Fall 2018, CMPSC 465

Homework Assignment #2

This homework is due **9 pm** on **September 12th**.

Collaboration is permitted on this homework. If you choose to collaborate, you are allowed to discuss each problem with at most three other students currently enrolled in class. Before working with others on a problem, you should think about it yourself for at least 45 minutes. *You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem.* You must also identify your collaborators. If you do not work with anyone, you should write "Collaborators: none." It is a violation of this policy to submit a problem solution that you cannot orally explain to an instructor or a TA. *Finding answers to problems on the web or from other outside sources (includes anyone not enrolled in the class) is strictly forbidden.*

You should aim to be as clear and concise as possible in your writeup of solutions. A simple and direct analysis is worth more points than a convoluted one. Each problem is worth 10 points. Points may be deducted for illegible handwriting. Partial credit will be given only for answers that make significant progress toward correct solution.

1

Consider the following divide-and-conquer algorithm.

A is a 1-indexed input array of size $n = 2^k$.

S is the (1-indexed) output array of size n initialized to all zeros.

```
ZigZagSum(A)
  if n = 1 then
  |  S[1] ← A[1];
  |  return;
  for i = 1 to n/2 do
  |  B[i] ← A[2i − 1] + A[2i];
  Recursively compute ZigZagSum(B) and store output in C;
  for i = 1 to n do
  |  if i is even then
  |  |  S[i] ← C[i/2];
  |  if i == 1 then
  |  |  S[1] ← A[1];
  |  if i is odd and i > 1 then
  |  |  S[i] ← C[(i−1)/2] + A[i];
```

- Show the steps of this algorithm for an array with two elements $[5, 6]$, and then show the steps for an array with four elements: $[2, 3, 1, 5]$. What does this algorithm do?
- Give the recurrence expression for the running time of this algorithm.
- Solve the recurrence equation using the Master theorem. What is the running time?
- Write pseudocode for a non-recursive algorithm that would give the same output as the above recursive algorithm. Your non-recursive algorithm must have the same asymptotic bounds as the recursive approach.

2

Suppose that each row of an $n \times n$ array A consists of 1's and 0's such that, in any row of A, all the 1's come before any 0's in that row. Assuming A is already in memory, describe a method running in O(n) time (not O(n^2) time) for finding the row of A that contains the most 1's.

3

An degree-n polynomial $p(x)$ is an equation of the form

$$p(x) = \sum_{i=0}^{n} a_i x^i,$$

where x is a real number and each a_i is a constant.

- Describe a simple $O(n^2)$-time method for computing $p(x)$ for a particular value of x.
- Consider now a rewriting of $p(x)$ as $p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \cdots + x(a_{n-1} + xa_n)\cdots)))$. Using the Big-Oh notation, characterize the number of multiplications and additions this method of evaluation uses.

4

Identify the mistake in the following proof.
Claim: All students in CMPSC 465 received the same grade for HW1.
Proof: We will use induction on n, the number of students, to prove this.
Base case: If there is only one student, there is only one grade.
Inductive step: Assume as induction hypothesis that within any set of $k < n$ students, there is only one grade. Now look at any set of $k = n$ students. Number them $1, 2, 3, ..., n$. Consider the sets $\{1, 2, 3, ..., n-1\}$ and $\{2, 3, 4, ..., n\}$. Each is a set of only $n-1$ students, therefore within each there is only one unique grade. But the two sets overlap, so there must be only one distinct grade for all n students.

5

Sort the following functions according to their asymptotic growth. That is, give an ordering $g_1, g_2, ...$ such that $g_i = \text{O}(g_{i+1})$.

$$n^{\log_2 6}, \quad 70n^2, \quad n \log n, \quad \sum_{i=1}^{n} 3^i, \quad \sqrt{n},$$

Indicate which functions are asymptotically equivalent (*i.e.*, indicate when $g_i = \Theta(g_{i+1})$) by drawing a brace ($\underbrace{\text{like this}}$) under the appropriate boxes.

6

Use the iterative substitution method to solve the recurrence $T(n) = T(\sqrt{n}) + \Theta(1)$.

7

Solve the recurrence equation $T(n) = 8T(n/4) + n^3$ using the recursion tree construction method. Express your answer in the Θ-notation. In the recursion tree method, you (i) draw a recursion tree, (ii) determine its height, (iii) estimate the work associated with nodes at each level, and (iv) simplify the summation to come up with a closed-form expression for the running time. Assume that $T(1) = \Theta(1)$.