Asymptotic analysis of Triple State Quicksort

Ammar Muqaddas

ammar.mu@ku.edu.kw

ammar.mu@gmail.com

Average case

On the average, a classical Quicksort uses:

$$\text{comparisons} \approx 2n\ln(n) - 2.8456n \text{ [3]} \qquad (1)$$

$$\text{swaps} \approx 0.33n\ln(n) - 0.58n \text{ [4] [5]} \qquad (2)$$

Before proceeding with the analysis for our algorithm, please note that there are a few assumptions to be made, refer to references [1] [2] [3] .

The analysis in [3] can be applied reasonably to Triple State Quicksort, leading to the same average number of comparisons *2n ln(n) – 2.8456n*.

For the number of swaps, Our algorithm does NOT do swaps literally but element copies instead (most of the time), the only way to compare it to other algorithms (that do swaps) is to have an equivalent measure. Since a swap is composed of 3 element copy operations, we can say that *swaps* = *copies* / 3. So we can convert the number of copy operations that our algorithm does to equivalent **virtual** swaps by dividing over 3. Surely this equivalence is not exact since one might argue that copying from/to a temporary variable (in a swap) is different from copying from/to memory locations. But it still gives a reasonable indication.

It has been shown in [1] that the number of swaps in a single Quicksort recursive stage is:

$$\frac{n}{6} + \frac{5}{6n} \qquad (3)$$

Since each swap involves two elements. The number of elements copied in our algorithm is twice as much swaps. Adding one final copy operation to copy back the **temp** element then converting the number of copies to virtual swaps by dividing by 3. Gives:

$$\frac{1}{3}\left[2\cdot\left(\frac{n}{6}+\frac{5}{6n}\right)+1\right] = \left(\frac{n}{9}+\frac{5}{9n}\right)+\frac{1}{3}$$

Notice here that we don't need to add 1 copy for copying back pivot **p** since that's already included in (3) part of a full swap. The average number of swaps S_n can be expressed as:

$$S_n = \left(\frac{n}{9}+\frac{5}{9n}\right)+\frac{1}{3}+\frac{1}{n}\sum_{k=0}^{n-1}(S_k + S_{n-1-k})$$

Since $\sum_{k=0}^{n-1} S_k = \sum_{k=0}^{n-1} S_{n-1-k}$, we can rewrite:

$$S_n = \left(\frac{n}{9}+\frac{5}{9n}\right)+\frac{1}{3}+\frac{2}{n}\sum_{k=0}^{n-1} S_k$$

Multiplying by *n*.

$$nS_n = n\left(\frac{n}{9} + \frac{5}{9n}\right) + \frac{n}{3} + 2\sum_{k=0}^{n-1} S_k \qquad (4)$$

substituting *n* by *n-1*

$$(n-1)S_n = (n-1)\left(\frac{n-1}{9} + \frac{5}{9(n-1)}\right) + \frac{n-1}{3} + 2\sum_{k=0}^{n-2} S_k \qquad (5)$$

Subtracting (5) from (4) and simplifying would finally yield

$$S_n = \left(1 + \frac{1}{n}\right)\left(\frac{2}{9} + S_{n-1}\right)$$

We know that $S_2 = 0.5$ since a two elements random array will have the same probability of being already sorted or reversed. Solving the recurrence relation above with initial condition $S_2 = 0.5$ results:

$$S_n = \frac{2}{9}(n+1)\sum_{k=1}^{n}\frac{1}{k} - \frac{1}{6}(n+1)$$

We know that $H_n = \sum_{k=1}^{n}\frac{1}{k}$ were H_n is the nth harmonic number. We can rewrite S_n as:

$$S_n = \frac{2}{9}(n+1)H_n - \frac{1}{6}(n+1) \qquad (6)$$

We also know that H_n can be approximated:

$$H_n \approx \ln(n) + \gamma + \frac{1}{2n}$$

Were $\gamma \approx 0.5772$ is the Euler–Mascheroni constant. Substituting back in (6).

$$S_n = \frac{2}{9}(n+1)\left(\ln(n) + \gamma + \frac{1}{2n}\right) - \frac{1}{6}(n+1)$$

$$= \frac{2}{9}n\ln(n) + n\cdot\left(\frac{2\gamma}{9} - \frac{1}{6}\right) + \frac{2}{9}\ln(n) + \frac{1}{9n} + \frac{2\gamma}{9} - \frac{1}{18}$$

$$S_n \approx 0.222n\ln(n) - 0.038n \qquad (7)$$

It can be shown by trivial math that S_n in (7) is less than the average swaps in (2) of classical Quicksort for all $n > 139$.

Worst case

Quicksorts worst case is of $O(n^2)$. This is true for any Quicksort including our algorithm as long as it does pivot selection and has a divide and conquer algorithm. The worst case happens when the pivot is **consistently** equal or near the maximum or minimum of the elements in the array in every recursive stage. However, for an advanced Quicksort that handles equal elements correctly, the probability of the worst case happening is pretty slim in practice. For example, if the pivot selection was done at random or input array is a random permutation. Then the probability that the pivot is maximum or minimum in a **single** stage is ***2/n***. Since there are about ***n*** recursive stages in a Quicksort binary tree. The probability of all stages selecting minimum or maximum pivot will be:

$$p[worst\_case] \approx \left(\frac{2}{n}\right)^n = \frac{2^n}{n^n}$$

n^n is much larger than 2^n for even as low as $n=5$. Hence the extremely low probability.

REFERENCES

[1] Hoare, C. A. R. "Quicksort". Comput. J. vol. 5, 1962, pp. 10–16.

[2] Sebastian Wild, Markus E. Nebel, Ralph Neininger. “Average Case and Distributional Analysis of Java 7’s Dual Pivot Quicksort”

[3] JACEK CICHON, "QUICK SORT - AVERAGE COMPLEXITY"

[4] Robert Sedgewick “The analysis of Quicksort programs.” Acta Inf. 7(4) (1977) pp 327–355

[5] Sebastian Wild and Markus E. Nebel. “Average Case Analysis of Java 7’s Dual Pivot Quicksot”.