Design and Analysis of Algorithms January 21, 2016

A Note on Asymptotics

Lecturer: Daniel A. Spielman

Notation

In this class, we will use asymptotic notation to describe the running times of algorithms. This note, and the first problem on Problem Set 1, are intended to help clear up some common confusions about asymptotics. The most important notion is that of *polynomial time*. An algorithm runs in polynomial time if on inputs of size n it takes time at most $O(n^k)$, where k is some absolute constant. Think of k as being 1, 2, or 3. No matter what, k *cannot* vary with n. Now, for the details...

While the standard in asymptotic notation is to write things like $f(n) = O(g(n))$, I prefer the notation $f(n) \leq O(g(n))$. As explained in Section 2.2 of the book by Kleinberg and Tardos, we will write $f(n) \leq O(g(n))$ if there is a positive constant C and an integer n_0 such that for all $n \geq n_0$,

$$f(n) \leq Cg(n).$$

I make the same adjustment to the big-Ω notation. I write $f(n) \geq \Omega(g(n))$ if there is a positive constant $C > 0$ and an integer n_0 such that for all $n \geq n_0$

$$f(n) \geq Cg(n).$$

Consider an algorithm that has two parts, the first of which takes time $f_1(n)$ and the second of which takes time $f_2(n)$. If $f_1(n) \leq O(g(n))$ and $f_2(n) \leq O(g(n))$, then the entire time of the algorithm, which is $f_1(n) + f_2(n)$, is also at most $O(g(n))$.

Claim 1. *If $f_1(n) \leq O(g(n))$ and $f_2(n) \leq O(g(n))$, then*

$$f_1(n) + f_2(n) \leq O(g(n)).$$

Proof. From the assumptions of the claim, we know that there are positive constants c_1, c_2, n_1, and n_2 so that

$$\begin{aligned} f_1(n) &\leq c_1 g(n), \text{ for all } n \geq n_1, \text{ and} \\ f_2(n) &\leq c_2 g(n), \text{ for all } n \geq n_2. \end{aligned}$$

Let $c_0 = c_1 + c_2$ and let $n_0 = \max(n_1, n_2)$. We then have that

$$f_1(n) + f_2(n) \leq c_0 g(n) \text{ for all } n \geq n_0.$$

□

The following property of O-notation is useful when we are analyzing loops. Consider using it to analyze a loop that is executed $h(n)$ times in which each execution requires time $f(n)$.

Claim 2. *If $f(n) \leq O(g(n))$ and if $h(n) \geq 0$ for $n \geq 1$, then*

$$f(n)h(n) \leq O(g(n)h(n)).$$

This is immediate: if there is a C and an n_0 such that $f(n) \leq Cg(n)$ for all $n \geq n_0$, then for all $n \geq n_0$

$$f(n)h(n) \leq Cg(n)h(n).$$

Inequalities

Kleinberg and Tardos give some useful inequalities concerning the big-O notation. In particular, they tell us

$$\text{For every } b > 1 \text{ and every } x > 0, \log_b n \leq O(n^x) \tag{2.8}$$

and

$$\text{For every } r > 1 \text{ and every } d > 0, n^d \leq O(r^n) \tag{2.9}$$

But, I don't think that they do an adequate job of explaining why. I will. We begin by recalling the fundamental properties of logarithms:

$$\begin{aligned}
\log_b 1 &= 0 && \text{for } b > 0 && (1)\\
\log_b b &= 1 && \text{for } b > 0 && (2)\\
\log_b x &= (\log_2 x)/(\log_2 b) && \text{for } b > 0 \text{ and } x > 0 && (3)\\
\log_b(xy) &= (\log_b x) + (\log_b y) && \text{for positive } b, x, \text{ and } y && (4)\\
\log_b(x^p) &= p \log_b x && \text{for positive } b \text{ and } x, \text{ and all real } p && (5)\\
\log_b x &< \log_b y && \text{for } b > 1 \text{ and } 0 < x < y. && (6)
\end{aligned}$$

We now prove an elementary inequality about the logarithm that we will use to derive all the others we need.

Lemma 3. *For all $x > 0$, $\log_2 x \leq x$.*

Proof. First observe that for $0 < x \leq 1$, $\log_2 x \leq 0 < x$. Similarly, for $1 < x \leq 2$, $\log_2 x \leq 1 < x$. We will now prove that for every non-negative integer k and every x such that $2^k < x \leq 2^{k+1}$, $\log_2 x < x$. We will prove this by induction on k, having already established the base case when $k = 0$. For $k \geq 1$, and $2^k < x \leq 2^{k+1}$, we know that $2^{k-1} < x/2 \leq 2^k$. So, we can apply the inductive hypothesis to $x/2$. This gives

$$\log_2 x = 1 + \log_2(x/2) < 1 + x/2 < x,$$

where the first inequality follows from the inductive hypothesis and the second follows from $x > 2^k \geq 2$. $\square$

On page 41, Kleinberg and Tardos say that for every $n \geq 1$ $\log n \leq n$. One should be careful to specify the base of the logarithm when making such statements, as they are not true for all bases. It seems to me that $\log_b x \leq x$ for all x for all b greater than some number close to 1.445. I'm not quite sure what that number is.

We will now use Lemma 3 to derive strengthening of (2.8) and (2.9). We begin with a seemingly weaker statement.

Lemma 4. *For every $c > 0$ there is an n_0 so that for all $n > n_0$,*

$$\log_2 cn \leq n. \tag{7}$$

Proof. We know from fact (4) and Lemma 3 that

$$\log_2(cn) = \log_2(2c) + \log_2(n/2) \leq \log_2(2c) + n/2.$$

So, if $\log_2(2c) \leq n/2$, then (7) holds. This implies that is suffices to set $n_0 = 2\log_2(2c)$. $\square$

Lemma 5. *For every $b > 1$ and $p > 0$, there is an integer n_0 so that for all $n > n_0$,*

$$log_b n \leq n^p. \tag{8}$$

Proof. We prove the lemma by applying a change of variables. If we set $x = n^p$, so $n = x^{1/p}$, then we need to show that for sufficiently large x

$$\log_b x^{1/p} \leq x.$$

Using facts (3) and (5) we can show

$$\log_b x^{1/p} = (\log_2 x)/(p \log_2 b). \tag{9}$$

So, it suffices to show that for sufficiently large x

$$\log_2 x \leq (p \log_2 b)x.$$

Setting $y = (p \log_2 b)x$, this is equivalent to showing that for y sufficiently large

$$\log_2(y/p \log_2 b) \leq y.$$

Lemma 4 tells us that this holds for all y larger than some constant, so (9) holds for all x larger than some other constant and (8) holds for all n larger than yet another constant. □

We now derive a strengthening of (2.9).

Lemma 6. *For every $r > 1$ and every $d > 0$, there is an integer n_0 so that for all $n > n_0$,*

$$n^d \leq r^n. \tag{10}$$

Proof. Taking logarithms base 2, this is equivalent to saying that for all $n > n_0$,

$$d \log_2 n \leq n \log_2 r,$$

which is equivalent to

$$\log_2 n \leq n(\log_2 r/d).$$

Setting $x = n(\log_2 r)/d$, this becomes equivalent to

$$\log_2 (x(d/\log_2 r)) \leq x.$$

Lemma 4 tells us that this holds for all sufficiently large x, which implies that (10) holds for all sufficiently large n. □