Experiment 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.

Software Requirements: 64-bit Open source Linux
Python IDE like spyder

Hardware Requirement: C2D, 2GB RAM, 500 GB HDD.

Objectives: To understand implementation of quick sort using python programming

Outcomes: Student will able to understand efficiency of quick sort over other sorting algorithms

Theory:

Quicksort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quicksort that pick pivot in different ways.

1. Always pick first element as pivot.
2. Always pick last element as pivot (implemented below)
3. Pick a random element as pivot.
4. Pick median as pivot.

Here, Pivot element = last eiement

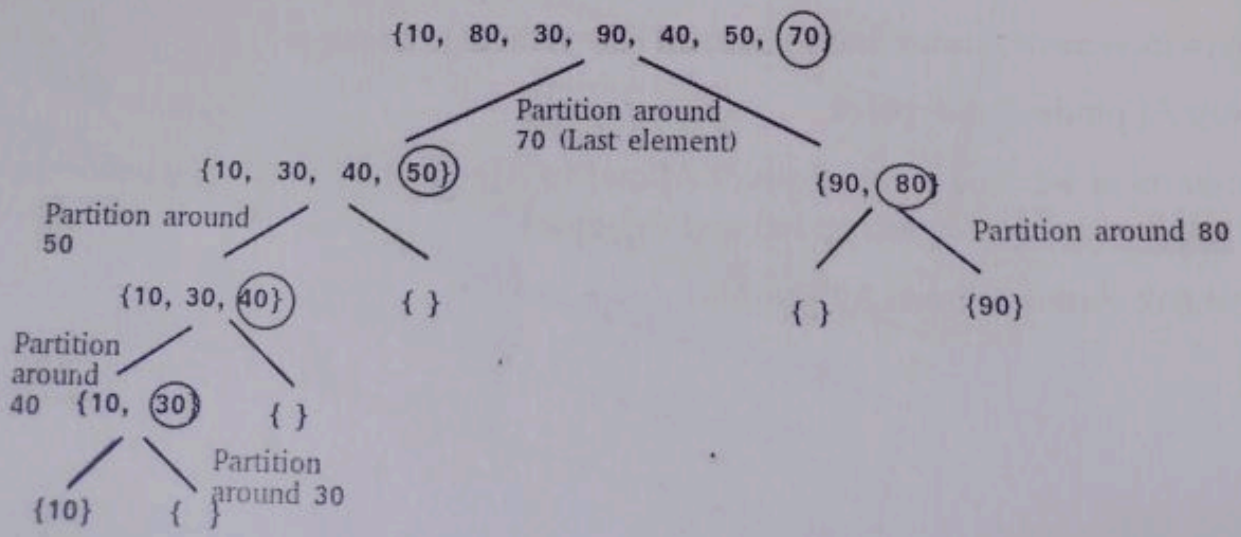
Partitioning: getting position of pivot such that all the element to its right are greater and all the element smaller than pivot will be at left side

Pindex =indexposition of pivot after partitioning

Consider left and right part of pivot and apply Quick sort again on both is same way using partitioning. If low > high then stop

arr[] = {10, 80, 30, 90, 40, 50, 70}

Indexes: 0 1 2 3 4 5 6



low = 0, high = 6, pivot = arr[h] = 70
Initialize index of smaller element, i = -1
Traverse elements from j = low to high-1
j = 0 : Since arr[j] <= pivot, do i++ and swap(arr[i], arr[j])
i = 0
arr[] = {10, 80, 30, 90, 40, 50, 70} // No change as i and j are same
j = 1 : Since arr[j] > pivot, do nothing // No change in i and arr[]
j = 2 : Since arr[j] <= pivot, do i++ and swap(arr[i], arr[j])
i = 1
arr[] = {10, 30, 80, 90, 40, 50, 70} // We swap 80 and 30
j = 3 : Since arr[j] > pivot, do nothing// No change in i and arr[]
j = 4 : Since arr[j] <= pivot, do i++ and swap(arr[i], arr[j])
i = 2
arr[] = {10, 30, 40, 90, 80, 50, 70} // 80 and 40 Swapped
j = 5 : Since arr[j] <= pivot, do i++ and swap arr[i] with arr[j]
i = 3
arr[] = {10, 30, 40, 50, 80, 90, 70} // 90 and 50 Swapped We come out of loop because j is now equal to high-1.
Finally we place pivot at correct position by swapping
arr[i+1] and arr[high] (or pivot)
arr[] = {10, 30, 40, 50, 70, 90, 80} // 80 and 70 Swapped
Now 70 is at its correct place. All elements smaller than 70 are before it and all elements greater than 70 are after it.

Algorithm:

1. Last element selected as a pivot element from given list .[You can choose any element from the list as the pivot element.]
2. Consider low=0 and high=n-2
3. Consider i=low if low>=high goto step 8
4. Compare all j^{th} elements from low to high end with pivot element
5. if Aj] < pivot increment pindex and increment j if j=n then go to step 6
6. Finally swap A[pindex] and pivot
7. make two parts as left and right of pivot A[low] to A[pindex-1] AND A[pindex+1] to A[high] and apply step 3 to 7 again on left and right part
8. Display first five elements from A[0] to A[4]
9. Stop

Conclusion: Thus implemented Quick sort algorithm

Questions:

1. Which design strategy is used by Quick sort algorithm?

- Divide and conquer design strategy

2. What is best case and worst case time complexity of Quick sort?

Best case - $O(n \log n)$
Worst case - $O(n^2)$

3. If input is already sorted then which case of time complexity is there?

$O(n \log n)$

4. What is partitioning?

Getting position of pivot such that all the elements to its right side are greater than all the elements similar than pivot will be left side.

Staff Signature & Date