Lecture Notes, Fundamental Algorithms: Some big Oh bounds

Theorem 1. $n = o(2^n)$.

Lemma 1. *For all* $n \geq 1$, $n \leq 2^n$. *Similarly, for all* $n \geq 4$, $n \leq 2^{n/2}$.

Proof. We show the first claim by induction. For the base case, $n = 1$, clearly $1 < 2 = 2^1$.

For larger n, suppose inductively that the bound holds for $n = k$, where $k \geq 1$. Then $k + 1 \leq 2k \leq 2 \cdot 2^k = 2^{k+1}$.

For the second claim, the base case is at $n = 4$. Here, $4 = 2^2 \leq 2^{4/2}$.

For larger n, suppose inductively that the bound holds for $n = k$, where $k \geq 4$. Then $k + 1 \leq \frac{5}{4}k \leq \sqrt{2} \cdot 2^{k/2} = 2^{(k+1)/2}$. □

Proof. (Of Theorem 1.) Let $c > 0$ be a constant. We now determine a value n_c such that for all $n \geq n_c$, $cn \leq 2^n$.

Let $d = \lceil \log c \rceil$. By Lemma 1, for $n \geq 4$, $n \leq 2^{n/2}$. So $cn \leq 2^d n \leq 2^{d+n/2} \leq 2^n$, if $n/2 \geq d$, i.e. if $n \geq 2d$. So for $n \geq n_c = \max\{4, 2\lceil \log c \rceil\}$, $cn \leq 2^n$. □

Theorem 2. $\log n = o(n)$.

As we will see this result follows from the previous bound.

Proof. We set $m = \log n$. We know that $m = o(2^m)$, meaning that for every constant $c > 0$, there is a value m_c such that for all $m \geq m_c$, $cm \leq 2^m$. In other words, $c \log n \leq 2^{\log n} = n$.

Strictly speaking, this bound only holds for integer values of m, i.e. for values of n that are an integer power of 2. But it is easy to extend the result to all large enough integer values of n. Let $2^k \leq n < 2^{k+1}$, for some integer $k \geq \max\{1, k_{2c}\}$, where the bound $2c \cdot \log 2^k \leq 2^k$ holds for all $k \geq k_{2c}$. The bound $k \geq 1$ will ensure that $\log 2^k \geq 1$.

Then, for $k \geq \max\{1, k_{2c}\}$, $c \log n \leq c \log 2^{k+1} = c(\log 2 + \log 2^k) = c(1 + \log 2^k) \leq 2c \log 2^k \leq 2^k \leq n$. □