Algorithms – CMSC 272

Evaluation of Recurrent Inequalities:
The Method of Reverse Inequalities.
The Fibonacci numbers

László Babai

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. We illustrate the method on a specific recurrent inequality which will occur in class in the analysis of a "branch-and-bound" algorithm.

Suppose we have an algorithm which takes at most $T(n)$ steps on all inputs of size n. Suppose in addition that $T(n)$ is known to satisfy the following recurrent inequality:

$$T(n) \le T(n-1) + T(n-2) \qquad (n \ge 2) \tag{1}$$

How can we use this information to give a good upper bound on $T(n)$?

Theorem 1. Suppose the function $g(n) > 0$ has the following property: there exists a threshold value n_0 such that for all $n \ge n_0$ the inequality

$$g(n) \ge g(n-1) + g(n-2) \tag{2}$$

holds. Then $T(n) = O(g(n))$.

Warning. Note that inequality (2) is the same as inequality (1) except that it goes in the **opposite direction!**

Proof: First, let us choose a positive constant C such that

(a) $Cg(n_0) \ge T(n_0)$, and

(b) $Cg(n_0+1) \ge T(n_0+1)$.

(The choice $C := \max\{T(n_0)/g(n_0),\ T(n_0+1)/g(n_0+1)\}$ will do.) Now we claim that for all $n \ge n_0$, the inequality

$$T(n) \le Cg(n) \tag{3}$$

holds. This clearly justifies the $T(n) = O(g(n))$ claim. (Note that on the basis of the data given, we have no information about the magnitude of the constant C implicit in the big-oh notation.)

The proof of inequality (3) is an easy application of *mathematical induction.* By inequalities (a) and (b) we know that inequality (3) holds for $n = n_0$ and $n = n_0 + 1$ (*base cases*).

For the *inductive step*, let now $n \geq n_0+2$ and assume that $T(k) \leq Cg(k)$ holds for all k in the interval $n_0 \leq k \leq n-1$. Under this *inductive hypothesis* we need to prove that $T(n) \leq Cg(n)$.

Indeed,

$$T(n) \leq T(n-1) + T(n-2) \leq Cg(n-1) + Cg(n-2) \leq Cg(n). \qquad (4)$$

(The first inequality is just inequality (1); the second inequality holds by the inductive hypothesis; and the third inequality comes from inequality (2).) This completes the inductive step, and thereby the inductive proof of Theorem 1. Q. E. D. [End of proof, from the Latin *"Quod erat demonstrandum"* ("which was to be demonstrated").]

Questions.

1. Why did we need to verify *two* base cases ($n = n_0, n_0+1$) (rather than just one base case)?
2. Why did $g(n)$ need to satisfy the *reverse* of the recurrent inequality satisfied by $T(n)$? (Where and how did the proof exploit this condition?)
3. Prove that $T(n) \leq 2T(n-1)$ for $n \geq 4$.
4. Infer from item 3 that $T(n) = O(2^n)$.
5. Prove that $g(n) \geq 2g(n-2)$ for $n \geq n_0 + 3$.
6. Infer from item 5 that $g(n) = \Omega(2^{n/2})$.

The next step in evaluating the recurrence (1) is to find some function $g(n)$ satisfying inequality (2). Item 4 above suggests the choice $g(n) := 2^n$. Indeed, $2^n > 2^{n-1} + 2^{n-2}$ (why?). But this may not be the smallest such $g(n)$. The upper bound in item 4 and the lower bound in item 6 together suggest that we should be looking for an exponential function, of the form $g(n) = \alpha^n$ (geometric progression) for some fixed $\alpha > 1$, and try to find the smallest such α in order to obtain the best possible upper bound. (From item 6 we know that any such α must be $\geq \sqrt{2}$.) The condition then is

$$\alpha^n \geq \alpha^{n-1} + \alpha^{n-2}; \qquad (5)$$

or equivalently (dividing by α^{n-2}),

$$\alpha^2 \geq \alpha + 1. \qquad (6)$$

Recall that we wish to minimize α. The smallest value of $\alpha > 1$ that satisfies (6) actually gives equality:

$$\alpha^2 = \alpha + 1. \tag{7}$$

This is a quadratic equation. Of the two solutions, one is negative and therefore has no meaning in our context; the other solution is

$$\alpha = \phi := \frac{1+\sqrt{5}}{2} \approx 1.6180339 \tag{8}$$

(this is the *golden ratio*).

The following summarizes our conclusion:

Theorem 2. If the function $T(n) \geq 0$ satisfies inequality (1) then

$$T(n) = O(\phi^n)$$

where ϕ is the golden ratio (equation (8)).

Comment. Of course $T(n)$ could be much smaller; what we have is an upper bound. However, this is the *best upper bound* that can be inferred from the information given (inequality (1)), as demonstrated by the example of the Fibonacci sequence in the role of $T(n)$.

Remark. The Fibonacci sequence is the sequence defined by the recurrence $F_n = F_{n-1} + F_{n-2}$ with *initial values* $F_0 = 0$ and $F_1 = 1$. (So the first few terms of the sequence are $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89$.) The asymptotic value of F_n is $F_n \sim \phi^n/\sqrt{5}$.

* * * * *

"Divide and Conquer" algorithms are the most frequent sources of recurrent inequalities. We illustrate the method on the recurrence arising from the Karatsuba integer multiplication algorithm.

For simplicity we assume $n = 2^k$. Let $B(n)$ be the number of digit operations (addition, subtraction, multiplication, and copying of digits) required by the Karatsuba algorithm to multiply two n-digit integers. The inequality then is:

$$B(n) \leq 3B(n/2) + O(n). \tag{9}$$

(The $O(n)$ term accounts for the cost of the recursive step: addition/substraction and copying of a few numbers with at most n digits.)

We cannot handle the $O(n)$ term directly because it is not a well-defined quantity: all we know about it is that it is between $-Cn$ and Cn for some positive constant C (for all $n \geq 1$). With this (unknown, but constant) value of C we obtain the inequality

$$B(n) \leq 3B(n/2) + Cn. \tag{10}$$

Now all terms of the inequality are sufficiently well defined to allow us to do basic operations with them.

Theorem 3. If the function $B(n) \geq 0$ satisfies inequality (10) then $B(n) = O(n^\alpha)$ where $\alpha = \log 3 \approx 1.58$.

Again, we need to find a function $g(n)$ which satisfies the reverse of inequality (10) for $n \geq n_0$ and $g(n) > 0$ for *all* $n \geq 1$. Then, as before, we shall know that $B(n) = O(g(n))$.

We try to find $g(n)$ in the form $g(n) = An^\alpha - Dn$ for some constants $A > 0$ and D. The reason for this choice will be apparent from the calculations below; here we only note that we are entitled to look for $g(n)$ in any form we like; the eventual success justifies a good choice.

For our choice to be good, we need to be able to find a value of the constant D such that

$$An^\alpha - Dn \geq 3\left(A(n/2)^\alpha - Dn/2\right) + Cn$$

holds for all n.

Observing that $n^\alpha = 3(n/2)^\alpha$ (this equality motivated the choice of the exponent α), our inequality reduces to

$$-Dn \geq -3Dn/2 + Cn,$$

i. e.,

$$Dn/2 \geq Cn.$$

Let us therefore choose $D := 2C$, and the required inequality is satisfied. Note that this holds *regardless of the value of* A. We now set the value of A sufficiently large such that $g(n) > 0$ for all $n \geq 1$.

Exercise. Prove that such a choice of A is possible.

Therefore $B(n) = O(g(n)) = O(An^\alpha - Dn) = O(n^\alpha)$. This concludes the proof of Theorem 3. Q. E. D.

Original January 12, 1998. Updated January 14, 2020.