Page No.
Date

Assignment No. 6

Sumit Gubb Bhamare
SEComp A08
Sub - DSL

Aim :- To illustrate Quick Sort

Problem Statement: 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 & display top five scores.

Learning Objectives:
To understand concept of sorting.
To find time complexity of Quick Sort.

Learning Outcome: Students will be able to analyze problems to apply suitable searching & sorting algorithm to various applications.

Theory of Quick Sort:

Quicksort is a fast sorting algorithm, which is used not only for educational purposes, but widely applied in practice. On the average, it has O(n log n) complexity, making quicksort suitable for sorting big data volumes. The idea of the algorithm is quite simple & once you realize it, you can write quicksort as fast as bubble sort.

Page No.
Date

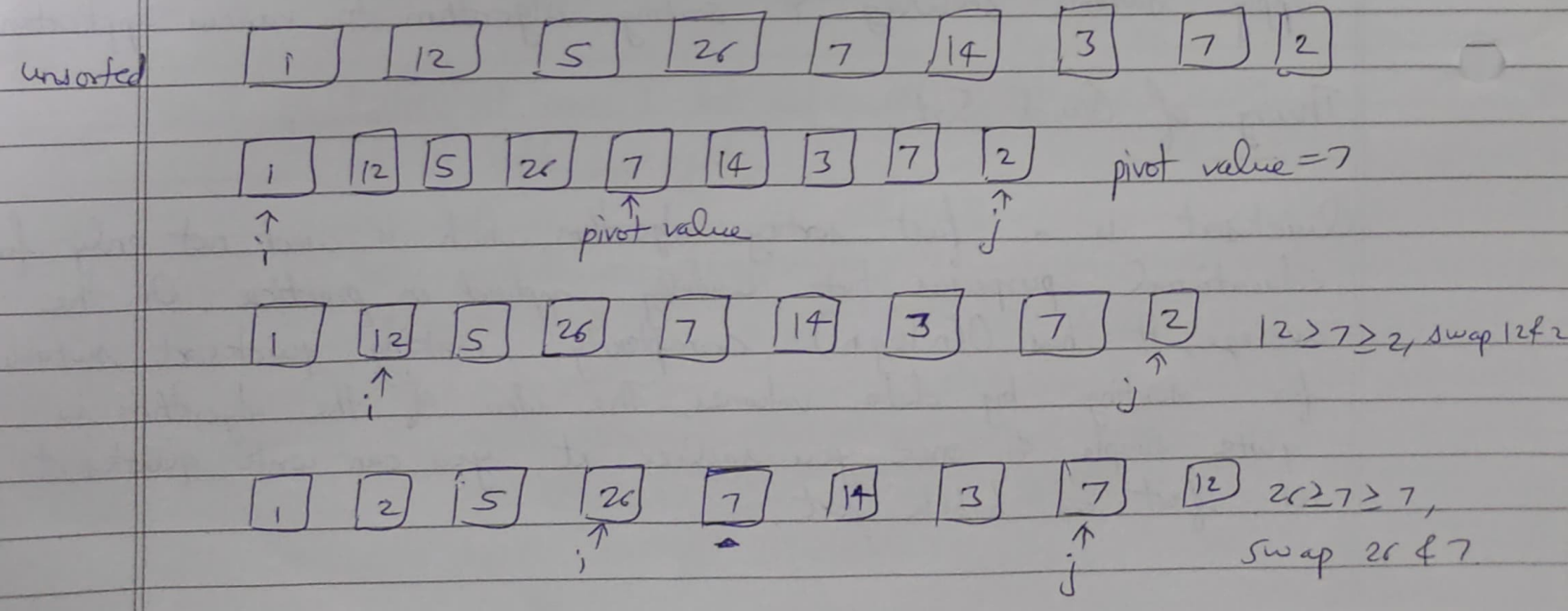
Algorithm:

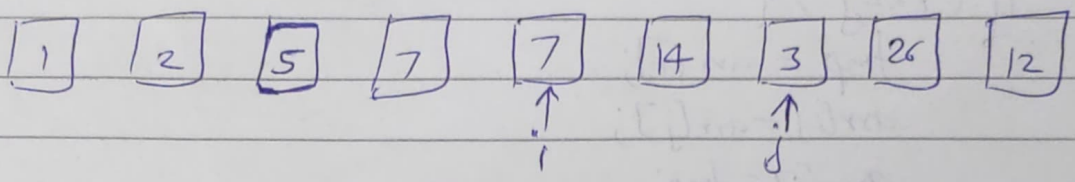
The divide-and-conquer strategy is used in quicksort. Below the recursion step is described:

- Choose a pivot value. We take the value of the middle element as pivot value, but it can be any value, which is in range of sorted values, even if it doesn't present in the array.

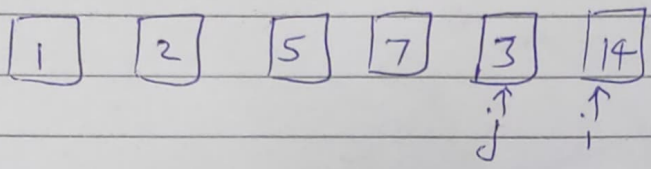
- Partition. Rearrange elements in such a way that all elements which are lesser than the pivot go to the left part of the array & all the elements greater than the pivot, go to the right part of the array. Values equal to the pivot can stay in any part of the array. Notice, that array may be divided in non-equal parts.
- Sort both parts. Apply quicksort algorithm recursively to the left & the right parts.

Example. Sort {1, 12, 5, 26, 7, 14, 3, 7, 2} using quicksort.

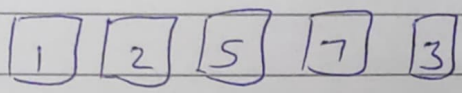




7≥7≥3, swap 7 & 3

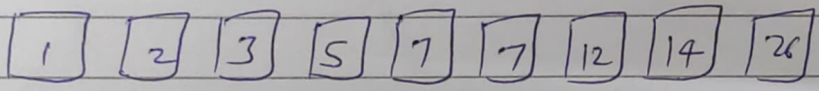


i>j, stop partition



14 7 26 12 run quick sort recursively

....



sorted.

Time complexity : O(nlogn)

Input: Enter the first-year percentage of students

Output: Sorting by Quick sort

Algorithm / Pseudo Code:

```
void quickSort(int arr[], int left, int right){
    int i= left , j=right;
    int tmp;
    int pivot= arr[(left+right)/2];
    /* partition */
    while(i<=j){
        while(arr[i]<pivot)
            i++;
        while(arr[j]>pivot)
            j--;
```

```
        if (i<=j) {
            tmp = arr[i];
            arr[i] = arr[j];
            arr[j] = tmp;
            i++;
            j--;
        }
    };
```

```
/* recursion */
  if (left<j)
      quickSort (arr, left, j);
  if (i<right)
      quick Sort (arr, i, right);
```

Conclusion: Thus, we have studied the implementation of Quick Sort.