**R-8.3**

**R-8.4 Hint:** Consider how the pivots work on a sorted sequence.

**Solution:** $O(n \log n)$ time. This is because the pivot splits a sorted sequence in half each time, since it is the median.

**C-8.3 Solution:** Merge sequences $A$ and $B$ into a new sequence $C$ (i.e., call merge$(A, B, C)$). Do a linear scan through the sequence $C$ removing all duplicate elements (i.e., if the next element is equal to the current element, remove it).

**C-8.5 Solution:** For the red and blue elements, we can order them by doing the following. Start with a marker at the beginning of the array and one at the end of the array. While the first marker is at a blue element, continue incrementing its index. Likewise, when the second marker is at a red element, continue decrementing its index. When the first marker has reached a red element and the second a blue element, swap the elements. Continue moving the markers and swapping until they meet. At this point, the sequence is ordered. With three colors in the sequence, we can order it by doing the above algorithm twice. In the first run, we will move one color to the front, swapping back elements of the other two colors. Then we can start at the end of the first run and swap the elements of the other two colors in exactly the same way as before. Only this time the first marker will begin where it stopped at the end of the first run.

**A-8.2 Hint:** Sort first, choosing the appropriate key for comparisons.

**Solution:** First sort the sequence $S$ by the candidate's ID. Then walk through the sorted sequence, storing the current max count and the count of the current candidate ID as you go. When you move on to a new ID, check it against the current max and replace the max if necessary.

**R-9.1 Hint:** Review each of these comparison-based algorithms.

**Solution:** The bubble-sort and merge-sort algorithms are stable.

**R-9.2 Hint:** Review the case for $d = 2$ and generalize this approach.

**Solution:** Preform the stable bucket sort on $m$, $l$, and then $k$. In general, perform the stable bucket sort on $k_d$ down to $k_1$.