Written Assignment 1

Vancouver Summer Program – Algorithms – UBC

- You should work with a partner.
- You must typeset your solutions.
- **Notation.** $\mathbb{N} = \{1,2,\dots\} \subset \{0,1,2,\dots\} = \mathbb{Z}_+$, and $\mathbb{R}_+ = [0,\infty)$.

---

1. (Enter Fibonacci) The Fibonacci sequence is defined as follows: $F_0 = F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for all integers $n \geq 2$.

   (a) You are to derive an efficient algorithm to compute the nth Fibonacci number. Observe that

   $$\begin{aligned} F_n &= F_{n-1} + F_{n-2} \\ F_{n-1} &= F_{n-1} + 0 \cdot F_{n-2}. \end{aligned}$$

   If we write this linear system in terms of matrices, we have

   $$\begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_{n-1} \\ F_{n-2} \end{bmatrix}$$

   Using this linear relation, derive an algorithm to compute F_n. Your algorithm should run in time $O(\log n)$.
   Hint: Use repeated squaring to compute matrix powers.

   (b) Now suppose that writing every bit of the output to memory counts as an operation that we wish to account for in our running-time analysis (in the previous part, we disregard the time required to write the output to memory). Can you compute F_n in time that is bounded by a polynomial in the size of the input? Justify your answer.

   (c) (**Bonus**) Find a if a and b are integers such that $x^2 - x - 1$ is a factor of $ax^{17} + bx^{16} + 1$. **Hint**: The answer is F_n for some $n \geq 1$. It is enough to show this and find n explicitly; you do not need to compute F_n.

2. (Time Complexity)

   (a) Algorithms A and B spend exactly $T_A(n) = 0.1n^2 \log_{10}(n)$ and $T_B(n) = 2.5n^2$ microseconds, respectively, for a problem of size n. Choose the algorithm, which is better in the Big-Oh sense, and find out a problem size n_0 such that for any larger size $n > n_0$ the chosen algorithm outperforms the other. If your problems are of the size $n \leq 10^9$, which algorithm will you recommend to use?

   (b) Let $f(n) = (\log n)^{\log n}$ and $g(n) = 2^{(\log_2 n)^2}$. Determine whether $f \in O(g)$, $f \in \Omega(g)$, or both (in which case $f \in \Theta(g)$).

   (c) Show that for any $f, g : \mathbb{Z}_+ \to \mathbb{R}_+$, $O(f+g) = O\big(\max\{f,g\}\big)$. Recall that $O(\cdot)$ is a set (see notes #1), and therefore one has to show both $O(f+g) \subset O\big(\max\{f,g\}\big)$ and $O\big(\max\{f,g\}\big) \subset O(f+g)$.