Section 3 -- DIVIDE AND CONQUER ALGORITHMS: The next set of lectures discusses three non-trivial examples of the divide and conquer algorithm design paradigm. The first is for counting the number of inversions in an array. This problem is related to measuring similarity between two ranked lists, which in turn is relevant for making good recommendations to someone based on your knowledge of their and others' preferences ("collaborative filtering"). The second algorithm is Strassen's mind-blowing recursive algorithm for matrix multiplication, which improves over the obvious iterative method. The third algorithm, which is more advanced and is optional material, is for computing the closest pair of points in the plane.

Section 4 -- THE MASTER METHOD: These lectures cover a "black-box" method for solving recurrences. You can then immediately determine the running time of most of the divide-and-conquer algorithms that you'll ever see! (Including Karatsuba's integer multiplication algorithm from Week 1.) The proof is a nice generalization of the recursion tree method that we used to analyze MergeSort. Ever wonder about the mysterious three cases of the Master Method? Watch these videos and hopefully all will become clear.

HOMEWORK: Problem Set #2 has five questions that should give you practice with divide-and-conquer algorithms and the Master Method. Programming assignment #2 asks you to implement the counting inversions algorithm (from Part III) in whatever programming language you please, run it on a quite large input, and enter the answer.

SUGGESTED READING FOR WEEK 2: Algorithms Illuminated (Part 1), Chapters 3 and 4.