LAB ASSIGNMENT 1

NAME: ROHAN GHOSH

BATCH: BCS2B

ID: 1810010011122

SUBJECT: Design and Analysis of Algorithms

Question:
Create an integer list 5000 random numbers and apply the following algorithms.
a. Insertion sort
b. Merge sort
c. Quick sort with last element as pivot
d. Quick sort with first element as pivot
e. Quick sort with random element as pivot
Measure CPU time for each algorithm and also check their stability.

In [1]:
```
import time import
random import
numpy
```

Insertion Sort

In [81]:
```
unsorted_array = [random.randint(1,5000) for i in range(5000)] before = time.time()
def insertionSort(arr): stable = 1
    for i in range(1, len(arr)): c = 0
            d = 0
            for a in range(i):
                if arr[a]==arr[i]: c+=1
            key = arr[i] j = i-1
            while (j >=0) and (key < arr[j]): arr[j+1] = arr[j]
                j -= 1
            arr[j+1] = key
            for a in range(j+1):
                if arr[a]==arr[j+1]: d+=1
            if c!=d:
                stable = 0 return
    arr,stable

sorted_array,stability = insertionSort(unsorted_array) print("Sorted array
is: ", end=" ")
for i in range(20): print(sorted_array[i], end=' ')
end = time.time()
print("\n\nTOTAL EXECUTION TIME: {}\n".format(end-before)) if stability:
    print("Insertion Sort is STABLE") else:
    print('Insertion Sort is NOT STABLE')
```

Sorted array is: 2 2 3 3 5 6 6 7 7 8 9 9 10 11 12 12 13 13 14 18 TOTAL EXECUTION

TIME: 5.142578363418579

Insertion Sort is STABLE

Merge Sort

In [82]:
```
unsorted_array = [random.randint(1,5000) for i in range(5000)] before = time.time()
def      mergeSort(arr):
    stable = 1
    if len(arr) >1:
            mid  =  len(arr)//2  L  =
            arr[:mid]
            R = arr[mid:]
            stability  =  mergeSort(L)  stability
            = mergeSort(R) i = j = k = 0
            while (i < len(L)) and (j < len(R)): c = d = 0
                if L[i] <= R[j]:
                    arr[k] = L[i] i+= 1
                else:
                    arr[k] = R[j] j+= 1
                k+= 1
            while i < len(L): c = d =
                0
                for a in range(i): if
                    arr[a]==L[i]:
                        c+=1
                arr[k] = L[i]
                for a in range(k):
                    if arr[a]==arr[k]: d+=1
                if c!=d:
                    stable = 0
                i+= 1
                k+= 1
            while j < len(R): c = d =
                0
                for a in range(mid+j): if
                    arr[a]==R[j]:
                        c+=1
                arr[k] = R[j]
                for a in range(k):
                    if arr[a]==arr[k]: d+=1
                if c!=d:
                    stable = 0
                  j+= 1
                  k+= 1
            return stable

stability = mergeSort(unsorted_array) print("Sorted
array is: ", end=' ') for i in range(20):
    print(unsorted_array[i], end=' ') end = time.time()
print("\n\nTOTAL EXECUTION TIME: {}\n".format(end-before)) if stability:
    print("Merge Sort is STABLE") else:
    print('Merge Sort is NOT STABLE')
```

Sorted array is: 1 1 2 3 4 4 4 6 6 9 10 11 11 12 12 16 17 18 19 19 TOTAL EXECUTION

TIME: 0.1025388240814209

Merge Sort is STABLE

Quick Sort (1st element as pivot)

In [83]:
```
unsorted_array = [random.randint(1,5000) for i in range(5000)] before = time.time()

def swap(array,a,b): c = d =
    e = f = 0 stable = 1
    for i in range(a):
        if array[a]==array[i]: c+=1
    for i in range(b):
        if array[b]==array[i]: e+=1
    array[a],array[b] = array[b],array[a] for i in range(b):
        if array[b]==array[i]: d+=1
    for i in range(a):
        if array[a]==array[i]: f+=1
    if (c!=d) | (e!=f): stable = 0
    return stable

def partition(array,start,end): stable = 1
    left = start + 1 pivot =
    array[start]
    for right in range(start+1,end): if pivot >
        array[right]:
            stable = swap(array,left,right) left = left + 1
    stable = swap(array,start,left-1) return left-1,stable
def quickSort(array,start,end): stable = 1
    if start < end:
        splitPoint,stable = partition(array,start,end) quickSort(array,start,splitPoint)
        quickSort(array,splitPoint+1,end)
    return stable


stability = quickSort(unsorted_array,0,len(unsorted_array)) print("Sorted array is: ", end=' ')
for i in range(20): print(unsorted_array[i], end=' ')
end = time.time()
print("\n\nTOTAL EXECUTION TIME: {}\n".format(end-before)) if stability:
    print("Quick Sort is STABLE") else:
    print('Quick Sort is NOT STABLE')
```

Sorted array is: 2 2 3 4 7 9 9 10 10 10 11 15 15 16 16 19 20 20 22 23 TOTAL EXECUTION TIME:

63.9921875

Quick Sort is NOT STABLE

Quick sort(last element as pivot)

In [84]:
```
unsorted_array = [random.randint(1,5000) for i in range(5000)] before = time.time()

def swap(array,a,b): c = d =
    e = f = 0 stable = 1
    for i in range(a):
        if array[a]==array[i]: c+=1
    for i in range(b):
        if array[b]==array[i]: e+=1
    array[a],array[b] = array[b],array[a] for i in range(b):
        if array[b]==array[i]: d+=1
    for i in range(a):
        if array[a]==array[i]: f+=1
    if (c!=d) | (e!=f): stable = 0
    return stable

def partition(arr,low,high): stable = 0
    i = ( low-1 ) pivot =
    arr[high]
    for j in range(low , high): if arr[j] <=
        pivot:
            i = i+1 swap(arr,i,j)
    stable = swap(arr,i+1,high) return
    i+1,stable
def quickSort(arr,low,high): stable = 0
    if low < high:
        pi,stable = partition(arr,low,high) quickSort(arr,
        low, pi-1) quickSort(arr, pi+1, high)
    return stable


stability = quickSort(unsorted_array,0,len(unsorted_array)-1) print("Sorted array is: ", end=' ')
for i in range(20): print(unsorted_array[i], end=' ')
end = time.time()
print("\n\nTOTAL EXECUTION TIME: {}\n".format(end-before)) if stability:
    print("Quick Sort is STABLE") else:
    print('Quick Sort is NOT STABLE')
```

Sorted array is: 1 4 5 6 6 7 7 7 7 8 9 9 9 11 12 12 14 16 17 19 TOTAL EXECUTION

TIME: 72.23339819908142

Quick Sort is NOT STABLE

Quick sort(random element as pivot)

In [87]:
```
unsorted_array = [random.randint(1,5000) for i in range(5000)] before = time.time()

def swap(array,a,b): c = d =
    e = f = 0 stable = 1
    for i in range(a):
        if array[a]==array[i]: c+=1
    for i in range(b):
        if array[b]==array[i]: e+=1
    array[a],array[b] = array[b],array[a] for i in range(b):
        if array[b]==array[i]: d+=1
    for i in range(a):
        if array[a]==array[i]: f+=1
    if (c!=d) | (e!=f): stable = 0
    return stable

def partition(arr,low,high): stable = 0
    random_idx = random.randint(low,high)
    arr[high], arr[random_idx] = arr[random_idx], arr[high]

    pi = low
    for i in range(low,high): if arr[i] <
        arr[high]:
            arr[i],arr[pi] = arr[pi],arr[i] pi += 1
    stable = swap(arr,pi,high) return
    pi,stable

def quickSort(arr,low,high): stable = 0
    if low < high:
        pi,stable = partition(arr,low,high) quickSort(arr,
        low, pi-1) quickSort(arr, pi+1, high)
    return stable

stability = quickSort(unsorted_array,0,len(unsorted_array)-1) print("Sorted array is: ", end=' ')
for i in range(20): print(unsorted_array[i], end=' ')
end = time.time()
print("\n\nTOTAL EXECUTION TIME: {}\n".format(end-before)) if stability:
    print("Quick Sort is STABLE") else:
    print('Quick Sort is NOT STABLE')
```

Sorted array is: 1 2 3 5 7 8 10 10 11 11 12 13 15 15 17 19 22 23 23 23 TOTAL EXECUTION TIME:

5.4375

Quick Sort is NOT STABLE

In [ ]: