Helwan University
Faculty of Computers and Information
Computer Science Department

2nd Level
CS 316 Algorithms

Algorithms HW#3

1.
   a. Write pseudocode for a divide-and-conquer algorithm for finding the position of the largest element in an array of n numbers.
   b. What will be your algorithm's output for arrays with several elements of the largest value?
   c. Set up and solve a recurrence relation for the number of key comparisons made by your algorithm.
   d. How does this algorithm compare with the brute-force algorithm for this problem?

2.
   a. Write pseudocode for a divide-and-conquer algorithm for finding values of both the largest and smallest elements in an array of n numbers.
   b. Set up and solve (for $n=2^k$) a recurrence relation for the number of key comparisons made by your algorithm.
   c. How does this algorithm compare with the brute-force algorithm for this problem?

3.
   a. Write pseudocode for a divide-and-conquer algorithm for the exponentiation problem of computing a^n where n is a positive integer.
   b. Set up and solve a recurrence relation for the number of multiplications made by this algorithm.
   c. How does this algorithm compare with the brute-force algorithm for this problem?

4. Is mergesort a stable sorting algorithm?

5. Give traces, showing how the following keys are sorted with mergesort.
   a) <Q, U, E, S, T, I, O, N> b) <5,4,2,1,3,2>

6. Solve the following recurrence relations
   a. $x(n) = x(n-1) + 5$ for $n > 1$, $x(1) = 0$
   b. $x(n) = 3x(n-1)$ for $n > 1$, $x(1) = 4$
   c. $x(n) = x(n-1) + n$ for $n > 0$, $x(0) = 0$
   d. $x(n) = x(n/2) + n$ for $n > 1$, $x(1) = 1$
   e. $x(n) = x(n/3) + 1$ for $n > 1$, $x(1) = 1$

7. Consider the following recursive algorithm

   Algorithm S(n)
   //Input: A positive integer n
   //Output: The sum of the first n cubes
   if n = 1 return 1
   else return $S(n-1) + n * n * n$

   a. Set up and solve a recurrence relation for the number of times the algorithm's basic operation is executed.
   b. How does this algorithm compare with the straightforward nonrecursive algorithm for computing this function?

8. Consider the following recursive algorithm

   ALGORITHM Riddle(A[0..n − 1])
   //Input: An array A[0..n − 1] of real numbers
   if n = 1 return A[0]
   else temp← Riddle(A[0..n − 2])
   if temp ≤ A[n − 1] return temp
   else return A[n − 1]

   a. What does this algorithm compute?
   b. Set up a recurrence relation for the algorithm's basic operation count and solve it.

9. Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 3T(\lfloor n/2 \rfloor) + n$. Use the master method to verify your answer.

10. Draw the recursion tree for $T(n) = 4T(\lfloor n/2 \rfloor) + cn$, where c is a constant, and provide a tight asymptotic bound on its solution. Verify your bound by the master method.

11. Consider the recurrence $T(n) = 3T(n/4) + cn^2$, where c is a constant, Find asymptotic bound using iteration tree method.

Good Luck