Notes, 5(b)

ECE 606

Quicksort

Divide: Partition (rearrange) the array $A[p\,..\,r]$ into two (possibly empty) subarrays $A[p\,..\,q-1]$ and $A[q+1\,..\,r]$ such that each element of $A[p\ ..\ q-1]$ is less than or equal to $A[q]$, which is, in turn, less than or equal to each element of $A[q+1\,..\,r]$. Compute the index q as part of this partitioning procedure.

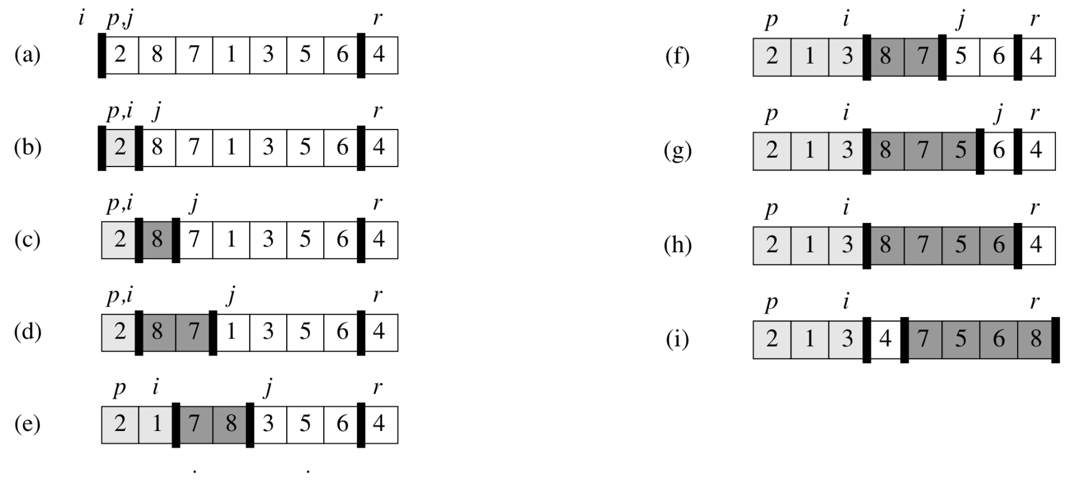
Conquer: Sort the two subarrays $A[p\,..\,q-1]$ and $A[q+1\,..\,r]$ by recursive calls to quicksort.

Combine: Since the subarrays are sorted in place, no work is needed to combine them: the entire array $A[p\,..\,r]$ is now sorted.

```
QUICKSORT(A, p, r)
1  if p < r
2     then q ← PARTITION(A, p, r)
3          QUICKSORT(A, p, q − 1)
4          QUICKSORT(A, q + 1, r)
```

```
PARTITION(A, p, r)
1  x ← A[r]
2  i ← p − 1
3  for j ← p to r − 1
4      do if A[j] ≤ x
5            then i ← i + 1
6                 exchange A[i] ↔ A[j]
7  exchange A[i + 1] ↔ A[r]
8  return i + 1
```

Partition, via example:



(a) i p,j r 2 8 7 1 3 5 6 4
(b) p,i j r 2 8 7 1 3 5 6 4
(c) p,i j r 2 8 7 1 3 5 6 4
(d) p,i j r 2 8 7 1 3 5 6 4
(e) p i j r 2 1 7 8 3 5 6 4
(f) p i j r 2 1 3 8 7 5 6 4
(g) p i j r 2 1 3 8 7 5 6 4
(h) p i r 2 1 3 8 7 5 6 4
(i) p i r 2 1 3 4 7 5 6 8

Claim 1. *The worst-case number of comparisons/exchanges in* $\textsc{Partition}(A, p, r)$ *is* $\Theta(n)$, *where* $n = r - p + 1$.

So, worst-case time-efficiency of Quicksort:

$$T(n) = \max_{0 \le q \le n-1} \{T(q) + T(n-q-1)\} + \Theta(n)$$

Claim 2. *The solution to the above recurrence is* $T(n) = \Theta(n^2)$.

We first prove that $T(n) = O(n^2)$. Towards this, we claim that $T(n) \le cn^2$ for some $c \in \mathbb{R}^+$, and sufficiently large n. We prove by induction on n.

For the step, because $q < n$ and $n - q - 1 < n$, by the induction assumption, we have:

$$\begin{aligned} T(n) &= \max_{0 \le q \le n-1} \{T(q) + T(n-q-1)\} + \Theta(n) \\ &\le \max_{0 \le q \le n-1} \left\{cq^2 + c(n-q-1)^2\right\} + \Theta(n) \\ &= c \cdot \max_{0 \le q \le n-1} \left\{q^2 + (n-q-1)^2\right\} + \Theta(n) \end{aligned}$$

Now let $f(q) = q^2 + (n-q-1)^2$, where n is treated as a constant, and $0 \le q \le n-1$. We seek to maximize $f(q)$.

$f(q) = q^2 + (n-1)^2 + q^2 - 2(n-1)q$	binomial expansion of $(n-1-q)^2$
$f'(q) = 4q - 2(n-1)$	first derivative of $f(q)$
$f''(q) = 4$	second derivative of $f(q)$

Thus:

$$\begin{aligned} f'(q) = 0 &\iff q = (n-1)/2, \text{ and,} \\ f''(q) > 0 &\implies q = (n-1)/2 \text{ is a minima} \\ &\implies f(q) \text{ maximized at one of the end-points} \end{aligned}$$

$f(0) = f(n-1)$, so pick either end-point to maximize f. So, $T(n) \le c \cdot (n-1)^2 + \Theta(n)$.

Now, we can prove that worst-case time-efficiency of Quicksort, $T(n) = O(n^2)$. Because:

$$T(n) \le cn^2 - c(2n-1) + c'n, \text{ for all } n > \text{some } n_0$$

Where the c' and n_0 correspond to Partition.

Now, we observe that we can pick c large enough so that $c(2n-1)$ dominates $c'n$. Specifically:

$$\begin{aligned} c(2n-1) > c'n &\iff c > c' \times \frac{1}{2 - 1/n} \\ &\Longleftarrow c > c', \text{ for all } n \ge 1 \end{aligned}$$

So, if the induction assumption is true, then for all $n > n_0$, $T(n) \le cn^2 \implies T(n) = O(n^2)$.

For the base-case, it depends on base-case that is specified for the recurrence. We could, for example, assume that the base-case is specified for all $n \le n_0$. And then, all we would need to do is pick c sufficiently large so that $T(n_0) \le cn_0^2$. That is, we pick $c > \max\{c', T(n_0)/n_0^2\}$.

If, on the other hand, the base-case in the recurrence is, for example, $T(1) = \Theta(1)$ and $n_0 > 1$, then we simply pick $c > \max\{c', T(n_0)/n_0^2\}$, where $T(n_0) = \max_{0 \le q \le n_0 - 1}\{T(q) + \underset{T}{\cancel{1}}(q - n_0 - 1)\} + c'n_0$.

To prove that $T(n) = \Omega(n^2)$ in the worst-case, all we need is to prove that there exists a run of Quicksort that elicits such a time-efficiency. For that, suppose Partition always chooses the smallest item as pivot, i.e., $q = 0$. Then, we observe that the recurrence is:

$$T(n) = \begin{cases} T(0) + T(n-1) + \Theta(n) & \text{if } n > 1 \\ \Theta(1) & \text{otherwise} \end{cases}$$

The solution the to above recurrence is $T(n) = \Theta(n^2)$. Thus, in the worst-case, the time-efficiency of Quicksort is $\Theta(n^2)$.

For the expected case, suppose we assume that all items in the array are distinct, and PARTITION chooses a pivot uniformly from amongst all possibilities.

A somewhat informal way to intuit what happens in expectation is to ask how well PARTITION splits the array in expectation. That is, think of PARTITION as splitting an input array of length n into a "smaller piece," and a "bigger piece." We ask: what is the expected length of the smaller piece?

If S is the random variable that is the length of the smaller piece, then:

$$\begin{aligned} E[S] &= \frac{2}{n} \times 0 + \frac{2}{n} \times 1 + \ldots + \frac{2}{n} \times \frac{n}{2} \\ &= \frac{2}{n}\left(0 + 1 + \ldots + \frac{n}{2}\right) \\ &= \frac{2}{n} \times \frac{\frac{n}{2}\left(\frac{n}{2} - 1\right)}{2} \approx \frac{n}{4} \end{aligned}$$

So, in expectation, the time-efficiency of Quicksort can be written as the following recurrence:

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + \Theta(n)$$

If we draw a recursion tree, we observe that the longest path from the root to a leaf has length $\log_{4/3} n$. The shortest has length $\log_4 n$. Therefore, there exist constants call them c_1, c_2 such that $c_1 \cdot n \log_4 n \leq T(n) \leq c_2 \cdot n \log_{4/3} n$.

Therefore, $T(n) = \Theta(n \lg n)$.

Your textbook, i.e., CLRS, has a more rigorous approach, which is the following. Suppose we count the number of comparisons that are made in a run of Quicksort; let X be the corresponding random variable.

We first observe: given any two items in the input array, call them z_i and z_j, we compare them at most once in a run of Quicksort. Because we compare them only if: (i) they both end up in the same partition at some point, (ii) one of them is chosen as a pivot, and, (iii) if one of them, say z_i, is indeed chosen as a pivot, after one comparison with each other item in the partition, z_i is never again compared with anything else.

Actually, we can say something stronger. Suppose the items in the input array A in sorted order are $z_1, z_2, \ldots, z_n$. Let $Z_{i,j} = \{z_i, z_{i+1}, \ldots, z_j\}$, for $j > i$.

Claim 3. *(i) Some member of every $Z_{i,j}$, where $j > i$, must be chosen as pivot at some point in a run of Quicksort, and, (ii) z_i and z_j are compared in a run of Quicksort if and only if the first item to be chosen as a pivot from $Z_{i,j}$ is either z_i or z_j.*

Note: the "if and only if," i.e., both directions of implication, is very important in (ii) in the above claim. An implication in one direction does not suffice for us.

Given the above claim, things are straightforward. Let $X_{i,j} = I\{z_i \text{ is compared to } z_j\}$. Then:

$$
\begin{aligned}
X &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j} \\
E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{i,j}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr\{z_i \text{ is compared to } z_j\} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr\{z_i \text{ or } z_j \text{ is first pivot chosen from } Z_{i,j}\} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\Pr\{z_i \text{ is first pivot chosen from } Z_{i,j}\} + \Pr\{z_j \text{ is first pivot chosen from } Z_{i,j}\}\right) \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\frac{1}{j-i+1} + \frac{1}{j-i+1}\right) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \\
&< \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k} = \sum_{i=1}^{n-1} O(\lg n) = O(n \lg n)
\end{aligned}
$$