COMP 3804 — Winter 2026 — Problem Set 2

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: 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 2026. (The k output numbers do not have to be in sorted order. The output may not be unique.)

For example, if $k = 3$ and

$$A = (2028, 10, 2, 2022, 1949, -16, 2025, 2030),$$

then both $(2028, 2025, 2022)$ and $(2028, 2025, 2030)$ 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 2: 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.

Question 3: Justin Bieber is really impressed by the analysis of the expected running time of the randomized selection algorithm:

Algorithm RSELECT(S, k):
Input: Sequence S of numbers, integer k with $1 \leq k \leq |S|$
Output: k-th smallest number in S
if $|S| = 1$
then return the only element in S
else $p =$ uniformly random element in S;
    by scanning S and making $|S| - 1$ comparisons, divide it into
    $L = \{x \in S : x < p\}$,
    $M = \{x \in S : x = p\}$,
    $R = \{x \in S : x > p\}$;
    if $k \leq |L|$
    then RSELECT(L, k)
    else if $k \geq 1 + |L| + |M|$
        then RSELECT$(R, k - |L| - |M|)$
        else return p
        endif
    endif
endif

Let n be the size of the sequence in the first call to RSELECT. In class, the entire computation was divided into *phases*: For any integer $i \geq 0$, a call to algorithm RSELECT is in phase i, if

$$(3/4)^{i+1} \cdot n < \text{ the length of the sequence in this call } \leq (3/4)^i \cdot n.$$

Justin wonders why the number 3/4 is used. He thinks that it is much more natural to say that a call to algorithm RSELECT is in *Bieber-phase* i, if

$$(1/2)^{i+1} \cdot n < \text{ the length of the sequence in this call } \leq (1/2)^i \cdot n.$$

- Use Bieber-phases to prove that the expected running time of algorithm RSELECT on an input sequence of length n is $O(n)$.

Question 4: Let $n \geq 2$ be an integer, and let $A[1 \ldots n]$ be an array storing n pairwise distinct numbers.

It is easy to compute the two smallest numbers in the array A: Using $n-1$ comparisons, we find the smallest number in A. Then, using $n-2$ comparisons, we find the smallest number among the remaining $n-1$ numbers. The total number of comparisons made is $2n-3$. By a similar argument, we can find the smallest and largest numbers in A using $2n-3$ comparisons. In this question, you will show that the number of comparisons can be improved.

(4.1) Consider the following algorithm TWOSMALLEST(A, n), which computes the two smallest numbers in the array A:

```
Algorithm TwoSmallest(A, n):
if A[1] < A[2]      (*)
then smallest = A[1]; secondsmallest = A[2]
else smallest = A[2]; secondsmallest = A[1]
endif;
for i = 3 to n
do if A[i] < smallest      (**)
    then secondsmallest = smallest; smallest = A[i]
    else if A[i] < secondsmallest      (***)
         then secondsmallest = A[i]
         endif
    endif
endfor;
return smallest and secondsmallest
```

In each of the lines (*), (**), and (***), the algorithm *compares* two input numbers.

Assume that A stores a uniformly random permutation of the set $\{1, 2, \ldots, n\}$. Let X be the total number of *comparisons* made when running algorithm TwoSmallest(A, n). Observe that X is a *random variable*. Prove that the expected value of X satisfies

$$\mathbb{E}(X) = 2n - \Theta(\log n).$$

Hint: For each $i = 3, 4, \ldots, n$, use an indicator random variable X_i that indicates whether or not line (***) is executed in iteration i.

(4.2) Consider the following algorithm SmallestLargest(A, n), which computes the smallest and largest numbers in the array A:

```
Algorithm SmallestLargest(A, n):
if A[1] < A[2]      (*)
then smallest = A[1]; largest = A[2]
else smallest = A[2]; largest = A[1]
endif;
for i = 3 to n
do if A[i] < smallest      (**)
    then smallest = A[i]
    else if A[i] > largest      (***)
         then largest = A[i]
         endif
    endif
endfor;
return smallest and largest
```

Observe that this algorithm is very similar to algorithm TwoSmallest(A, n).

Assume that A stores a uniformly random permutation of the set $\{1, 2, \ldots, n\}$. Let Y be the total number of *comparisons* made when running algorithm SmallestLargest(A, n). Observe that Y is a *random variable*. Argue, in a few sentences, that the same analysis as for **(4.1)** implies that

$$\mathbb{E}(Y) = 2n - \Theta(\log n).$$

(4.3) Assume that $n \geq 2$ is a power of 2. Furthermore, assume that the array $A[1 \ldots n]$ stores an arbitrary sequence of n pairwise distinct numbers. Describe, in English, an algorithm that computes the two smallest numbers in the array A, and that makes $n + \log n - 2$ comparisons. Justify your answer.

Hint: Consider a tennis tournament with n players. The players are numbered from 1 to n. For each player i, the number $A[i]$ is the ATP-ranking of this player.

The n players play as they do in any Grand Slam tournament: They play against each other in pairs; after any game, the winner goes to the next round and the loser goes home. This is a special tournament: If player i plays against player j, then the player with the smaller A-value (i.e., higher ATP ranking) is guaranteed to win.

In this way, the smallest number in A corresponds to the tennis player who wins the tournament. The second smallest number in A corresponds to the second best player. This second best player must lose some game. Who can beat the second best player?

(4.4) Assume that $n \geq 2$ is an even integer. Furthermore, assume that the array $A[1 \ldots n]$ stores an arbitrary sequence of n pairwise distinct numbers. Describe, in English, an algorithm that computes the smallest and largest numbers in the array A, and that makes $\frac{3}{2}n - 2$ comparisons. Justify your answer.
Hint: If $A[1] < A[2]$, is it possible that $A[1]$ is the largest number? If $A[1] > A[2]$, is it possible that $A[1]$ is the smallest number? If $A[3] < A[4]$, is it possible that $A[3]$ is the largest number? If $A[3] > A[4]$, is it possible that $A[3]$ is the smallest number?