1.Python Program for Topological Sorting

In [1]:
```
from collections import defaultdict

class Graph:
    def __init__(self,vertices):
        self.graph = defaultdict(list)
        self.V = vertices

    def addEdge(self,u,v):
        self.graph[u].append(v)

    def topologicalSortUtil(self,v,visited,stack):

        visited[v] = True

        for i in self.graph[v]:
            if visited[i] == False:
                self.topologicalSortUtil(i,visited,stack)

        stack.insert(0,v)

    def topologicalSort(self):

        visited = [False]*self.V
        stack =[]

        for i in range(self.V):
            if visited[i] == False:
                self.topologicalSortUtil(i,visited,stack)

        print (stack)

g= Graph(6)
g.addEdge(5, 2);
g.addEdge(5, 0);
g.addEdge(4, 0);
g.addEdge(4, 1);
g.addEdge(2, 3);
g.addEdge(3, 1);

print ("Following is a Topological Sort of the given graph")
g.topologicalSort()
```

```
Following is a Topological Sort of the given graph
[5, 4, 2, 3, 1, 0]
```

2. Python Program for Radix Sort.

In [2]:
```
def countingSort(arr, exp1):

    n = len(arr)

    output = [0] * (n)

    count = [0] * (10)

    for i in range(0, n):
        index = (arr[i]/exp1)
        count[int((index)%10)] += 1

    for i in range(1,10):
        count[i] += count[i-1]

    i = n-1
    while i>=0:
        index = (arr[i]/exp1)
        output[ count[ int((index)%10) ] - 1] = arr[i]
        count[int((index)%10)] -= 1
        i -= 1

    i = 0
    for i in range(0,len(arr)):
        arr[i] = output[i]

def radixSort(arr):

    max1 = max(arr)

    exp = 1
    while max1/exp > 0:
        countingSort(arr,exp)
        exp *= 10

arr = [ 170, 45, 75, 90, 802, 24, 2, 66]
radixSort(arr)

for i in range(len(arr)):
    print(arr[i],end=" ")
```

```
2 24 45 66 75 90 170 802
```

3. Python Program for Binary Insertion Sort.

In [4]:
```
def insertion_sort(arr):
    for i in range(1, len(arr)):
        temp = arr[i]
        pos = binary_search(arr, temp, 0, i) + 1
        for k in range(i, pos, -1):
            arr[k] = arr[k - 1]
        arr[pos] = temp
def binary_search(arr, key, start, end):
    if end - start == 1:
        if key < arr[start]:
            return start - 1
        else:
            return start
    mid = (start + end)//2
    if arr[mid] < key:
        return binary_search(arr, key, mid, end)
    elif arr[mid] > key:
        return binary_search(arr, key, start, mid)
    else:
        return mid
arr = [1,5,3,4,8,6,3,4]
n = len(arr)
insertion_sort(arr)
print("Sorted array is:")
for i in range(n):
    print(arr[i],end=" ")
```

```
Sorted array is:
1 3 3 4 4 5 6 8
```

4. Python Program for Bitonic Sort.

In [5]:
```
def compAndSwap(a, i, j, dire):
    if (dire==1 and a[i] > a[j]) or (dire==0 and a[i] < a[j]):
        a[i],a[j] = a[j],a[i]

def bitonicMerge(a, low, cnt, dire):
    if cnt > 1:
        k = cnt//2
        for i in range(low , low+k):
            compAndSwap(a, i, i+k, dire)
        bitonicMerge(a, low, k, dire)
        bitonicMerge(a, low+k, k, dire)

def bitonicSort(a, low, cnt_dire):
    if cnt > 1:
        k = cnt//2
        bitonicSort(a, low, k, 1)
        bitonicSort(a, low+k, k, 0)
        bitonicMerge(a, low, cnt, dire)

def sort(a,N, up):
    bitonicSort(a,0, N, up)

a = [3, 7, 4, 8, 6, 2, 1, 5]
n = len(a)
up = 1

sort(a, n, up)
print ("\nSorted array is")
for i in range(n):
    print("%d" %a[i],end=" ")
```

```
Sorted array is
1 5 2 6 3 7 4 8
```

5. Python Program for Comb Sort.

In [2]:
```
def combsort(num):
    gap = len(num)
    swaps = True
    while gap > 1 or swaps:
        gap = max(1, int(gap / 1.25))
        swaps = False
        for i in range(len(num) - gap):
            j = i+gap
            if num[i] > num[j]:
                num[i], num[j] = num[j], num[i]
                swaps = True

num_list = [75, 16, 55, 19, 48, 14, 2, 61, 22, 100]
print("Before: ", num_list)
combSort(num_list)
print("After:  ", num_list)
```

```
Before:  [75, 16, 55, 19, 48, 14, 2, 61, 22, 100]
After:   [2, 14, 16, 19, 22, 48, 55, 61, 75, 100]
```

6. Python Program for Pigeonhole Sort.

In [3]:
```
def pigeonhole_sort(a):

    my_min = min(a)
    my_max = max(a)
    size = my_max - my_min + 1

    holes = [0] * size

    for x in a:
        assert type(x) is int, "integers only please"
        holes[x - my_min] += 1

    i = 0
    for count in range(size):
        while holes[count] > 0:
            holes[count] -= 1
            a[i] = count + my_min
            i += 1

a = [8, 3, 2, 7, 4, 6, 8]
print("Sorted order is : ", end =" ")

pigeonhole_sort(a)

for i in range(0, len(a)):
    print(a[i], end =" ")
```

```
Sorted order is :  2 3 4 6 7 8 8
```

7. Python Program for Cocktail Sort.

In [5]:
```
def cocktail_shaker_sort(alist):
    def swap(i, j):
        alist[i], alist[j] = alist[j], alist[i]

    upper = len(alist) - 1
    lower = 0

    no_swap = False
    while (not no_swap and upper - lower > 1):
        no_swap = True
        for j in range(lower, upper):
            if alist[j + 1] < alist[j]:
                swap(j + 1, j)
                no_swap = False
        upper = upper - 1

        for j in range(upper, lower, -1):
            if alist[j - 1] > alist[j]:
                swap(j - 1, j)
                no_swap = False
        lower = lower + 1


alist = input('Enter the list of numbers: ').split()
alist = [int(x) for x in alist]
cocktail_shaker_sort(alist)
print('Sorted list: ', end='')
print(alist)
```

```
Enter the list of numbers: 3 18 5 2 10 0 7 4
Sorted list: [0, 2, 3, 4, 5, 7, 10, 18]
```

8. Python Program for Gnome Sort.

In [7]:
```
def gnomeSort( arr, n):
    index = 0
    while index < n:
        if index == 0:
            index = index + 1
        if arr[index] >= arr[index - 1]:
            index = index + 1
        else:
            arr[index], arr[index-1] = arr[index-1], arr[index]
            index = index - 1
    return arr
# main
arr = [1,4,2,3,6,5,8,7]
n = len(arr)
arr = gnomeSort(arr, n)
print ("Sorted sequence is:")
for i in arr:
    print (i,end=" ")
```

```
Sorted sequence is:
1 2 3 4 5 6 7 8
```

9. Python Program for Odd-Even Sort / Brick Sort.

In [8]:
```
def oddEvenSort(arr, n):

    isSorted = 0
    while isSorted == 0:
        isSorted = 1
        temp = 0
        for i in range(1, n-1, 2):
            if arr[i] > arr[i+1]:
                arr[i], arr[i+1] = arr[i+1], arr[i]
                isSorted = 0

        for i in range(0, n-1, 2):
            if arr[i] > arr[i+1]:
                arr[i], arr[i+1] = arr[i+1], arr[i]
                isSorted = 0

    return

arr = [34, 2, 10, -9]
n = len(arr)

oddEvenSort(arr, n);
for i in range(0, n):
    print(arr[i], end =" ")
```

```
-9 2 10 34
```

10. Python Program for BogoSort or Permutation Sort.

In [9]:
```
import random

def bogoSort(a):
    n = len(a)
    while (is_sorted(a)== False):
        shuffle(a)

def is_sorted(a):
    n = len(a)
    for i in range(0, n-1):
        if (a[i] > a[i+1] ):
            return False
    return True

def shuffle(a):
    n = len(a)
    for i in range (0,n):
        r = random.randint(0,n-1)
        a[i], a[r] = a[r], a[i]

a = [3, 2, 4, 1, 0, 5]
bogoSort(a)
print("Sorted array :")
for i in range(len(a)):
    print ("%d" %a[i]),
```

```
Sorted array :
0
1
2
3
4
5
```

In [ ]:
```

```