Algorithms – CMSC 37200

The Method of Reverse Inequalities: Evaluation of Recurrent Inequalities

László Babai

Updated 01-21-2015

In this handout, we discuss a typical situation in the analysis of algorithms: the number of steps required by the algorithm satisfies some recurrent inequality; from this we want to infer an upper bound on the order of magnitude of the number of steps, and we seek the best upper bound (in term of rate of growth) that can be inferred from the given recurrent inequality.

We explain how to do this using the *method of reverse inequalities.* We illustrate the method on two recurrences that occur in the analysis of "Divide-and-Conquer" algorithms.

Our first recurrence arises from the *Karatsuba algorithm* to multiply integers and to multiply polynomials. Both algoriothms lead to the same recurrent inequality.

For simplicity we assume $n = 2^k$. Let $f(n)$ be the number of real number operations (multiplication, addition, subtraction) and bookkeeping operations required by the Karatsuba algorithm to multiply two polynomials of degree $\leq n - 1$. The inequality then is:

$$f(n) \leq 3f(n/2) + O(n). \tag{1}$$

Let us first ignore the $O(n)$ term. Then we have $f(2^k) \leq 3f(2^{k-1})$, from which it follows by easy induction on k that $f(2^k) \leq 3^k f(1)$. Now $3^k = (2^k)^\alpha$ wher $\alpha = \log 3 \approx 1.585$, so we have $f(n) \leq f(1)n^\alpha = O(n^\alpha)$ (because $f(1)$ is a constant).

Let us now consider the full inequality (1). Somewhat surprisingly, it turns out that we shall still have $f(n) = O(n^\alpha)$.

Theorem 1. *If the function $f(n) \geq 0$ satisfies inequality* (1) *then $f(n) = O(n^\alpha)$ where $\alpha = \log 3 \approx 1.585$.*

The big-Oh notation is ill-suited for evaluation in a recurrence, so we make the $O(n)$ term explicit by replacing it by Cn where C is an unspecified constant. So we have

$$f(n) \leq 3f(n/2) + Cn. \tag{2}$$

Our strategy is to guess a function $g(n)$ that satisfies the inequalities

$$g(n) \geq 3g(n/2) + Cn \qquad \text{and} \qquad g(1) \geq f(1). \tag{3}$$

Note that the inequality $g(n) \geq 3g(n/2) + Cn$ goes in the opposite direction than inequality (2) (hence the name of the method) and this is crucial for our inductive argument.

Theorem 2. *Suppose the function $f(n)$ satisfies Eq. (2) and the function $g(n)$ satisfies Eq. (3). Then, for all all values n that are powers of 2, we have $f(n) \leq g(n)$.*

Proof. Let $n = 2^k$. We need to show $f(2^k) \leq g(2^k)$ for all k. We proceed by induction on k.
Base case. For $k = 0$ we need $f(1) \leq g(1)$ which is true by the second inequality in Eq. (3).
Inductive step. Let now $k > 0$ and assume the inequality $f(2^\ell) \leq g(2^\ell)$ holds for all $\ell < k$ (Inductive Hypothesis). We have

$$f(2^k) \leq 3f(2^{k-1}) + C2^k \leq 3g(2^{k-1}) + C2^k \leq g(2^k). \tag{4}$$

Here the first inequality is Eq. (2), the second is true by the Inductive Hypothesis, and the third by Eq. (3). □

Our next job is to guess the function $g(n)$. Our target is $g(n) = O(n^\alpha)$ where $\alpha = \log 3$, so let us try to find $g(n)$ in the form of An^α for some constant A that we may determine later.

We need $g(n) \geq 3g(n/2) + Cn$. In fact if we choose $g(n) = An^\alpha$ then $g(n) = 3g(n/2)$ (verify!) so the desired inequality never holds. But the nature of the failure suggests that adjusting our guess at $g(n)$ with a linear term may succeed.

Let us therefore try to find $g(n)$ in the form $g(n) = An^\alpha + Bn$ for some constants A and B.

For our choice to be good, we need to be able to find values of the constants A and B such that Eq. (3) holds. In other words, we need

$$An^\alpha + Bn \geq 3\left(A(n/2)^\alpha + Bn/2\right) + Cn \tag{5}$$

for all n and we need

$$A + B \geq f(1). \tag{6}$$

to cover both parts of Eq. (3).

We have two degrees of freedom, being free to choose both A and B.

Observing that $n^\alpha = 3(n/2)^\alpha$ (by the definiotn of $\alpha = \log 3$), inequality (5) reduces to

$$Bn \geq 3Bn/2 + Cn,$$

i. e.,

$$B \leq -2C.$$

It may be surprising at first that we are forced to give B a negative value. Let us choose $B := -2C$ (the "least negative" value permitted); then inequality (5) is satisfied.

Note that this holds *regardless of the value of* A. Next we invoke the other degree of freedom we have: we now set the value of A sufficiently large to satisfy the initial value condition (6): $A + B \geq f(1)$, i. e., $A \geq f(1) - B = f(1) + 2C$. The smallest value of A that satifies this is $A = f(1) + 2C$. So we choose

$$A = f(1) + 2C \qquad \text{and} \qquad B = -2C. \tag{7}$$

With this choice of the constants A and B, both conditions in Eq. (3) hold and therefore $f(n) \leq g(n) = An^\alpha - 2Cn \leq An^\alpha = O(n^\alpha)$. This concludes the proof of Theorem 1. □

How do we know that $f(n) = O(n^\alpha)$ is the best possible upper bound on $f(n)$ inferable from the recurrence (1)? The answer is simple: the function $f(n) = n^\alpha$ satisfies (1).

Let us now consider the recurrence

$$t(n) \leq 2t(n/2) + n \tag{8}$$

with the initial value $t(1) = 0$. This recurrence arises in the study of MERGE-SORT; $t(n)$ denotes the number of comparisons made when we sort a list of n data.

Theorem 3. *If the function* $t(n) \geq 0$ *satisfies inequality* (8) *then* $f(n) \leq n \log n$ *where* log *is to the base 2.*

In order to apply the method of reverse inequalities to evaluating this recurrence, we need to guess a function $g(n)$ such that

$$g(n) \geq 2g(n/2) + n \qquad \text{and} \qquad g(1) \geq 0. \tag{9}$$

Exercise 4. Suppose the function $t(n)$ satisfies Eq. (8) and the function $g(n)$ satisfies Eq. (9). Then, for all all values n that are powers of 2, we have $t(n) \leq g(n)$.

Now $g(n)$ is easy to guess: $g(n) = n \log n$ satisfies Eq. (9) (verify!). This proves Theorem 3. $\square$

Exercise 5. Suppose the function $t(n) \geq 0$ satisfies the condition

$$t(n) \leq 2t(n/2) + O(n). \tag{10}$$

Prove: $t(n) = O(n \log n)$.

Exercise 6. Assume $f(n) \geq 0$ satifies the inequality

$$f(n) \leq 3f(\lfloor n/2 \rfloor) + O(n). \tag{11}$$

Prove: $f(n) = O(n^\alpha)$ where $\alpha = \log 3$. Prove the required upper bound for all values of n, not only for powers of 2.

Exercise 7. Assume $f(n) \geq 0$ satifies the inequality

$$f(n) \leq 3f(\lceil n/2 \rceil) + O(n). \tag{12}$$

Prove: $f(n) = O(n^\alpha)$ where $\alpha = \log 3$. Prove the required upper bound for all values of n, not only for powers of 2.

Exercise 8. Assume $T(n) \leq T(0.2n) + T(0.7n) + O(n)$. Prove: $T(n) = O(n)$. (This recurrence arises in analysing a linear-time algorithm for the median.) (a) Ignore rounding. (b) Don't ignore rounding.