**HW11a: Sorting and Searching part 1**

1. In this problem you will investigate the running time and asymptotic class of several different implementations of the Selection Sort algorithm.

a. Run the SelectionSortExperiment program contained in the comp132.examples.sorting package of the 132SampleCode project. Perform the following tasks based on the results reported:

i. Give a formula describing the running time of the selection sort implementation in the iterativeSort method as a function of the problem size (n). To get this formula you can use the process that we used in class:

* Copy and paste the results into Excel.
* Create an x-y scatter plot of the data.
* Add a polynomial trend line to the graph with its equation.

ii. Express the running time from part i using Big-O notation.

b. Repeat part a using the recursive implementation of selection sort that is provided in the recursiveSort method of the SelectionSort class. Note: You may need to eliminate some of the largest size arrays to avoid recursive call stack overflow.

c. Briefly compare the running times and the asymptotic classes of the two distinct implementations of selection sort. Which sort is faster? Do they belong to the same asymptotic class?

2. True or False: An implementation of a O(n) algorithm will always run faster than an implementation of a O(n2) algorithm, regardless of the problem size. Briefly but fully explain your answer.

3. For each of the following running times give the name of the asymptotic class to which the algorithm belongs (e.g. constant, linear, etc…) and express the running time using Big-O notation.

a. 3n2-2n+273

b. 14n + 22 n lg n + 1

c. 0.0001n + 100123

d. (n – 4) (2n – 5)

e. 123456 + 10 lg n

f. 232-1

4. Imagine a program that processes 5000 input values in 10 seconds. About how long would it take the program to process 20,000 input values if the algorithm implemented by the program is:

a. A O(n) algorithm.

b. A O(n2) algorithm.

5. Consider the following program that swaps the maximum and minimum values in an array.

**public** **void** swapMaxAndMin(**int**[] vals) {

**int** mindex = 0;

**int** maxdex = 0;

**for** (**int** i=1; i<vals.length; i++) {

**if** (vals[i] < vals[mindex]) {

mindex = i;

}

**if** (vals[i] > vals[maxdex]) {

maxdex = i;

}

}

**int** tmp = vals[mindex];

vals[mindex] = vals[maxdex];

vals[maxdex] = tmp;

}

a. Give a function describing the operation count for this program if an array access is counted as the basic operation.

b. Express the function from part a using Big-O notation.

6. We know that once the problem becomes large enough a O(n) algorithm will be faster than a O(n2) algorithm. Consider the following running time equations for programs implementing two different algorithms for the problem:

A: f(n) = 2n2 – 3n + 70

B: f(n) = 25n + 100

How large would the problem need to be in order for program B to be faster than program A? Show your work and explain how you found your solution.