Lecture 5: The Linear Time Selection in the worst case

In the last lecture, we discussed a *randomized* selection algorithm that runs in $O(n)$ in average. In this class, we discuss a *deterministic* algorithm that runs in $O(n)$ in the worst case.

Observation and Intuition

If we follow the `Partition` idea to solve the selection problem, which step(s) make the worst case running time becomes $O(n^2)$?

We make a 'bad' split in each iteration. So the trick here is in each iteration, we 'pick' a good element such that it 'guarantees' a good split.

How to get such a 'good' element in each iteration?

`DSelection(A, p, r, i)`

1. Divide the $n = p - r + 1$ items into $\lceil n/5 \rceil$ sets in which each, except possibly the last, contains 5 items. $O(n)$

2. Find *median* of each of the $\lceil n/5 \rceil$ sets. $O(n)$

3. Take these $\lceil n/5 \rceil$ medians and put them in another array. Use `DSelection()` to *recursively* calculate the *median* of these medians. Call this x. $T(n/5)$

4. Partition the original array using x as the pivot. Let q be index of x, i.e., x is the $k = q - p + 1$'st smallest element in original array. $O(n)$

5. If $i = k$ return x
   If $i < k$ return `DSelection(A,p,q-1,i)`
   If $i > k$ return `DSelection(A,q+1,r,i-k)`
   $T(\max(q - p, r - q))$

<u>Termination condition:</u>
If $n \leq 5$ sort the items and return the ith largest.

The algorithm returns the correct answer because lines 4 and 5 will always return correct solution, no matter which x is used as pivot.

The reason for lines 1, 2, and 3 is to guarantee that
x is “near” the center of the array $\Rightarrow$ a ’good’ split.

How many elements in A are greater (less) than x?. Answer (proven next page): At least

$$\frac{3n}{10} - 6.$$

Assuming that $T(n)$ is non-decreasing this implies that time used by step 5 is at most

$$T\left(\frac{7n}{10}\right) + 6.$$

Lemma: At least

$$\frac{3n}{10} - 6$$

elements are greater (less) than x.

Proof: We assume that all elements are distinct (not needed but makes the analysis a bit cleaner).

At least 1/2 of the $\lceil \frac{n}{5} \rceil$ medians in step 2 are *greater* than x.

Ignoring the group to which x belongs and the (possibly small) final group this leaves $\frac{1}{2}\left\lceil \frac{n}{5} \right\rceil - 2$ groups whose medians are greater than x.

Each such group has *at least* 3 items greater than x. Then, number of items greater than x is *at least*

$$3\left(\frac{1}{2}\left\lceil \frac{n}{5} \right\rceil - 2\right) \geq \frac{3n}{10} - 6$$

Analysis of number less than x is exactly the same!

Running Time of Algorithm

Assume any input with $n \le 140$ uses $O(1)$ time.

Let a be such that Steps 1,3,4 need at most an time.

Assume that $T(n)$ is non-decreasing. Then

$$T(n) \le \begin{cases} \Theta(1) & \text{if } n \le 140 \\ T(\lceil n/5 \rceil) + T(7n/10 + 6) + an & \text{if } n > 140 \end{cases}$$

We will show, by induction that $T(n) \le cn$, $\forall n > 0$.
Choose c large enough that
$\forall n \le 140$, $T(n) \le cn$.
By induction hypothesis

$$\begin{aligned} T(n) &\le T(\lceil n/5 \rceil) + T(7n/10 + 6) + an \\ &\le c\lceil n/5 \rceil + c(7n/10 + 6) + an \\ &\le cn/5 + c + 7cn/10 + 6c + an \\ &= 9cn/10 + 7c + an \\ &= cn + (-cn/10 + 7c + an) \end{aligned}$$

Have already seen that

$$T(n) \le cn + (-cn/10 + 7c + an).$$

We want to show that $T(n) \le cn$ so we would be finished if, $\forall n \ge 140$

$$\begin{aligned} 0 &\ge -cn + 70c + 10an \\ &= -c(n - 70) + 10an \end{aligned}$$

or

$$c \ge 10a(n/(n-70)).$$

Since $n \ge 140$ we have $n/(n-70) < 2$ so this will be true for any $c \ge 20a$ and we have shown that $T(n) \le cn$ for all $n \ge 140$ and

$$T(n) = O(n).$$