CS31 (Algorithms), Spring 2020 : Lecture 3 Supplement

Date:

Topic: Divide and Conquer

Disclaimer: These notes have not gone through scrutiny and in all probability contain errors. Please discuss in Piazza/email errors to deeparnab@dartmouth.edu

---

1 Proof of the Master Theorem

Theorem 1. Consider the following recurrence:

$$T(n) \leq a \cdot T(\lceil n/b \rceil) + O(n^d)$$

where a, b, d are non-negative integers. Then, the solution to the above is given by

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

Proof. As before, we replace the $O(n^d)$ by $\leq C \cdot n^d$ for some constant $C > 0$. We also replace the $\lceil n/b \rceil$ by the simpler n/b and worry about how to handle these later. Once again, you should try drawing the recursion tree and see how the kitty grows. Here we give the "opening the brackets and consolidating" method.

$$\begin{aligned}
T(n) &\leq aT(n/b) + C \cdot n^d \\
&\leq a\left(aT(n/b^2) + C \cdot (n/b)^d\right) + C \cdot n^d \\
&= a^2 T(n/b^2) + Cn^d \cdot \left(1 + \frac{a}{b^d}\right) \\
&\leq a^2\left(aT(n/b^3) + C \cdot \left(n/b^2\right)^d\right) + Cn^d \cdot \left(1 + \frac{a}{b^d}\right) \\
&= a^3 T(n/b^3) + Cn^d \cdot \left(1 + \frac{a}{b^d} + \left(\frac{a}{b^d}\right)^2\right) \\
&\vdots \\
&\leq a^k T(n/b^k) + Cn^d \cdot \left(1 + \frac{a}{b^d} + \left(\frac{a}{b^d}\right)^2 + \cdots + \left(\frac{a}{b^d}\right)^{k-1}\right) \qquad (1) \\
& \qquad (2)
\end{aligned}$$

To finish the proof, we need to argue about the Geometric Sum in (1). Note that when $a = b^d$, the geometric sum evaluates to k. When $a < b^d$, then if we denote $1 > \rho = a/b^d$, the Geometric Sum evaluates to $\leq \frac{1}{1-\rho} = O(1)$ since a, b, d are constants. When $a > b^d$, then if we denote $1 < \rho = a/b^d$, the Geometric Sum evaluates to $\frac{\rho^k - 1}{\rho - 1}$. In sum, we get for any k,

$$T(n) = a^k T(n/b^k) + Cn^d \cdot \begin{cases} O(k) & \text{if } a = b^d \\ O(1) & \text{if } a < b^d \\ O((a/b^d)^k) & \text{if } a > b^d \end{cases}$$

To finish the proof, we set $k = \log_b n$ in which case the first term $a^k T(n/b^k)$ evaluates to $O(a^{\log_b n}) = O(n^{\log_b a})$. When $a \le b^d$, we get $n^{\log_b a} \le n^d$. Thus in Case 1 and Case 2 above, we see that the Master theorem follows. The third case of the above cases evaluates to

$$\left(\frac{a}{b^d}\right)^k = \left(b^{\log_b (a/b^d)}\right)^k = \left(b^{\log_b n}\right)^{\log_b (a/b^d)} = n^{(\log_b a - d)} = n^{\log_b a}/n^d$$

Therefore, in the third case the second summand evaluates to $O(n^{\log_b a})$. □

2 Taking care of ceilings and floors.

Let's address the icky detail of "forgetting" the ceilings and floors. We will prove this for the mergesort recurrence and to see you have understood it, you should try it to prove the master theorem. This is left as an exercise below. Let us recall the true mergesort recurrence

$$T(n) \le T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n) \tag{3}$$

First, we will use *monotonicity* of $T()$. That is, we will use the fact for any $a \le b$ we have $T(a) \le T(b)$. Why is that the case? This follows from the definition of $T - T(a)$ is the worst case running time among all instances of size $\le a$. These clearly are a subset of all instances of size $\le b$. Therefore, we will simplify by stating

$$T(n) \le 2T(\lceil n/2 \rceil) + O(n) \le 2T(n/2 + 1) + O(n)$$

where we have used the fact $\lfloor x \rfloor \le \lceil x \rceil < x + 1$ for any x.

Next, we explicitly write what $O(n)$ means (recall the lecture on Big-Oh, section on abuse of notation). We get that there exists positive constants A, B such that

$$T(1) \le B, \quad \text{and} \quad T(n) \le 2T(n/2 + 1) + A \cdot n, \qquad \text{for all } n \ge 2 \tag{4}$$

Next, we do the change-of-function trick. We define a new function $S(n) := T(n+2)$. We establish the following property.

Claim 1. $S(1) = O(1)$, and for all $n > 1$, $S(n) \le 2S(n/2) + O(n)$

Proof. Note that $S(1) = T(3) = O(1)$.

To see the recurrence on S, observe

$$\begin{aligned} S(n) = T(n+2) &\le 2T\left(\frac{n+2}{2} + 1\right) + A(n+2) \\ &= 2T\left(\frac{n}{2} + 2\right) + (A \cdot n + 2A) \\ &= 2S(n/2) + O(n) \end{aligned}$$

□

Thus, we have moved from a recurrence (3) with ceilings and floors to a recurrence on a different function without them. And this latter one, we did solve in class. We know that

$$S(n) = O(n \log n)$$

Noting that $T(n) = S(n-2)$ and using that $(n-2)\cdot\log(n-2) = O(n \log n)$, we get $T(n) = O(n \log n)$ as well. Which is what we claimed. ✍

Exercise:

- *Complete the proof of the Master Theorem with ceilings using the transformation* $S(n) = T(n+b)$.
- *Solve the recurrence* $T(n) \leq T(\lfloor n/3 \rfloor) + T(\lceil 2n/3 \rceil) + O(n)$.
- *Solve the recurrence* $T(n) \leq \sqrt{n} \cdot T(\lceil \sqrt{n} \rceil) + O(n)$. ☕☕
  Hint: Think of the 'transformation' : $S(n) = (1 + \frac{4}{n}) \cdot T(n+4)$