

K. Abhishek
1002281717

DSGN & ANALY ALGORITHMS

5311 - Hands-On-6

③ Mathamatically derive average runtime complexity of non-random Pivot Version of quickSort.

Ans: Usually the runtime complexity depends on the choice of pivot & how balanced partitions are.

→ If pivot splits the array into two nearly equal halves, recurrence relation is

$$T(n) = T(n/2) + T(n/2) + O(n)$$

Also, which can be simplified as

$$T(n) = 2T(n/2) + O(n)$$

→ Expected Runtime Recurrence Relation which gives average pivot selection, the recurrence becomes

$$T(n) = T(\alpha n) + T((1-\alpha)n) + O(n)$$

→ The Solving Recurrence by summing up to work done at each level of recursion.

In all total no. of levels is $O(\log n)$, & total work at each level remains $O(n)$, overall complexity sums to $O(n \log n)$

→ The Average-case time complexity is $O(n \log n)$

This is because:

→ The pivot on average results in nearly balanced partitions.

→ The total depth of recursion is $O(\log n)$ and work per level is $O(n)$

Even through worst-case complexity $O(n^2)$, in practical scenarios, QuickSort performs close to $O(n \log n)$, making it one of fastest sorting algorithms.

At first, the array is divided into two parts, a smaller one and a larger one.

$$T(n) = T(n-1) + T(1) + O(n)$$

Now, the smaller part is further divided.

$$T(n) = T(n-2) + T(2) + O(n)$$

Repeating this process, the array is divided into smaller and smaller parts.

$$T(n) = T(n-1) + T(1) + O(n)$$

At each level of recursion, the work done is $O(n)$.

The total number of levels is $O(\log n)$.

Overall complexity is $O(n \log n)$.

The average-case time complexity is $O(n \log n)$.

This is because:

The pivot is chosen randomly, so the array is divided into two parts.

The total work at each level is $O(n)$.

Work at each level is $O(n)$.