CSE331: Introduction to Algorithms
Notes on Lecture 7: Solving Recurrences I

Antoine Vigneron

September 18, 2020

1 Comments on the slide "Avoiding Pitfalls"

We know that the proof is wrong, because it would prove that the function given by

$$T(n) = 2T(\lfloor n/2 \rfloor) + n$$

is $O(n)$, but we know that it is $\Theta(n \log n)$. (This function is similar to the running time of Merge Sort.) It shows that you must be careful when making a proof by induction: Always use an exact form with explicit constants, do not use the $O(\cdot)$ notation. This notation is good to state the final result, but not for writing the proof.

The proof breaks down here because you accumulate errors at each level of recursion, and in the end the error becomes large. To see why, we can unfold the recursion and see what happens. To simplify the presentation, suppose that m and n are powers of 2. So the approach on the slide is based on the fact that, for any constant $c > 0$,

$$T(m) \leqslant cm \quad \text{implies} \quad \begin{aligned} T(2m) &\leqslant 2cm + 2m \\ &\leqslant (c+1)2m. \end{aligned} \tag{1}$$

Let $T(1) = c_1$. Then the relation above shows that $T(2) \leqslant 2(c_1 + 1)$. It follows that $T(4) \leqslant 4(c_1 + 2)$, $T(8) \leqslant 8(c_1 + 3)$ ... Finally we get $T(n) \leqslant n(c_1 + \log n)$ and thus $T(n) = O(n \log n)$. It shows that the error has accumulated over the levels of recursion, and in the end we have a $\log n$ factor instead of a constant factor. A correct proof based on (1) yields the correct upper bound $O(n \log n)$.