CSCI3160 Design and Analysis of Algorithms (2025 Fall)

Quiz 1

Date: Oct. 02, 2025

Name: **123456789** Student ID: **Xiao LIANG**

Please read the following instructions carefully.

- Please fill in your *Name* and *Student ID* in the corresponding fields above.
- This question booklet contains 3 questions across 4 pages, for the total of 100 points. Check to see if any pages are missing.
- This question booklet is printed single-sided. If necessary, feel free to use the blank space on the back of each page for your answers.

This table is intended for grading use only. Exam participants are requested to leave it blank.

Question:	1	2	3	Total
Points:	30	30	40	100
Score:				

Exam Questions:

1. In each of the following situations, indicate whether $f = O(g)$, or $f = \Omega(g)$, or both (in which case $f = \Theta(g)$). You can simply write the final answer without providing any justification or explanation.

   (a) (10 points) $f(n) = n \log n$ and $g(n) = 10n \log(10n)$.

   (b) (10 points) $f(n) = n^{1.01}$ and $g(n) = n \log^2 n$.

   (c) (10 points) $f(n) = (\log n)^{\log n}$ and $g(n) = n / \log n$.

Solution:

(a) $f(n) = \Theta(g(n))$.

(b) $f(n) = \Omega(g(n))$.

(c) $f(n) = \Omega(g(n))$.

2. Recall the randomized algorithm for k-selection introduced in the lecture. Given an array S of size n and a target rank k, it repeatedly invokes a randomized subroutine A_{sub} that reduces the problem size by a factor of $2/3$ at each step. The pseudocode for A_{sub} is shown in Algo. 1 below.

Algorithm 1: The Randomized Sub-Algorithm A_{sub}

Input: An array S and a desired rank k.

1. Take an element $v \in S$ uniformly at random.
2. Divide S into S_1 and S_2 where
   - S_1 = the set of elements in S less than or equal to v;
   - S_2 = the set of elements in S greater than v.
3. If $|S_1| \geq k$, then return $S' = S_1$ and $k' = k$;
   else return $S' = S_2$ and $k' = k - |S_1|$.

The algorithm **succeeds** if $|S'| \leq \frac{2}{3}|S|$, or **fails** otherwise.
Repeat the algorithm until it **succeeds**.

We now modify Algo. 1 by changing the success condition from "$|S'| \leq \frac{2}{3}|S|$" to "$|S'| \leq \frac{9}{10}|S|$" while keeping everything else the same. Answer the following questions regarding this modified version:

(a) (20 points) Recall that we proved in lecture that a single run of the original Algo. 1 succeeds with probability at least $\frac{1}{3}$. After the modification above, what lower bound can we give for the success probability? Provide a formal proof of your answer.

(b) (10 points) Recall that we proved in lecture that the randomized k-selection algorithm using Algo. 1 runs in $O(n)$ expected time. Now, if we replace Algo. 1 with the modified version in this k-selection algorithm, what would its expected running time be? Please provide an explanation for your answer.

Solution:

For (a) The lower bound is $\frac{4}{5}$.

Proof. Let $|S| = n$ and let the pivot's rank r be uniformly distributed in $\{1, 2, \ldots, n\}$. Fix the target rank k. After partitioning:

- If $r \geq k$, then $S' = S_1$ and $|S'| = r$.
- If $r < k$, then $S' = S_2$ and $|S'| = n - r$.

A run fails iff $|S'| > 0.9n$. Thus:

- If $r \geq k$, failure implies $r > 0.9n$.
- If $r < k$, failure implies $n - r > 0.9n$, i.e., $r < 0.1n$.

Therefore failure can occur only when r lies in the lowest 10% or the highest 10% of ranks.

Thus, the answer is $1 - 0.1 - 0.1 = 0.8$. □

For (b): The expected running time is still $O(n)$. We can re-do the "geometric series" analysis as we did in the lecture. The only difference is that the multiplicative factor changes from $\frac{2}{3}$ to $\frac{9}{10}$. But this does not change the asymptotic behavior of the geometric series.

3. Consider running the "counting inversion" algorithm on the array $A = (70, 28, 43, 60, 80, 12, 30, 50)$. Recall that the algorithm divides A into two equal halves at the middle, and recursively solves the subproblems corresponding to the two halves, respectively. Answer the following questions:

   (a) (10 points) What are the outputs of the two subproblems, respectively?

   (b) (15 points) After recursion, the algorithm will count the number of "crossing inversions." How many crossing inversions are there in A?

   (c) (15 points) In the class, we used an $O(n \log n)$-time method to count the number of crossing inversions and proved that the whole algorithm ran in $O(n \log^2 n)$ time. Assume that Mr. Goofy decides to replace our $O(n \log n)$-time method with his own method that runs in $O(n \log^2 n)$ time. What is the worst-case time of the whole algorithm now? You need to explain the derivation of your answer.

Solution:

For (a): 3 and 3.

For (b): 9.

For (c): $f(n) = 2 \cdot f(n/2) + O(n \log^2 n)$, which solves to $f(n) = O(n \log^3(n))$ by the Master Theorem.