Practical 1

Write a Program for Randomized Selection Algorithm

In [1]:
```
from random import randrange
def partition(x, pivot_index = 0):
    i = 0
    if pivot_index !=0: x[0],x[pivot_index] = x[pivot_index],x[0]
    for j in range(len(x)-1):
        if x[j+1] < x[0]:
            x[j+1],x[i+1] = x[i+1],x[j+1]
            i += 1
    x[0],x[i] = x[i],x[0]
    return x,i

def RSelect(x,k):
    if len(x) == 1:
        return x[0]
    else:
        xpart = partition(x,randrange(len(x)))
        x = xpart[0]
        j = xpart[1]
        if j == k:
            return x[j]
        elif j > k:
            return RSelect(x[:j],k)
        else:
            k = k - j - 1
            return RSelect(x[j+1:], k)

x = [3,1,8,4,7,9]
for i in range(len(x)):
    print (RSelect(x,i))
```

```
1
3
4
7
8
9
```

Practical 2

Write a Program for Heap Sort Algorithm

In [2]:
```
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]

        heapify(arr, n, largest)

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

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

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

Practical 3

Write a Program to perform Radix Sort Algorithm

In [11]:
```
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)
    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]),
```

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

Practical 4

Write a Program to Perform Bucket Sort Algorithm

In [12]:
```
def insertionSort(b):
    for i in range(1, len(b)):
        up = b[i]
        j = i - 1
        while j >=0 and b[j] > up:
            b[j + 1] = b[j]
            j -= 1
        b[j + 1] = up
    return b

def bucketSort(x):
    arr = []
    slot_num = 10
    for i in range(slot_num):
        arr.append([])

    for j in x:
        index_b = int(slot_num * j)
        arr[index_b].append(j)

    for i in range(slot_num):
        arr[i] = insertionSort(arr[i])

    k = 0
    for i in range(slot_num):
        for j in range(len(arr[i])):
            x[k] = arr[i][j]
            k += 1
    return x

x = [0.897, 0.565, 0.656,
     0.1234, 0.665, 0.3434]
print("Sorted Array is")
print(bucketSort(x))
```

```
Sorted Array is
[0.1234, 0.3434, 0.565, 0.656, 0.665, 0.897]
```

Practical 5

Write a Program to Perform Folyd-Warshall algorithm

In [13]:
```
V = 4
INF  = 99999

def floydWarshall(graph):


    dist = list(map(lambda i : list( map(lambda j : j , i)) , graph))
    for k in range(V):
        for i in range(V):
            for j in range(V):
                dist[i][j] = min(dist[i][j] , dist[i][k]+ dist[k][j] )
    printSolution(dist)


def printSolution(dist):
    print( "Following matrix shows the shortest distances\ between every pair of vertices" )
    for i in range(V):
        for j in range(V):
            if(dist[i][j] == INF):
                print ("%7s" %("INF"))
            else:
                print ("%7d\t" %(dist[i][j]))
            if j == V-1:
                print ("" )



"""
          10
     (0)------>(3)
      |         /|\
    5 |          |
      |          | 1
     \|/         |
     (1)------>(2)
         3           """
graph = [[0,5,INF,10],
         [INF,0,3,INF],
         [INF, INF, 0,   1],
         [INF, INF, INF, 0]
        ]

floydWarshall(graph);
```

```
Following matrix shows the shortest distances\ between every pair of vertices
      0
      5
      8
      9

    INF
      0
      3
      4

    INF
    INF
      0
      1

    INF
    INF
    INF
      0
```

Practical 6

Write a Program for Counting Sort Algorithm in python

In [15]:
```
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 = "geeksforgeeks"
ans = countSort(arr)
print ("Sorted character array is %s" %(""".join(ans)))
```

```
Sorted character array is eeeefggkkorss
```

Practical 7

Write a program for Set Covering Problem

In [16]:
```
def set_cover(universe, subsets):
    """Find a family of subsets that covers the universal set"""
    elements = set(e for s in subsets for e in s)
    if elements != universe:
        return None
    covered = set()
    cover = []
    while covered != elements:
        subset = max(subsets, key=lambda s: len(s - covered))
        cover.append(subset)
        covered |= subset

    return cover

def main():
    universe = set(range(1, 11))
    subsets = [set([1, 2, 3, 8, 9, 10]),
               set([1, 2, 3, 4, 5]),
               set([4, 5, 7]),
               set([5, 6, 7]),
               set([6, 7, 8, 9, 10])]
    cover = set_cover(universe, subsets)
    print(cover)

if __name__ == '__main__':
    main()
```

```
[{1, 2, 3, 8, 9, 10}, {4, 5, 7}, {5, 6, 7}]
```

Practical 8

Write a Program for found a subset with given sum

In [17]:
```
def isSubsetSum(set, n, sum) :

    if (sum == 0) :
        return True
    if (n == 0 and sum != 0) :
        return False

    if (set[n - 1] > sum) :
        return isSubsetSum(set, n - 1, sum);

    return isSubsetSum(set, n-1, sum) or isSubsetSum(set, n-1, sum-set[n-1])

set = [3, 34, 4, 12, 5, 2]
sum = 9
n = len(set)
if (isSubsetSum(set, n, sum) == True) :
    print("Found a subset with given sum")
else :
    print("No subset with given sum")
```

```
Found a subset with given sum
```

In [ ]:
```

```