

Quicksort with the First Element as Pivot Is Slowest on Sorted Input, and Choosing the Pivot at Random Makes Its Average Cost Independent of the Input

— on sorted input the number of comparisons becomes the number of all pairs, but with a random pivot the average returns to the same formula for every input — the separator is whether the procedure relies on the input for disorder or makes its own with random numbers — [Paper 927]

Yuuki YAMAGISHI (山岸 勇輝)

Yamagishi Complexity Torus Lab, Japan

September 2026

YOI: <https://www.yamagishiyuuki.com/yoi/927>

— *From thinking to syncing.* —

Abstract

Quicksort repeatedly chooses one value as a pivot and splits the rest into values smaller and larger than it. This paper counts the comparisons of a version using the first value as pivot and a version choosing the pivot at random, on shuffled and on sorted input, and sets them beside the average formula and the comparison lower bound. No new theorem or law is claimed. First, on shuffled input, even the first-element pivot averages as the formula says: at $n = 100, 500, 1000$ and 2000 the mean comparisons were **656.9, 4793.8, 11051.2** and **24602.1**, in ratios **1.013969, 0.997366, 1.005943** and **0.994835** to the formula $2(n+1)H_n - 4n$ (Section 2). Second, on sorted input, the first-element pivot is slowest: the comparisons equal the number of all pairs, $n(n-1)/2$: at $n = 2000$, **1999000**, which is **81.253226** times the shuffled-input mean (Section 3). Third, with a random pivot, the average on sorted input returns to the formula: on sorted input it took **647.3, 4795.0, 10949.2** and **24744.2** comparisons, in ratios from **0.99666** to **1.000583** to the formula (Section 3). Fourth, the average stays below 1.3 times the comparison lower bound: the lower bound $\log_2 n!$ is **524.8, 3767.4, 8529.4** and **19053.0**, the ratio of the mean to it runs from **1.251798** to **1.295660**, and the limiting coefficient is $2 \ln 2 = 1.386294$ (Section 4). Fifth, the separator is whether the procedure relies on the input for disorder or makes its own with random numbers: the first-element pivot relies on the input being disordered, so ordered input becomes its worst case. The random pivot makes its own disorder, so the average is the same for every input, and slowness moves from the input to the luck of the random numbers (Section 5). **Keywords:** Quicksort, number of comparisons, average-case complexity, worst-case complexity, randomized algorithm, comparison lower bound.

Scope of this paper (scope note): No new theorem or law is claimed: Quicksort, its average comparisons $2(n+1)H_n - 4n$, the $n(n-1)/2$ comparisons of the first-element pivot on sorted input, the independence of the average from the input under a random pivot, and the comparison lower bound $\log_2 n!$ are all known. Implementation speed is not discussed: memory reads and writes, constant factors and recursion depth are not treated. Only the number of comparisons is counted. One partitioning scheme is fixed: the pivot is placed first and each remaining value is compared with it once (one partition compares as many times as there are remaining values). Other pivot choices and dual pivots are not treated. The luck of the random numbers is not bounded: how small the probability of a slow run with a random pivot is, is not claimed; only means over trials are set down. Relation to earlier papers: Paper 843 showed that the lower bound on comparisons comes from the number of answers, not from the input. Here it is counted that, while the lower bound does not depend on the input, the count of the first-element-pivot procedure changes by a factor of 81.253226 with the order of the input. Paper 759 showed that, comparing equally often, the number of slow-memory reads is set by the tree's branching. Here, not memory but the number of comparisons itself is counted as changing with the order of the input and the choice of pivot. What is added: counting two versions on two kinds of input, confirming that the count on sorted input equals $n(n-1)/2$, showing that a random pivot returns sorted input to the formula, setting down the ratio of the average to the lower bound, and placing the separator on "relying on the input for disorder, or making one's own with random numbers".

1. Notation and the Question

Setting: a list of length n is sorted by placing the pivot first, comparing each remaining value with it once to split the list, and repeating on the two sides. The pivot is either the first value or a randomly chosen value swapped to the front. The average number of comparisons on shuffled input and the comparison lower bound are:

$$C(n) = 2(n+1)H_n - 4n, \quad H_n = \sum_{k=1}^n \frac{1}{k}, \quad L(n) = \log_2 n!$$

The check averages over 200 trials for $n = 100, 500$ and 1000 and 60 trials for $n = 2000$, and confirms that every version's output is sorted.

Question: what in the input and the choice of pivot decides how slow Quicksort is?

2. Shuffled Input

Computation 2.1: first-element pivot, mean on shuffled input and the formula.

n	mean	formula $C(n)$	ratio
100	656.9	647.9	1.013969
500	4793.8	4806.4	0.997366
1000	11051.2	10985.9	1.005943
2000	24602.1	24729.8	0.994835

Observation 2.2

on shuffled input, even the first-element pivot averages as the formula says. The first value of a shuffled input behaves like a randomly chosen value: the first-element pivot borrows that disorder from the input.

3. Sorted Input

Computation 3.1: sorted input.

n	first-element pivot	$n(n-1)/2$	random pivot (mean)	random pivot over formula
100	4950	4950	647.3	0.999081
500	124750	124750	4795.0	0.99762
1000	499500	499500	10949.2	0.99666
2000	1999000	1999000	24744.2	1.000583

At $n = 2000$, the first-element pivot on sorted input took **81.253226** times the shuffled-input mean.

Observation 3.2

ordered input is the worst case for the first-element pivot. The first value of a sorted list is the smallest, so each split leaves one side empty and the remaining length shrinks by only one: choosing the pivot at random returned the average on the same sorted input to the formula.

4. Ratio to the Lower Bound

Computation 4.1: the comparison lower bound $\log_2 n!$ is 524.8, 3767.4, 8529.4 and 19053.0, and the ratios of the first-element-pivot shuffled-input means to it are 1.251798, 1.272445, **1.295660** and 1.291246. The limiting coefficient of $n \log_2 n$ in the average is $2 \ln 2 = 1.386294$.

Observation 4.2

the average stays below 1.3 times the lower bound, close to a bound that does not depend on the input. The lower bound of Paper 843 is set by the number of answers alone and does not depend on the order of the input: by contrast, the first-element pivot's count at the same $n = 2000$ changes by a factor of 81.253226 with the order of the input.

5. The Separator

Observation 5.1

the separator is whether the procedure relies on the input for disorder or makes its own with random numbers. The first-element pivot borrows disorder from the input, so ordered input, having none, cannot repay the loan, and the comparisons become the number of all pairs: the random pivot makes its own disorder, so the average follows the same formula for every input, and whether a run is slow moves from the input to the luck of the random numbers.

Observation 5.2

what is not said. Implementation speed, the memory hierarchy, other pivot choices and bounds on the probability of a slow run with a random pivot are not treated. What is shown is that the first-element pivot is slowest on ordered input, and that choosing the pivot at random returns the average to the formula for every input.

6. Roll Call and Closing

1. $C(n) = 2(n+1)H_n - 4n$, $L(n) = \log_2 n!$ (Section 1): average and lower bound.
2. shuffled input ratio 0.994835 to 1.013969 (Computation 2.1): as the formula says.

3. **1999000 comparisons on sorted input** (Computation 3.1): $n(n-1)/2$.
4. **81.253226 times the shuffled mean** (Computation 3.1): **the worst case**.
5. **random pivot ratio 0.99666 to 1.000583** (Computation 3.1): **independent of input**.
6. **mean over lower bound up to 1.295660** (Computation 4.1): **limit $2 \ln 2$** .
7. **relying on the input, or on its own random numbers** (Observation 5.1): **the separator; Papers 843 and 759**.

Closing: Quicksort with the first element as pivot is slowest on sorted input, and choosing the pivot at random makes its average cost independent of the input. On shuffled input the first-element pivot averaged as the formula $2(n+1)H_n - 4n$ says, but on sorted input its comparisons became the number of all pairs, 81.253226 times as many at $n = 2000$: choosing the pivot at random returned the average on sorted input to the formula. The average stayed at most 1.295660 times the input-independent lower bound $\log_2 n!$: the separator is whether the procedure relies on the input for disorder or makes its own with random numbers. Placed among the earlier papers: beside the input-independent lower bound of Paper 843, a procedure whose count changes with the order of the input was set, and apart from the memory reads of Paper 759, how the number of comparisons itself leans on the input was counted. To be honest, which sorting procedure is fastest is not said. Whether the input or the random numbers decide the slowness, and no further.

References

1. Hoare, C. A. R. (1962). Quicksort. *The Computer Journal*.
 2. Sedgewick, R. (1977). The analysis of Quicksort programs. *Acta Informatica*.
 3. Yamagishi, Y. (2026). *Comparing Equally Often, the Number of Slow-Memory Reads Is Set by the Tree's Branching* [Paper 759]. YOI: <https://www.yamagishiyuuki.com/yoi/759>; DOI: 10.5281/zenodo.23015187
 4. Yamagishi, Y. (2026). *The Lower Bound on Comparisons Comes From the Number of Answers, Not From the Input* [Paper 843]. YOI: <https://www.yamagishiyuuki.com/yoi/843>
-

— From thinking to syncing. —

Yuuki YAMAGISHI (山岸 勇輝)

© 2026 Yamagishi Complexity Torus Lab, Japan · Licensed under Creative Commons Attribution 4.0 International (CC BY 4.0)