Name: Sai Rohit Kalyan Gandham

StudentID: 1002070724

EmailID: [sxg0724@mavs.uta.edu](mailto:sxg0724@mavs.uta.edu)

7.2)

d) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RANDOMIZED\_QUICKSORT()\*\*\*\*\*\*\*\*\*

import random

def quicksortPivotAsFirstElement(arrayToSort, startPointer , endPointer):

if(startPointer < endPointer):

pivotindex = partitionForRandomNumber(arrayToSort, startPointer, endPointer)

quicksortPivotAsFirstElement(arrayToSort , startPointer , pivotindex-1)

quicksortPivotAsFirstElement(arrayToSort, pivotindex + 1, endPointer)

def partitionForRandomNumber(arrayToSort , startPointer, endPointer):

randpivot = random.randrange(startPointer, endPointer)

arrayToSort[startPointer], arrayToSort[randpivot] = arrayToSort[randpivot], arrayToSort[startPointer]

return partition(arrayToSort, startPointer, endPointer)

def partition(arrayToSort,startPointer,endPointer):

pivot = startPointer

intialIndex = startPointer + 1

for secondIndex in range(startPointer + 1, endPointer + 1):

quicksortPivotAsFirstElement.x += 1

if arrayToSort[secondIndex] <= arrayToSort[pivot]:

arrayToSort[intialIndex] , arrayToSort[secondIndex] = arrayToSort[secondIndex] , arrayToSort[intialIndex]

intialIndex = intialIndex + 1

arrayToSort[pivot] , arrayToSort[intialIndex - 1] = arrayToSort[intialIndex - 1] , arrayToSort[pivot]

pivot = intialIndex - 1

return (pivot)

if \_\_name\_\_ == '\_\_main\_\_':

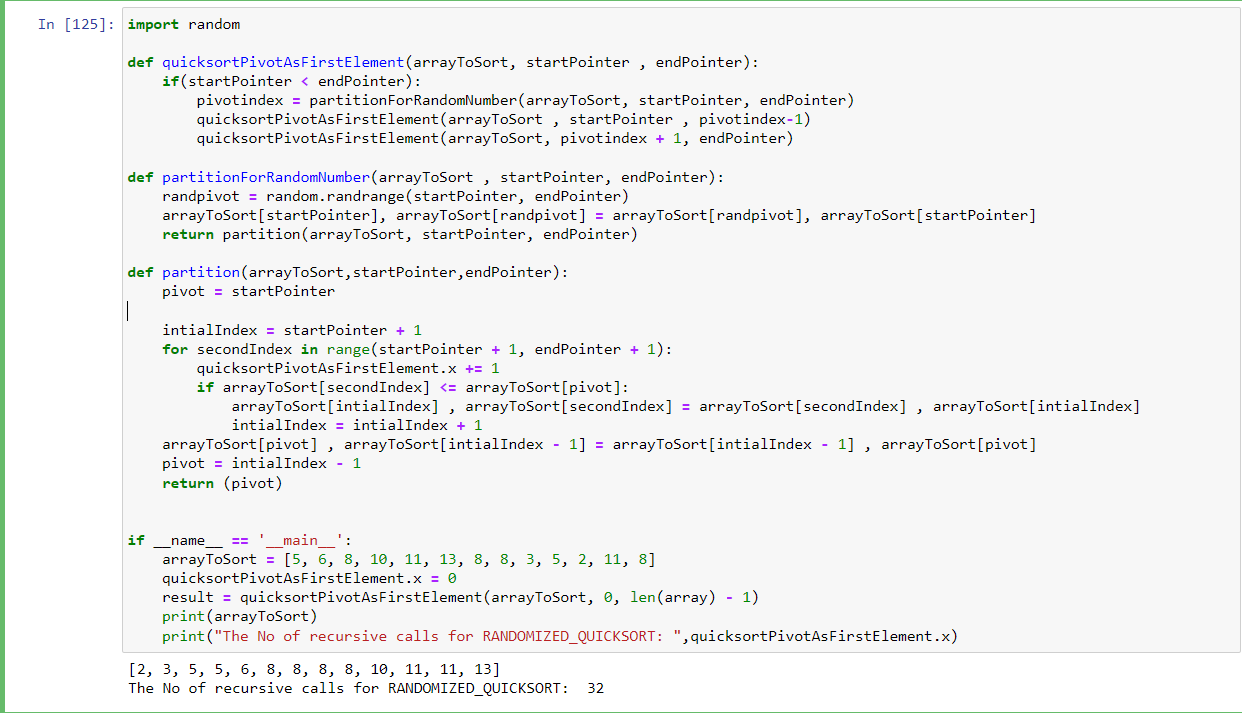
arrayToSort = [5, 6, 8, 10, 11, 13, 8, 8, 3, 5, 2, 11, 8]

quicksortPivotAsFirstElement.x = 0

result = quicksortPivotAsFirstElement(arrayToSort, 0, len(array) - 1)

print(arrayToSort)

print("The No of recursive calls for RANDOMIZED\_QUICKSORT: ",quicksortPivotAsFirstElement.x)



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RANDOMIZED\_QUICKSORT()`\*\*\*\*\*\*\*\*\*

import random

random.seed()

def swap(arrayToSort, firstElement, secondElement):

tempVariable = arrayToSort[firstElement]

arrayToSort[firstElement] = arrayToSort[secondElement]

arrayToSort[secondElement] = tempVariable

def partition(arrayToSort, startPointer, endPointer):

pivot = random.randint(startPointer, endPointer)

mark = startPointer

swap(arrayToSort, pivot, endPointer)

for i in range(startPointer, endPointer):

if arrayToSort[i] <= arrayToSort[endPointer]:

quicksortPivotAsLastElement.y += 1

swap(arrayToSort, i, mark)

mark += 1

swap(arrayToSort, mark, endPointer)

return mark

def do\_quicksortPivotAsLastElement(arrayToSort, startPointer, endPointer):

if startPointer < endPointer:

pivot = partition(arrayToSort, startPointer, endPointer)

do\_quicksortPivotAsLastElement(arrayToSort, startPointer, pivot - 1)

do\_quicksortPivotAsLastElement(arrayToSort, pivot + 1, endPointer)

def quicksortPivotAsLastElement(arrayToSort):

do\_quicksortPivotAsLastElement(arrayToSort, 0, len(arrayToSort) - 1)

if \_\_name\_\_ == "\_\_main\_\_":

arrayToSort = [5, 6, 8, 10, 11, 13, 8, 8, 3, 5, 2, 11, 8]

quicksortPivotAsLastElement.y = 0

quicksortPivotAsLastElement(arrayToSort)

print(arrayToSort)

print("The No of recursive calls for RANDOMIZED\_QUICKSORT``: ", quicksortPivotAsLastElement.y)



7-4)

d)

y\_tailNormalQuickSort = []

y\_optimizedTailQuickSort = []

def partition(arrayToSort, startPointer, endPointer):

pivot = arrayToSort[endPointer]

i = startPointer - 1

for j in range(startPointer, endPointer):

if arrayToSort[j] <= pivot:

i = i + 1

(arrayToSort[i], arrayToSort[j]) = (arrayToSort[j], arrayToSort[i])

(arrayToSort[i + 1], arrayToSort[endPointer]) = (arrayToSort[endPointer], arrayToSort[i + 1])

return i + 1

def quickSortNormal(arrayToSort, startPointer, endPointer):

while (startPointer < endPointer):

quickSort.x += 1

pi = partition(arrayToSort, startPointer, endPointer)

y\_tailNormalQuickSort.append(0)

quickSort(arrayToSort, startPointer, pi - 1)

y\_tailNormalQuickSort.append(1)

low = pi+1

def quickSort(arrayToSort, startPointer, endPointer):

while (startPointer < endPointer):

pi = partition(arrayToSort, startPointer, endPointer);

if (pi - startPointer < endPointer - pi):

quickSort.x += 1

y\_optimizedTailQuickSort.append(0)

quickSort(arrayToSort, startPointer, pi - 1);

y\_optimizedTailQuickSort.append(0)

startPointer = pi + 1;

else:

quickSort.x += 1

y\_optimizedTailQuickSort.append(0)

quickSort(arrayToSort, pi + 1, endPointer);

y\_optimizedTailQuickSort.append(1)

endPointer = pi - 1;

if \_\_name\_\_ == '\_\_main\_\_':

arrayToSort = [5, 6, 8, 10, 11, 13, 8, 8, 3, 5, 2, 11, 8]

quickSort.x = 0

data = quickSort(arrayToSort, 0, len(arrayToSort) - 1)

print(arrayToSort)

tailNormalQuickSort = quickSort.x

print(tailNormalQuickSort)

print(".............Normal Quick Sort........")

arrayToSort2 = [5, 6, 8, 10, 11, 13, 8, 8, 3, 5, 2, 11, 8]

data = quickSortNormal(arrayToSort2, 0, len(arrayToSort2) - 1)

print(arrayToSort2)

optimizedTailQuickSort = quickSort.x

print(optimizedTailQuickSort)

x\_pointsNormal = []

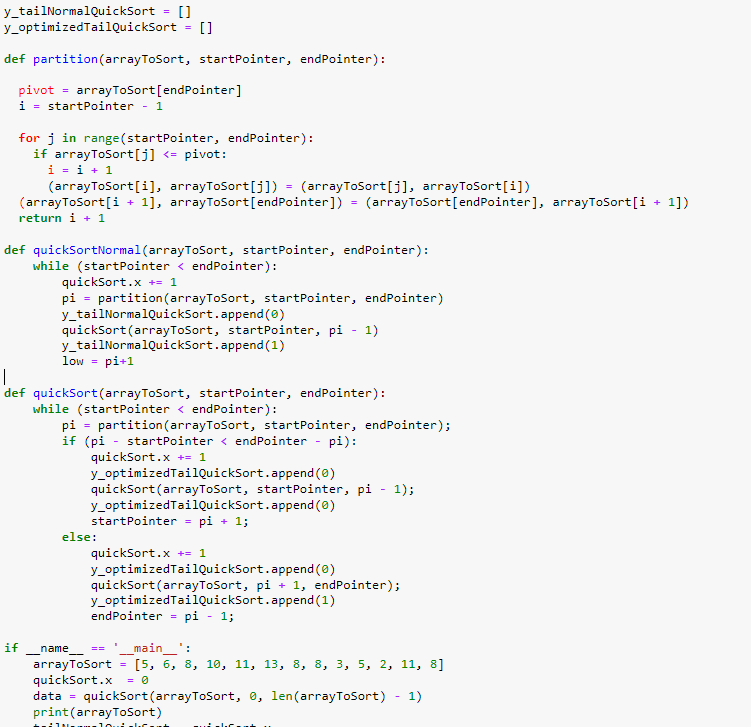
x\_pointsOptimesed = []

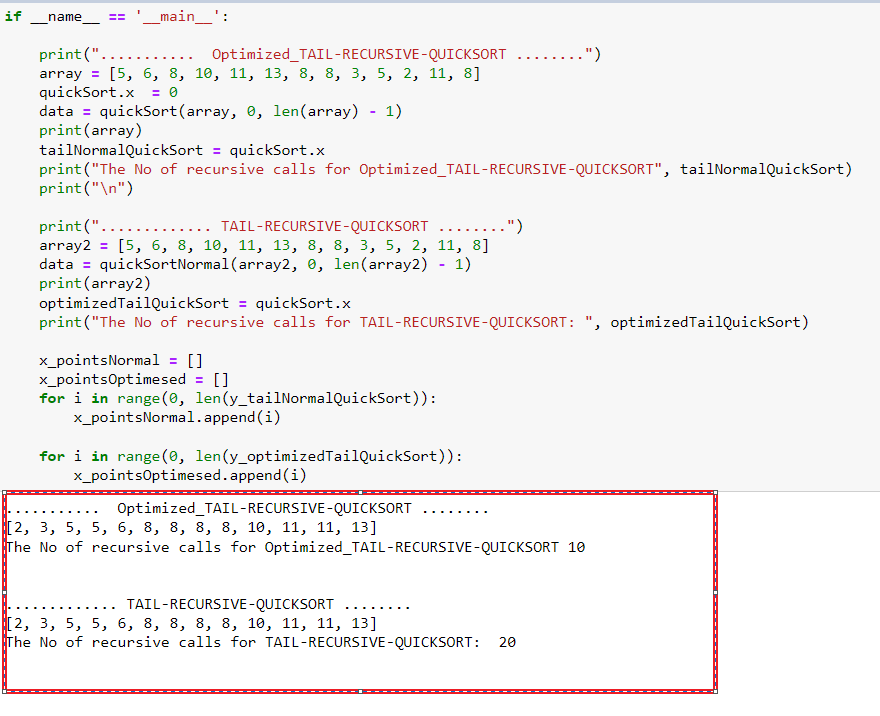
for i in range(0, len(y\_tailNormalQuickSort)):

x\_pointsNormal.append(i)

for i in range(0, len(y\_optimizedTailQuickSort)):

x\_pointsOptimesed.append(i)





……..Ploting………..

import matplotlib.pyplot as plt

plt.rcParams["figure.figsize"] = (15,10)

plt.plot(x\_pointsNormal, y\_tailNormalQuickSort, 'g', label='Insertion Sort')

plt.plot(x\_pointsOptimesed, y\_optimizedTailQuickSort, 'b', label='Merge Sort')

plt.title('Length of list vs Execution Time')

plt.xlabel('Length of list')

plt.ylabel('Execution Time')

plt.legend()

plt.show()

