CS_332: Designing Algorithms **Faron Moller**
CS_M32: Algorithm Design and Analysis

Analysis of QuickSort

$\textsc{QuickSort}(A,p,r)$
if $p<r$ **then**
  $q \leftarrow \textsc{Partition}(A,p,r)$
  $\textsc{QuickSort}(A,p,q-1)$
  $\textsc{QuickSort}(A,q+1,r)$

$\textsc{Partition}(A,p,r)$
$x \leftarrow A[r]$
$i \leftarrow p-1$
for $j \leftarrow p$ **to** $r-1$ **do**
  if $A[j] \le x$ **then** $i \leftarrow i+1$
    $A[i] \leftrightarrow A[j]$
$A[i+1] \leftrightarrow A[r]$
return $i+1$

Informal Analysis: $\textsc{Partition}$ performs $\Theta(n)$ comparisons (where $n = r-p+1$).

Worst case: already sorted array, giving unbalanced partition:

$$T(n) = \Theta(n) + T(n-1) \implies T(n) = \underline{\underline{\Theta(n^2)}}.$$

Best case: even split every time:

$$T(n) = \Theta(n) + T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil -1) \implies T(n) = \underline{\underline{\Theta(n \lg n)}}.$$

"Average case": probably $\Theta(n \lg n)$, since even with a bad 99-1 split every time, we get

$$T(n) = \Theta(n) + T(\tfrac{99n}{100}) + T(\tfrac{n}{100}) \implies T(n) = \underline{\underline{\Theta(n \lg n)}}.$$

Randomized QuickSort: To avoid the worst case behaviour, we can exploit ***randomization*** in one of two ways:

1. randomly permute the array $A[1\ldots n]$ before calling $\textsc{QuickSort}(A,1,n)$.
2. partition with a randomly chosen partition element:

$\textsc{RandomizedPartition}(A,p,r)$
  $i \leftarrow \text{random}(p..r)$
  $A[r] \leftrightarrow A[i]$
  return $\textsc{Partition}(A,p,r)$

Average Case Analysis: Assume that all elements are distinct, and $\textsc{RandomizedPartition}$ chooses the kth-smallest element for the pivot with probability $1/n$.

$$\begin{aligned} T(n) &= \Theta(n) + \frac{1}{n} \sum_{0<k\le n} \Big[\underbrace{T(k-1) + T(n-k)}_{\text{pivot is } k^{\text{th}} \text{ smallest}} \Big] \\ &= \Theta(n) + \frac{2}{n} \sum_{0<k<n} T(k). \end{aligned}$$

That is, for some $c, n_0 > 0$, $T(n) \le cn + \frac{2}{n} \sum_{0<k<n} T(k)$ for all $n \ge n_0$.

Let $S(n) = \begin{cases} T(n), & \text{for } n < n_0; \\ cn + \frac{2}{n} \sum\limits_{0<k<n} S(k), & \text{for } n \geq n_0. \end{cases}$

Then $T(n) \leq S(n)$ for all n. *[Proof by induction on n.]*

For $n > n_0$ we get the following.

$$
\begin{aligned}
nS(n) &= cn^2 + 2 \sum_{0<k<n} S(k) && \textit{[multiply by } n \textit{]} \\
(n-1)S(n-1) &= c(n-1)^2 + 2 \sum_{0<k<(n-1)} S(k) && \textit{[replace } n \textit{ by } n-1 \textit{]} \\
nS(n) - (n-1)S(n-1) &= c[n^2 - (n-1)^2] + 2S(n-1) && \textit{[subtract]}
\end{aligned}
$$

$$
\begin{aligned}
nS(n) &= (n+1)S(n-1) + 2cn - c && \textit{[simplify]} \\
&\leq (n+1)S(n-1) + 2c(n+1)
\end{aligned}
$$

$$
\frac{S(n)}{n+1} \leq \frac{S(n-1)}{n} + \frac{2c}{n} \qquad \textit{[divide by } n(n+1) \textit{]}
$$

Let $F(n) = \frac{S(n)}{n+1}$. Then

$$
\begin{aligned}
F(n) &\leq F(n-1) + \frac{2c}{n} \\
&\leq \frac{2c}{n} + \frac{2c}{n-1} + \frac{2c}{n-2} + \cdots + \frac{2c}{n_0} + F(n_0-1) \\
&= 2c\left(\frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + \cdots + \frac{1}{n_0}\right) + \frac{S(n_0-1)}{n_0} \\
&\leq 2c\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right) + \frac{S(n_0-1)}{n_0} \\
&= 2cH_n + \frac{T(n_0-1)}{n_0} \\
&= 2c\,O(\lg n) + O(1) = O(\lg n).
\end{aligned}
$$

[Note: In the above, H_n are the *Harmonic numbers* $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = O(\lg n)$.]

Hence $S(n) = (n+1)F(n) = (n+1)O(\lg n) = O(n \lg n)$.

Thus $T(n) = O(n \lg n)$.

Since the *best* case runtime is $\Omega(n \lg n)$, the average case runtime is $T(n) = \Theta(n \lg n)$.