Recursive algorithms that run in linear time

Amit Chakrabarti
Computer Science
Dartmouth College

Imagine an algorithm that is given an input of size n. It works by doing a linear amount of work (at most cn units, for some constant c), and calling itself recursively on *one* input of size at most $n/2$. These two steps need not occur in this order; the algorithm could first do some work, then call itself recursively *once*, then do some more work. So long as the total amount of work is at most cn units, we can use the recurrence inequality $T(n) \leq T(\lceil n/2 \rceil) + cn$ for the running time of the algorithm. Letting $U(n)$ be a sequence such that $U(1) = T(1)$ and $U(n) = U(\lceil n/2 \rceil) + cn$ for $n > 1$, we can show by an easy induction that $T(n) \leq U(n)$. The master theorem tells us that $U(n) = \Theta(n)$. Therefore, $T(n) = O(n)$. This proves that our algorithm runs in linear time.

In class, we generalized the above result. We observed that it's not particularly important that our algorithm's recursive call cut the input size in half. It might decrease the input size from n to at most $2n/3$ or $9n/10$ or in fact αn for any positive constant $\alpha < 1$. However, we now need a new proof because the master theorem no longer helps us. We have the new recurrence inequality

$$T(n) \ \leq \ T(\lceil \alpha n \rceil) + cn \, . \tag{1}$$

Theorem 1 *For any positive sequence $T(n)$ satisfying* (1)*, we have $T(n) = O(n)$.*

Proof: We have to show that there exist constants k and n_0 such that for all integers $n \geq n_0$ we have $T(n) \leq kn$. We shall prove this using strong induction. Choose n_0 and k so that they satisfy

$$n_0 \ \geq \ \frac{2}{1-\alpha} \, , \tag{2}$$

$$k \ \geq \ \frac{1}{n_0} \cdot \max_{1 \leq i < n_0} \{T(i)\} + c \, , \tag{3}$$

$$k \ \geq \ \frac{2c}{1-\alpha} \, . \tag{4}$$

The above inequalities were not born out of thin air; instead, they are the result of some calculations on scratch paper. But to keep the proof slick I have hidden this work! You should feel free to do the same when writing your proofs.

Before embarking on the proof by induction, we make two important claims:

CLAIM 1: For $n \geq n_0$, we have $\lceil \alpha n \rceil < n$.

CLAIM 2: For $n \geq n_0$, we have $(1-\alpha)n - 1 \geq \dfrac{(1-\alpha)n}{2}$.

PROOF OF CLAIM 1: Using (2), $(1-\alpha)n \geq (1-\alpha)n_0 \geq 2$. We can rewrite that as $\alpha n \leq n - 2$. Now using this, $\lceil \alpha n \rceil < \alpha n + 1 \leq n - 2 + 1 < n$.

PROOF OF CLAIM 2: Using (2), $\frac{1}{n} \leq \frac{1}{n_0} \leq \frac{1-\alpha}{2}$. Thus,

$$\frac{(1-\alpha)n - 1}{n} \ = \ (1-\alpha) - \frac{1}{n} \ \geq \ (1-\alpha) - \frac{1-\alpha}{2} \ = \ \frac{1-\alpha}{2} \, .$$

Multiplying out by n gives us the inequality we claimed.

Now that we have proved our claims, let us resume the proof of the theorem. For our base case, we use $n = n_0$. By claim 1, the integer $\lceil \alpha n_0 \rceil$ is less than n_0. So, by (3), $k \geq \frac{1}{n_0} T(\lceil \alpha n_0 \rceil) + c$. Thus,

$$kn_0 \ \geq \ T(\lceil \alpha n_0 \rceil) + cn_0 \ \geq \ T(n_0) \, ,$$

where the last inequality follows directly from the given condition (1). This finishes the base case.

For the inductive step, suppose we have shown $T(i) \leq ki$ for all integers i with $n_0 \leq i < n$. We shall show that $T(n) \leq kn$ as well. Claim 1 tell us that $\lceil \alpha n \rceil < n$, so the inductive hypothesis applies and we get

$$T(n) \;\leq\; T(\lceil \alpha n \rceil) + cn \;\leq\; k\lceil \alpha n \rceil + cn \;<\; k(\alpha n + 1) + cn \;=\; kn - k((1-\alpha)n - 1) + cn\,.$$

Now, using Claim 2*, we obtain

$$T(n) \;\leq\; kn - k\frac{(1-\alpha)n}{2} + cn\,.$$

Next, using (4)†, we obtain

$$T(n) \;\leq\; kn - \frac{2c}{1-\alpha}\frac{(1-\alpha)n}{2} + cn \;=\; kn - cn + cn \;=\; kn\,.$$

This completes the inductive step and the proof of the theorem. □

Using the above theorem we see that recursive algorithms that make one recursive call and do a linear amount of additional work will run in linear time provided the input size in the recursive call is *smaller than n by at least a constant factor*. This principle is fundamental to the design of many many linear time algorithms.

*Do you see why we made such an outlandish looking claim?
†Do you now see why we wanted k to satisfy inequality (4)?