insertionSort([5,2,7,9,1,0,3,3,3,4,8,6])
```

Out[11]:

```
[0, 1, 2, 3, 3, 3, 4, 5, 6, 7, 8, 9]
```

selection sort

The selection sort algorithm sorts list by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm or logic will maintains two sublists in a given list.

1. The sublist which is already sorted.

2. Remaining sublist which is unsorted.

In [9]:

```
class Solution:
    def selectionsort(self,arr):
        for i in range(len(arr)):
            ind=i
            for j in range(i+1,len(arr)):
                if arr[j]<arr[ind]:
                    ind=j
            arr[i],arr[ind]=arr[ind],arr[i]
        return arr
obj=Solution()
obj.selectionsort([5,2,7,9,1,0,3,3,3,4,8,6])
```

Out[9]:

```
[0, 1, 2, 3, 3, 3, 4, 5, 6, 7, 8, 9]
```

merge sort

In [2]:

```
def merge(left,right):
    result=[]
    i,j=0,0
    while(i<len(left) and j<len(right)):
        if (left[i]<right[j]):
            result+=[left[i]]
            i+=1
        else:
            result+=[right[j]]
            j+=1
    result+=left[i:]+right[j:]
    return result


def mergesort(arr):
    if(len(arr)<=1):
        return arr
    mid=len(arr)//2
    left=mergesort(arr[:mid])
    right=mergesort(arr[mid:])
    return merge(left,right)

arr=[1,2,5,-1,-4,0,8,5]
mergesort(arr)
```

Out[2]:

```
[-4, -1, 0, 1, 2, 5, 5, 8]
```

Quick Sort using list comprehension

In [41]:

```
def quicksort(arr):
    if len(arr)<=1:
        return arr
    else:
        pivot=arr[0]
        left=[i for i in arr[1:] if i<pivot]
        right=[i for i in arr[1:] if i>=pivot]
        return quicksort(left)+[pivot]+quicksort(right)
arr=[1,2,5,-1,-4,0,8,5]
quicksort(arr)
```

Out[41]:

```
[-4, -1, 0, 1, 2, 5, 5, 8]
```

Qucik sort

In [34]:

```
def quicksort(arr,left,right):
    if left<right:
        partition_pos=partition(arr,left,right)
        quicksort(arr,left,partition_pos-1)     # calling quick sort on left array
        quicksort(arr,partition_pos+1,right)    # calling quick sort on right array
    return arr
def partition(arr,left,right):    # returns index of pivot element
    i=left
    j=right-1
    pivot=arr[right]
    while i<j:
        while arr[i]<pivot and i<right:
            i+=1
        while arr[j]>=pivot and j>=i:
            j-=1
        if i<=j:
            arr[i],arr[j]=arr[j],arr[i]
    if arr[i]>pivot:
        arr[i],arr[right]=arr[right],arr[i]
    return i
arr=[22,11,88,66,55,77,33,44]
quicksort(arr,0,len(arr)-1)
```

Out[34]:

```
[11, 22, 33, 44, 55, 66, 77, 88]
```

In [42]:

```
def partition(array, low, high):
    pivot = array[high]
    i = low-1
    for j in range(low, high):
        if array[j] <= pivot:
            i = i + 1
            (array[i], array[j]) = (array[j], array[i])
    (array[i + 1], array[high]) = (array[high], array[i + 1])
    return i + 1
def quickSort(array, low, high):
    if low < high:
        pi = partition(array, low, high)
        quickSort(array, low, pi - 1)
        quickSort(array, pi + 1, high)
data = [1, 7, 4, 1, 10, 9, -2]
size = len(data)
quickSort(data, 0, size - 1)
print(data)
```

```
[-2, 1, 1, 4, 7, 9, 10]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [40]:

```
class Node:
    def __init__(self,val):
        self.val=val
        self.next=None

class LinkedList():
    def __init__(self):
        self.head=None

    def print(self):
        if not self.head: return 'empty'
        itr=self.head
        lstr=''
        while itr:
            lstr+=str(itr.val)+'-->' if itr.next else str(itr.val)
            itr=itr.next
        return lstr
    def reverse(self):
        if not self.head: return 'empty'
        curr=self.head
        prev=None
        while(curr):
            next=curr.next
            curr.next=prev
            prev=curr
            curr=next
        self.head=prev
        return self.print()
SLL=LinkedList()
```

In [41]:

```
arr=[3,1,5,2,7,8,4]
SLL.head=Node(arr[0])
temp=SLL.head
for i in range(1,len(arr)):
    temp.next=Node(arr[i])
    temp=temp.next
```

In [42]:

```
SLL.print()
```

Out[42]:

```
'3-->1-->5-->2-->7-->8-->4'
```

In [43]:

```
SLL.reverse()
```

Out[43]:

```
'4-->8-->7-->2-->5-->1-->3'
```

In [ ]:

```

```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: