Exercise:
Prove the correctness of QuickSort by induction on the length of the subarray \( A[p..r] \); \( n = r-p+1 \).

The runtime of QuickSort depends heavily on the value of the partition. If the subarrays \( A[p..(r-1)] \) and \( A(r+1..r) \) are not balanced (i.e., of roughly equal size) then performance is inferior. In this case our recursive call to QuickSort is on a subarray which is inordinately long.

The worst case occurs when the array is already sorted. Then partition returns

\[
A[p...(r-1)] \leq A[r] \leq \ldots \text{ empty}
\]

Let \( T(n) \) denote the worst case run time of QuickSort (i.e., with \( A[p..r] \) already sorted.)
Then $T(n)$ satisfies

$$T(n) = \begin{cases} 
\Theta(1) & n=0,1 \\
T(n-1) + \Theta(n) & n \geq 2
\end{cases}$$

Simplify this to $T(n) = T(n-1) + c_n$ for definiteness. By the iteration method,

$$T(n) = c_n + T(n-1)$$
$$= c_n + c_{n-1} + T(n-2)$$
$$= c_n + c_{n-1} + c_{n-2} + T(n-3)$$
$$\vdots$$
$$= c \sum_{i=0}^{k-1} (n-i) + T(n-k)$$
$$= c n k - \frac{1}{2} c k (k-1) + T(n-k).$$

Choose $k$ such that $n-k=1$, i.e., $k=n-1$.

Then

$$T(n) = c n(n-1) - \frac{1}{2} c (n-1)(n-2) + \text{const}$$
$$= \Theta(n^2).$$

So what's so quick about Quicksort?
The advantage of Quicksort is in its average case.

We assume that all $n!$ permutations of the input array $A[1 \ldots n]$ are equally likely, i.e., that any given permutation occurs with probability $\frac{1}{n!}$.

We choose as basic operation (i.e., parameter) the comparison of numerical values on line 3 of Partition. Let $t(n)$ denote the average number of comparisons performed by Quicksort on an input array $A[1 \ldots n]$ of length $n$, i.e.

$$t(n) = \frac{\sum \text{(\# of comparisons performed on given permutation)}}{n!}$$

We wish to determine a recurrence for $t(n)$. Our assumption implies that the pivot $A[3]$ is equally likely to be placed in any of the $n$ locations in $A[1 \ldots n]$.

Thus the return value $q$ of Partition has probability $\frac{1}{n}$ of being any one of the $n$ values: $q = 1, 2, \ldots, n$. 

Observe that Partition itself does $(n-1)$ comparisons on $A[1...n]$. Also

$$\text{Length}[A[1...(n-1)]] = n-1$$

and

$$\text{Length}[A[(n+1)...n]] = n-q.$$

Thus

$$t(n) = \sum_{q=1}^{n} \left( (n-1) + t(4-1) + t(n-q) \right)$$

$$t(n) = (n-1) + \frac{1}{n} \sum_{q=1}^{n} \left( t(q-1) + t(n-q) \right)$$

The initial values $t(0) = 0, t(1) = 0, t(2) = 1$ can be seen by inspection. Thus

$$t(n) = (n-1) + \frac{1}{n} \left( \sum_{q=1}^{n-1} t(q) + \sum_{q=1}^{n-1} t(n-q) \right)$$

We have

$$t(n) = (n-1) + \frac{2}{n} \sum_{q=1}^{n-1} t(q)$$
To solve this recurrence we resort to some tricks. Let

\[ x_n = \sum_{t=1}^{n-1} t(q), \quad x_1 = 0. \]

Then

\[ x_{n+1} - x_n = \sum_{q=1}^{n} t(q) - \sum_{q=1}^{n-1} t(q) = t(n), \]

and so

\[ x_{n+1} - x_n = (n-1) + \frac{2}{n} \cdot x_n. \]

\[ x_{n+1} - \left(\frac{n+2}{n}\right) x_n = n-1. \]

Multiply by the magic number \( \frac{1}{(n+1)(n+2)} \)

\[ \frac{x_{n+1}}{(n+1)(n+2)} - \frac{x_n}{n(n+1)} = \frac{n-1}{(n+1)(n+2)} = \frac{3}{n+2} - \frac{2}{n+1}. \]

Replace \( n \) by \( r \).

\[ \frac{x_{r+1}}{(r+1)(r+2)} - \frac{x_r}{r(r+1)} = \frac{3}{r+2} - \frac{2}{r+1}. \]
Summation from 0 to n-1:

\[
\sum_{r=1}^{n-1} \left( \frac{x_{n+r}}{r(n+1)} - \frac{x_r}{r(r+1)} \right) = \sum_{r=1}^{n-1} \left( \frac{1}{r+2} + \frac{2}{r+2} - \frac{2}{r+1} \right)
\]

\[
\sum_{r=2}^{n} \frac{x_r}{r(n+1)} - \sum_{r=1}^{n-1} \frac{x_r}{r(n+1)} = \sum_{r=3}^{n+1} \frac{1}{r} + \sum_{r=3}^{n+1} \frac{2}{r} - \sum_{r=2}^{n} \frac{2}{r}
\]

\[
\frac{x_n}{n(n+1)} - \frac{x_1}{2} = \sum_{r=3}^{n+1} \frac{1}{r} + \frac{2}{n+1} - 1
\]

\[
= \sum_{r=1}^{n} \frac{1}{r} + \frac{1}{n+1} - 1 - \frac{1}{2} + \frac{2}{n+1} - 1
\]

\[
= \sum_{r=1}^{n} \frac{1}{r} + \frac{3}{n+1} - \frac{5}{2}
\]

Recall \( x_1 = 0 \). Define \( H_n = \sum_{r=1}^{n} \frac{1}{r} \) (called the \( n \)th harmonic number). Then

\[
\frac{x_n}{n(n+1)} = \frac{3}{n+1} - \frac{5}{2} + H_n
\]

\[
\therefore \quad x_n = 3n - \frac{5}{2} n(n+1) + n(n+1) H_n
\]

\[
\therefore \quad t(n) = (n-1) + \frac{2}{n} x_n
\]
\[ t(n) = -4n + 2(n+1) \cdot H_n \]

We must estimate the size of \( H_n \) to determine the asymptotic order of \( t(n) = \pm \).
Thus

\[ \Omega(\ln(n)) = \ln(n+1) \leq H_n \leq 1 + \ln(n) = O(\ln(n)) \]

\[ \therefore H_n = \Theta(\ln(n)) = \Theta(\log(n)) \]

Thus

\[ t(n) = \Theta(n \log(n)) \]

which is better than the worst case \( \Theta(n^2) \) run time.

Our assumption that all permutations of \( A[1 \ldots n] \) are equally likely may not be well founded in practice. We may wish to sort a pre-sorted array, or one that is nearly sorted more often than not.

There are several ways to randomize Quicksort to compensate for this.

One way is to simply apply a randomly chosen permutation to \( A[1 \ldots n] \) before calling Quicksort.