

Midterm Review

CS 561 – Fall 2026

Jared Saia

University of New Mexico

Logistics

- **Thursday, October 15, 2026**, during regular class time: 12:30–1:45 p.m. (75 minutes).
- **5 problems.**
- Closed-book, except for **two cheat sheets**: two regular-sized sheets of paper, handwritten on both sides.
- No electronic devices or calculators.

During the Exam

- Read through the problems first and budget your time.
- Work both quickly and carefully. If you get stuck, record useful progress and return to the problem later.
- Define your notation and explain why your method works.
- Check base cases, boundary cases, and the direction of inequalities.

Topics: Foundations

- **Asymptotics and induction:** O , Ω , Θ , and related notation; proofs by induction.
- **Recurrences:** recursion trees, the Master Method and its conditions, annihilators, and change of variables.
- **Probability:** indicator random variables, linearity of expectation, union bounds, Markov's inequality, and the tail-sum identity.

Review Chapters 3–5 of CLRS (fourth edition), lecture notes, and the corresponding homework problems.

Topics: Algorithms

- **Randomized algorithms and data structures:** randomized quicksort, bucket sort, skip lists, Bloom filters, and Count-Min Sketch.
- **Dynamic programming:** edit distance, matrix-chain multiplication, and maximum-weight independent set on trees.
- **Greedy algorithms:** activity selection, fractional versus 0–1 knapsack, exchange arguments, and counterexamples.

Review lecture notes and homework, plus CLRS Chapters 5 (randomized algorithms), 14 (dynamic programming), and 15 (greedy algorithms).

Problem: Short Answer

Practice true/false, matching, and short-answer questions.

- Give tight bounds in Θ notation when requested.
- Know the time and space bounds of the algorithms covered, and when you would use each one.
- Distinguish worst-case bounds from expected bounds and high-probability guarantees.
- State the assumptions behind a bound, such as the input distribution for bucket sort.

Induction and Recurrences

Induction checklist:

- State the claim and prove the base case(s).
- State the induction hypothesis precisely.
- Prove the induction step using that hypothesis.

Recurrence checklist:

- Form the recurrence and its base cases.
- Choose an applicable method and derive a tight bound.

Problem: Dynamic Programming

1. Define in words: what is each smaller problem?
2. Give the recurrence and all necessary base cases.
3. Describe how to fill in solutions to the smaller problems and say which value is returned as the answer.

Practice adapting problems from lecture and homework to new constraints.

Problem: Greedy Algorithms

- State the greedy rule precisely, including how ties are handled when relevant.
- For a correct algorithm, use an **exchange argument**: modify an optimal solution to agree with the greedy choice without making it worse, then handle the remaining instance.
- For an incorrect algorithm, give a **counterexample**: show the greedy output and a strictly better feasible solution.

Problem: Probability

- Define the random variable or events you want to analyze. Indicators are often useful to “break up” a hard random variable.
- **Linearity of expectation** does not require independent random variables.
- The **union bound** does not require independent events.
- **Markov’s inequality** requires $X \geq 0$ and $a > 0$:

$$\Pr(X \geq a) \leq \frac{\mathbb{E}[X]}{a}.$$

Randomized Data Structures

- **Skip lists:** exact answers; expected $O(\log n)$ time per search, insertion, or deletion, and expected $O(n)$ space ($n \geq 1$).
- **Bloom filters:** $O(k)$ insertion/query time, m bits. False positives are possible; no false negatives after insertions.
- **Count-Min:** T unit updates; never underestimates. For a fixed stream and query independent of the hashes,

$$\Pr\left(f_i(T) \leq \hat{f}_i(T) \leq f_i(T) + \varepsilon T\right) \geq 1 - \delta$$

when $w = \lceil e/\varepsilon \rceil$, $d = \lceil \ln(1/\delta) \rceil$. Uses dw counters and $O(d)$ time per update or query.

Assumptions are as in lecture; $0 < \varepsilon, \delta < 1$.

Approaching a Harder Problem

- Try small cases and identify what makes the problem difficult.
- Define subproblems or random variables that capture the information you need.
- Look for a connection to an induction, recurrence, or algorithmic pattern from class.

How to Study

Solve problems! Start with this year's homework and worked lecture examples.

1. Attempt the problem without looking at a solution.
2. If stuck, reveal only a small hint, then continue on your own.
3. Compare your work with the solution and identify the missing idea or mistake.
4. Re-solve missed problems from scratch, without hints.
5. Make a timed attempt at a past midterm, using only your allowed cheat sheets. Then check and revise your work.

Past Midterms

Click a term to open its exam PDF.

CS 561	CS 362
Fall 2025	Spring 2026
Fall 2024	Spring 2025
Fall 2023	Spring 2024

Scope note: Skip CS 561 Fall 2023, Question 1(e), on amortized analysis.

Practice using methods on new problems, rather than memorizing solutions to old problems.

More Practice

- Prioritize this year's homework, lecture examples, and the linked past midterms.
- Use relevant worked examples and exercises from CLRS and [Jeff Erickson's free *Algorithms* book](#) for additional practice.
- [LeetCode](#) can provide extra DP and greedy problems, but practice defining states, proving correctness, and analyzing complexity as well as writing code.