Part VIII --- LINEAR-TIME SELECTION (Required). These lectures study the problem of computing the ith smallest element of an input array (e.g., the median). It's easy to solve this problem in O(n log n) time using sorting, but we can do better! The required material goes over a super-practical randomized algorithm, very much in the spirit of QuickSort, that has \*linear\* expected running time. Don't forget that it takes linear time just to read the input! The analysis is somewhat different than what we studied for QuickSort, but is equally slick. Basically, there's a cool way to think about the progress the algorithm makes in terms of a simple coin-flipping experiment. Linearity of expectation (yes, it's back...) seals the deal.

Part VIII --- LINEAR-TIME SELECTION (Optional). I couldn't resist covering some advanced related material. The first is an algorithm that has more Turing Award-winning authors than any other algorithm that I know of. It is a deterministic (i.e., no randomization allowed) linear-time algorithm for the Selection problem, based on an ingenious "median-of-medians" idea for guaranteeing good pivot choices. (There are some accompanying lectures notes for this part, available for download underneath each video.) The second optional topic answers the question "can we do better?" for sorting, unfortunately in the negative. That is, a counting argument shows that there is no "comparison-based" sorting algorithm (like MergeSort, QuickSort, or HeapSort) with worst-case running time better than n log n.

Part IX --- GRAPHS AND THE CONTRACTION ALGORITHM. The second set of lectures for this week is a segue from randomized algorithms to graph algorithms. We first review graphs and the most standard ways of representing them (most commonly, by adjacency lists). We then consider the random contraction algorithm, discovered by Karger "only" 20ish years ago (while a PhD student here at Stanford). This algorithm solves the minimum cut problem --- given an undirected graph, separate the vertices into two non-empty groups to minimize the number of "crossing edges". Such problems come up when reasoning about, for example, physical networks, social networks, and images. This algorithm was perhaps the first strong evidence that graph problems could be added to the long list of "killer applications" of random sampling. Don't tune out before the final plot twist --- a simple but useful trick for transforming an algorithm that almost always fails into one that almost always succeeds.

HOMEWORK: Problem Set #4 has five questions about the randomized selection algorithm, cuts in graphs, and the contraction algorithm. Programming Assignment #4 asks you to implement the contraction algorithm and use it to compute the min cut of the graph that we provide.

SUGGESTED READINGS FOR WEEK 4: Algorithms Illuminated (Part 1), Chapter 6.

For a textbook treatment of the contraction algorithm, see Kleinberg and Tardos, Algorithm Design, Sections 13.2 and 13.5.