1. Explain what you think the worst-case, Big-O complexity and the best-case, Big-O complexity of bubble sort is. Why do you think that?
2. Explain what you think the worst-case, Big-O complexity and the best-case, Big-O complexity of selection sort is. Why do you think that?
3. Does selection sort require any additional storage (i.e. did you have to allocate any extra memory to perform the sort?) beyond the original array?
4. Would the Big-O complexity of any of these algorithms change if we used a linked list instead of an array?
5. Explain what you think Big-O complexity of sorting algorithm that is built into the C libraries is. Why do you think that?
6. The worst-case is both O(n^2). Because the algorithm needs to make n-1 passes through the array and during each pass, it needs to compare and swap elements of the array, which takes O(n) time. Therefore, the total time taken would be n \* O(n) = O(n^2). In the best case, if the array is already sorted, the algorithm needs n-1 passes through the array, and no swapping takes place, so the time taken would be O(n).
7. Both the worst-case and best-case Big-O complexity of selection sort are O(n2). Because in the worst case, this algorithm needs to make n-1 passes to find the minimum element, which takes O(n) time. Therefore, the result is n \* O(n) = O(n^2). In the best case, as the array is already sorted, it needs to make n-1 passes as well, but the minimum element is always the first element and then the time O(n2) as well.
8. Selection sort does not need any additional storage as its sorting is in-place.
9. No. The time complexity of sorting is determined by the number of comparisons and swaps made, and this is independent of the data structure used to store the data.
10. O(n log n). Since