8 Algorithms (fms27)

Reminder:

$$f(n) \in \Theta(g(n))$$

$$\iff$$

$$\exists\ n_0, c_1, c_2 \in \mathbb{R}_{>0} \text{ such that } \forall n > n_0 : 0 < c_1 g(n) \leq f(n) \leq c_2 g(n).$$

(*a*) For the bubblesort algorithm, state its best-case Θ complexity and describe, for any given input of arbitrary size n, one permutation that would trigger this best-case behaviour. Then give the corresponding permutation of `[0, 1, 2, 3, 4, 5, 6, 7, 7, 7]`. Then repeat the above for the worst case: state the Θ complexity, saying when it is achieved, and exhibit as a concrete example a permutation of the 10 numbers given. [4 marks]

(*b*) Repeat Part (*a*) for the heapsort algorithm. [4 marks]

(*c*) Repeat Part (*a*) for the basic quicksort algorithm, where the pivot is simply chosen as the last element in the range. [4 marks]

(*d*) Write clear and efficient pseudocode to eliminate all duplicates from a linked list of n elements, without changing the order of the remaining elements. Then derive and justify its Θ complexity. [4 marks]

(*e*) [*Hint:* The Ω notation, like the Θ notation, is typically used to describe the asymptotic behaviour of a worst-case cost function $f(n)$. When we say, by extension, that a certain task has a complexity bound of $\Omega(g(n))$, we mean that this bound applies to the worst-case cost function of every possible algorithm that could solve that task.]

Give a formula for $f(n) \in \Omega(g(n))$, in a format similar to that of the Reminder above, and briefly explain it. Then derive, with a clear justification, a tight Ω complexity bound for the task of eliminating all duplicates from a linked list of n elements. [4 marks]