QuickSort Algorithm

Wednesday, May 20, 2020 12:06 PM

Pseudocode:

QuickSort(A, low, high) {
If(low < high){

Pivot = partition(A,low,high)	//O(N)
QuickSort(A,low,pivot-1)	//T(k)
QuickSort(A, pivot+1, high)	//T(n-k)

}
}

How it works?

8	2	7	1	11

Let's assume that 8 is our pivot. Then the array should be like this

2	7	1	8	11

1	2	7

8 11

1 2 7 8 11

Partition Method:

Algorithm

```
PARTITION(A, p, q) ▷ A[p..q]
x ← A[p]          ▷ pivot = A[p]
i ← p
for j ← p+1 to q
    do if A[j] ≤ x
        then i ← i+1
            exchange A[i] ↔ A[j]
exchange A[p] ↔ A[i]
return i
```

A[0...4]

0	1	2	3	4
8	11	1	7	2

Exchange will be based on this

X = A[0] = 8 //partition = 8
I = 0
For -> from j = 1 to 4
(When j = 1)
If(11 <= 8) -> NO

0	1	2	3	4
8	1	11	7	2

(When j = 2)
If(1 <= 8) -> YES
I = 1
Exchange 11,1

After exchange

0	1	2	3	4
8	1	7	11	2

(When j = 3)
If(7<=8) -> YES
I = 2
Exchange 11,7

0	1	2	3	4
8	1	7	2	11

(When j =4)
If(2 <= 8) -> YES
I = 3
Exchange 2,11

2	1	7	8	11

Exchange 8,2

Recurrence Formula:

T(N) = T(k) + T(n-k) + O(n)

Here, k symbolizes the position of the pivot.

If pivot is in the middle -> k = n/2

T(N) = T(n/2) + T(n-n/2) + O(n)
= T(n/2) + T(n/2) + O(n)

This would have a time complexity $n\log_2(n)$ -> BEST CASE

QuickSort has two types of Time Complexity.
If pivot is in the middle, then that is the best case.
If pivot is in the start or end index, then that is the worst case.

QuickSort Worst Case