COMP 3804 — Winter 2025 — Assignment 2

Due: Sunday February 16, 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_a2.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.

Question 1: Write your name and student number.

Question 2: 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
> $S_< = \{x \in S : x < p\}$,
> $S_= = \{x \in S : x = p\}$,
> $S_> = \{x \in S : x > p\}$;
>   **if** $k \leq |S_<|$
>   **then** RSELECT$(S_<, k)$
>   **else if** $k \geq 1 + |S_<| + |S_=|$
>   **then** RSELECT$(S_>, k - |S_<| - |S_=|)$
>   **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 3: Let $n \geq 2$ be an integer, and let $A[1 \dots n]$ 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.

(3.1) 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 $\{1, 2, \dots, n\}$. 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 $i = 3, 4, \dots, n$, use an indicator random variable X_i that indicates whether or not line (***) is executed in iteration i.

(3.2) 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 **(3.1)** implies that

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

(3.3) 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?

(3.4) 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.

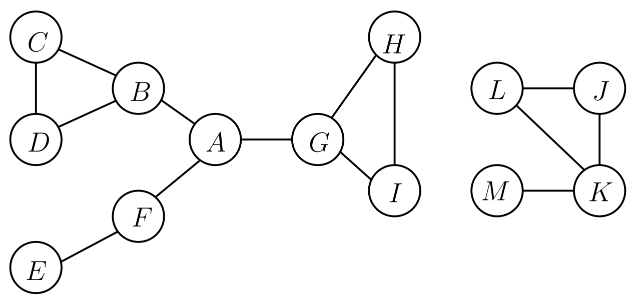
Hint: If $A[1] < A[2]$, is it possible that $A[1]$ is the largest number? If $A[1] > A[2]$, is it possible that $A[1]$ is the smallest number? If $A[3] < A[4]$, is it possible that $A[3]$ is the largest number? If $A[3] > A[4]$, is it possible that $A[3]$ is the smallest number?

Question 4: Let $G = (V, E)$ be an undirected graph. A *vertex coloring* of G is a function $f : V \to \{1, 2, \ldots, k\}$ such that for every edge $\{u, v\}$ in E, $f(u) \neq f(v)$. In words, each vertex u gets a "color" $f(u)$, from a set of k "colors", such that the two vertices of each edge have different colors.

Assume that the graph G has exactly one cycle with an odd number of vertices. (The graph may contain cycles with an even number of vertices.)

What is the smallest integer k such that a vertex coloring with k colors exists? As always, justify your answer.

Question 5: Consider the following undirected graph:



Question 5.1: Draw the DFS-forest obtained by running algorithm DFS on this graph. Recall that algorithm DFS uses algorithm EXPLORE as a subroutine.

In the forest, draw each tree edge as a solid edge, and draw each back edge as a dotted edge.

Whenever there is a choice of vertices (see the two lines labeled (*)), pick the one that is alphabetically first.

Question 5.2: Do the same, but now, whenever there is a choice of vertices (see the two lines labeled (*)), pick the one that is alphabetically last.

Algorithm DFS(G):
for each vertex u
do $visited(u) = false$
endfor;
$cc = 0$;
for each vertex v (*)
do if $visited(v) = false$
    then $cc = cc + 1$
            EXPLORE(v)
    endif
endfor

```
Algorithm EXPLORE(v):
visited(v) = true;
ccnumber(v) = cc;
for each edge {v, u}        (*)
do if visited(u) = false
    then EXPLORE(u)
    endif
endfor
```