**Queens College Computer Science Department**

**Algorithms (CSCI 323 & 700)**

**Spring 2016 - Homework #4 - Solutions**

1. [5 points] If binary search is repeatedly used to find the insertion point in Insertion Sort, the total time for that would be log 1 + log 2 + log 3 + … + log (n-1) = log (n-1)! = O(n log n).
2. [5 points] A brute-force approach to solving the selection problem is to pre-sort the array using a sorting algorithm such as the ones discussed in class. Then simply return a[i]. Total time is O(1) + time for sorting.
3. [5 points] Using code similar to Selection Sort, if i < n/2, repeatedly select the next remaining minimum i times, while if i < n/2, repeatedly select the next remaining maximum i times. The number of comparisons will be (n-1) + (n-2) + … + (n-k) where k is either i for the former scenario (finding minimums) or for the first scenario, or n-i+1 for the latter scenario (fining maximums). When i is constant, this will amount to O(n); when i is a function of n, then this will amount to O(n2).
4. [5 points] To solve the selection problem, partition the array into two sets of elements, those less than a pivot value x, and those greater than x. Then, depending on the index of the pivot compared to i – the index we want to select, recursively call the algorithm either only on the left part or only the right part. The worst case time complexity is O(n2) as it becomes like a Selection Sort. The average case time complexity is actually O(n). (Consider that the solution to T(n) = T(n/2) + n, where at each iteration the problem is half the size of the previous iteration, is O(n).)
5. [10 points = 5 for explanation of recurrence, 5 for solution to recurrence] When there is only one element, no comparisons are required, so T(1) = 0. For n > 1, the recurrence is T(n) = 3(n/3) + 2n-3 since there are three sub-problems, each one-third the original size. To merge the three sublists, the first n-2 elements require 2 comparisons each as they are copied into the auxiliary array, the second to last requires one comparison, and the last requires no comparisons , for a total of 2(n-2) + 1 + 0 = 2n-3.

To solve the recurrence, Assume n = 3k and k = log3n. Use a domain transformation, S(k) = T(n) = 3T(n/3) + 2n-3 = 3S(k-1) + 2•3k – 3, S(0) = T(1) = 0. Use a range transformation, R(k) = S(k)/3k = 1/3k x (3S(k-1) + 2•3k – 3) = R(k-1) + (2•3k – 3)/3k = R(k-1) + 2 – 1/3k-1 Using telescoping – adding up the system of equations - we get R(k) – R(0) = 2k - Σ i=0..k-1 (1/3)I . R(k) = 2k – [(1/3)k – 1][1/3 – 1] = 2k – [1 – (1/3)k]/(2/3) = 2k – (3/2) [1 – (1/3)k]. So S(k) = 3k R(k) = 3k [2k – (3/2) [1 – (1/3)k]] = 3k 2k - (3/2) 3k  + (3/2) 3k 1/3)k . And T(n) = 3log n 2(log n) - (3/2) 3klog n  + (3/2) = 2nlog n - 3n/2 + 3/2 = O(n log n).

1. [5 points] The recurrence is T(n) = 3(n/3) + 2n-3, so a = 3, b = 3, and d = 1. Since log33 = 1, case 2 of the Master Theorem applies, so T(n) = O(n1 log n) = O(n log n).
2. [5 points] The recurrence is T(n) = T(n/2) + 1, so a = 1, b = 2, and d = 0. Since log21 = 0, case 2 of the Master Theorem applies, so T(n) = O(n0 log n) = O(log n).