Vertex Institute of Technology

Undergraduate Teaching Assistant (TA) Grading Guideline

Course: CS 201 — Data Structures & Algorithms (Fall 2026)

Dr. Eleanor Vance
Lead Instructor
evance@vertex.edu

Sarah Jenkins
Head Teaching Assistant
sjenkins@vertex.edu

Last Updated: August 2026

Abstract

This document serves as the official grading guidelines for Undergraduate Teaching Assistants (TAs) in CS 201 (Data Structures & Algorithms) at the Vertex Institute of Technology. TAs play a crucial role in student learning by providing objective, consistent, and timely evaluations of student homework submissions. This guideline outlines general grading policies, detailed rubrics for Homework 3, best practices for writing constructive feedback, and procedures for addressing grade discrepancy and student regrade requests.

---

1 Introduction

As a Teaching Assistant for CS 201, you are one of the primary points of contact for students navigating complex algorithmic concepts. Grading is not merely an administrative task; it is an essential component of student learning. Your feedback helps students identify logical errors, improve code quality, and reinforce key theoretical concepts [1].

Our grading goals are:

- **Consistency:** Every student submission must be evaluated under the same rigorous standard, regardless of which TA grades the paper.
- **Objectivity:** Grades must be based strictly on the rubrics provided, without personal bias or arbitrary deductions.
- **Constructive Feedback:** Every deduction must be accompanied by a clear, polite explanation of what was incorrect and how to correct it.

1.1 Grading Workflow

To maintain consistency and meet grading deadlines, all TAs must follow the structured workflow illustrated below. Submissions must be graded and released within seven calendar days of the as-

signment deadline.

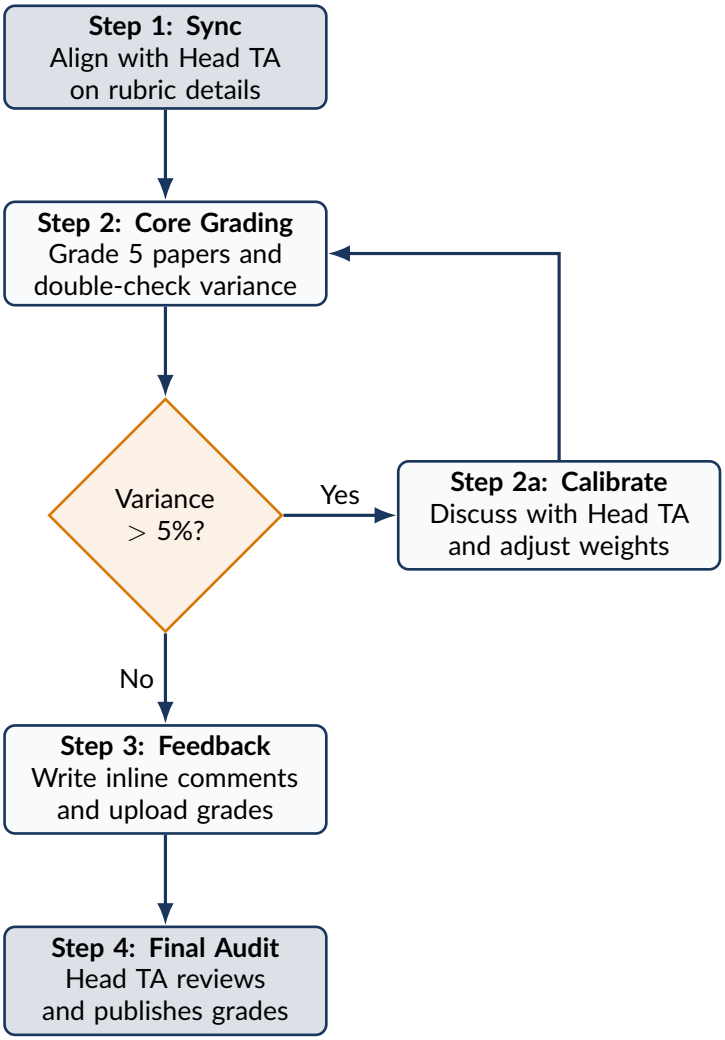

Figure 1: CS 201 Sequential Grading and Calibration Workflow.

By synchronizing grading early, we prevent downstream regrade requests and ensure fairness across all discussion sections [4].

2 General Grading Policies

To maintain academic standards and ensure fair treatment for all students, you must adhere to the following course policies. Any exceptions to these policies must be approved directly by Dr. Vance [3].

2.1 Late Submission Policy

CS 201 has a strict late policy managed automatically by the Nexa LMS.

- Submissions up to 24 hours late receive a flat **15% deduction**.
- Submissions between 24 and 48 hours late receive a flat **30% deduction**.
- Submissions exceeding 48 hours late receive a **score of 0**.

As a TA, you do not need to manually calculate these deductions. The Nexa LMS will apply the late penalty to the final grade. Your job is to grade the assignment as if it were submitted on time, and the system will handle the discount.

2.2 Academic Integrity and Plagiarism

Academic integrity is taken very seriously at Vertex. We use the Moss (Measure of Software Similarity) system to screen all programming assignments.

⚠ Handling Suspected Plagiarism

If you suspect plagiarism or notice code that is identical to another student's work or an online solution:

1. **Do NOT grade** the submission or write feedback.
2. **Do NOT contact** the student or accuse them of cheating.
3. **Flag the submission** on the Synapse grading portal and email the student's name, ID, and matching evidence to Dr. Vance immediately.

The instructional team will handle all academic integrity investigations.

2.3 Regrade Requests

Students have exactly seven days from the release of grades to submit a formal regrade request via the Synapse portal.

- **Minor Errors:** If you made a simple clerical error (e.g., miscounting points), correct it immediately and notify the Head TA.
- **Conceptual Disagreements:** If a student disputes a rubric deduction, they must submit a written justification referencing the lecture slides or textbook. In such cases, the student's entire assignment will be subject to a complete re-evaluation, which may result in a higher, identical, or lower score.

Never argue with a student. If a discussion becomes tense, politely direct the student to the Head TA or Dr. Vance.

3 Homework 3 Grading Rubric

Homework 3 focuses on the implementation and analysis of self-balancing binary search trees (specifically AVL trees) and their performance comparison with standard Binary Search Trees (BST). The assignment is graded out of 100 points, broken down into three categories.

3.1 Detailed Point Breakdown

Table 1 outlines the grading criteria. When grading, apply deductions incrementally. Do not deduct more than the maximum allocated points for any sub-component.

Component	Grading Criteria	Max Points	Type
1. AVL Implementation	Correct single and double rotations during node insertion	20	Autograded
	Tree height maintenance and balance factor updates	15	Autograded
	Correct implementation of tree deletion (successor swap)	15	Autograded
2. Complexity Analysis	Analytical proof of AVL tree search time complexity $O(\log n)$	15	Manual (TA)
	Correct identification of worst-case BST height scenarios	10	Manual (TA)
3. Code Quality & Style	Proper encapsulation, use of comments, and variable naming	15	Manual (TA)
	Absence of memory leaks (verified using Valgrind)	10	Autograded

Table 1: Homework 3 Detailed Point Distribution and Grading Modality.

3.2 Common Deductions Guide

To ensure grading uniformity, apply these standard deductions for manual components:

- **Complexity Analysis (Section 2):**
  - Deduct **5 points** if the proof lacks mathematical induction or does not reference the Fibonacci lower bound for AVL nodes (see Cormen et al. [2]).
  - Deduct **3 points** if worst-case BST height is stated as $O(n)$ but lacks a descriptive diagram or verbal explanation of skewed trees.
- **Code Quality (Section 3):**
  - Deduct **2 points** for poor indentation or inconsistent brace style (up to a maximum of 5 points).
  - Deduct **5 points** if raw pointers are used where smart pointers (e.g., `std::unique_ptr`) were explicitly requested in the prompt.

4 Feedback Guidelines

Writing high-quality feedback is one of your most important responsibilities. Students read feedback to understand their mistakes and improve their future work. Good feedback is specific, actionable, and encouraging.

4.1 Tone and Style

Always maintain a professional, polite, and encouraging tone. Never use sarcasm, exclamation marks in deductions, or judgmental language.

Examples of Grading Feedback

Below are comparative examples of poor and high-quality feedback.

Example 1: Logical Error in AVL Rotations

- ✗ **Poor:** "Your rotation code is completely broken. -10 points. Check the slides."
- ✓ **Good:** "-10 points. Your implementation of right-left double rotation fails because you did not update the parent pointer of the root node after the first rotation. Refer to Lecture 6, slide 14 for the correct pointer update sequence."

Example 2: Code Quality and Comments

- ✗ **Poor:** "No comments. Bad code quality. -5."
- ✓ **Good:** "-5 points. While your code compiles and passes all tests, there are no docstrings or inline comments explaining the search helper function. Adding descriptive comments makes your code maintainable and easier to debug."

4.2 Feedback Writing Checklist

When writing feedback in the Synapse system, ensure you:

1. State the **exact rubric item** and points deducted.
2. Describe the **specific bug or omission** in the student's submission.
3. Direct the student to a **resource** (e.g., lecture slide, textbook chapter, or discussion notes) where they can find the correct implementation.
4. Write in complete, grammatically correct sentences.

5 Target Grade Distribution

While we grade strictly according to the rubric, we monitor the overall class distribution to ensure the exam difficulty and grading standards are aligned with student performance.

5.1 Historical Distribution

Figure 2 shows the historical and target grade distribution for CS 201 over the last three semesters at the Vertex Institute of Technology.

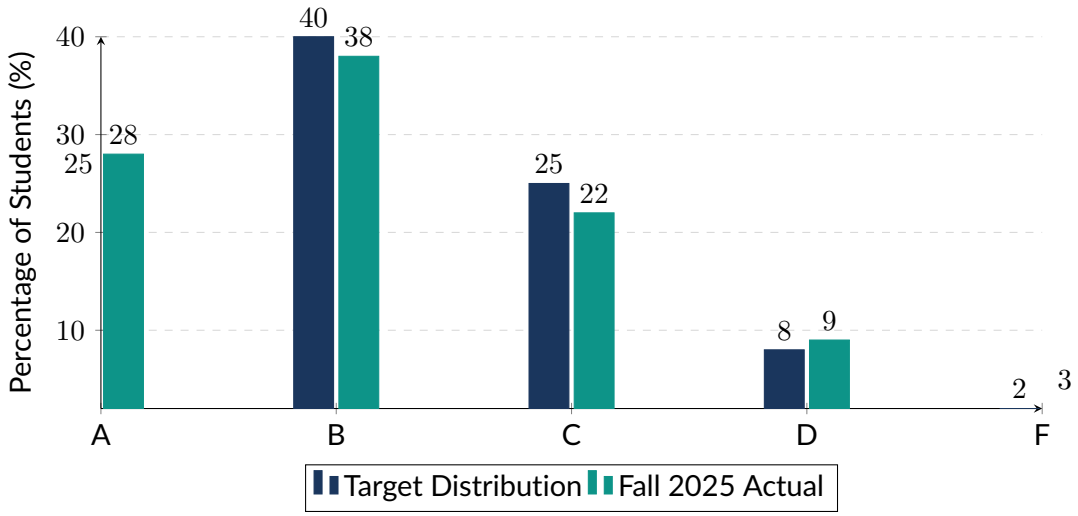

Figure 2: Target vs. Actual Grade Distribution for CS 201.

5.2 Grade Inflation and Calibration

If you notice that more than 60% of students in your assigned grading bucket are receiving an “A” on a manual component, pause your grading and contact the Head TA. We may need to recalibrate the manual criteria to prevent grade inflation. Conversely, if more than 20% of students are failing a component, we will schedule an emergency review session.

References

[1] Lillian Barker and Keith A. Shaffer. Effective feedback in computer science education: A study of ta written comments. *Journal of Educational Computing Research*, 56(4):512–528, 2018.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 4th edition, 2022.

[3] Sarah Jenkins and Eleanor Vance. Pedagogical guidelines and best practices for undergraduate teaching assistants. Technical Report VIT-CSE-2025-04, Vertex Institute of Technology, Department of Computer Science & Engineering, 2025.

[4] Alistair Smith, Beatrice Vance, and Marcus Sterling. Calibrating grading rubrics across large-scale discussion sections. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE)*, pages 891–897, 2021.