10.7
Quick Sort
Quick Sort

Quick Sort [Hoare]

1. Pick a pivot element from array
2. Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself. Linear scan of array does it. Time is $O(n)$
3. Recursively sort the subarrays, and concatenate them.
Quick Sort

Quick Sort [Hoare]

1. Pick a pivot element from array
2. Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself. Linear scan of array does it. Time is $O(n)$
3. Recursively sort the subarrays, and concatenate them.
Quick Sort

Quick Sort [Hoare]

1. Pick a pivot element from array
2. Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself. Linear scan of array does it. Time is $O(n)$
3. Recursively sort the subarrays, and concatenate them.
Quick Sort [Hoare]

1. Pick a pivot element from array
2. Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself. Linear scan of array does it. Time is $O(n)$
3. Recursively sort the subarrays, and concatenate them.
Quick Sort: Example

array: 16, 12, 14, 20, 5, 3, 18, 19, 1

pivot: 16
Let $k$ be the rank of the chosen pivot. Then,

$$T(n) = T(k - 1) + T(n - k) + O(n)$$
Let $k$ be the rank of the chosen pivot. Then,
\[ T(n) = T(k - 1) + T(n - k) + O(n) \]

If $k = \lceil n/2 \rceil$ then
\[ T(n) = T(\lceil n/2 \rceil - 1) + T(\lfloor n/2 \rfloor) + O(n) \leq 2T(n/2) + O(n). \]
Then,
\[ T(n) = O(n \log n). \]
Let $k$ be the rank of the chosen pivot. Then,
\[ T(n) = T(k - 1) + T(n - k) + O(n) \]

If $k = \lceil n/2 \rceil$ then
\[ T(n) = T(\lceil n/2 \rceil - 1) + T(\lfloor n/2 \rfloor) + O(n) \leq 2T(n/2) + O(n). \]
Then,
\[ T(n) = \mathcal{O}(n \log n). \]

Median can be found in linear time.
Let $k$ be the rank of the chosen pivot. Then,
\[
T(n) = T(k - 1) + T(n - k) + O(n)
\]

If $k = \lceil n/2 \rceil$ then
\[
T(n) = T(\lceil n/2 \rceil - 1) + T(\lfloor n/2 \rfloor) + O(n) \leq 2T(n/2) + O(n).
\]
Then,
\[
T(n) = O(n \log n).
\]
Median can be found in linear time.

Typically, pivot is the first or last element of array. Then,
\[
T(n) = \max_{1 \leq k \leq n} (T(k - 1) + T(n - k) + O(n))
\]

In the worst case $T(n) = T(n - 1) + O(n)$, which means $T(n) = O(n^2)$. Happens if array is already sorted and pivot is always first element.
THE END

...(for now)