Searching And Sorting Worksheet

1. Identify the following search algorithms as Sequential Search, Binary Search, or "none of the above."

   (a) Examine consecutive elements until the target is found.

   (b) Examine the middle element; if the target is smaller, apply the same algorithm to the left half; if the target is larger, apply the same algorithm to the right half.

   (c) Examine the middle element; if the target is smaller, examine consecutive elements in the left half; if the target is larger, examine consecutive elements in the right half.

2. In his paper presented at SIGCSE 2003 [1], Owen Astrachan examined the history and origin of popularity of Bubble Sort (which is no longer taught in most computer science courses). Read the paper, including pseudocode for Bubble Sort (Section 2.1 on Page 2). Answer the following questions.

   (a) Is Bubble Sort, on average, more or less efficient than Selection Sort?

   (b) What is the big-O for the best case for Bubble Sort?

   (c) Is Bubble sort's code presented in the paper shorter or longer that a typical implementation of selection Sort?

   (d) Is it esier to implement Bubble Sort with iterations or recursively?

   (e) The author states: "Here we show that by several measures bubble sort is not simpler to code than other sorts and that its performance is terrible." Why?

   (f) In Section 2.4 of his paper, Owen Astrachan writes "A 1988 SIGCSE paper [16] notes that bubble sort 'while once the best known sort, seems relegated to the status of an example of inefficiency'. Fifteen years later this sort is still with us, and students continue to use it."

3. Identify the following sorting algorithms as Selection Sort, Bubble Sort, Insertion Sort, and Mergesort, or "none of the above."

   (a) If the array size is less than or equal to 2, just swap the elements if necessary. Otherwise, split the array into two halves. Recursively sort the first half and the second half, then merge the two sorted halves.

   (b) Choose a pivot element and partition the array, so that all the elements on the left side of pivot are less than or equal to the pivot and all the elements on the right side of pivot are greater than or equal to the pivot; then recursively sort each part.

   (c) Set *n* to the size of the array. While *n* is greater than 2, repeat the following: find the largest element among the first *n*, swap it with the *n*-th element of the array, and decrement *n* by one.

   (d) Keep the first *n* elements sorted. Starting at $n = 2$, repeat the following: find the place for `a[n]` in order among the first $n - 1$ elements, shift the required number of elements to the right to make room, and insert `a[n]`. Increment *n* by one.

   (e) Swap consecutive pairs of elements if they are out of order. Repeat n times.

4. Describe in your own words and write pseudocode for the following sorting algorithms:

   (a) Selection Sort
   (b) Insertion Sort
   (b) Bubble Sort
   (d) Mergesort

5. Arrange the following algorithms in order of their average time efficiency: Selection Sort, Insertion Sort, Mergesort, Bubble Sort, Quicksort.

6. Which of the searching algorithms, whose implementation is presented in Java Methods, require temporary arrays?

7. Which of the sorting algorithms, whose implementation is presented in Java Methods, require temporary arrays?

8. Is the big-O time efficiency different for iterative and recursive implementations of Binary Search? Selection Sort?

9. Describe the big-O time efficiency for average, best, and worst case scenarios for the following algorithms:

   (a) Sequential Search
   (b) Binary Search
   (c) Selection Sort
   (d) Insertion Sort
   (e) Mergesort
   (f) Quicksort