Expected Number of Comparisons in Randomized Quicksort

Manish Acharya

1 Introduction

Randomized Quicksort is one of the most celebrated examples where randomness transforms a fragile worst-case algorithm into one that is efficient and robust in expectation. Unlike deterministic Quicksort, which fixes a pivot-selection rule and can suffer $\Theta(n^2)$ behavior on structured inputs, Randomized Quicksort differs only in selecting pivots uniformly at random.

Although its expected running time is often stated as $O(n \log n)$, the most illuminating analysis does not proceed through recurrence relations or recursion trees. Instead, the key idea is to analyze a more fundamental quantity: the *expected number of comparisons* performed during the execution of the algorithm.

This approach has several advantages. It:

- avoids solving complicated recurrences,
- highlights the probabilistic structure of the algorithm,
- and introduces powerful tools such as indicator random variables and linearity of expectation.

In this note, we present a structured and concept-driven analysis of the expected number of comparisons in Randomized Quicksort, with particular emphasis on the intuition behind each lemma and probabilistic argument.

2 Why Count Comparisons?

In comparison-based sorting algorithms, comparisons dominate the running time. Randomized Quicksort differs from deterministic Quicksort only in how it selects pivots, yet this small change dramatically alters its expected performance.

A key observation is:

> *The total running time of Quicksort is proportional to the total number of element comparisons performed.*

Thus, instead of analyzing recursion depth or worst-case splits, we analyze the expected number of comparisons directly.

3 Model and Assumptions

We assume:

- All elements are *distinct*.
- The pivot in each recursive call is chosen *uniformly at random* from the current subarray.

Let the input elements be

$$z_1 < z_2 < \cdots < z_n,$$

where the indices refer to positions in the *sorted order*, not the *input order*.

4 A Key Structural Property of Quicksort

Lemma 1. *No two elements are ever compared more than once during the execution of Quicksort.*

Proof. Two elements are compared only when one of them is chosen as a pivot while both lie in the same recursive subproblem.

Once a pivot element is chosen, the partition procedure permanently separates the remaining elements into two disjoint subarrays: those smaller than the pivot and those larger than it. Any two elements that fall into different subarrays after this partition will never appear together in any future recursive call.

Since the pivot itself is removed from all future recursive calls, any comparison involving that pivot can occur only at the moment it is chosen. Consequently, once two elements are separated by a pivot, or one of them is chosen as a pivot, they can never be compared again. □

This property allows us to count comparisons on a *pair-by-pair* basis.

5 When Are Two Elements Compared?

Let's fix two elements z_i and z_j with $i < j$.

Lemma 2. *Elements z_i and z_j are compared if and only if one of them is chosen as the* first pivot *among the elements*

$$\{z_i, z_{i+1}, \ldots, z_j\}.$$

Proof. Initially, all elements in $\{z_i, \ldots, z_j\}$ belong to the same recursive call.

- If some z_k with $i < k < j$ is chosen first as a pivot, then z_i and z_j fall on opposite sides of the partition and are never compared.
- If either z_i or z_j is chosen first, it is compared against every other element in the set, including the other one.

Thus, the comparison occurs if and only if z_i or z_j is the first pivot chosen from the set. □

6 Probability of a Comparison

Lemma 3. *For $i < j$, the probability that z_i and z_j are compared is*

$$\Pr[z_i \text{ is compared with } z_j] = \frac{2}{j - i + 1}.$$

Proof. The set $\{z_i, \ldots, z_j\}$ has size $j - i + 1$.

Each element is equally likely to be the first pivot chosen from this set. Only two outcomes (z_i or z_j) cause a comparison.

Thus,

$$\Pr[z_i \text{ is compared with } z_j] = \frac{2}{j - i + 1}.$$

□

This probability computation captures the core randomness of the algorithm.

7 Indicator Random Variables

Let's define, for $1 \leq i < j \leq n$,

$$X_{ij} = \begin{cases} 1 & \text{if } z_i \text{ is compared with } z_j, \\ 0 & \text{otherwise.} \end{cases}$$

Since each pair of distinct elements can be compared at most once, we index comparisons by unordered pairs (i, j) with $i < j$.

Let

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

be the total number of comparisons performed by the algorithm.

8 Main Result

Theorem 1. *The expected number of comparisons performed by Randomized Quicksort on n distinct elements is $O(n \log n)$.*

Proof. We have

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij},$$

where X_{ij} is the indicator of the event that z_i is compared with z_j. By linearity of expectation,

$$\mathbb{E}[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}[X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr[X_{ij} = 1].$$

By the previous lemma, $\Pr[X_{ij} = 1] = \frac{2}{j-i+1}$, hence

$$\mathbb{E}[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j - i + 1}.$$

Now substitute $k = j - i$ (so $k \in \{1, 2, \ldots, n - i\}$):

$$\mathbb{E}[X] = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \leq \sum_{i=1}^{n-1} \sum_{k=1}^{n-1} \frac{2}{k+1} = (n-1) \sum_{k=1}^{n-1} \frac{2}{k+1}.$$

Using the harmonic-series bound $\sum_{t=1}^{m} \frac{1}{t} = O(\log m)$, we get

$$\sum_{k=1}^{n-1} \frac{1}{k+1} = \sum_{t=2}^{n} \frac{1}{t} \leq \sum_{t=1}^{n} \frac{1}{t} = O(\log n).$$

Therefore,

$$\mathbb{E}[X] = O(n \log n),$$

as claimed. □

9 Conclusion

Randomized Quicksort provides a canonical example of how randomness can dramatically simplify both algorithm design and analysis. By focusing on the expected number of comparisons rather than recursive structure or worst-case inputs, we obtain a clean and intuitive proof of the $O(n \log n)$ expected running time.

The central idea is to decompose the algorithm's behavior into pairwise events and to analyze each event independently using indicator random variables. This approach avoids recurrence relations altogether and highlights a powerful general principle: expected performance can often be understood by counting how frequently simple events occur.

Beyond Quicksort, the techniques used in this analysis—linearity of expectation, indicator variables, and pairwise reasoning—recur throughout the study of randomized algorithms. They appear in analyses of hashing, randomized search trees, and selection algorithms, where similar probabilistic decompositions lead to transparent and robust performance guarantees.

Viewed in this light, Randomized Quicksort is not only an efficient sorting algorithm, but also an instructive case study in probabilistic algorithm analysis.

References

[CLRS09] T. H. Cormen et al., *Introduction to Algorithms*, 3rd ed., MIT Press, 2009.

[KT06] J. Kleinberg and É. Tardos, *Algorithm Design*, Pearson, 2006.

[SW11] R. Sedgewick and K. Wayne, *Algorithms*, 4th ed., Addison-Wesley, 2011.