# $O$ vs $\Theta$, Solutions

## Problems

1. *Let $f(n) = 5 + 2n + 3n^2$ and $g(n) = n^2$. Confirm that $f = \Theta(g)$ and $g = \Theta(f)$ are both true.*

   We have:

   $$1 \times n^2 \leqslant 5 + 2n + 3n^2 \leqslant 10 \times n^2$$

   for all $n \geqslant 1$ since $1, n \leqslant n^2$ when that is the case. So

   $$1 \times g(n) \leqslant f(n) \leqslant 10 \times g(n)$$

   for all $n \geqslant 1$ and hence $f = \Theta(g)$. But we also have:

   $$(1/10) \times f(n) \leqslant g(n) \leqslant 1 \times f(n),$$

   so $g = \Theta(f)$ as well (see below for a more general argument about this).

2. True or false:

   - $n^2 = O(3^n)$: **True** (see induction notes)
   - $3^n = O(n^2)$: **False** ($3^n$ is eventually much larger, see induction notes)
   - $n^2 = \Theta(3^n)$: **False** (follows from preceding, since $3^n$ is not $O(n^2)$ we can't find a $B > 0$ such that $B \times 3^n \leqslant n^2$ for then we'd also have $3^n \leqslant (1/B) \times n^2$.)
   - $3^n = \Theta(n^2)$: **False** (similar)
   - $3^n + n^2 = O(3^n)$: **True** (both terms on left are $O(3^n)$ and so their sum will be as well)
   - $3^n + n^2 = O(n^2)$: **False** (since the first term on the left grows too quickly)
   - $3^n + n^2 = \Theta(3^n)$: **True** (follows from $n^2 = O(3^n)$ so we get $3^n$ as a lower bound and some multiple of $3^n$ as an upper bound)
   - $3^n + n^2 = \Theta(n^2)$: **False** (left hand side gets too big)
   - $2^n = O(3^n)$: **True** (just from $2^n \leqslant 3^n$)
   - $2^n = \Theta(3^n)$: **False** (any constant multiple of $3^n$ will eventually be bigger than $2^n$ no matter how small the constant)
   - $n = O(n \log n)$: **True** (the part inside the O is larger)

- $n = \Theta(n \log n)$: **False** (we can't get a matching lower bound because $\log n$ grows arbitrarily large, albeit very slowly).

3. *Is it true in general that if $f = \Theta(g)$ then $g = \Theta(f)$? If not, give a counterexample. If so, explain why.*

   This is true. If, $f = \Theta(g)$ then there are constants $0 < B < A$ such that for all sufficiently large $n$:

   $$B \times g(n) \leqslant f(n) \leqslant A \times g(n).$$

   But, rearranging the two sides this gives:

   $$(1/A) \times f(n) \leqslant g(n) \leqslant (1/B) \times f(n).$$

   And $0 < 1/A < 1/B$ are also constants, so $g = \Theta(f)$.

4. *Both selection sort and insertion sort have run times bounded above by (a constant multiple of) $n^2$ and so are $O(n^2)$ algorithms. Do both have $\Theta(n^2)$ run times? Does either one?*

   Selection sort is $\Theta(n^2)$ since at each pass the maximum (or minimum depending on implementation) of the remaining items is found – the work done does not change. Insertion sort is not $\Theta(n^2)$ since, properly implemented, on sorted data it has run time proportional to $n$ (all it does is check that each item is at least as large as the preceding one).

5. *(A little bit subtle). Suppose that $f_1 = \Theta(g)$ and $f_2 = O(g)$. Can we say that $f_1 + f_2 = O(g)$? Can we say that $f_1 + f_2 = \Theta(g)$?*

   We're assuming throughout (as we always do) that everything is non-negative. What we are given is that there are constants $0 < B < A$ and $0 < C$ such that for all sufficiently large $n$:

   $$B \times g(n) \leqslant f_1(n) \leqslant A \times g(n),$$
   $$f_2(n) \leqslant C \times g(n).$$

   We can certainly say that $f_1 + f_2 = O(g)$ since we have,

   $$f_1(n) + f_2(n) \leqslant (A + C) \times g(n).$$

   But, we can *also* say that $f_1 + f_2 = \Theta(g)$ since

   $$B \times g(n) \leqslant f_1(n) \leqslant f_1(n) + f_2(n) \leqslant (A + C) \times g(n).$$