COMP 3804 — Winter 2025 — Assignment 1

Due: Thursday January 30, 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. $1 + 2 + 3 + \cdots + n = n(n+1)/2$.

2. for any real number $x > 0$, $x = 2^{\log x}$.

3. For any real number $x \neq 1$ and any integer $k \geq 1$,

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

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

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

Master Theorem:

1. Let $a \geq 1$, $b > 1$, $d \geq 0$, and

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ a \cdot T(n/b) + \Theta(n^d) & \text{if } n \geq 2. \end{cases}$$

2. If $d > \log_b a$, then $T(n) = \Theta(n^d)$.

3. If $d = \log_b a$, then $T(n) = \Theta(n^d \log n)$.

4. If $d < \log_b a$, then $T(n) = \Theta(n^{\log_b a})$.

Question 1: Write your name and student number.

Question 2: After having attended the first lecture of COMP 3804, Justin Bieber is intrigued by the recursive algorithm $\text{FIB}(n)$ that computes the n-th Fibonacci number in exponential time. He is convinced that a simple modification should run much faster. Here is Justin's algorithm.

> **Algorithm** $\text{FIBBIEBER}(n)$:
> **comment:** $n \geq 0$ is an integer
> initialize an array $f(0 \dots n)$;
> **for** $i = 0, 1, \dots n$ **do** $f(i) = -1$
> **endfor**;
> $\text{BIEBER}(n)$;
> return $f(n)$
>
> **Algorithm** $\text{BIEBER}(m)$:
> **comment:** $0 \leq m \leq n$, this algorithm has access to the array $f(0 \dots n)$
> **if** $m = 0$
> **then** $f(0) = 0$
> **endif**;
> **if** $m = 1$
> **then** $f(0) = 0$; $f(1) = 1$
> **endif**;
> **if** $m \geq 2$
> **then if** $f(m-2) = -1$
>   **then** $\text{BIEBER}(m-2)$
>   **endif**;
> $x = f(m-2)$;
>   **if** $f(m-1) = -1$
>   **then** $\text{BIEBER}(m-1)$
>   **endif**;
> $y = f(m-1)$;
> $f(m) = x + y$
> **endif**

- Is algorithm FIBBIEBER correct? That is, is it true that for every integer $n \geq 0$, the output of algorithm $\text{FIBBIEBER}(n)$ is the n-th Fibonacci number? As always, justify your answer.
- What is the running time of algorithm $\text{FIBBIEBER}(n)$? You may assume that two integers can be added in constant time. As always, justify your answer.

Question 3: Taylor Swift is not impressed by Justin's algorithm in the previous question. Taylor is convinced that there is a much simpler algorithm. Here is Taylor's algorithm:

Algorithm FIBSWIFT(n):
comment: $n \geq 0$ is an integer
initialize an array $f(0 \dots n)$;
for $i = 0, 1, \dots n$ **do** $f(i) = -1$
endfor;
SWIFT(n);
return $f(n)$

Algorithm SWIFT(m):
comment: $0 \leq m \leq n$, this algorithm has access to the array $f(0 \dots n)$
if $m = 0$
then $f(0) = 0$
endif;
if $m = 1$
then $f(0) = 0$; $f(1) = 1$
endif;
if $m \geq 2$
then SWIFT($m - 1$);
        $f(m) = f(m-1) + f(m-2)$;
endif

- Is algorithm FIBSWIFT correct? That is, is it true that for every integer $n \geq 0$, the output of algorithm FIBSWIFT(n) is the n-th Fibonacci number? As always, justify your answer.

- What is the running time of algorithm FIBSWIFT(n)? You may assume that two integers can be added in constant time. As always, justify your answer.

Question 4: Consider the following recurrence, where n is a power of 7:

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ n^3 + 12 \cdot T(n/7) & \text{if } n \geq 7. \end{cases}$$

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

- Solve this recurrence using the *Master Theorem*.

Question 5: You are given an array $A(1 \dots n)$ of n distinct numbers. This array has the following property: There is an index i with $1 \leq i \leq n$, such that

1. the subarray $A(1 \dots i)$ is sorted in increasing order, and

2. the subarray $A(i \dots n)$ is sorted in decreasing order.

Describe a recursive algorithm that returns, in $O(\log n)$ time, the largest number in the array A. (At the start of the algorithm, you do not know the above index i.)

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 sequence $S = (a_1, a_2, \ldots, a_n)$ of n distinct numbers. A pair (a_i, a_j) is called *Out-of-Order*, if $i < j$ and $a_i > a_j$; in words, a_i is to the left of a_j and a_i is larger than a_j.

If the sequence S is sorted then the number of Out-of-Order pairs is zero. On the other hand, if S is sorted in decreasing order, then there are $\binom{n}{2}$ Out-of-Order pairs.

Describe a comparison-based divide-and-conquer algorithm that returns, in $O(n \log n)$ time, the number of Out-of-Order pairs in the sequence S.

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.

Hint: Think of Merge-Sort.

Question 7: You are given an array $A(1 \ldots n)$ of n distinct numbers, and an integer k with $1 \leq k \leq n$.

Describe a comparison-based algorithm that returns, in $O(n)$ time, k numbers in A that are closest to the number 2025. (The k output numbers do not have to be in sorted order. The output may not be unique.)

For example, if $k = 3$ and

$$A = (2027, 9, 1, 2021, 1948, -17, 2024, 2029),$$

then both $(2027, 2024, 2021)$ and $(2027, 2024, 2029)$ are valid outputs.

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

Question 8: Consider the following recurrence, where $n \geq 1$ is an integer:

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 1 + T(\lfloor \sqrt{n} \rfloor) & \text{if } n \geq 2. \end{cases}$$

Solve this recurrence, i.e., use Big-O notation to express $T(n)$ as a function of n.