A&W

QuickSort: Expected Runtime is $O(n \log n)$

Detailed Walkthrough

Goal of these notes. We prove that randomized QuickSort makes $O(n \log n)$ comparisons in expectation. The argument has three reusable techniques: (i) collapsing a multi-variable expected runtime into a single sequence t_n via a symmetry argument, (ii) the shift-and-subtract trick to turn a recurrence with a sum into a clean two-term recurrence, and (iii) a multiplicative-to-additive substitution so the recurrence telescopes into a harmonic sum.

Contents

---

1 Setup and notation

The objects we work with throughout:

- The array $A[1..n]$ of *distinct* elements (the distinctness assumption simplifies the analysis; the bound holds in general).
- The pivot $p \in \{\ell, \ell+1, \ldots, r\}$, chosen *uniformly at random* on each recursive call.
- The cost measure: $T_{\ell,r}$, the random number of comparisons made by QuickSort(A, ℓ, r).

1.1 The algorithm

QuickSort(A, ℓ, r) proceeds as follows. If $\ell < r$:

1. Pick a random pivot index $p \sim \text{Uniform}(\{\ell, \ldots, r\})$.
2. Call Partition(A, ℓ, r, p), which rearranges $A[\ell..r]$ so that elements smaller than $A[p]$ end up to its left and elements larger end up to its right. Returns the pivot's final position t.
3. Recurse on $A[\ell..t-1]$ and $A[t+1..r]$.

Remark 1.1: Cost of Partition

A single call to Partition on a slice of size $n = r - \ell + 1$ uses exactly $n - 1$ comparisons — it must compare each non-pivot element to the pivot once. This is the only place comparisons happen, so $T_{\ell,r}$ is the sum of $(r_i - \ell_i)$ over all recursive calls.

1.2 The destination

Theorem 1.1 (Expected runtime of randomized QuickSort)**.** *Let $T_{1,n}$ be the random number of comparisons made by* QuickSort$(A, 1, n)$ *on an array of n distinct elements. Then*

$$\mathbb{E}[T_{1,n}] \;\le\; 2(n+1)\ln n + O(n) \;=\; O(n \log n).$$

The next three sections build up to this bound.

2 Reducing to a single variable: t_n

2.1 The symmetry observation

Look at $\mathbb{E}[T_{\ell,r}]$. The pivot is uniform on $\{\ell, \ldots, r\}$, and only the *relative order* of distinct elements matters — not their values, not the absolute positions. So $\mathbb{E}[T_{\ell,r}]$ depends only on the size $n = r - \ell + 1$.

Key Idea 2.1: Reduce to one variable

Define $t_n := \mathbb{E}[T_{\ell,r}]$ for any ℓ, r with $r - \ell + 1 = n$. The whole proof now lives in the world of the sequence $(t_n)_{n \ge 0}$, not in pairs (ℓ, r).

2.2 Deriving the recurrence

Base case. If $n \le 1$, nothing to sort: $t_n = 0$.

Recursive case ($n \ge 2$). Run one step of QuickSort on a slice of size n:

- Partition costs $n - 1$ comparisons.
- Suppose the pivot ends up with i elements on its left. Then $n - i - 1$ end up on its right (since one slot is the pivot itself: $i + 1 + (n - i - 1) = n$).

- Because the pivot is uniformly random on n slots, $\mathbb{P}[\text{left has } i] = \frac{1}{n}$ for each $i \in \{0, \dots, n-1\}$.
- Conditional on the split being $(i, n-i-1)$, the two recursive calls cost t_i and t_{n-i-1} in expectation.

By the law of total expectation:

$$t_n = \sum_{i=0}^{n-1} \underbrace{\tfrac{1}{n}}_{\mathbb{P}[\text{split}=i]} \cdot \big((n-1) + t_i + t_{n-i-1}\big) = \frac{1}{n} \sum_{i=0}^{n-1} \big((n-1) + t_i + t_{n-i-1}\big).$$

Definition 2.1 (The sequence t_n)**.**

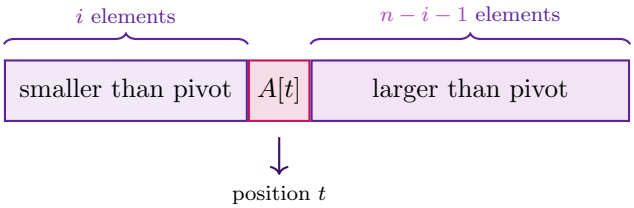
$$t_n = \begin{cases} 0 & \text{if } n \le 1, \\ \dfrac{1}{n} \displaystyle\sum_{i=0}^{n-1} \big((n-1) + t_i + t_{n-i-1}\big) & \text{if } n \ge 2. \end{cases}$$

Example 2.1: Sanity check at $n = 4$

The pivot lands at each rank with probability $1/4$:

Pivot rank	Left size i	Right size $n-i-1$	Cost
1st (smallest)	0	3	$3 + t_0 + t_3$
2nd	1	2	$3 + t_1 + t_2$
3rd	2	1	$3 + t_2 + t_1$
4th (largest)	3	0	$3 + t_3 + t_0$

Average: $t_4 = \frac{1}{4} \sum_{i=0}^{3} (3 + t_i + t_{3-i})$, matching the formula.



3 Manipulating the recurrence

We now massage the recurrence into something solvable. Strategy: kill the sum.

3.1 Step 1 — multiply through by n

$$n \cdot t_n = \sum_{i=0}^{n-1} \big((n-1) + t_i + t_{n-i-1}\big). \tag{$\star$}$$

Remark 3.1: Why multiply by n?

We're about to subtract the $n-1$ version of this equation from the n version. If we kept the $\frac{1}{n}$ in front, the two equations would have different prefactors ($\frac{1}{n}$ vs $\frac{1}{n-1}$) and the algebra would be messy. Multiplying through gives clean integer coefficients on both sides.

3.2 Step 2 — write the same equation for $n-1$

Replace n with $n-1$ everywhere in $(\star)$:

$$(n-1)\cdot t_{n-1} \;=\; \sum_{i=0}^{n-2}\big((n-2)+t_i+t_{n-i-2}\big). \tag{$\star\star$}$$

3.3 Step 3 — the shift-and-subtract trick

Key Idea 3.1: The shift-and-subtract trick

Whenever you see $n\cdot a_n = \sum_{i<n} f(a_i)$, write the same equation for $n-1$ and subtract. Most of the sum cancels, leaving a clean two-term recurrence.

Expand the right-hand side of $(\star)$ into three sums:

$$n\cdot t_n \;=\; \sum_{i=0}^{n-1}(n-1)+\sum_{i=0}^{n-1}t_i+\sum_{i=0}^{n-1}t_{n-i-1}.$$

The first sum has n copies of $(n-1)$, equal to $n(n-1)$. The third sum, with substitution $j=n-i-1$, becomes $\sum_{j=0}^{n-1}t_j$ — the same as the second. So:

$$n\cdot t_n \;=\; n(n-1)+2\sum_{i=0}^{n-1}t_i. \tag{$\star'$}$$

By the same reasoning applied to $(\star\star)$:

$$(n-1)\cdot t_{n-1} \;=\; (n-1)(n-2)+2\sum_{i=0}^{n-2}t_i. \tag{$\star\star'$}$$

Subtract. The two large sums differ by exactly one term: $\sum_{i=0}^{n-1}t_i-\sum_{i=0}^{n-2}t_i=t_{n-1}$. So:

$$n\cdot t_n-(n-1)\cdot t_{n-1} \;=\; \big[n(n-1)-(n-1)(n-2)\big]+2t_{n-1}.$$

Simplify the bracket: $n(n-1)-(n-1)(n-2)=(n-1)[n-(n-2)]=2(n-1)$. Therefore:

$$\boxed{n\cdot t_n-(n-1)\cdot t_{n-1} \;=\; 2(n-1)+2t_{n-1}.} \tag{$\dagger$}$$

The sum is gone.

3.4 Step 4 — solve $(\dagger)$ for t_n

Move $(n-1)\,t_{n-1}$ to the right and combine the two t_{n-1} terms:

$$n\cdot t_n=(n-1)t_{n-1}+2t_{n-1}+2(n-1)=(n+1)\,t_{n-1}+2(n-1).$$

Divide by n:

$$t_n \;=\; \frac{n+1}{n}\cdot t_{n-1}+\frac{2(n-1)}{n}. \tag{$\ddagger$}$$

3.5 Step 5 — weaken to a clean inequality

Note that $\frac{2(n-1)}{n} = 2 - \frac{2}{n} \le 2$. So:

$$\boxed{t_n \;\le\; \frac{n+1}{n} \cdot t_{n-1} + 2.}$$

Remark 3.2: Why drop the $-\frac{2}{n}$?

We only need an *upper* bound, and the cleaner constant 2 makes the next step tidier. The discarded $\frac{2}{n}$ doesn't affect the final $O(n \log n)$ bound.

4 Solving the recurrence

We have $t_n \le \frac{n+1}{n}\, t_{n-1} + 2$. The multiplicative factor $\frac{n+1}{n}$ depends on n, so the recurrence does not directly telescope. We change variables to flatten it.

4.1 The substitution $s_n := t_n/(n+1)$

Key Idea 4.1: Multiplicative → additive substitution

For a recurrence $a_n \le \dfrac{f(n)}{f(n-1)}\, a_{n-1} + d_n$, the substitution $b_n := a_n/f(n)$ *exactly cancels* the multiplicative factor, leaving $b_n \le b_{n-1} + d_n/f(n)$. Here $f(n) = n+1$, so we set $s_n := t_n/(n+1)$.

Plug in $t_n = (n+1)\, s_n$ and $t_{n-1} = n\, s_{n-1}$:

$$(n+1)\, s_n \;\le\; \frac{n+1}{n} \cdot n\, s_{n-1} + 2 \;=\; (n+1)\, s_{n-1} + 2.$$

Divide both sides by $(n+1)$:

$$\boxed{s_n \;\le\; s_{n-1} + \frac{2}{n+1}.}$$

4.2 Telescope

Unrolling the additive recurrence down to the base:

$$\begin{aligned} s_n &\le s_{n-1} + \tfrac{2}{n+1} \\ &\le s_{n-2} + \tfrac{2}{n} + \tfrac{2}{n+1} \\ &\;\;\vdots \\ &\le s_2 + 2\sum_{i=3}^{n+1} \frac{1}{i}. \end{aligned}$$

Direct computation gives $t_2 = 1$, hence $s_2 = 1/3 \le 2\sum_{i=3}^{3} \frac{1}{i} = 2/3$. By induction (base $n = 2$ verified, step is the recurrence above), for all $n \ge 2$:

$$s_n \;\le\; 2\sum_{i=3}^{n+1} \frac{1}{i}.$$

4.3 The harmonic sum

Recall $H_n = \sum_{i=1}^{n} \frac{1}{i} = \ln n + O(1)$. Therefore:

$$\sum_{i=3}^{n+1} \frac{1}{i} \;=\; H_{n+1} - 1 - \tfrac{1}{2} \;=\; \ln(n+1) + O(1) \;=\; \ln n + O(1).$$

So $s_n \le 2\ln n + O(1)$.

4.4 Convert back to t_n

Multiply by $(n+1)$:

$$t_n \;=\; (n+1)\, s_n \;\le\; (n+1)\big(2\ln n + O(1)\big) \;=\; 2(n+1)\ln n + O(n).$$

Combined with $\mathbb{E}[T_{1,n}] = t_n$ (Section 2):

$$\boxed{\mathbb{E}[T_{1,n}] \;\le\; 2(n+1)\ln n + O(n) \;=\; O(n \log n).} \qquad \square$$

Watch out 4.1: Big-O notation: $\le$ vs $=$

$O(n \log n)$ is a *set* of functions, so writing $X \le O(\cdots)$ is technically a category error. The convention is to use $=$, read as "is": $f(n) = O(g(n))$ means f grows at most like g. The boxed conclusion above is correct because the first $\le$ compares functions and the second $=$ is the standard big-O "is".

Watch out 4.2: Common confusions

- "Why does $T_{\ell,r}$ depend only on $r-\ell+1$?" The random pivot doesn't see absolute positions, only the size of the slice. Formally, by induction on $r - \ell$.
- "$(n-1) + t_i + t_{n-i-1}$ — where do these three terms come from?" Three sources of comparisons in one recursive call: the partition step $(n-1)$, the left recursion (t_i), the right recursion (t_{n-i-1}). The subscript on t_{n-i-1} is forced by sizes adding up: $i+1+(n-i-1) = n$.
- "Why divide by $n+1$?" The multiplicative factor was $(n+1)/n$, and $s_n := t_n/(n+1)$ exactly cancels it.
- Worst case is still $\Theta(n^2)$. The bound is on the *expected* number of comparisons, where the expectation is over the random pivot, not over the input. The worst case (pivot always lands at the extreme) still has runtime $\Theta(n^2)$ — but it has vanishing probability.

5 Closing summary

The structure of the argument in one line:

$$\mathbb{E}[T_{\ell,r}] \xrightarrow[\text{symmetry}]{\text{step 1}} t_n \text{ recurrence} \xrightarrow[\text{shift-and-subtract}]{\text{step 2}} t_n \le \tfrac{n+1}{n} t_{n-1} + 2 \xrightarrow[s_n = t_n/(n+1)]{\text{step 3}} \text{harmonic sum} \xrightarrow[H_n = \ln n + O(1)]{\text{step 4}} O(n \log n).$$

Three reusable tricks.

1. Symmetry → single variable. If the random source treats positions interchangeably, expected runtime depends only on size.
2. Shift-and-subtract. Whenever $n \cdot a_n = \sum_{i<n} f(a_i)$, write the same equation for $n-1$ and subtract.
3. Multiplicative-to-additive substitution. For $a_n \leq \frac{f(n)}{f(n-1)}\, a_{n-1} + d_n$, set $b_n = a_n / f(n)$.