Dharwad
ज्ञानेन विकासः

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

INTRODUCTION TO ALGORITHMS

EC351

ASSIGNMENT 3

Merge Sort and Quick Sort

Submitted by:

TEAM 7

Abdul Rahman	18BEC001
Ajmal A	18BEC003
Aryan Kumar	18BEC004
B.K Likhith Kumar	18BEC005
Safwan Mohammad	18BEC028

Submitted To:

DR. UMA S
27 Sept 2020

QUESTION: Find out Time complexity for the arrays using Quick Sorting and Merge Sorting Algorithms

1. **A[2.5, 4.5, 3.0,1.2,6.5,8.9,7.4,6.3]**
2. **B[5,3,6,3,4,5,4,6,4]**

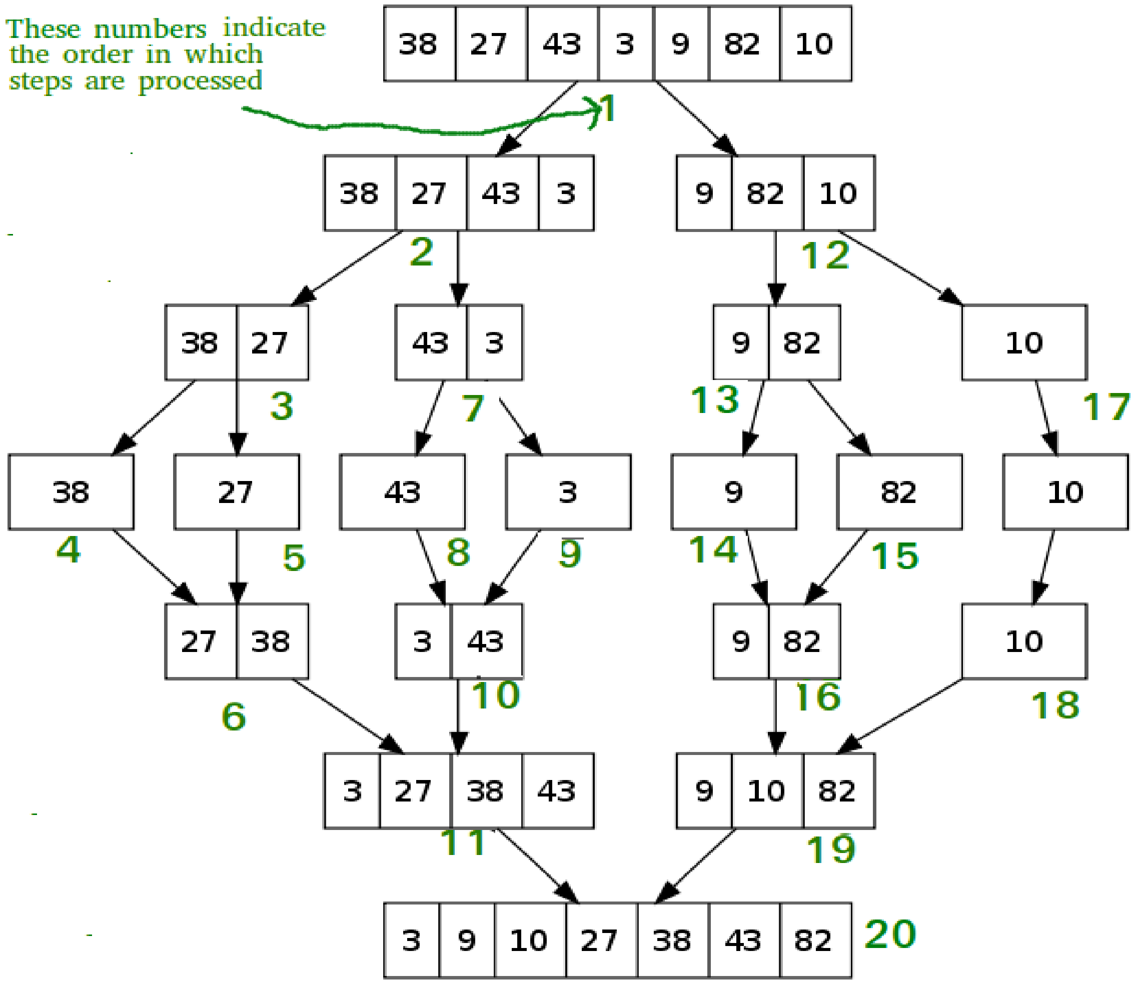
- **MergeSorting:**

Sorting arrays on different machines. Merge Sort is a recursive algorithm and time complexity can be expressed as following recurrence relation.

T(n) = 2T(n/2) + O(n)

The above recurrence can be solved either using Recurrence Tree method or Master method. It falls in case II of Master Method and solution of the recurrence is O(nLogn).

Time complexity of Merge Sort is O(n*logn) in all 3 cases (worst, average and best) as merge sort always divides the array into two halves and take linear time to merge two halves.



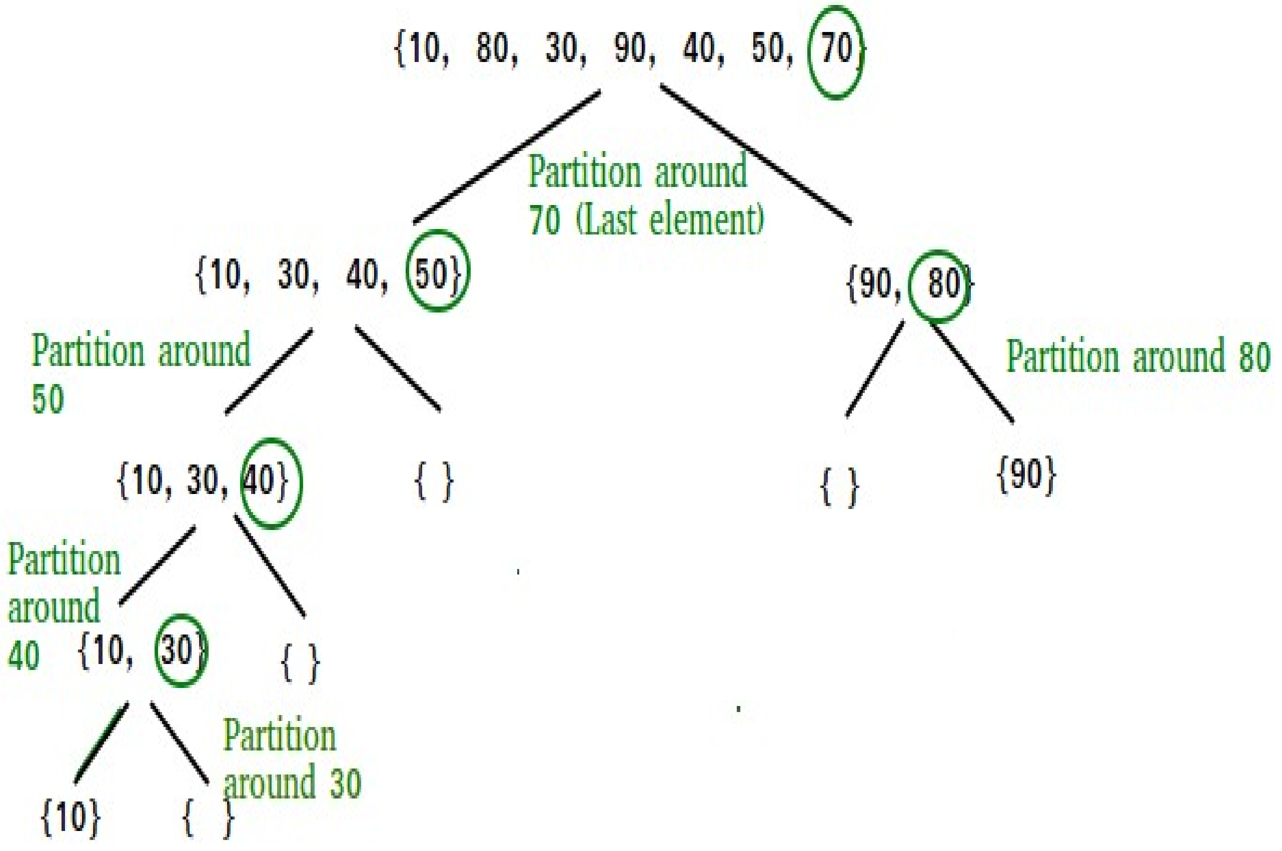
- **Quick Sorting:**

- Time taken by QuickSort in general can be written as
  T(n) = T(k) + T(n-k-1) + O(n).

- The best case occurs when the partition process always picks the middle element as pivot. Following is recurrence for best case is O(n*log).

- The worst case occurs when the partition process always picks greatest or smallest element as pivot. If we consider above partition strategy where last element is always picked as pivot, the worst case would occur when the array is already sorted in increasing or decreasing order. Following is recurrence for worst case is O(n*n)

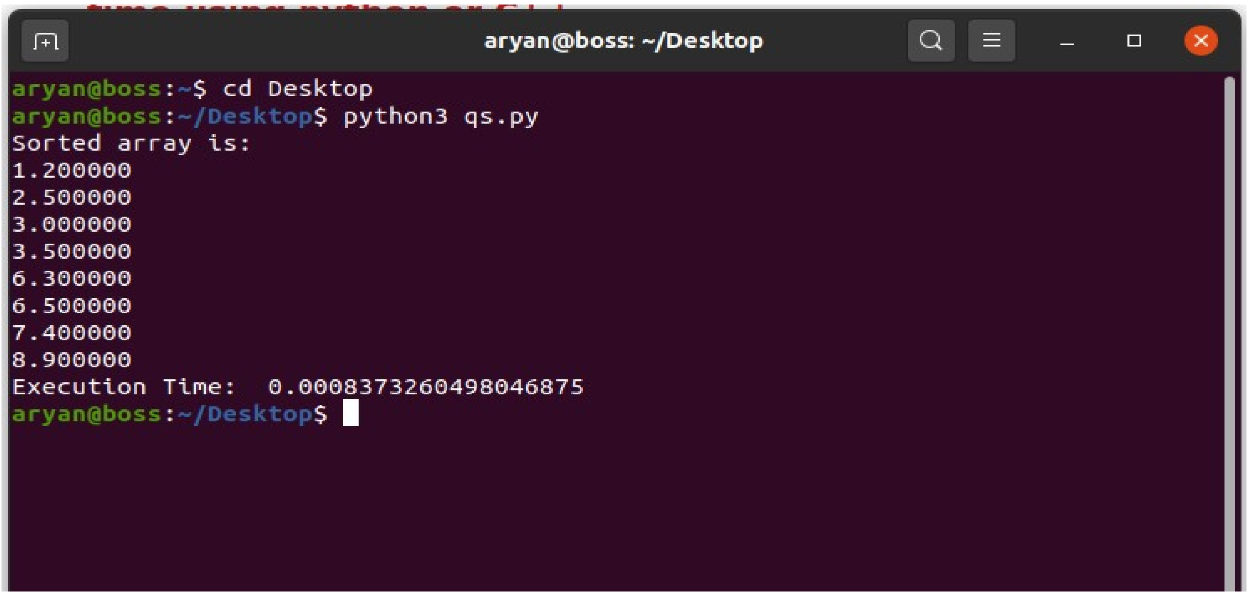
- To do average case analysis, we need to consider all possible permutation of array and calculate time taken by every permutation which doesn't look easy. We can get an idea of average case by considering the case when partition puts O(n/9) elements in one set and O(9n/10) elements in other set. Following is recurrence for this case **O(n*logn).**



Question 2: Find out Arrays Sorting program execution time using python or C++.

Quick Sort Algorithm

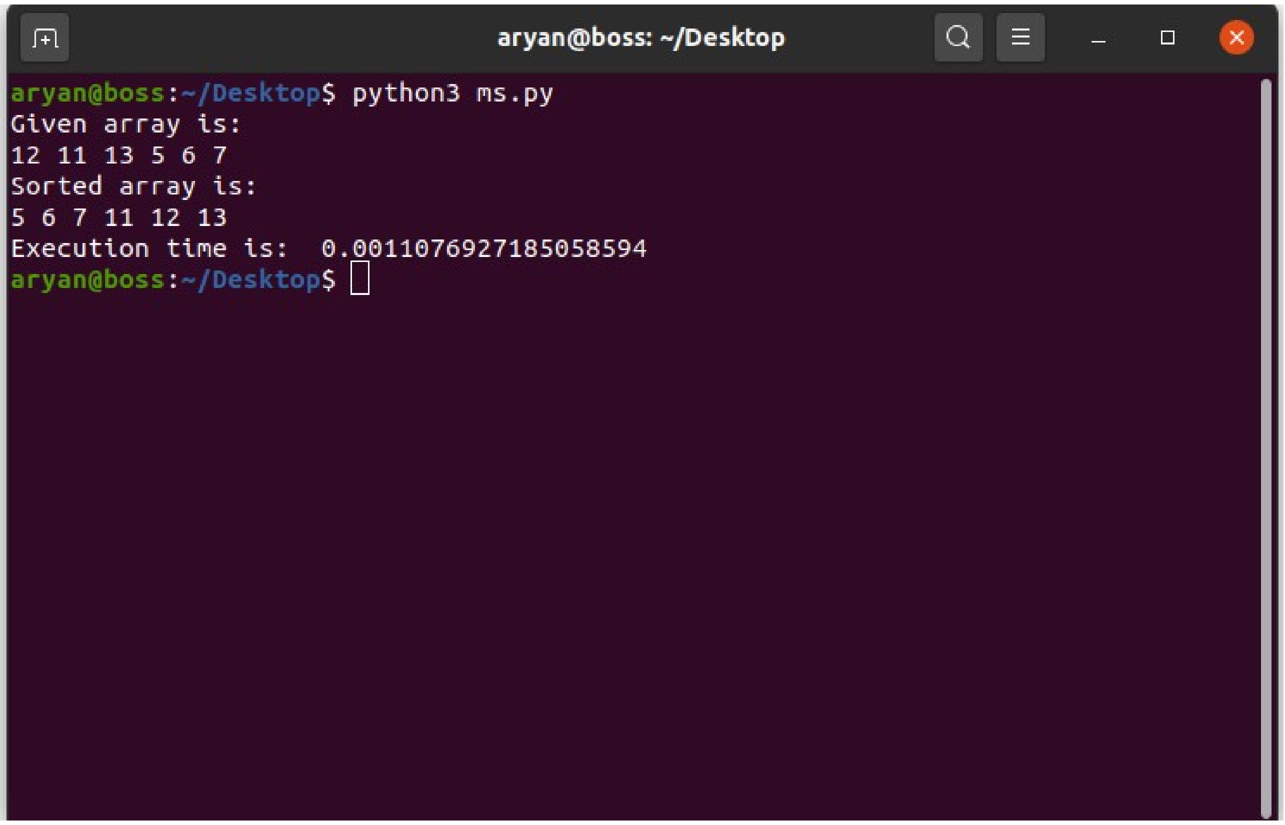
```
import time
def partition(arr, low, high):
    i = (low-1)         # index of smaller element
    pivot = arr[high]     # pivot
    for j in range(low, high):

        if arr[j] <= pivot:
            i = i+1
            arr[i], arr[j] = arr[j], arr[i]

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

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

arr = [2.5,3.5,3.0,1.2,6.5,8.9,7.4,6.3]
n = len(arr)
start = time.time()
time.sleep(1)
quickSort(arr, 0, n-1)
end = time.time()
print("Sorted array is:")
for i in range(n):
    print("%f" % arr[i])
print("Execution Time: ",end – start-1)
```



```
aryan@boss:~$ cd Desktop
aryan@boss:~/Desktop$ python3 qs.py
Sorted array is:
1.200000
2.500000
3.000000
3.500000
6.300000
6.500000
7.400000
8.900000
Execution Time:  0.0008373260498046875
aryan@boss:~/Desktop$ 
```

Merge Sort Algorithm

```
import time
def mergeSort(arr):
    if len(arr) >1:
        mid = len(arr)//2
        L = arr[:mid]
        R = arr[mid:]
        mergeSort(L)
        mergeSort(R)
        i = j = k = 0

        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i+= 1
            else:
                arr[k] = R[j]
                j+= 1
            k+= 1

        while i < len(L):
            arr[k] = L[i]
            i+= 1
            k+= 1

        while j < len(R):
            arr[k] = R[j]
            j+= 1
            k+= 1

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

if __name__ == '__main__':
    arr = [12, 11, 13, 5, 6, 7]
    print ("Given array is: ", end ="\n")
    printList(arr)
    start = time.time()
    mergeSort(arr)
    time.sleep(1)
    end = time.time()
    print("Sorted array is: ", end ="\n")
    printList(arr)
    print("Execution time is: ",end-start-1)
```



```
aryan@boss:~/Desktop$ python3 ms.py
Given array is:
12 11 13 5 6 7
Sorted array is:
5 6 7 11 12 13
Execution time is:  0.0011076927185058594
aryan@boss:~/Desktop$ 
```