1. Python Program for Recursive Insertion Sort

In [1]:
```
def insertionSortRecursive(arr,n):
    # base case
    if n<=1:
        return

    insertionSortRecursive(arr,n-1)
    '''Insert last element at its correct position
        in sorted array.'''
    last = arr[n-1]
    j = n-2

    while (j>=0 and arr[j]>last):
        arr[j+1] = arr[j]
        j = j-1

    arr[j+1]=last

def printArray(arr,n):
    for i in range(n):
        print(arr[i],end=" ")

arr = [12,11,13,5,6]
n = len(arr)
insertionSortRecursive(arr, n)
printArray(arr, n)
```

```
5 6 11 12 13
```

2. Python Program for QuickSort

In [2]:
```
def partition(l, r, nums):

    pivot, ptr = nums[r], l
    for i in range(l, r):
        if nums[i] <= pivot:

            nums[i], nums[ptr] = nums[ptr], nums[i]
            ptr += 1

    nums[ptr], nums[r] = nums[r], nums[ptr]
    return ptr

def quicksort(l, r, nums):
    if len(nums) == 1:
        return nums
    if l < r:
        pi = partition(l, r, nums)
        quicksort(l, pi-1, nums)
        quicksort(pi+1, r, nums)
    return nums


example = [4, 5, 1, 2, 3]
result = [1, 2, 3, 4, 5]
print(quicksort(0, len(example)-1, example))

example = [2, 5, 6, 1, 4, 6, 2, 4, 7, 8]
result = [1, 2, 2, 4, 4, 5, 6, 6, 7, 8]

print(quicksort(0, len(example)-1, example))
```

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

3.Python Program for Iterative Quick Sort

In [3]:
```
def partition(arr,l,h):
    i = ( l - 1 )
    x = arr[h]

    for j in range(l , h):
        if   arr[j] <= x:

            i = i+1
            arr[i],arr[j] = arr[j],arr[i]

    arr[i+1],arr[h] = arr[h],arr[i+1]
    return (i+1)

def quickSortIterative(arr,l,h):

    size = h - l + 1
    stack = [0] * (size)

    top = -1

    top = top + 1
    stack[top] = l
    top = top + 1
    stack[top] = h

    while top >= 0:

        h = stack[top]
        top = top - 1
        l = stack[top]
        top = top - 1

        p = partition( arr, l, h )

        if p-1 > l:
            top = top + 1
            stack[top] = l
            top = top + 1
            stack[top] = p - 1

        if p+1 < h:
            top = top + 1
            stack[top] = p + 1
            top = top + 1
            stack[top] = h

arr = [4, 3, 5, 2, 1, 3, 2, 3]
n = len(arr)
quickSortIterative(arr, 0, n-1)
print ("Sorted array is:")
for i in range(n):
    print ("%d" %arr[i]),
```

```
Sorted array is:
1
2
2
3
3
3
4
5
```

4. Python Program for Selection Sort.

In [4]:
```
import sys
A = [64, 25, 12, 22, 11]

for i in range(len(A)):

 min_idx = i
 for j in range(i+1, len(A)):
  if A[min_idx] > A[j]:
   min_idx = j

 A[i], A[min_idx] = A[min_idx], A[i]

print ("Sorted array")
for i in range(len(A)):
 print("%d" %A[i]),
```

```
Sorted array
11
12
22
25
64
```

5.Python Program for Bubble Sort .

In [5]:
```
def bubbleSort(arr):
    n = len(arr)

    swapped = False

    for i in range(n-1):

        for j in range(0, n-i-1):

            if arr[j] > arr[j + 1]:
                swapped = True
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

        if not swapped:

            return

arr = [64, 34, 25, 12, 22, 11, 90]

bubbleSort(arr)

print("Sorted array is:")
for i in range(len(arr)):
    print("%d" % arr[i], end=" ")
```

```
Sorted array is:
11  12  22  25  34  64  90
```

6. Python Program for Merge Sort.

In [6]:
```
def merge(arr, l, m, r):
    n1 = m - l + 1
    n2 = r - m

    L = [0] * (n1)
    R = [0] * (n2)

    for i in range(0, n1):
        L[i] = arr[l + i]

    for j in range(0, n2):
        R[j] = arr[m + 1 + j]

    i = 0
    j = 0
    k = l

    while i < n1 and j < n2:
        if L[i] <= R[j]:
            arr[k] = L[i]
            i += 1
        else:
            arr[k] = R[j]
            j += 1
        k += 1

    while i < n1:
        arr[k] = L[i]
        i += 1
        k += 1

    while j < n2:
        arr[k] = R[j]
        j += 1
        k += 1


def mergeSort(arr, l, r):
    if l < r:

        m = l+(r-l)//2

        mergeSort(arr, l, m)
        mergeSort(arr, m+1, r)
        merge(arr, l, m, r)

arr = [12, 11, 13, 5, 6, 7]
n = len(arr)
print("Given array is")
for i in range(n):
    print("%d" % arr[i],end=" ")

mergeSort(arr, 0, n-1)
print("\n\nSorted array is")
for i in range(n):
    print("%d" % arr[i],end=" ")
```

```
Given array is
12 11 13 5 6 7

Sorted array is
5 6 7 11 12 13
```

7. Python Program for Iterative Merge Sort.

In [7]:
```
def merge(left, right):
    if not len(left) or not len(right):
        return left or right

    result = []
    i, j = 0, 0
    while (len(result) < len(left) + len(right)):
        if left[i] < right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
        if i == len(left) or j == len(right):
            result.extend(left[i:] or right[j:])
            break

    return result

def mergesort(list):
    if len(list) < 2:
        return list

    middle = int(len(list)/2)
    left = mergesort(list[:middle])
    right = mergesort(list[middle:])

    return merge(left, right)

seq = [12, 11, 13, 5, 6, 7]
print("Given array is")
print(seq);
print("\n");
print("Sorted array is")
print(mergesort(seq))
```

```
Given array is
[12, 11, 13, 5, 6, 7]


Sorted array is
[5, 6, 7, 11, 12, 13]
```

8. Python Program for Heap Sort.

In [8]:
```
def heapify(arr, n, i):
    largest = i
    l = 2 * i + 1
    r = 2 * i + 2

    if l < n and arr[i] < arr[l]:
        largest = l

    if r < n and arr[largest] < arr[r]:
        largest = r

    if largest != i:
        arr[i],arr[largest] = arr[largest],arr[i] # swap

        heapify(arr, n, largest)

def heapSort(arr):
    n = len(arr)

    for i in range(n // 2 - 1, -1, -1):
        heapify(arr, n, i)

    for i in range(n-1, 0, -1):
        arr[i], arr[0] = arr[0], arr[i] # swap
        heapify(arr, i, 0)

arr = [ 12, 11, 13, 5, 6, 7]
heapSort(arr)
n = len(arr)
print ("Sorted array is")
for i in range(n):
    print ("%d" %arr[i]),
```

```
Sorted array is
5
6
7
11
12
13
```

9.Python Program for Counting Sort.

In [10]:
```
def countSort(arr):

    output = [0 for i in range(256)]

    count = [0 for i in range(256)]

    ans = ["" for _ in arr]

    for i in arr:
        count[ord(i)] += 1

    for i in range(256):
        count[i] += count[i-1]

    for i in range(len(arr)):
        output[count[ord(arr[i])]-1] = arr[i]
        count[ord(arr[i])] -= 1

    for i in range(len(arr)):
        ans[i] = output[i]
    return ans

arr = "geodsforgood"
ans = countSort(arr)
print ("Sorted character array is %s" %("".join(ans)))
```

```
Sorted character array is ddfgoeoooors
```

10. Python Program for ShellSort.

In [13]:
```
def shell_sort(my_list, list_len):
    interval = list_len // 2
    while interval > 0:
        for i in range(interval, list_len):
            temp = my_list[i]
            j = i
            while j >= interval and my_list[j - interval] > temp:
                my_list[j] = my_list[j - interval]
                j -= interval
            my_list[j] = temp
        interval //= 2

my_list = [ 45, 31, 62, 12, 89, 5, 9, 8]
list_len = len(my_list)
print("\nThe list before sorting is :")
print(my_list)
shell_sort(my_list, list_len)
print ("\nThe list after performing shell sorting is :")
print(my_list)
```

```
The list before sorting is :
[45, 31, 62, 12, 89, 5, 9, 8]

The list after performing shell sorting is :
[5, 8, 9, 12, 31, 45, 62, 89]
```

In [ ]: