Divide and Conquer Algorithms: Quicksort

- A key part of Quicksort is a partition algorithm. Read about **partitions** from the middle of page 158 to the bottom of page 159, about partitions in general and Lomuto's algorithm in particular.
  - Describe the overall plan in Lomuto's algorithm.
  - What is the role of the indices s and i in Lomuto's algorithm? What do they each keep track of?
- Read 5.2
  - Mergesort separates the array elements based on their position. What does **Quicksort** do instead?
  - What do we refer to as an *array partition*?
  - What is the overall approach of the Quicksort algorithm?
    - Why do we not include the s position in the two recursive calls?
    - Both MergeSort and Quicksort do two key things: two recursive calls, and a merge/partition. In what ways to they differ?
  - Understand Hoare's partition algorithm.
    - What do the indices i, j represent? Where do they start at? Which way do they move as the algorithm progresses?
    - There are two different ways for the algorithm to terminate. What are they?
    - Work out Hoare's algorithm for the word EXAMPLE
    - Explain the need for the last swap. What would happen if that swap did not occur?
  - What is the best-case running time for the Quicksort algorithm? Explain.
  - What will happen to Quicksort if the array is already sorted?
  - What is the worst-case running time for the Quicksort algorithm?
  - What is the average-case running time for Quicksort?
  - What are some improvements that have been developed for Quicksort?
  - Is Quicksort a stable algorithm?
  - Practice problems: 5.1, 5.2, 5.3, 5.7