Precept Outline
- Course introduction.
- Review of Lectures 1 and 2.
- Overview of Assignment 1 (Percolation).

Relevant Book Sections
- 1.4 (Analysis) and 1.5 (Union-Find)
- 1.1 and 1.2 (Java review)

A. INTRODUCTION

We're pretty psyched for you to see what we've got in store, but, before we let you loose, here are just a few words about the format of this precept:

These exercises are meant to be done in pairs. We want to encourage you to talk about the details of algorithms and data structures with a peer so you can help fill in each other's blind spots.

These exercises are not graded. Though your answers in Ed are marked as "correct" or "incorrect," you'll be able to see the solution immediately after your first attempt. You can go back to correct your answer when you get a question wrong, or, you can just leave it as-is and move on.

For more practice or a challenge, try the optional problems. These handouts and Ed lessons may contain additional problems. You **are** expected to do all of the exercises in the handout, except when clearly marked optional. We encourage you to try the others – some are quite fun. (Really!) If you or your precept doesn't make it through every problem, that's OK; just make sure to solve them before the next.

Credit earned for collaborative participation in precept. Your first two absences will be automatically waived. Beyond that, a waiver will be granted only via a request in Ed (using the *Accommodations* → *Precept* tag) with accompanying documentation.

The course staff is here to help. Your preceptor is in your corner. If you get stuck anywhere in these exercises, please don't hesitate to call us over. We'd be happy to explain anything that's not making sense or just review the material with you if you're ever feeling rusty on something.

B. ANALYSIS

Part 1: Experimental Analysis

Suppose that you collect the following timing data for a program as a function of the input size n.

n	$T(n)$
125	0.03 sec
1,000	1.00 sec
8,000	32.00 sec
64,000	1,024.00 sec
512,000	32,768.00 sec

Estimate the running time of the program as a function of n and use tilde notation. (Remember that we assume the *power law hypothesis*.)

Part 2: Mathematical Analysis

Runtime analysis can be tricky business; even short and simple-looking code can be hard to analyze correctly. The key to mastering this skill is practice, practice, practice. That's what we'll do in this part.

Solve Exercises 1 to 7 of the "Analysis Drills" Ed lesson.

C. UNION-FIND

Solve all the exercises in the "Union-Find" Ed lesson.

Challenge Problem (optional): Suppose you are given a sequence of n positive integers. Define a "group" as a contiguous subsequence of elements. The length of the group is the number of elements in it, and its value is its smallest element. (E.g., the sequence $(5, 3, 4, 1)$ has a single group of length 4 and value 1; two groups of length 3, with values 3 and 1; etc.)

For each $\ell = 1, \ldots, n$, determine the maximum value among all groups of length ℓ. Your algorithm's runtime should be smaller than $\Theta(n^2)$.

D. ASSIGNMENT OVERVIEW: PERCOLATION

Your preceptor will introduce and give an overview of your first assignment. Please don't hesitate to ask questions!