Assignment no	**6**
Aim	**Write a python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using quick sort and display top five scores.**
Objective	1. To understand the standard and abstract data representation Methods. 2. To identify the appropriate data structure and algorithm design method for a specified application. 3. To get the sorted array of percentage of first year students using quick sort. 4. To get the top five score of students after quick sort.
Outcome	1. To understand ,design and implement quick sort using list or array in python. 2. To analyse the quick sort time complexity 3. To design the algorithms to solve the programming problems.
OS/Programming tools used	(64-Bit) 64-BIT Fedora 17 or latest 64-BIT Update of Equivalent Open source OS or latest 64-BIT Version and update of Microsoft Windows 7 Operating System onwards Programming Tools (64-Bit) Eclipse with Python plugin

Theory related to assignment:

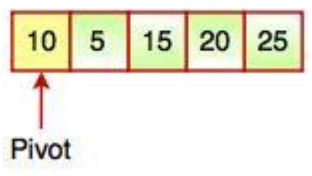
In this assignment we will implements quick sort. List or array as primitive data structure can be used to perform quick sort.

Quick Sort

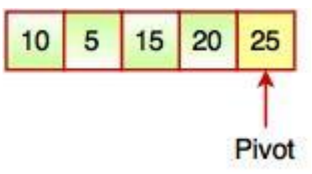
- Quick sort is also known as **Partition-exchange sort** based on the rule of **Divide and Conquer.**
- It is a highly efficient sorting algorithm.
- Quick sort is the quickest comparison-based sorting algorithm.
- It is very fast and requires less additional space, only O(n log n) space is required.
- Quick sort picks an element as pivot and partitions the array around the picked pivot.

There are different versions of quick sort which choose the pivot in different ways:

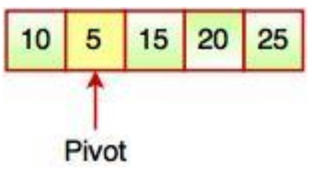
1. First element as pivot



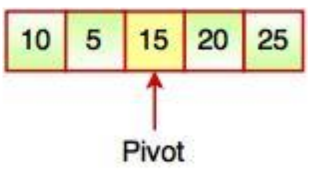
2. Last element as pivot



3. Random element as pivot



4. Median as pivot



Algorithm for Quick Sort

Step 1: Choose the highest index value as pivot.

Step 2: Take two variables to point left and right of the list excluding pivot.

Step 3: Left points to the low index.

Step 4: Right points to the high index.

Step 5: While value at left < (Less than) pivot move right.

Step 6: While value at right > (Greater than) pivot move left.

Step 7: If both Step 5 and Step 6 does not match, swap left and right.

Step 8: If left = (Less than or Equal to) right, the point where they met is new pivot.

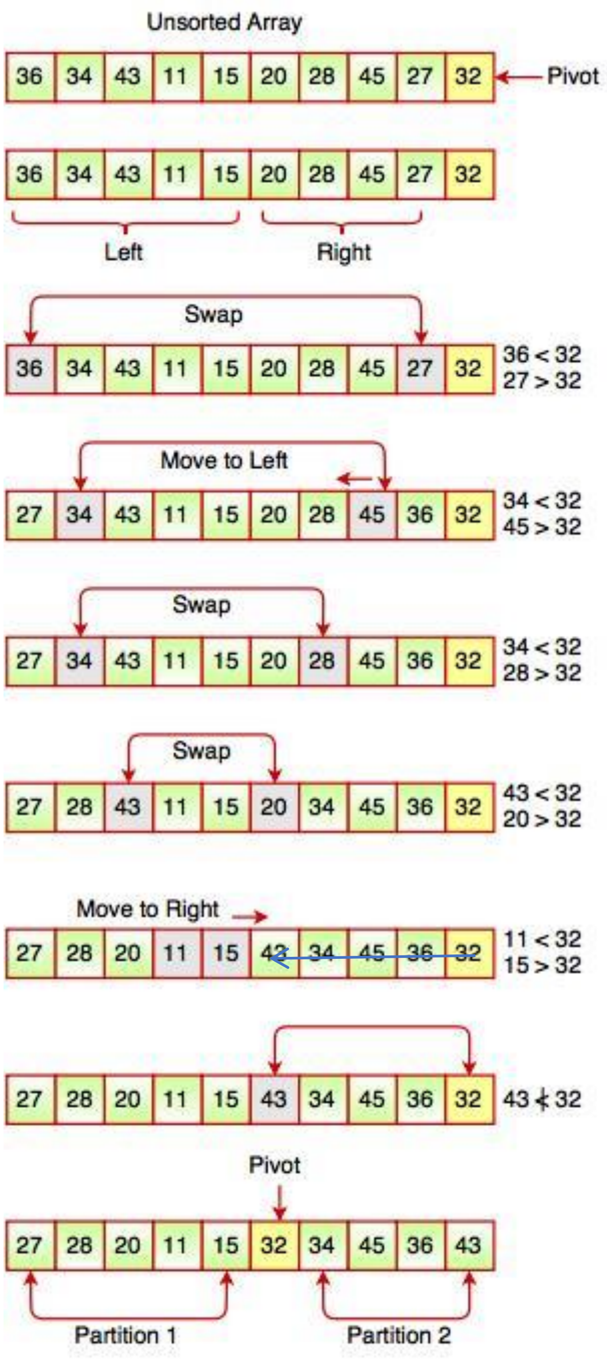


Fig. Finding Pivot Value in an Array

Pseudo Code for recursive QuickSort function:

/* low –> Starting index, high –> Ending index */

```
quickSort(arr[], low, high) {
    if (low < high) {
        /* pi is partitioning index, arr[pi] is now at right place */
        pi = partition(arr, low, high);
        quickSort(arr, low, pi – 1);  // Before pi
        quickSort(arr, pi + 1, high); // After pi
    }
}
```

Pseudo code for partition ()

/ This function takes last element as pivot, places the pivot element at its correct position in sorted array, and places all smaller (smaller than pivot) to left of pivot and all greater elements to right of pivot */*

```
partition (arr[], low, high)
{
    // pivot (Element to be placed at right position)
pivot = arr[high];
 i = (low – 1)  // Index of smaller element and indicates the
// right position of pivot found so far
for (j = low; j <= high- 1; j++){
 // If current element is smaller than the pivot
if (arr[j] < pivot){
i++;    // increment index of smaller element
 swap arr[i] and arr[j]
     }
 }
    swap arr[i + 1] and arr[high])
return (i + 1)
```

Time Complexity: O(n log n)

Space Complexity: O(logn)

Conclusion:

Quick sort implemented successfully for getting top 5 students.

Review Questions:

1. Explain the sorting?
2. What are the different types of sorts in data structures?
3. Define the quick sort?
4. How many passes are required in quick sort?
5. What is the time complexity of quick sort?