Notes, 5(c)

ECE 606

Order Statistics

The "ith order statistic" is the i^{th} smallest element in a set of items. If the set has n items, then:

- If $i = 1$, we mean the smallest item in the set.
- If $i = n$, we mean the largest item.
- If $i = \lfloor (n+1)/2 \rfloor$, we mean the (lower) median.

Our problem, call it "selection":

- Inputs:
    1. an array $A[1, \ldots, n]$ of distinct intgers, not necessarily sorted, and,
    2. $i \in \{1, \ldots, n\}$.
- Output: the (index of the) ith order statistic of $A[1, \ldots, n]$.

An algorithm: sort A, return $A[i]$. Time-efficiency: $\Omega(n \lg n)$.

Challenge: an algorithm which runs in time $o(n \lg n)$, perhaps even $O(n)$.

We assume p < r, and i is between 1 and r - p + 1. Because between p and r in the array, we only have this many items.

RANDOMIZED-SELECT(A, p, r, i)

if $p = r$
    then return $A[p]$
$q \leftarrow$ RANDOMIZED-PARTITION(A, p, r)
$k \leftarrow q - p + 1$
if $i = k$ $\triangleright$ the pivot value is the answer
    then return $A[q]$
elseif $i < k$
    then return RANDOMIZED-SELECT$(A, p, q - 1, i)$
else return RANDOMIZED-SELECT$(A, q + 1, r, i - k)$

This means that the pivot is chosen uniformly amongst the items between A[p] and A[r]. And q is the index in which the pivot ends up.

In the worst-case, a recurrence for time-efficiency is:

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

We have done theta(n) work in this particular recursive call.

A solution is $T(n) = \Theta(n^2)$.

In expectation, however, PARTITION chooses a pivot such that an input array is split into 1/4 and 3/4. So, in expectation, $T(n) \leq T\left(\frac{3n}{4}\right) + \Theta(n)$.

And that has solution:

cn is the representative function for theta(n)

$$
\begin{aligned}
T(n) &\leq T\left(\frac{3n}{4}\right) + cn \\
&\leq T\left(\left(\frac{3}{4}\right)^2 \cdot n\right) + \frac{3}{4}cn + cn \\
&\leq T\left(\left(\frac{3}{4}\right)^3 \cdot n\right) + \left(\frac{3}{4}\right)^2 cn + \left(\frac{3}{4}\right)^1 cn + \left(\frac{3}{4}\right)^0 cn \\
&\dots \\
&\leq T\left(\left(\frac{3}{4}\right)^{\log_{4/3} n} \cdot n\right) + \left(\frac{3}{4}\right)^{(\log_{4/3} n)-1} cn + \dots + \left(\frac{3}{4}\right)^1 cn + \left(\frac{3}{4}\right)^0 cn \\
&= T(1) + \left(\frac{3}{4}\right)^{(\log_{4/3} n)-1} cn + \dots + \left(\frac{3}{4}\right)^1 cn + \left(\frac{3}{4}\right)^0 cn \\
&= c + cn\left(\left(\frac{3}{4}\right)^{(\log_{4/3} n)-1} + \dots + \left(\frac{3}{4}\right)^1 + \left(\frac{3}{4}\right)^0\right) \\
&= c + cn\left(\frac{1 - \left(\frac{3}{4}\right)^{\log_{4/3} n}}{1 - \frac{3}{4}}\right) \\
&= c + cn\left(\frac{1 - 1/n}{1/4}\right) = c + cn \cdot \frac{4(n-1)}{n} \\
&= c + 4c(n-1) = \Theta(n)
\end{aligned}
$$

In which we exploit:

$$
\begin{aligned}
\left(\frac{3}{4}\right)^{\log_{4/3} n} &= \left(\frac{4}{3}\right)^{-\log_{4/3} n} && \because a^x = (1/a)^{-x} \\
&= \left(\frac{4}{3}\right)^{\log_{4/3} (1/n)} && \because -\log x = \log(1/x) \\
&= \frac{1}{n} && \because a^{\log_a x} = x
\end{aligned}
$$

It turns out that we can carry out selection in worst-case time $O(n)$.

SELECT($A[1,\ldots,n], i$)

1 If $n \leq$ some constant, solve by brute-force and return
2 Perceive $A[1,\ldots,n]$ of being made up of $n/5$ groups of 5 elements each
3 Find the median of each group of 5; let these be $M = [m_1, m_2, \ldots, m_{n/5}]$
4 Call SELECT recursively to find the median, m, of M
5 PARTITION A with m as the pivot
6 Suppose PARTITION returns the index q where the pivot m ends up
7 **if** $i = q$ **then return** $A[i]$
8 **else if** $i < q$ **then return** SELECT($A[1,\ldots,q-1], i$)
9 **else return** SELECT($A[q+1,\ldots,n], i-q$)

The key insight the algorithm exploits is that m, "the median of medians," turns out to be a very good pivot to use in PARTITION. That is, if we partition around m as pivot, we get a good split.

To see why, we ask: how large could $A[1,\ldots,q-1]$ possibly be?

Answer: because m is the median of M, half the items in M are smaller than m and half are larger.

Now consider the group of 5 items of which m_i is the median. If $m_i < m$, then all 5 of those items may be smaller than m. If $m_i > m$, then at most 2 items of those 5 may be smaller than m. Thus, the maximum number of items in $A[1,\ldots,n]$ that can be smaller than m is:

$$5 \times \frac{1}{2} \times \frac{n}{5} + 2 \times \frac{1}{2} \times \frac{n}{5} = \frac{7}{10} \times n$$

Similarly, the maximum number of items that may be larger than m is also $7n/10$. So immaterial of whether we recurse in Line (8) or Line (9), we recurce on an array whose size is at worst 7/10 the size of the input array. So recurrence for running-time in the worst-case:

We recursively invoke SELECT on array M.

$$\begin{aligned} T(n) &= T(n/5) + T(7n/10) + \Theta(n) \\ &\approx T(9n/10) + \Theta(n) \end{aligned}$$

Line2, 3, 5,.. All of these together are theta(n).

And in the worst case, we might recurs on something of size as large as 7n/10

Which has solution $T(n) = O(n)$.