Running Time of the Selection Algorithm

By definition, the find-pivot functions takes time $\Theta(n) + T(|M|)$. Therefore, the total running time of the selection problem, in the form of a recurrence, is $T(n) = \Theta(n) + T(|M|) + \max\{T(|A_1|), T(|A_2|)\}$.

We now bound the size of $|M|$ and $\max\{|A_1|, |A_2|\}$. Clearly, $|M| = n/5$, as the number of subarrays is $n/5$. Think: how many numbers in A are *guaranteed* smaller than x (this number then gives a lower bound of $|A_1|$)? First, in these $n/5$ medians, half of them, i.e., $n/10$ numbers, are smaller than x, as x is the median of these medians. Second, consider these $n/10$ subarrays whose median is smaller than x: clearly in each of these subarrays, at least two numbers are smaller than x. This is because the median of this subarray is smaller than x and there are two numbers in this subarray that are smaller than its median. Combined, we have a lower bound of $|A_1|$: $|A_1| \geq n/10 + 2 \cdot n/10 = 3n/10$. This gives an upper bound of $|A_2|$: $|A_2| = n - |A_1| \leq 7n/10$.

Symmetrically, we have that $|A_2| \geq 3n/10$ and hence $|A_1| = n - |A_2| \leq 7n/10$. This is because, in these $n/5$ medians, $n/10$ of them are larger than x, and in these corresponding $n/10$ subarrays whose median is larger than x, there are in total $2 \cdot n/10$ numbers larger than x.

Combined, we have $\max\{|A_1|, |A_2|\} \leq 7n/10$. The above recurrence becomes $T(n) \leq \Theta(n) + T(n/5) + T(7n/10)$. How to solve this recurrence? Here is the conclusion (you will see its prove via assignment). For a more generalized version is this recurrence: $T(n) = \Theta(n) + T(c_1 n) + T(c_2 n)$, where $0 < c_1, c_2 < 1$, we have $T(n) = \Theta(n)$ if $c_1 + c_2 < 1$; $T(n) = \Theta(n \log n)$ if $c_1 + c_2 = 1$. For the selection problem we have $c_1 + c_2 = 1/5 + 7/10 < 1$. Hence its running time $T(n) = \Theta(n)$.

Choices of the Size of Subarrays

How about we partition A into subarrays of size 3? Note, in this case the algorithm is still correct. But will the algorithm still run in linear time? Let's analyze it. Now we have $|M| = n/3$, as the number of subarrays is $n/3$. In these $n/3$ medians, half of them, i.e., $n/6$ numbers, are smaller than x, and in these corresponding $n/6$ subarrays whose median is smaller than x, there are in total $1 \cdot n/6$ numbers smaller than x. This gives that $|A_1| \geq n/6 + n/6 = n/3$, which gives $|A_2| \leq 2n/3$. Symmetrically we can prove $|A_1| \leq 2n/3$ and combined we have $\max\{|A_1|, |A_2|\} \leq 2n/3$. The recursion in this case, will be $T(n) = \Theta(n) + T(n/3) + T(2n/3)$. In fact, now $T(n) = \Theta(n \log n)$ as $1/3 + 2/3 = 1$. In sum, choosing subarrays of size 3 won't give a linear time algorithm. (Note: by using the idea of "median-of-median-of-medians", a linear-time algorithm can still be obtained in this case; see assignment.)

How about we partition A into subarrays of size 7? Now we have $|M| = n/7$, as the number of subarrays is $n/7$. In these $n/7$ medians, half of them, i.e., $n/14$ numbers, are smaller than x, and in these corresponding $n/7$ subarrays whose median is smaller than x, there are in total $3 \cdot n/14$ numbers smaller than x. This gives that $|A_1| \geq n/14 + 3n/14 = 2n/7$, which gives $|A_2| \leq 5n/7$. Symmetrically we can prove $|A_1| \leq 5n/7$ and combined we have $\max\{|A_1|, |A_2|\} \leq 5n/7$. The recursion in this case, will be $T(n) = \Theta(n) + T(n/7) + T(5n/7)$. So $T(n) = \Theta(n)$ as $1/7 + 5/7 < 1$. In fact, any odd size that is larger than 5 will lead to a linear-time algorithm. But bigger size will result in bigger factor in sorting these subarrays. For example, compare size of 7 and size of 5: it takes $n/7 \cdot 7 \cdot \log 7 = \log 7 \cdot n$ time to sort in the case of size 7, which is larger than $\log 5 \cdot n$ in the case of size 5.

Randomized Algorithm for Selection Problem

We now design a *randomized algorithm* for selection problem. The idea is simply pick the pivot uniformly at random from A. The pseudo-code is given below.

```
function find-pivot-random (A)
    pick pivot x uniformly at random from A;
end function ;
```

First, note that the selection algorithm combined with above random function to pick pivot is correct, i.e., it will still find the k-th smallest number of A. We now analyze its running time. Again, let $T(n)$ be the running time of selection (A, k), with above random function to select pivot, when $|A| = n$. Define random variable $Z := \max\{|A_1|, |A_2|\}$. Hence we can write $T(n) = \Theta(n) + T(Z)$. Again, here Z is a random variable, $T(Z)$ is a random variable, and therefore $T(n)$ is also a random variable.

We aim to calculate the expected running time, a common practice in analyzing randomized algorithms. We first estimate the distribution of Z. Think: what's the probability for event $Z \leq 3n/4$? Answer: at least $1/2$. Why? This is because we pick x uniformly at random from x. Therefore, the probability of event of $\{x$ is between 25-percentile and 75-percentile of $A\}$ is $1/2$. And this event is equivalent to the event that $Z \leq 3n/4$, according to the definition of Z. Hence, $\Pr(Z \leq 3n/4) = 1/2$.

We now calculate its expected running time. We start with recursion $T(n) = \Theta(n) + T(Z)$. We first take expectation over Z on both sides: $\mathcal{E}_Z[T(n)] = \Theta(n) + \mathcal{E}_Z[T(Z)]$. Note that $T(n)$ does not contain Z (although $T(n)$ is a random variable), we have $\mathcal{E}_Z[T(n)] = T(n)$. That is $T(n) = \Theta(n) + \mathcal{E}_Z[T(Z)]$.

We now estimate $\mathcal{E}_Z[T(Z)]$.

$$
\begin{aligned}
\mathcal{E}_Z[T(Z)] &= \sum_{k=n/2}^{n} \Pr(Z = k) \cdot T(k) \\
&= \sum_{k=n/2}^{3n/4} \Pr(Z = k) \cdot T(k) + \sum_{k=3n/4}^{n} \Pr(Z = k) \cdot T(k) \\
&\leq T(3n/4) \cdot \sum_{k=n/2}^{3n/4} \Pr(Z = k) + T(n) \cdot \sum_{k=3n/4}^{n} \Pr(Z = k) \\
&= T(3n/4) \cdot \Pr(Z \leq 3n/4) + T(n) \cdot \Pr(Z \geq 3n/4) \\
&\leq T(3n/4) \cdot 1/2 + T(n) \cdot 1/2.
\end{aligned}
$$

Hence, now we have $T(n) \leq \Theta(n) + T(3n/4)/2 + T(n)/2$, which gives $T(n) \leq \Theta(n) + T(3n/4)$. We now take expectation, over $T(n)$, on both sides: $\mathcal{E}_T[T(n)] = \Theta(n) + \mathcal{E}_T[T(3n/4)]$. By using master's theorem, we have that $\mathcal{E}_T[T(n)] = \Theta(n)$.