8 Algorithms (FMS)

(a) Transform the following recurrence

\[ f(x) = f(\sqrt{x}) + c \]

into a closed-form expression for the function \( f \) (that is, an expression that does not contain \( f \)). Having done that, give the asymptotic complexity of \( f \) using big-O notation. [4 marks]

(b) (i) Explain the programming technique known as *memoization*, detailing the cases to which it applies. [4 marks]

(ii) In a few lines of pseudocode, write a memoized recursive function to compute the \( i \)th Fibonacci number \( F(i) \), with \( i \in \mathbb{N} \setminus \{0\} \). Recall that \( F(1) = 1, F(2) = 1, \ldots \) [4 marks]

(c) Computing a recursive function \( f \) on arrays, when called on an array of size \( n \), results in \( 2^n \) recursive calls to \( f \). After memoizing \( f \), on an array of a specific size \( n_0 \) we observe that about 90\% of the calls to \( f \) return a memoized result rather than invoking \( f \) recursively. Is either of the following statements correct? Justify your answers.

(i) “The number of recursive calls goes down by a factor of ten; so it will take 1/10 of the time it used to, that is, it will run 10 times faster.” [2 marks]

(ii) “Previously, the function did \( 2^n \) recursive calls. Now it does \( 0.9 \cdot c_1 + 0.1 \cdot 2^n \) recursive calls. That is still \( O(2^n) \), so the asymptotic complexity of the function is still the same (even after memoization).” [2 marks]

(d) Some implementations of the Quicksort algorithm select the pivot at random, rather than taking the last entry in the input array.

(i) Discuss the advantages and disadvantages of such a choice. [1 mark]

(ii) How would you construct an input to trigger quadratic running time for this randomised Quicksort, without having access to the state of the random number generator? [3 marks]