CS 201 Homework 4 – Written Answers

Name: Benjamin Allan-Rahill

Problem 1: Identifying Sorting Algorithms

Part a. Based on Bud's data, fill in the asymptotic notation in the table below that best describes the running time of the routine on a particular type of array. You should use one of the following for every entry of the table: O(1), O(log(n)), O(n), O(nlog(n)), $O(n^2)$, $O(n^3)$. (Note: because the numbers are taken from an actual implementation, they may not fit any category exactly; if the category is ambiguous, say so!)

	Sorted Array	Unsorted Array
sort1	O(n)	$O(n^2)$
sort2	$O(n^2)$	O(nlog(n))
sort3	$O(n^2)$	$O(n^2)$
sort4	O(n) OR O(nlogn)	O(n) OR O(nlogn)

Part b. Based on the table from part a, determine for each of Bud's four routines, which of the four sorting algorithms it uses. Briefly explain your reasoning.

sort1: Insertion sort - If the array is already sorted, the algorithm does not have to make as many comparisons and swappings, indicating a linear relationship.

sort2: Quick sort - The worst case for quick sort occurs when the array is already sorted or the pivot is too low or high. Ideally, the algorithm would like to partition the array in two (have the pivot in the middle), but with a sorted array, it compares the pivot to every other element and nothing changes.

sort3: Selection Sort - Even though the array is sorted, the algorithm will still run all the comparisons, giving the algorithm $O(n^2)$ run time.

sort4: Merge Sort - Like selection sort, even if the array is sorted, the algorithm will run through all of the comparisons, giving it the same run times for both scenarios.

Part c. Answer the following for the four sorting algorithms that Bud has implemented:

i. Which sorting algorithm(s) does much better on sorted arrays than on random arrays? Why?

Insertion sort - If the array is already sorted, the algorithm does not have to make as many comparisons and swappings, indicating a linear relationship.

ii. Which sorting algorithm(s) does much worse on sorted arrays than on random arrays? Why?

Quick Sort - The worst case for quick sort occurs when the array is already sorted or the pivot is too low or high. Ideally, the algorithm would like to partition the array in two (have the pivot in the middle), but with a sorted array, it compares the pivot to every other element and nothing changes.

iii. Which sorting algorithm(s) takes about the same amount of time on both sorted and random arrays? Why?

Selection - This algorithm will run through all of the same comparisons wether the array is sorted or unsorted. It will still select select a single max and “place” it at the end. This will result in O(n^2) run time.

Merge Sort - Like selection sort, even if the array is sorted, the algorithm will run through all of the comparisons, giving it the same run times for both scenarios.