Section 3.1 – Asymptotic notation

3.1-1 Let $f(n)$ and $g(n)$ be asymptotically nonnegative functions. Using the basic definition of Θ-notation, prove that $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.

Since $f(n)$ and $g(n)$ are both asymptotically nonnegative,

$$\exists\, n_0 \mid f(n) \geq 0\ g(n) \geq 0\ \forall\, n \geq n_0.$$

From the definition of $\Theta(\cdot)$, we have

$$\exists\, c_1\ c_2\ n_0 \in \mathbb{R}^+ \mid c_1 f(n) + c_1 g(n) \leq \max(f(n), g(n)) \leq c_2 f(n) + c_2 g(n)\ \forall\, n \geq n_0.$$

If $f(n) \geq g(n)$, we have

$$c_1 f(n) + c_1 g(n) \leq f(n) \leq c_2 f(n) + c_2 g(n).$$

The right-hand-side inequality is trivially satisfied with $c_2 = 1$. To find c_1, we notice that,

$$f(n) + g(n) \leq 2f(n),$$

and say,

$$c_1 = \frac{1}{2}.$$

The demonstration is similar for $g(n) > f(n)$, with $c_1 = 1/2$ and $c_2 = 1$.

3.1-2 Show that for any real constants a and b, where $b > 0$, $(n + a)^b = \Theta(n^b)$.

From the definition of $\Theta(\cdot)$, we have

$$(n+a)^b = \Theta(n^b)\ \equiv\ \exists\, c_1\ c_2\ n_0 \in \mathbb{R}^+ \mid c_1 n^b \leq (n+a)^b \leq c_2 n^b\ \forall\, n \geq n_0,$$

and from the binomial theorem, we have

$$(n+a)^b = \binom{b}{0} n^b a^0 + \binom{b}{1} n^{b-1} a^1 + \cdots + \binom{b}{b-1} n^1 a^{b-1} + \binom{b}{b} n^0 a^b.$$

To find c_1, we notice that for n big enough,

$$\binom{b}{i} n^{b-i} a^i - \binom{b}{i+1} n^{b-i+1} a^{i+1} \geq 0 \quad \forall\, i \in 0, 2, \ldots, b,$$

which implies

$$\binom{b}{0} n^b a^0 + \binom{b}{1} n^{b-1} a^1 \leq (n+a)^b,$$

and also for n big enough,

$$\frac{n^b}{2} \leq n^b + \binom{b}{1} n^{b-1} a^1,$$

which implies

$$\frac{n^b}{2} \leq (n+a)^b,$$

and say

$$c_2 = \frac{1}{2}.$$

To find c_2, we notice that for n big enough,

$$n^b = \binom{b}{0} n^b a^0 \geq \binom{b}{i} n^{b-i} a^i \quad \forall\, i \in 1, \ldots, b,$$

which implies

$$(n+a)^b \leq b n^b,$$

and say

$$c_2 = b.$$

3.1-3 Explain why the statement, "The running time of algorithm A is at least $O(n^2)$," is meaningless.

Because the O-notation only bounds from the top, not from the bottom.

3.1-4 Is $2^{n+1} = O(2^n)$? Is $2^{2n} = O(2^n)$?

From the definition of $O(\cdot)$, we have

$$2^{n+1} = O(2^n) \;\equiv\; \exists\, c\; n_0 \in \mathbb{R}^+ \mid 0 \leq 2^{n+1} \leq c \cdot 2^n \;\; \forall\, n \geq n_0.$$

To find c, we notice that,

$$2^{n+1} = 2 \cdot 2^n,$$

and say $c = 2$ and $n_0 = 0$.
From the definition of $O(\cdot)$, we have

$$2^{2n} = O(2^n) \;\equiv\; \exists\, c\; n_0 \in \mathbb{R}^+ \mid 0 \leq 2^{2n} \leq c \cdot 2^n \;\; \forall\, n \geq n_0.$$

To show that $2^{2n} \neq O(2^n)$, we notice that,

$$2^{2n} = 2^n \cdot 2^n,$$

which implies

$$c \geq 2^n,$$

which is not possible, since c is a constant and n is not.

3.1-5 Prove Theorem 3.1.

To prove

$$f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n)),$$

we need to show

$$f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n)) \rightarrow \Theta(n),$$

and

$$f(n) \neq O(g(n)) \text{ or } f(n) \neq \Omega(g(n)) \rightarrow f(n) \neq \Theta(g(n)).$$

From the definition of $O(\cdot)$, we have

$$f(n) = O(g(n)) \rightarrow \exists\, c_1\; n_1 \in \mathbb{R}^+ \mid 0 \leq f(n) \leq c_1 g(n) \;\; \forall\, n \geq n_1,$$

and from the definition of $\Omega(\cdot)$, we have

$$f(n) = \Omega(g(n)) \rightarrow \exists\, c_2\; n_2 \in \mathbb{R}^+ \mid 0 \leq c_2 g(n) \leq f(n) \;\; \forall\, n \geq n_2.$$

Putting the two above together, we show that

$$f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n)) \;\equiv\; \exists\, c_1\; c_2\; n_0 \in \mathbb{R}^+ \mid c_1 g(n) \leq f(n) \leq c_2 g(n) \;\; \forall\, n \geq n_0 \;\equiv\; f(n) = \Theta(g(n)).$$

3.1-6 Prove that the running time of an algorithm is $\Theta(g(n))$ if and only if its worst-case running time is $O(g(n))$ and its best-case running time is $\Omega(g(n))$.

Let $f_b(n)$ and $f_w(n)$ be the best and worst-case running times of algorithm A, respectivelly.
If the running time of A is $\Theta(g(n))$, we have

$$f_b(n) = \Theta(g(n)),$$

and

$$f_w(n) = \Theta(g(n)).$$

From Theorem 3.1,

$$f_b(n) = \Theta(g(n)) \iff f_b(n) = O(g(n)) \text{ and } f_b(n) = \Omega(g(n)),$$

and

$$f_w(n) = \Theta(g(n)) \iff f_w(n) = O(g(n)) \text{ and } f_w(n) = \Omega(g(n)).$$

3.1-7 Prove that $o(g(n)) \cap \omega(g(n))$ is the empty set.

From the definition of $o(\cdot)$, we have

$$o(g(n)) = \{f(n) : \forall\, c_1 > 0\ \exists\, n_1 \in \mathbb{R}^+ \mid 0 \leq f(n) \leq c_1 g(n)\ \ \forall\, n \geq n_1\},$$

and from the definition of $\omega(\cdot)$, we have

$$\omega(g(n)) = \{f(n) : \forall\, c_2 > 0\ \exists\, n_2 \in \mathbb{R}^+ \mid 0 \leq c_2 g(n) \leq f(n)\ \ \forall\, n \geq n_2\}.$$

Thus,

$$o(g(n)) \cap \omega(g(n)) = \{f(n) : \forall\, c_1 > 0\ \forall\, c_2 > 0\ \exists\, n_0 \in \mathbb{R}^+ \mid 0 \leq c_2 g(n) \leq f(n) \leq c_1 g(n)\ \ \forall\ \ n \geq n_2\},$$

which is the empty set, since for very large n $f(n)$ cannot be less than $c_1 g(n)$ and greater than $c_2 g(n)$ for all $c_1, c_2 > 0$.

3.1-8 We can extend our notation to the case of two parameters n and m that can go to infinity independently at different rates. For a given $g(n, m)$, we denote by $O(g(n, m))$ the set of functions

$O(g(n, m)) = \{f(n, m)$: there exist positive constants $c, n_0,$ and m_0 such that $0 \leq f(n, m) \leq cg(n, m)$ for all $n \geq n_0$ and $m \geq m_0\}$.

Give corresponding definitions for $\Omega(g(n, m))$ and $\Theta(g(n, m))$.

We denote by $\Omega(g(n, m))$ the set of functions

$$\Omega(g(n, m)) = \{f(n, m) : \exists\, c\ n_0\ m_0 \in \mathbb{R}^+ \mid 0 \leq cg(n, m)) \leq f(n, m)\ \ \forall\, n \geq n_0\ \ \forall\, m \geq m_0\}.$$

We denote by $\Theta(g(n, m))$ the set of functions

$$\Theta(g(n, m)) = \{f(n, m) : \exists\, c_1\ c_2\ n_0\ m_0 \in \mathbb{R}^+ \mid 0 \leq c_1 g(n, m) \leq f(n, m) \leq c_2 g(n, m)\ \ \forall\, n \geq n_0\ \ \forall\, m \geq m_0\}.$$