COMP 3804 — Assignment 1

Due: Thursday February 2, 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 6:

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ n^2 + 11 \cdot T(n/6) & \text{if } n \geq 6. \end{cases}$$

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

Question 3: Consider the following recurrence:

$$T(n) = n + T(n/5) + T(7n/10).$$

In class, we have seen that $T(n) = O(n)$. In this question, you will prove this using the *recursion tree method*.

Recall from class: The root represents the recursion tree on an input of size n. Consider a node u in the recursion tree that represents a recursive call on an input of size m. Then we write the value m at this node u, we give u a left subtree which is a recursion tree for an input of size $m/5$, and we give u a right subtree which is a recursion tree for an input of size $7m/10$. In this way, $T(n)$ is the sum of the values stored at all nodes in the entire recursion tree.

Below, we assume that the *levels* in the recursion tree are numbered $0, 1, 2, \ldots,$, where the root is at level 0. For each $i \geq 0$, let S_i be the sum of the values of all nodes at level i.

- Determine S_0.
- Determine S_1.
- Determine S_2.
- Use induction to prove the following claim: For every $i \geq 0$,

  $$S_i \leq (9/10)^i \cdot n.$$

  Hint: Consider level i, let $k = 2^i$, and let the values stored at the nodes at level i be $m_1, m_2, \ldots, m_k$. What are the values stored at the nodes at level $i + 1$?
- Complete the proof by showing that $T(n) = O(n)$.

Question 4: Zoltan is not only your friendly TA, he is also the owner of the popular budget airline ZoltanJet that offers flights in Canada. As you all know, there are n airports in Canada. We denote these airports, in order from west to east, by $A_1, A_2, \ldots, A_n$.

William, who is the CEO of ZoltanJet, has designed a *flight plan* which is a list of ordered pairs (A_i, A_j) of airports such that there is a direct flight from A_i to A_j. This flight plan has the following two properties:

- (P.1) Every flight is going eastwards[1]. In other words, if (A_i, A_j) is in the flight plan, then $i < j$.
- (P.2) For any two indices i and j with $1 \leq i < j \leq n$, it is possible to fly from A_i to A_j in at most two *hops*. In other words, either (A_i, A_j) is in the flight plan, or there is an index k such that both (A_i, A_k) and (A_k, A_j) are in the flight plan. Note that, because of (P.1), $i < k < j$.

Observe that ZoltanJet can guarantee (P.1) and (P.2) by offering direct flights between all $\binom{n}{2} = \Theta(n^2)$ pairs (A_i, A_j) of airports, where $1 \leq i < j \leq n$.

- Prove that ZoltanJet can guarantee (P.1) and (P.2) using a flight plan having only $O(n \log n)$ pairs of airports. You may assume that n is a power of two.

  Hint: Since this is the divide-and-conquer assignment, you probably have to use ...

Question 5: Professor Justin Bieber needs a fast algorithm that searches for an arbitrary element x in a sorted array $A[1 \ldots n]$ of n numbers. He remembers that there is something called "binary search", which maintains an interval $[\ell, r]$ of indices such that, if x is present in the array, then it is contained in the subarray $A[\ell \ldots r]$. In one iteration, the algorithm takes the middle index, say p, in the interval $[\ell, r]$. Then the algorithm either finds x at the position p, or it recurses in the interval $[\ell, p-1]$, or it recurses in the interval $[p+1, r]$. Unfortunately, Professor Bieber does not remember the expression[2] for p in terms of ℓ and r.

Professor Bieber does remember that, instead of choosing p in the middle of the interval $[\ell, r]$, it is often enough to choose p uniformly at random in this interval. Based on this, he obtains the following algorithm: The input consists of the sorted array $A[1 \ldots n]$, its size n, and a number x. If x is in the array, then the algorithm returns the index p such that $A[p] = x$. Otherwise, the algorithm returns "not present". We assume that all numbers in A are distinct.

[1] But how do I get home? A customer service representative will tell you "that is your problem".

[2] is it $\lfloor (r - \ell)/2 \rfloor$, or $\lceil (r - \ell)/2 \rceil$, or $\lfloor (r - \ell + 1)/2 \rfloor$, or $\lceil (r - \ell + 1)/2 \rceil$?

Algorithm BIEBERSEARCH(A, n, x):
$\ell = 1$; $r = n$;
while $\ell \leq r$
do $p =$ uniformly random element in $\{\ell, \ell + 1, \ldots, r\}$;
    if $A[p] < x$
    then $\ell = p + 1$
    else if $A[p] > x$
        then $r = p - 1$
        else return p
        endif
    endif
endwhile;
return "not present"

Let T be the running time of this algorithm on an input array of length n. Note that T is a random variable. Prove that the expected value of T is $O(\log n)$.
Hint: Most solutions that you find on the internet are wrong.

Question 6: You are given a sequence S consisting of n numbers; not all of these numbers need to be distinct.

Describe an algorithm, in plain English, that decides, in $O(n)$ time, whether or not this sequence S contains a number that occurs more than $n/4$ times.

You may use any result that was proven in class. Justify the correctness of your algorithm and explain why the running time is $O(n)$.
Hint: The algorithm must be comparison-based; you are not allowed to use hashing, bucket-sort, or radix-sort.