**10.8 Write a python function for merge/quick sort for integer list as parameter to it.**

**Objectives:**

1. To learn about python as scripting option.

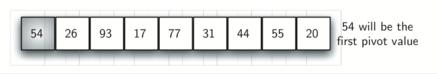
**Theory:**

**Quick Sort:**

The **quick sort** uses divide and conquer to gain the same advantages as the merge sort, while not using additional storage. As a trade-off, however, it is possible that the list may not be divided in half. When this happens, we will see that performance is diminished.

A quick sort first selects a value, which is called the **pivot value**. Although there are many different ways to choose the pivot value, we will simply use the first item in the list. The role of the pivot value is to assist with splitting the list. The actual position where the pivot value belongs in the final sorted list, commonly called the **split point**, will be used to divide the list for subsequent calls to the quick sort.

Figure shows that 54 will serve as our first pivot value. Since we have looked at this example a few times already, we know that 54 will eventually end up in the position currently holding 31. The **partition** process will happen next. It will find the split point and at the same time move other items to the appropriate side of the list, either less than or greater than the pivot value.



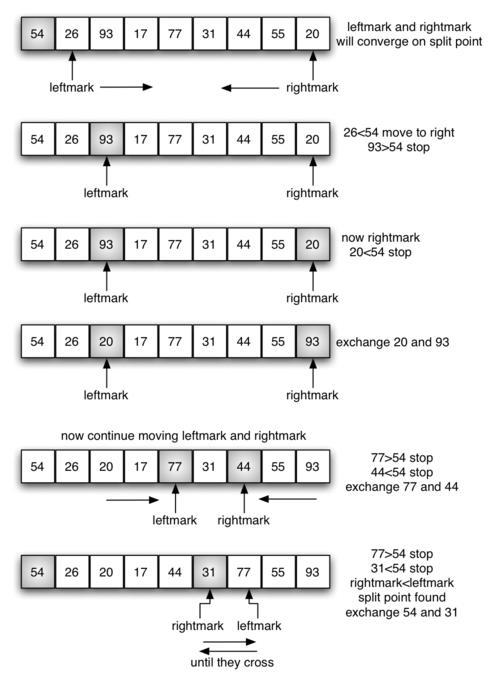


Fig 10.8 Merge Quick Sort

**Program:**

def quickSort(alist):

quickSortHelper(alist,0,len(alist)-1)

def quickSortHelper(alist,first,last):

if first<last:

splitpoint = partition(alist,first,last)

quickSortHelper(alist,first,splitpoint-1)

quickSortHelper(alist,splitpoint+1,last)

def partition(alist,first,last):

pivotvalue = alist[first]

leftmark = first+1

rightmark = last

done = False

while not done:

while leftmark <= rightmark and alist[leftmark] <= pivotvalue:

leftmark = leftmark + 1

while alist[rightmark] >= pivotvalue and rightmark >= leftmark:

rightmark = rightmark -1

if rightmark < leftmark:

done = True

else:

temp = alist[leftmark]

alist[leftmark] = alist[rightmark]

alist[rightmark] = temp

temp = alist[first]

alist[first] = alist[rightmark]

alist[rightmark] = temp

return rightmark

alist = list(map(int, input("Enter a list: ").split()))

quickSort(alist)

print(alist)

**Output:**

Enter a list: 485 982745 9827 9274

[485, 9274, 9827, 982745]

**Conclusion:**

1.Sorting a list using efficient algorithms like merge and quick sort was implemented

2.To enhance the application of basics of python

**References:**

[1] https://docs.python.org/3/