Fundamentals of
Data Structures using C

Quick Sort

B.Bhuvaneswaran, AP (SG) / CSE

9791519152
bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI
ENGINEERING COLLEGE

Quick Sort

- Quick Sort is the most efficient internal sorting technique.
- It possesses a very good average case behaviour among all the sorting techniques.
- It is also called partitioning sort which uses divide and conquer techniques.

Quick Sort

- The quick sort works by partitioning the array A[1], A[2] . . . A[n] by picking some key value in the array as a pivot element.
- Pivot element is used to rearrange the elements in the array.
- Pivot can be the first element of an array and the rest of the elements are moved so that the elements on left side of the pivot are lesser than the pivot, whereas those on the right side are greater than the pivot.
- Now, the pivot element is placed in its correct position.
- Now the quicksort procedure is applied for left array and right array in a recursive manner.

Example

- Consider an unsorted array as follows:
  - 40 20 70 14 60 61 97 30
- Here PIVOT = 40, i = 20, j = 30.
- The value of i is incremented till a[i] < Pivot and the value of j is decremented till a[j] > pivot, this process is repeated until i < j.
- If a[i] > pivot and a[j] < pivot and also if i < j then swap a[i] and a[j].
- If i > j then swap a[j] and a[pivot].
- Once the correct location for PIVOT is found, then partition array into left sub array and right subarray, where left sub array contains all the elements less than the PIVOT and right sub array contains all the elements greater than the PIVOT.

Passes of Quick Sort

40	20	70	14	60	61	97	30
Pivot	i						j

40	20	70	14	60	61	97	30
Pivot		i					j

As i < j, swap(a[i], a[j]), swap(70, 30).

Passes of Quick Sort

40	20	30	14	60	61	97	70
Pivot		i					j

40	20	30	14	60	61	97	70
Pivot			i				j

40	20	30	14	60	61	97	70
Pivot				i			j

40	20	30	14	60	61	97	70
Pivot				i		j	

40	20	30	14	60	61	97	70
Pivot				i	j		

40	20	30	14	60	61	97	70
Pivot				i j			

40	20	30	14	60	61	97	70
Pivot			j	i			

As i > j, swap(a[j], Pivot), swap(14, 40).

Passes of Quick Sort

14	20	30	40	60	61	97	70
			j	i			

- Now, the pivot element has reached its correct position.
- The elements lesser than the Pivot {14, 20, 30} is considered as left sub array.
- The elements greater than the pivot {60, 61, 97, 70} is considered as right sub array.
- Then the QuickSort procedure is applied recursively for both these arrays.

Routine

```
void QuickSort(int a[], int left, int right)
{
        int i, j, temp, pivot;
        if (left < right) {
                pivot = left;
                i = left + 1;
                j = right;
                while (i < j)
                {
                        while (a[i] < a[pivot])
                                i++;
                        while (a[j] > a[pivot])
                                j--;
                        if (i < j) {
                                temp = a[i];
                                a[i] = a[j];
                                a[j] = temp;
                        }
                }
                temp = a[pivot];
                a[pivot] = a[j];
                a[j] = temp;
                QuickSort(a, left, j - 1);
                QuickSort(a, j + 1, right);
        }
}
```

Analysis of Quick Sort

▪ Best-case analysis	:	$O(n \log n)$
▪ Average-case analysis	:	$O(n \log n)$
▪ Worst-case analysis	:	$O(n^2)$

Advantages of Quick Sort

- It is one of the fastest sorting algorithms.
- Its implementation does not require any additional memory.
- It has better cache performance and high speed.

Limitations of Quick Sort

- The worst case efficiency of $O(n^2)$ is not well suited for large sized lists.
- Its algorithm is considered as a little more complex in comparison to some other sorting techniques.

Queries?

Thank You!