DAA HANDS ON 6

Average runtime complexity of the non-random pivot version of quicksort.

1. Recurrence Relation for Quicksort

At each step of the algorithm, we:

1. Choose a pivot.
2. Partition the array around the pivot into two sub-arrays: one which will have element less than the pivot and the other will have elements greater than the pivot.
3. Recursively sort the two sub-arrays.

The partitioning, takes O(n) time for an array of size n.

Let T(n) be the time complexity to sort an array of size n. The recurrence relation is given by:

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

- $T(k)$ is the time to sort the left partition of size k.
- $T(n-k-1)$ is the time to sort the right partition of size $n-k-1$.
- $O(n)$ is the time to partition the array.

2. Best Case and Worst Case

- **Best Case**: When the pivot divides the array into two equal parts, i.e., $k \approx n/2$. The best-case time complexity is $O(n \log n)$.
- **Worst Case**: When the pivot always divides the array into the worst possible split. The time complexity in the worst case is $O(n^2)$.

3. Average Case

In the **average case**, we consider all possible positions for the pivot and the resulting splits they create. On average, the pivot will divide the array into two sub-arrays of roughly equal size.

The average-case recurrence relation becomes:

$$T(n) = \frac{1}{n} \sum_{k=0}^{n-1} \big(T(k) + T(n-k-1)\big) + O(n)$$

- Each partition takes O(n) time for the partitioning step.
- The summation $\sum_{k=0}^{n-1} \big(T(k) + T(n-k-1)\big) + O(n)$ accounts for all possible positions of the pivot.

Therefore, $T(n) \approx 2T\left(\frac{n}{2}\right) + O(n)$

4. Solving the Recurrence

Using **Master Theorem**:

$$T(n) = aT\left(\frac{n}{2}\right) + O(n^d)$$

For our case:

- a = 2 (two recursive calls),
- b = 2 (each call deals with half of the array),
- d = 1 (the partitioning step takes linear time O(n)).

Using the Master Theorem:

- We compare n^d with $n^{\log_b a}$, where $\log_b a = \log_2 2 = 1$.
- Since $d = \log_b a$, we are in **Case 2** of the Master Theorem, which gives us the time complexity:

$$T(n) = O(n^d \log n) = O(n \log n)$$

5. Conclusion

The average runtime complexity of the non-random pivot version of quicksort is O(n log n). This holds because, on average, the pivot will split the array into two approximately equal parts, leading to balanced recursion.

In summary:

- **Best Case**: $O(n \log n)$
- **Worst Case**: $O(n^2)$
- **Average Case**: $O(n \log n)$