CSI 3105 ASSIGNMENT 2 Fall 2014

Due Date: Wednesday, October 22, 2014, by 4:00 pm. You can hand in your assignment directly to me in class that day, or else you can bring it to my office (STE 5106) before 4:00 (put it under my door if I am not there). Assignments from 1 minute to 24 hours late will lose 6 out of 60 marks (10%). Assignments received after 4:00 pm on Oct. 23 will receive a grade of 0.

On this and all future assignments: You must be sure to include your name and student ID on the assignment, as well as the course code and the assignment number. Also, your assignment must be **STAPLED.** If you are missing any of these items, you may lose a mark. In your assignment, please be sure to explain all of your solutions CLEARLY, or you will lose marks. To be complete, in any worst case analysis you do you must state the input size, the basic operations being counted, and a general input of your fixed input size which gives the worst case (you should state these things in your solution even if some of them are given in the questions). Whenever possible, simplify your answers by using any appropriate formulas found in Appendix A. Also, you will be marked on the efficiency of any algorithms you design.

This assignment will be marked out of 60.

---

1. [10 marks total]

The maximum sum problem is as follows: Given an array X of integers, find the maximum sum found in any contiguous subarray of X. for example, for X = [31, -41, 59, 26, -53, 58, 97, -93, -23, 84] the answer is 187 (the sum of the numbers in positions 3, 4, 5, 6 and 7).

Consider the following recursive pseudocode algorithm for solving this problem for an array X of length n (of course a solution comes from a call to MaximumSum(1,n)):

```
void  MaximumSum(int L, int U)
{
 int Sum, MaxToLeft, MaxToRight, M, I, MaxCrossing, MaxInA, MaxInB;

if (L >= U) {  //base case
    if (L > U)    //Zero-element array
      return 0.0;
    if (L = =U)    //One-element array
      return max(0.0, X[L]);  //{***}
}
else {  //recursive case
  M=⌊ (L + U)/2 ⌋;  //A is X[L..M], B is X[M+1..U]

  //Find the max crossing to the left
  Sum= 0.0;  MaxToLeft= 0.0;
  for (I = M; I>=L; I--) {
       Sum:= Sum + X[I] ;   //{***}
       MaxToLeft= max(MaxToLeft, Sum)  //{***}
```

```
        }//endfor
        //Find the max crossing to the right
        Sum= 0.0;  MaxToRight= 0.0;
        for (I =M+1; I<=U; I++) {
           Sum= Sum + X[I];    //{***}
           MaxToRight = max(MaxToRight, Sum);  //{***}
        }//endfor
        Maxcrossing = MaxToLeft + MaxToRight ; //{***}
        MaxInA = MaximumSum(L, M);
        MaxInB = MaximumSum(M+1, U);
        return max(MaxCrossing, MaxInA, MaxInB);  //{***}
      }//endelse
    }
```

a) [9 marks]
Do a worst case analysis for this algorithm, assuming that n is a power of 2 (i.e. $n = 2^k$, $k>=0$). In this worst case analysis, use n as the input size, and count the number of additions involving elements of X and finding the max of a set of numbers (to be clear, I have marked these operations in the algorithm by {***}--these are exactly what you count). This analysis will be similar to that of Mergesort, but if you are getting the same recurrence relation, check again (because yours is wrong!). Be sure that in your worst case analysis you clearly do the following:

i) state the recurrence relation for W(n) for this algorithm
ii) "solve" this recurrence relation using back substitution to get an educated guess.

(NOTE: You do not need to verify your solution using induction)

b) [1 mark] State the complexity of this algorithm, based on your analysis from a).

2. [8 marks total]

a) [3 marks] Prove the following using limits and L'Hôpital's Rule: That lgn is O(sqrt(n)) (Note: sqrt(n) is square root of n).

b) [3 marks] Prove the following using limits and L'Hôpital's Rule: That $20n^2 + 20n + 7$ is $\Omega(n + 5)$.

c) [2 marks] Prove the following using the formal definition for "Big Oh" (i.e. find an appropriate c and n_o): $n^3 + 5000n^2 + 20 \in O(n^3)$.

3. [10 marks]
Do an average case analysis of Mergesort for n=3, and counting the number of comparisons of keys in S.

4. [6 marks total]

a) [3 marks] Find the exact total number of multiplications and additions and subtractions (all together) for the usual algorithm (improved) for matrix multiplication, and Strassen's algorithm for matrix multiplication for n=32 (show the formulas you used). Which is faster for this value of n?

b) [3 marks] Write the recursive formula for Strassen's algorithm, counting only multiplications, when the threshold used for the algorithm is n=2. Explain your

answer briefly.

5. [11 marks total]
a) [5 marks] Solve the following recurrence relation using back-substitution (assuming $n= 2^k$, $k>=1$):

For $n = 2$, $W(n) = 1$
For $n > 2$, $W(n) = 2W(n/2) + 4$.

b) [5 marks] Verify your solution using induction.
c) [1 mark] State the complexity of W(n).

6. [15 marks]
Suppose you are given an array S of n integers which are all > 0 and distinct, and that you are also given an integer P > 0. The numbers in S are sorted from smallest to largest. You wish to determine whether or not there are two numbers in S which, when multiplied together, give exactly the number P. Note that you cannot use one of the numbers twice, i.e. you cannot use S[i] *S[i].

Design a divide and conquer algorithm for this problem. You must explain the idea behind your algorithm clearly, and give a pseudocode description. You do not need to do a worst case analysis of your algorithm, but it should be a linear time algorithm.
(Hint: Multiply the first and last numbers in S and compare this product to P. Try to use this comparison to eliminate either the first number or the last number from consideration, and then use recursion to solve for a smaller list).