3.2. THE GROWTH OF FUNCTIONS

(27.) a) $n \log(n^2+1) + n^2 \log n$

$\log(n^2+1)$ and $\log n$ are in the same big O class since $\log n^2 = 2\log n$, therefore simplest good answer would be just $O(n^2 \log n)$

b) $(n \log n + 1)^2 + (\log n + 1)(n^2+1)$ – first term is $O(n^2 (\log n)^2)$, and the second term is $O(n^2 \log n)$. Therefore, answer is $O(n^2 (\log n)^2)$.

c) $n^{2^n} + n^{n^2} \Rightarrow O(n^{2^n})$, since $2^n > n^2$

(47.) $f(n)$ is not $O(g(n))$ and $g(n)$ is not a $O(f(n))$.

The key is that a function should be a big O of another function, and the inequality should hold for all the possible inputs.

Suppose $f(x) = x^2$ for even x and x for odd x.
Similarly $g(x) = x^2$ for odd x and x for even x.
then clearly neither $|f(x)| \leq C|g(x)|$ nor $|g(x)| \leq C|f(x)|$ will hold for all x, since for even x, the first function is much bigger, than the second one, while for odd x the second one is much bigger than the first.

3.3 Complexity of Algorithms

(27.) Used algorithm is going through the list once, doing some small number of operations at each step. In particular, between any two successive executions of the 'i := i+1' there are at most not more than six operations, therefore at most 6n steps are done, which means that the time complexity for all cases stays O(n).

(43.) Number of multiplications is given by the following expression is: $[1+2+\ldots+n]+[1+2\ldots+(n-1)]+\ldots+[1]$.

To simplify, the sum of the first k positive integers is $k(k+1)/2$, as well as the sum $1\cdot2+2\cdot3+3\cdot4\ldots+k(k+1) = k(k+1)(k+1)/3 \Rightarrow n(n+1)(n+2)/6$ is the total number of multiplications.