21118

Assignment No: 05 Name: Shubham Chemate

(SE-1 : E-1)

19 oct

Sorting Operation- ~~Insertion shell~~ Quick. sort

Problem Statement:

write a python program to store first year percentages of students in an array. write a function for sorting the array of floating point numbers in ascending order using quick-sort & display top-5 scores.

Objectives:

1) To understand the working of quicksort algorithm

2) To able to use lists in python to implement this algorithm

Outcomes:

1) To implement quicksort to sort the elements of a list.

2) To find the time & space complexity.

3) To write a menu-driven & modular program.

Software requirement:

Operating System: windows-10 home Single Laguage

Python version: 3.8.5

VS Code (text-editor): Sept-2020 Version.

Hardware requirement:

Manufacturer: Acer

Processor: Intel(R) Core. i5 - 8265U CPU @ 1.60 GHz

System Type: 64-bit Operating System, x-64 based processor.

Theory:-

Quick-Sort:

1) Quicksort is one of the very efficient sorting algorithm.

2) It is based on the divide & conquer technique. In DnC technique problem is reduced in two (or) multiple problems of same type each of which is solved individually.

3> In quicksort, large array is partitioned into two smaller subarrays based on the chosen pivot: one subarray contains elements smaller than pivot & other contain elements greater than pivot.
We ~~app~~ solve/sort this two subarrays indepondently.

4> The time complexity of quicksort algorithm is based on choice of pivot.
If we choose pivot randomly in each step of sorting the time complexity comes out to be O(nlogn).

Algorithm (Pseudo-code of quicksort)

Sorting Method:

```
Algorithm Quicksort (list, left, right)
    [illegible] right)
    Quicksort(list, left, pivotIndex-1)
    Quicksort(list, pivotIndex+1, right)
```

the exit condition of recursive function will be
if (left > right) return.

Partition Subroution:

pivotIndex = random index b/w left & right.

```
Algorithm Partition(list, left, right)
  swap( list[pivotIndex], list[right]), i ← left
  for j ← left to right:
      If (list[j] < pivot):
          i += 1
          swap(list[i], list[j])
swap.(list[i], list[high])
return i
```

ADT of class :

```
class Scoresheet {
    def constructor():
        // intitialize the list & store count
        mylist = []
        n=0
    def quicksort():
        // logic of quicksort algorithm
```

Time & space complexity Analysis:

1) By careful implementation the quicksort algorithm con be implemented in constant space. O(n) space is required for array

2) The avg time complexity of quicksort algorithm is O(nlogn) The avg is over the choice of pivot in each smaller subproblem.

Testcases:

Testcase No.	Testcase Description	Expected o/p		Actual o/p	
1.	list={90.22, 85.23,	rank	marks	rank	marks
	96.28, 53.55, 82.91,	1	96.28	1	96.28
	93.26}	2	93.28	2	93.26
	∴ list with no.	3	90.22	3	90.22
	duplicates.	4	85.23	4	85.23
		5	82.91	5	82.91
2.	list with duplicates	rank.	marks	rank	marks
	list={90.21, 88.56, 90.21,	1	93.83	1	93.83
	92.53, 88.26, 93.83}	2.	92.53	2	92.53
		3	90.21	3	90.21
		4.	88.56	4	88.56

Applications:

The quicksort is most commonly used sorting algorithm. Most standard librarges used quicksort as default sorting algorithm.

Conclusion:

After assignment I'm able to implement quicksort & able to do it's time & space complexity analysis.