Name: _____ Roll no. _____

CSL 7030: Algorithms for Big Data Quiz 2

Date: 10/02/2025 Time: 15 minutes

Instructions: Question 1: 4 points, Question 2: 6 points.
For the MCQ: if there are $1 \leq i \leq 4$ correct options, then each correct option **correct option** will give you $4/i$ points.
Failing to select a correct option, or selecting an incorrect option does not have a penalty.
Correct option without explanation or with wrong explanation gives -1 point.

1. We know that A has size 200. Obviously if A is sorted then it looks like a bunch of 0's followed by a bunch of 1's.
   In particular, this means that the the 10th 1 from the left must come after the 10th 0 from the right. So, option (b) is correct.
   However, it can happen that $\ell_1 > r_0$, but A is not sorted. So, (a) is not correct.
   Now suppose A is $\frac{1}{10}$-far from being sorted, which means that we need to modify at least 20 positions to convert A to a sorted array.
   In this case, we proved a lemma in the class that $\ell_1 < r_0$. Hence (c) is correct.
   If $\ell_1 < r_0$, then there is an inversion between at least $10 = \frac{1}{20} \cdot 200$ 0's and the same number of 1's. Hence, to get to a sorted array, we need to change at least 10 0's to 1's or vice versa. Hence, A is $\frac{1}{20}$-far from being sorted. Hence (d) is correct.
   Correct options: b, c, d.

2. $\mathcal{A}_{\text{new}}$ simply repeats $\mathcal{A}$ say t times **independently**. If $\mathcal{A}$ always says "sorted", then $\mathcal{A}_{\text{new}}$ says "sorted". Otherwise if at least one execution of $\mathcal{A}$ says "not sorted", then $\mathcal{A}_{\text{new}}$ says "not sorted.
   Each execution of $\mathcal{A}$ has time/query complexity $O(\log n)$, therefore the query complexity of $\mathcal{A}_{\text{new}}$ is $O(t \log n)$.
   If the array is sorted, then each execution of $\mathcal{A}$ says "sorted".
   Therefore, $\Pr(\mathcal{A}_{\text{new}}\text{says sorted when array is sorted}) = 1 \geq 1 - \delta$, since $0 < \delta < 1$.

$$
\begin{aligned}
&\Pr(\mathcal{A}_{\text{new}} \text{ says "sorted" when array is } \epsilon\text{-far from being sorted}) \\
&= \Pr(\text{All } t \text{ independent executions of } \mathcal{A} \text{ say "sorted" when array is } \epsilon\text{-far from being sorted}) \\
&= \prod_{i=1}^{t} \Pr(i\text{th execution of } \mathcal{A} \text{ says "sorted" when array is } \epsilon\text{-far from being sorted}) \\
&\leq \prod_{i=1}^{t} (1-\epsilon) = (1-\epsilon)^t \\
&\leq e^{-\epsilon t} && \text{(Since } 1 + x \leq e^x \text{ for all } x\text{)}
\end{aligned}
$$

   We want this probability to be at most δ. Therefore, $e^{-\epsilon t} \leq \delta$, which means that $-\epsilon t \leq \ln \delta$, or $t \geq \frac{1}{\epsilon} \ln \frac{1}{\delta}$. Therefore, we can set $t = \lceil \frac{1}{\epsilon} \ln \frac{1}{\delta} \rceil$. Therefore, overall time complexity of $\mathcal{A}_{new}$ is $O(\frac{1}{\epsilon} \log \frac{1}{\delta} \log n)$.