CS256: Algorithm Design and Analysis **Fall 2026**

Assignment 0 (due 09/16/2026)

Instructor: Shikha Singh *Solution template: Overleaf link*

Note. This is an individual assignment. It will not be graded on correctness but on completion—to get full points, you must attempt all the questions. This problem set is done individually.

The goal of this problem set is for you to check your familiarity with the background material from Data Structures (CS136) and Discrete Math (MATH200), and for you to get comfortable with LaTeX.

Submission guidelines. Sign up on Gradescope using the course code **XEVXD2**. When submitting the assignment PDF on Gradescope, you must match questions to pages in the PDF. This takes less than a minute and is crucial for efficient grading. Sometimes, you may need to mark pages approximately, e.g., for multi-part questions such as Problem 3.

Task 1. Please complete the following brief course survey, preferably by Monday Sept 14, 5 pm: `https://forms.gle/j6XrmpdsBEXpoU8C7`

Problem 1. Let A, B be sets. Prove by contradiction that $A \cap B = \emptyset \implies A \subseteq \overline{B}$.

Hint. When we prove by contradiction, we assume that the statement is false. Here we have an implication: $A \cap B = \emptyset$ implies $A \subseteq \overline{B}$. What does it mean for this implication to be false?

Solution.

$\square$

Problem 2. (**Practice Problem**) This problem includes two summations we will utilize frequently in class. You should review on your own how to prove them but do not need to submit a solution.

- Show that $1 + 2 + 3 + 4 + \ldots + n = n(n+1)/2$. Written another way, prove that $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

- Show that $1 + 2 + 2^2 + 2^3 + 2^4 \ldots + 2^{\log_2 n} = \Theta(n)$.

Solution. (*Optional*)

$\square$

> **Note.** Problem 3 relies on material you've likely seen before, but that we won't go over in class until Monday 09/14/26.

Problem 3. For each of the following, answer with the tightest upper bound from this list: $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$, $O(2^n)$. No justification necessary.

(a) The *depth* of a complete binary tree with n nodes:

Solution.

□

(b) The worst-case run time to sort n items using merge sort:

Solution.

□

(c) The number of distinct subsets of a set of n items:

Solution.

□

(d) The number of bits needed to represent the positive integer n:

Solution.

□

(e) The time to find the second largest number in a set of n (not necessarily sorted) numbers:

Solution.

□

Problem 4. (Induction on Graphs and Trees) In this course, correctness of graph algorithms is often proved using induction. In this question, we will practice proving the following property about trees using induction.

> **Claim 1.** *Any tree with n vertices has exactly $n-1$ edges.*

First, we will look at an incorrect induction attempt for Claim 1, which *feels* like a real induction but does not prove the claim above.

Proof by induction. (Incorrect). We prove the claim by an induction on the number of vertices n of T. A tree with $n = 1$ vertex has 0 edges (*Base case*). Assume that any arbitrary tree T with $n \geq 2$ vertices has $n-1$ edges (*Inductive hypothesis*).
Let $w \in T$ be a leaf node. A tree must always have at least one leaf node (node of degree 1), since otherwise it would contain a cycle. Suppose we add a vertex v to T via the edge (v, w). Let the new tree be T'. Then T' has $n+1$ vertices and one more edge than T. By the inductive hypothesis T has $n-1$ edges. Thus, T' has n edges, which proves the claim.

(a) The problem with the above attempt is that it proves the claim about a specific tree T' in the inductive step (rather than an arbitrary tree). To support this, give an example of a tree on $n+1 = 4$ vertices that cannot be constructed in this way.

Solution.

□

(b) To fix the proof, start with an *arbitrary tree* T on $n+1$ vertices and *remove* a leaf to obtain tree T' on n vertices. Complete this proof and explain why it works.

Solution.

□