COMP 3804 — Assignment 1

Due: Thursday February 1, 23:59.

Assignment Policy:

- Your assignment must be submitted as one single PDF file through Brightspace.

> Use the following format to name your file:
>
> `LastName_StudentId_a1.pdf`

- **Late assignments will not be accepted. I will not reply to emails of the type "my internet connection broke down at 23:57" or "my scanner stopped working at 23:58", or "my dog ate my laptop charger".**
- You are encouraged to collaborate on assignments, but at the level of discussion only. When writing your solutions, you must do so in your own words.
- Past experience has shown conclusively that those who do not put adequate effort into the assignments do not learn the material and have a probability near 1 of doing poorly on the exams.
- When writing your solutions, you must follow the guidelines below.
  - You must justify your answers.
  - The answers should be concise, clear and neat.
  - When presenting proofs, every step should be justified.

Some useful facts:

1. for any real number $x > 0$, $x = 2^{\log x}$.
2. For any real number $x \neq 1$ and any integer $k \geq 1$,

$$1 + x + x^2 + \cdots + x^{k-1} = \frac{x^k - 1}{x - 1}.$$

3. For any real number $0 < \alpha < 1$,

$$\sum_{i=0}^{\infty} \alpha^i = \frac{1}{1 - \alpha}.$$

Question 1: Write your name and student number.

Question 2: Consider the following recurrence, where n is a power of 8:

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ n + 64 \cdot T(n/8) & \text{if } n \geq 8. \end{cases}$$

- Solve this recurrence using the *unfolding method*. Give the final answer using Big-O notation.
- Solve this recurrence using the *Master Theorem*.

Question 3: Professor Justin Bieber has observed that only five multiplications are needed to compute the square of a 2×2 matrix:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a^2 + bc & b(a+d) \\ c(a+d) & bc + d^2 \end{pmatrix}$$

Professor Bieber remembers Strassen's algorithm from COMP 3804. Based on this, he claims that, for any given $n \times n$ matrix A, the matrix $A^2 = AA$ can be computed by a divide-and-conquer algorithm that makes five recursive calls, resulting in a running time of $O(n^{\log 5})$.

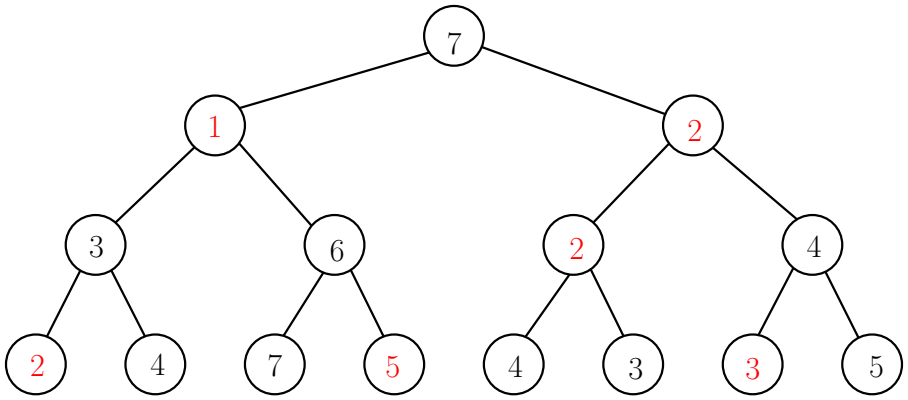
Is Professor Bieber's claim correct? As always, justify your answer.

Question 4: You are given an array $A[1 \ldots n]$ of n numbers. Describe a divide-and-conquer algorithm that returns, in $O(n \log n)$ time, the value

$$\max\{A[j] - A[i] : 1 \leq i < j \leq n\}.$$

You may describe your algorithm in plain English or in pseudocode. Justify the correctness of your algorithm and explain why the running time is $O(n \log n)$. You may use any result that was proven in class.

Question 5: You are given a complete binary tree (i.e., all leaves are at the same level and the bottom level is full). The levels are numbered $0, 1, 2, \ldots, d-1$ and the number n of nodes satisfies $n = 2^d - 1$. Each node u in this tree stores an integer $value(u)$.



A node u is called a *local minimum* if $value(u)$ is less than or equal to the values in its neighboring nodes. In the example above, all local minima are colored red.

Describe a recursive algorithm that returns, in $O(\log n)$ time, a local minimum in this tree.

You may describe your algorithm in plain English or in pseudocode. Justify the correctness of your algorithm and explain why the running time is $O(\log n)$. You may use any result that was proven in class.

Question 6: You are given a *sorted* array $A[1 \dots n]$ of n *distinct integers*. Describe a recursive algorithm that decides, in $O(\log n)$ time, if there is an index i such that $A[i] = i$. If such an index exists, the algorithm returns one such index. Otherwise, the algorithm returns "No".

You may describe your algorithm in plain English or in pseudocode. Justify the correctness of your algorithm and explain why the running time is $O(\log n)$. You may use any result that was proven in class.

Question 7: Let $\mathcal{A}$ be the fastest algorithm that takes as input two n-bit integers x and y, and returns the product xy. Let $T(n)$ be the running time (in terms of bit-operations) of this algorithm.

Let $\mathcal{A}'$ be the fastest algorithm that takes as input one n-bit integer x, and returns the square x^2. Let $T'(n)$ be the running time (in terms of bit-operations) of this algorithm.

- Explain, in at most two sentences, why $T'(n) = O(T(n))$.
- Professor Taylor Swift claims that $T'(n) = o(T(n))$, i.e., $T'(n)$ is asymptotically smaller than $T(n)$. Her reasoning is that the input to algorithm $\mathcal{A}'$ is just one integer, whereas the input to algorithm $\mathcal{A}$ consists of two integers.

  Is Professor Swift's claim correct? As always, justify your answer.

  Hint: $(x + y)^2$.