CS 325 - Homework Assignment 2
Due Monday at 11:59pm

Problem 1: Give the asymptotic bounds for T(n) in each of the following recurrences. Make your bounds as tight as possible and justify your answers.

a) $T(n) = T(n-2) + n$

b) $T(n) = T(n-1) + 3$

c) $T(n) = 2T\left(\frac{n}{4}\right) + n$

d) $T(n) = 4T\left(\frac{n}{2}\right) + n^2\sqrt{n}$

Problem 2: Suppose you are choosing between the following three algorithms:

- *Algorithm A* solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
- *Algorithm B* solves problems of size n by recursively solving two subproblems of size (n -1) and then combining the solutions in constant time.
- *Algorithm C* solves problems of size n by dividing them into nine subproblems of size $\frac{n}{3}$, recursively solving each subproblem, and then combining solutions in O(n^2) time.

What are the running times of each of these algorithms and which would you choose?

Problem 3: How many times as a function of n (in Θ form), does the following PHP function echo "Print"? Write a recurrence and solve it.

```
function foo( $n )   {
            if ($n > 1) {
               foo($n/2);
               foo($n/2);
               foo($n/2);
               foo($n/2);
                      for ($i = 1; $i <= $n; $i += 1) {
                               echo " Print  ".$i." <br> ";
                               }
                      echo " <br>";
            }  else {
               return 1;
            }
}
```

Problem 4: The ternary search algorithm is a modification of the binary search algorithm that splits the input not into two sets of almost-equal sizes, but into three sets of sizes approximately one-third. Write pseudo-code for the ternary search algorithm, give the recurrence for the ternary search algorithm and determine the asymptotic complexity of the algorithm. Compare the worst-case running time of the ternary search algorithm to that of the binary search algorithm.

Problem 5: A k-way merge operation. Suppose you have k sotred arrays, each with n elements, and you want to combine them into a single sorted array of kn elements.

a) One strategey: Using a merge procedure, merge the first two arrays, then merge in the third array, then merge in the fourth, and so on. What is the time complexity of this algorithm in terms of k and n?
b) Design a more efficient algorithm to solve this problem using divide and conquer. You can give a higher level description of the algorithm in "words" rather than code. However you must provide enough detail to determine time complexity in terms of k and n?

CS 325 - Homework Assignment 2
Due Monday at 11:59pm

Problem 6: Insertion sort on small arrays in merge sort

Although merge sort runs in $\Theta(n\lg n)$ worst-case time and insertion sort runs in $\Theta(n^2)$ worst-case time, the constant factors in insertion sort can make it faster in practice for small problem sizes on many machines. Thus, it makes sense to coarsen the leaves of the recursion by using insertion sort with merge sort when subproblems become sufficiently small, Consider a modification to merge sort in which n/k sublists of length k are sorted using insertion sort and then merged using the standard mechanism, where k is a value to be determined.

a) Show that insertion sort can sort n/k sublists, each of length k, in $\Theta(nk)$ worst-case time.

b) Show how to merge the sublists in $\Theta(n \lg(n/k))$ worst-case time.

c) Given that the modified algorithm runs in $\Theta(nk + n\lg(n/k))$ worst-case time, what is the largest value of k as a function of n for which the modified algorithm has the same running time as the standard merge sort, in terms of Θ-notation?

d) How should we choose k in practice?

Problem 7: Design and analyze a divide and conquer algorithm that determines the minimum and maximum value in an unsorted list (array). Write pseudo-code for the min_and_max algorithm, give the recurrence and determine the asymptotic complexity of the algorithm. Compare the running time of the recursive min_and_max algorithm to that of an iterative algorithm for finding the minimum and maximum values of an array.