Running Time of Quick Sort
[Notes for the Training Camp]

Yufei Tao

ITEE
University of Queensland
We will prove that quick sort (when executed on $n$ elements) finishes in $O(n \log n)$ expected time. We actually will give two proofs – one standard but tedious, and the other creative and interesting.
Recall:

**Quick Sort**

We will denote the input array as $A$.

**Inductive Case.** Otherwise, the algorithm runs the following steps:

1. Randomly pick an integer $p$ in $A$—call it the **pivot**.
   - This can be done in $O(1)$ time using $\text{RANDOM}(1, n)$.
2. Re-arrange the integers in an array $A'$ such that
   - All the integers smaller than $p$ are positioned before $p$ in $A'$.
   - All the integers larger than $p$ are positioned after $p$ in $A'$.
3. Sort the part of $A'$ before $p$ recursively.
4. Sort the part of $A'$ after $p$ recursively.
Analysis 1

Let $f(n)$ be the *expected* time of quick sort when performed on $n$ elements.

Clearly, $f(0)$ and $f(1)$ are constants.
Consider a general value of $n$. The algorithm finds a random pivot $p$, and spends $O(n)$ time to distribute the elements to the left and right of $p$.

Then, it recurs on the left and right of $p$, respectively. If we assume the left part of $p$ has $t$ elements, then the right part has $n - 1 - t$ elements. The recursion into those parts takes $f(t)$ and $f(n - 1 - t)$ expected time, respectively.

Remember that the algorithm is randomized. In other words, $t$ has $1/n$ probability to equal $0, 1, \ldots, n - 1$, respectively. This implies:

\[
\begin{align*}
f(n) &= O(n) + \frac{1}{n} \sum_{t=0}^{n-1} (f(t) + f(n - 1 - t)) \\
&= O(n) + \frac{2}{n} \sum_{t=0}^{n-1} f(t).
\end{align*}
\]
So it remains to solve the recurrence:

\[ f(n) \leq \alpha \cdot n + \frac{2}{n} \sum_{t=0}^{n-1} f(t) \]

where \( \alpha \) is a constant. This can be done using the substitution method, as shown below.

It suffices to find \( c \) such that \( g(n) \leq c \cdot n \ln n \) for sufficiently large \( n \). Inductively, we have

\[ g(n) \leq \alpha n + \frac{2}{n} \sum_{i=1}^{n-1} (c \cdot i \ln i) \]

\[ g(n) \leq \alpha n + \frac{2}{n} \int_1^n (c \cdot x \ln x) \, dx \]
Analysis 1

We know that, in general, \( \int (cx \ln x) \, dx = (c/2)x^2 \ln x - cx^2/4 \). Hence:

\[
\int_1^n (cx \ln x) \, dx = (c/2)n^2 \ln n - cn^2/4 + c/4
\]

Hence:

\[
g(n) \leq \alpha n + \frac{2}{n} \left( \frac{c}{2} n^2 \ln n - c \frac{n^2}{4} + \frac{c}{4} \right)
\]

\[
= cn \ln n - \left( \frac{c}{2} - \alpha \right) n + \frac{c}{2n}
\]

We want the above to be at most \( cn \ln n \), leading to the inequality of the next slide.
Analysis 1

\[
\frac{c}{2n} \leq \left(\frac{c}{2} - \alpha\right)n
\]

Setting \( c = 4\alpha \), the above holds for all \( n \geq 2 \).
Now we will give an alternative proof.

First, convince yourself that it suffices to analyze the number $X$ of comparisons. The running time is bounded by $O(n + X)$.

Next, we will prove that $E[X] = O(n \log n)$. 
Denote by $e_i$ the $i$-th smallest integer in $S$. Consider $e_i, e_j$ for any $i, j$ such that $i \neq j$.

What is the probability that quick sort compares $e_i$ and $e_j$?

This question – which seems to be difficult at first glance – has a surprisingly simple answer. Let us observe:

- Every element will be selected as a pivot precisely once.
- $e_i$ and $e_j$ are not compared, if any element between them gets selected as a pivot before them.

For example, consider $i = 7$ and $j = 12$. If $e_9$ is the pivot, then $e_i$ and $e_j$ will be separated by $e_9$. There is no chance that $e_i$ and $e_j$ can get compared in the subsequent execution.
Analysis 2

Therefore, $e_i$ and $e_j$ are compared if and only if either one is the first among $e_i, e_{i+1}, \ldots, e_j$ picked as a pivot.

The probability is $2/(j - i + 1)$ (random pivot selection).
Define random variable $X_{ij}$ to be 1, if $e_i$ and $e_j$ are compared. Otherwise, $X_{ij} = 0$. We thus have $\Pr[X_{ij} = 1] = 2/(j - i + 1)$. That is, $E[X_{ij}] = 2/(j - i + 1)$.

Clearly, $X = \sum_{i,j} X_{ij}$. Hence:

$$E[X] = \sum_{i,j} E[X_{ij}] = \sum_{i,j} \frac{2}{j - i + 1}$$

$$= 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j - i + 1}$$

$$= 2 \sum_{i=1}^{n-1} O(\log(j - i + 1))$$

$$= 2 \sum_{i=1}^{n-1} O(\log n) = O(n \log n).$$
As a final remark, the above analysis used the following fact:

\[ 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n} = O(\log n). \]