§1. Class Notes

In the last recitation, our Teaching Assistant (Ziyao Wei) covered three topics:

- Harmonic numbers H_n. It is shown that $H_n = \Theta(\lg n)$. This material is in ¶18 of Lecture II.
- Stirling's approximation for $n!$. There are many forms of this approximation, but Ziyao presented the one I found most useful, which is found in ¶19 of Lecture II.
- The Heuristic Method of comparing complexity functions by taking logarithms. Below, we will go over this method systematically.

¶1. The Heuristic of Taking Logs.

Jack:	*My algorithm has time complexity* $O((\lg n)^n)$.
Jill:	*Oh, with a new tweak, mine now runs in* $O(n^{\lg n})$.

Who has the faster algorithm – Jack or Jill? An effective way to compare the growth rates of two complexity functions is to compare the growth rates of their logarithms. To compare the running times of Jack and Jill,

$$(\lg n)^n \quad \text{versus} \quad n^{\lg n}, \tag{1}$$

let us compare their logs:

$$n \lg\lg n \quad \text{versus} \quad \lg^2 n. \tag{2}$$

Perhaps you already see that the former dominates the latter. If not, you could take logs again:

$$\lg n + \lg\lg\lg n \quad \text{versus} \quad 2\lg\lg n. \tag{3}$$

It is now clear that the left-hand side super-dominates the right-hand side, since

$$\lg n \ggcurly \lg\lg n. \tag{4}$$

Working backwards to the original comparison, we conclude that

$$(\lg n)^n \ggcurly n^{\lg n}. \tag{5}$$

Thus Jack's algorithm is slower than Jill's. Is this argument rigorous? Well, the above idea of taking logs amounts to an application of the following "backwards" inference rule:

$$(f \ggcurly g) \Leftarrow (\lg f \ggcurly \lg g). \tag{6}$$

Here, "$A \Leftarrow B$" reads "A holds *provided* B holds". Logically, $A \Leftarrow B$ and $B \Rightarrow A$ are equivalent, but the backwards formulation seems more natural in proofs of (super-)dominance, such as in (5). See Lecture I (Appendix A) for discussion of logical proofs.

Close, but not quite!

Unfortunately, the rule (6) is not sound. Here is a counter example: let $g = 1$ and $f = 2$. Then $1 = \lg f \ggcurly \lg g = 0$, but it is not true that $f \ggcurly g$. What is needed is some additional guarantee that $\lg f$ is growing fast enough. E.g., $\lg f \ggcurly 1$. We now prove this:

LEMMA 1. *Let f, g be complexity functions. If $\lg f$ super-dominates both 1 and $\lg g$, then f super-dominates g. In symbols,*

$$(f \succ\!\!\succ g) \Leftarrow (\lg f \succ\!\!\succ 1) \wedge (\lg f \succ\!\!\succ \lg g).$$

Proof.

$$\begin{array}{rcl} (f \succ\!\!\succ g) & \Leftrightarrow & (\forall C > 0)[Cf \geq g \text{ (ev.)}] \\ & \Leftrightarrow & (\forall C > 0)[\lg C + \lg f \geq \lg g \text{ (ev.)}] \\ & \Leftarrow & (\forall C > 0)[\lg C + \frac{1}{2}\lg f \geq 0 \text{ (ev.)}] \wedge (\frac{1}{2}\lg f \geq \lg g \text{ (ev.)}) \quad \text{(Split } \lg f \text{ in half!)} \\ & \Leftarrow & (\lg f \succ\!\!\succ 1) \wedge (\lg f \succ\!\!\succ \lg g). \end{array}$$

Q.E.D.

Returning to our heuristic argument that Jill's algorithm is better than Jack's, we see that the heuristic rule (6) just needs an additional precondition that "$\lg f \succ\!\!\succ 1$" holds. These preconditions amount to $n \lg\lg n \succ\!\!\succ 1$ and $\lg n \succ\!\!\succ 1$. But we knew these to be true. In general, we need the following fact as the base case:

LEMMA 2. *For all $k \geq 1$,*

$$\lg^{(k)} n \succ\!\!\succ \lg^{(k+1)} n \succ\!\!\succ 1.$$