CS 161: Recitation 2 Solutions

Selection, Big-O, and Recurrences

1. Recall the "Master Method" theorem from lecture:

   Given a recurrence $T(n) = aT(\frac{n}{b}) + O(n^d)$ with $a \geq 1$, and $b > 1$, and $T(1) = \Theta(1)$, then

$$T(n) = \begin{cases} O(n^d \log n) & \text{if } a = b^d \\ O(n^d) & \text{if } a < b^d \\ O(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

   Describe the intuition for the three cases, and the intuitive interpretation of the corresponding bounds on $T(n)$ for those three cases.

   Solution. For the first case, each level in our tree has the same amount of work, and there are $\log n$ levels.

   For the second case, each level is a multiple smaller than the previous level, meaning that $(\frac{a}{b^d})^i \to 0$ as $i \to \infty$, and we know that decreasing geometric series are finite, bounded by some constant factor times the first term. Hence the total amount of work is, up to a constant factor, the amount of work done at the top level.

   In the third case, the amount of work at each level grows as we travel down the computation tree. It makes sense, then that the amount done total is a (up to a constant multiple) just the work done at the leaves of the computation tree. (You should convince yourself that there is $O(n^{\log_b a})$ work done at the leaves!)

2. Find a runtime bound for $T(n) = 3T(\frac{n}{2}) + O(n^2)$, with $T(1) = 1$.

   Solution. For this problem, we can apply the master method directly. In this case, $a = 3$, $b = 2$, and $d = 2$. Since $a = 3 < b^d = 4$, we fall into case 2 of the master theorem. So, the runtime is $O(n^d) = O(n^2)$.

3. Find a runtime bound for $T(n) = T(n-1) + O(\frac{1}{n})$, with $T(1) = 1$

   Solution. Notice that this problem does not fit the formula for the master method. So, instead of trying to apply the master method, we will expand this recurrence and try to find a pattern:

$$T(n) = O(\frac{1}{n}) + T(n-1)$$

$$= O(\frac{1}{n}) + O(\frac{1}{n-1}) + T(n-2)$$

$$= O(\frac{1}{n}) + O(\frac{1}{n-1}) + O(\frac{1}{n-2}) + T(n-3)$$

$$= \sum_{i=1}^{n} O(\frac{1}{i})$$

This is the harmonic series, and the harmonic series is $\Theta(\log n)$, so the upper bound on the runtime is $O(\log n)$. (Challenge: prove that the harmonic series is $\Theta(\log n)$!)

4. Our `select` algorithm from class divided the n-length array into $\frac{n}{5}$ lists of 5. Analyze the runtime of `select` if we instead divided the n-length array into $\frac{n}{7}$ lists of 7.

Solution We have $\lceil n/7 \rceil$ medians (including one extra group with fewer than 7 elements). Half of these medians are guaranteed to contain at least 4 elements that are greater than the median of medians. The only exceptions are (1) the extra group and (2) the group containing fewer than 7 elements. Therefore, at least

$$4\left(\lceil \frac{1}{2} \lceil \frac{n}{7} \rceil \rceil - 2\right) \geq \frac{2n}{7} - 8$$

elements are guaranteed to be greater than the median of medians. Therefore, we discard at least $2n/7 - 8$ elements and recurse on subproblems of at most $5n/7 + 8$.

The recursive call to Select that finds the median of medians takes $T(\lceil n/7 \rceil)$ time, and recursing takes $\leq T(\lceil 5n/7 + 8 \rceil)$. The rest of the steps take $O(n)$ time. The recurrence is

$$T(n) \leq \begin{cases} O(1) & \text{if } n < n_0 \\ T(\lceil n/7 \rceil) + T(\lceil 5n/7 + 8 \rceil) + O(n) & \text{if } n \geq n_0 \end{cases}$$

We use the substitution method to solve the recurrence. Specifically, we will show that $T(n) \leq cn$ for some constant c and all $n > 0$.

For the base case, we choose c large enough that $T(n) \leq cn$ for all $n < n_0$. Since n_0 is a constant, we can achieve this by declaring c to be a relatively large constant.

Now for the inductive step, we assume $T(k) \leq ck$ for all $k < n$. Also, let a be the constant from the $O(n)$ term (i.e. a constant such that the function described by the $O(n)$ term is bounded above by an for all $n > 0$).

By the inductive hypothesis, we have

$$\begin{aligned} T(n) &\leq c\lceil n/7 \rceil + c(\lceil 5n/7 \rceil + 8) + an \\ &\leq cn/7 + c + 5cn/7 + c + 8c + an \\ &= 6cn/7 + 10c + an \\ &= cn + (-cn/7 + 10c + an) \end{aligned}$$

which is at most cn if $-cn/7 + 10c + an \leq 0$.