quick sort (Continued.)

With the partition algorithm, the result is

\[ \text{partition}(A[0, \ldots, n-1], p) \]
\[ A[0], \ldots, A[i-1] \leq p \]
\[ A[i], \ldots, A[n-1] > p \]

Based on the partition algorithm, the quicksort algorithm can be implemented as:

\[ \text{quicksort}(A[0, \ldots, n-1]) \]
\[ \text{if } n \leq 1 \]
\[ \text{return} \]
\[ \text{else} \]
\[ i \leftarrow \text{random}(0, 1, \ldots, n-1) \]
\[ \text{swap}(A[i], A[n-1]) \]
\[ t = \text{partition}(A[0, \ldots, n-2], A[n-1]) \]
\[ \text{swap}(A[t], A[n-1]) \]
\[ \text{quicksort}(A[0, \ldots, t-1]) \]
\[ \text{quicksort}(A[t+1, \ldots, n-1]) \]

**Corollary 1.** Suppose every partition splits array exactly into half, then

\[ T(n) = 2T\left(\frac{n}{2}\right) + O(n) \Rightarrow T(n) = O(n \log n) \]

For the worst case

\[ T(n) = T(n-1) + O(n) \Rightarrow T(n) = O\left(\sum_{i=1}^{n} i\right) = O(n^2). \]
**Definition 2.** A pivot of an array with length $n$ is said to be “good”, if

$$\frac{n}{4} \leq \text{rank of pivot} \leq \frac{3n}{4}$$

and correspondingly

$$P[\text{pivot is good}] = \frac{1}{2}.$$

**Corollary 3.** In a quick sort process of an array with length $n$, it takes at most $\log_4 n$ good pivots to reach the end of the path.

**Proof:**

Based on the property of good pivot, size of subarray after $k$th partition by good pivot is at most $\left(\frac{3}{4}\right)^k n$.

$$\left(\frac{3}{4}\right)^k n = 1 \Rightarrow k = \log_4 n. \square$$

**Corollary 4.** The length of each path in quick sort is $O(\log n)$ w.h.p.

**Proof:**

Assume the length of a path is $l$

$$X_i = \begin{cases} 0 & \text{if } i\text{th pivot is good} \\ 1 & \text{if } i\text{th pivot is bad} \end{cases}$$

and $X = \sum_{i=1}^{\ell} X_i$ is the number of bad pivot.

$$\mu = E[X] = \frac{l}{2}$$

then, based on Chernoff bound, we have the conclusion

$$X \leq \mu + O\left(\sqrt{\mu \log n}\right) \text{ w.h.p}$$

$$= \frac{l}{2} + O\left(\sqrt{\frac{l}{2} \log l}\right) \text{ w.h.p.}$$

The number of good pivots

$$l - X \geq \frac{l}{2} - O\left(\sqrt{\frac{l}{2} \log l}\right) \text{ w.h.p.}$$

Based on Corollary 3, a path needs at most $\log_4 n$ good pivots

$$\frac{l}{2} - O\left(\sqrt{\frac{l}{2} \log l}\right) \geq \log_4 n \text{ w.h.p.} \Rightarrow l \geq 2 \log_4 n \text{ w.h.p.}$$

$$\Rightarrow l = O(\log n) \text{ w.h.p.} \square$$
Theorem 5. Randomized quick sort run in \(O(n \log n)\) w.h.p.

Proof:
Every path is ended with an element in the array. There are \(n\) elements in
the array, thus there are \(n\) corresponding paths. Since the length of each path
in quick sort is \(O(\log n)\) w.h.p., randomized quick sort run in \(O(n \log n)\) w.h.p.

Next, we introduce how to find the \(k\)th rank element of an array with modiﬁed quick sort algorithm:

\[
kth(A[0, \ldots, n-1], k)
\]
\[
if \ n \leq 1
return A[0]
else
i \leftarrow random (0, 1, \ldots, n-1)
swap(A[i], A[n-1])
t = partition(A[0, \ldots, n-2], A[n-1])
swap(A[t], A[n-1])
if t > k
kth(A[0, \ldots, t-1], k)
else
kth(A[t, \ldots, n-1], k-t)
\]