Problem 1-1. Asymptotic Growth: Part 1

Decide whether these statements are True or False. You must justify all your answers to receive full credit by either giving a short proof (1-2 sentences) or exhibiting a counter-example.

(a) \( f(n) = O(g(n)) \) and \( g(n) = o(h(n)) \) implies that \( f(n) = o(h(n)) \).

Solution: True. For any small constant \( \varepsilon \), and for all sufficiently large \( n \geq N_\varepsilon \), we have that \( g(n) \leq \varepsilon \cdot h(n) \), and that \( f(n) \leq C \cdot g(n) \), where \( C \) is an absolute constant. Thus, \( f(n) \leq C\varepsilon \cdot h(n) \) and we can make \( C\varepsilon \) as small as we want by setting \( \varepsilon \) correspondingly.

(b) \( f(n) = O(g(n)) \) and \( g(n) = \Omega(h(n)) \) implies that \( f(n) = \Theta(h(n)) \).

Solution: False. Let \( f(n) = 1, g(n) = n, \) and \( h(n) = \sqrt{n} \).

(c) \( f(n) = \omega(g(n)) \) implies there exists some \( h(n) \) such that \( f(n) = \omega(h(n)) \) and \( h(n) = \omega(g(n)) \).

Solution: True. Take \( h(n) = \sqrt{f(n)g(n)} \).
(d) It must always hold that either \( f(n) = O(g(n)) \) or \( g(n) = O(f(n)) \). What if \( f \) and \( g \) are also guaranteed to be monotonically non-decreasing?

**Solution:** False. Take \( f(n) = n \cdot ((-1)^n + 1) \) and \( g(n) = n \cdot (1 - (-1)^n) \). Even if \( f(n) \) and \( g(n) \) are monotonically non-decreasing, the statement remains false: take

\[
f(n) = \begin{cases} 
  n!, & \text{if } n \text{ is odd} \\
  (n-1)!, & \text{if } n \text{ is even}
\end{cases}
\]

and

\[
g(n) = \begin{cases} 
  (n-1)!, & \text{if } n \text{ is odd} \\
  n!, & \text{if } n \text{ is even}.
\end{cases}
\]
Problem 1-2. Asymptotic Growth: Part 2

Rank the following functions by increasing order of growth; that is, find an arrangement $g_1, g_2, \ldots, g_{14}$ of the functions satisfying $g_1 = O(g_2), g_2 = O(g_3), \ldots, g_{13} = O(g_{14})$. Partition your list into equivalence classes such that $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$. All the logs are in base 2.

$$
\sum_{i=1}^{n} \frac{i^2+3i+9}{i^3+i^2+4}, \quad (\log n)\sqrt{\log n}, \quad (n!)^{1/n}, \\
\binom{n}{n/2}, \quad \sqrt{n}, \quad n \frac{\log n}{\log n}, \quad \binom{n}{100}, \\
\sum_{i=1}^{n} i^{99}, \quad 1/n, \quad 2^n, \\
3^{\sqrt{n}}, \quad 1/5, \quad \sum_{i=1}^{n} \frac{1}{i}, \\
n \log n, \quad \log(n!).
$$

Solution: The ordering is:

\[
\begin{align*}
\frac{1}{n} & , \\
\frac{1}{5} , \\
\sum_{i=1}^{n} \frac{i^2+3i+9}{i^3+i^2+4} , \\
\binom{n}{n/2} , \\
(\log n)\sqrt{\log n} , \\
(n!)^{1/n} , \\
n \log n & , \\
\binom{n}{100} , \\
3^{\sqrt{n}} , \\
2^n , \\
\frac{n}{n/2} : \sqrt{n}.
\end{align*}
\]

Problem 1-3. Correctness of Horner’s rule

The following code fragment implements Horner’s rule for evaluating a polynomial

\[
P(x) = \sum_{k=0}^{n} a_k x^k \\
= a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + x a_n) \cdots))
\]

given the coefficients $\langle a_0, a_1, \ldots, a_n \rangle$ and a value for $x$:  

1 \ y \leftarrow 0
2 \ i \leftarrow n
3 \textbf{while } i \geq 0
4 \quad \textbf{do } y \leftarrow a_i + x \cdot y
5 \quad i \leftarrow i - 1

(a) What is the asymptotic running time of Horner’s rule? Compare this with the running time of a naïve polynomial evaluation algorithm that computes each term of the polynomial from scratch.

\textbf{Solution: } The asymptotic running time of Horner’s rule is $\Theta(n)$. Each iteration of the loop performs a constant amount of work (one multiplication, one addition, and one subtraction) and iterates $n$ times.

The following code evaluates a polynomial at $x$ by computing each term of the polynomial from scratch.

1 \ y \leftarrow 0
2 \ i \leftarrow n
3 \textbf{while } i \geq 0
4 \quad \textbf{do } z \leftarrow 1
5 \quad j \leftarrow i
6 \quad \textbf{while } j > 0
7 \quad \quad \textbf{do } z \leftarrow z \cdot x
8 \quad \quad j \leftarrow j - 1
9 \quad y \leftarrow y + a_i \cdot z
10 \quad i \leftarrow i - 1

For this algorithm, computing the term $a_k x^k$ requires $k + 1$ multiplications, $k + 1$ subtractions, and one addition. Thus evaluating the whole polynomial takes time $\sum_{k=0}^{n} 2k + 3 = 2(n(n + 1)/2) + 3(n + 1) = \Theta(n^2)$
A smarter algorithm for computing \( x^k \) by successive squaring requires only \( \Theta(\lg k) \) multiplications per term. Using this method takes \( \Theta(n \lg n) \) time to evaluate the polynomial.

**b)** Prove that if the loop invariant

\[
y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k
\]

holds at the beginning of the previous iteration of the while loop (line 3) then it holds at the beginning of the next iteration.

**Solution:** We show the inductive step of the proof of the loop invariant (by induction on number of iterations, \( l \)). The inductive hypothesis is

\[
y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k
\]

at the beginning of the \( l \)th iteration. Since it has to be proved that this condition holds after one more iteration, it also assumed that the loop is not about to terminate, that is, \( i \geq 0 \). Let \( y' \) and \( i' \) be the values of \( y \) and \( i \) at the beginning of the \((l+1)\)st iteration, respectively. We are required to show that

\[
y' = \sum_{k=0}^{n-(i'+1)} a_{k+i'+1} x^k
\]

Since \( y' = a_i + x \cdot y \) and \( i' = i - 1 \), we have

\[
y' &= a_i + x \cdot y \\
&= a_i + x \cdot \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k \quad \text{(inductive hypothesis)} \\
&= a_{i'+1} + x \cdot \sum_{k=0}^{n-((i'+1)+1)} a_{k+(i'+1)+1} x^k \\
&= a_{i'+1} + \sum_{k=0}^{n-((i'+1)+1)} a_{k+(i'+1)+1} x^{k+1} \\
&= \sum_{k=0}^{n-(i'+1)} a_{k+i'+1} x^k
\]

and so the condition holds at the beginning of the \((l+1)\)st iteration.
(c) We wish to prove the postcondition

\[ y = \sum_{k=0}^{n} a_k x^k \]  

holds upon termination of the while loop. State the weakest precondition needed at line 3 to construct such a proof. Justify your answer by exhibiting the proof. (You should assume invariant Equation (1).)

**Solution:** The precondition states what must be initially true of the variables in order for the program to execute correctly. The question asks for the weakest set of conditions that are needed at line 3 for the loop to evaluate the polynomial correctly. Thus, the precondition establishes the basis of the inductive proof of the loop invariant. The weakest precondition needed to establish the invariant is \( \text{Pre}: y = 0 \) and \( i = n \). Given that \( \text{Pre} \) is true at the beginning of the first iteration of the algorithm, the condition

\[ y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k \]

holds.

To prove the postcondition, we demonstrate that for invariant \( I \), loop guard \( B \), and postcondition \( Post \), \( I \) and \( \neg B \Rightarrow Post \). So we want to show

\[ y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k \text{ and } i < 0 \Rightarrow y = \sum_{k=0}^{n} a_k x^k. \]

To actually, complete the proof we need to strengthen the condition \( \neg B \) to \( i = -1 \). That can be done in two ways. First, we can change line 3 of the code so that the loop guard is \( i \not= -1 \). Alternatively, we can argue that since initially \( i \) is a positive integer and since \( i \) is decremented by 1 on each iteration, \( i \) will first violate the guard when \( i = -1 \). So, now we want to show

\[ y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k \text{ and } i = -1 \Rightarrow y = \sum_{k=0}^{n} a_k x^k. \]

Simple substitution of \(-1\) for \( i \) in \( I \) completes the proof.

(d) Conclude by arguing the given code fragment correctly evaluates a polynomial characterized by the coefficients \( \langle a_0, a_1, \ldots, a_n \rangle \).

**Solution:** We show that the given iterative algorithm correctly establishes the postcondition \( Post: \sum_{k=0}^{n} a_k x^k \) when executed after some code that establishes precondition \( \text{Pre}: y = 0 \) and \( i = n \). Part b provided a condition \( I: y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k \). We
proved inductively that $I$ is a loop invariant established by precondition $\text{Pre}$ in parts b (inductive case) and c (base case). In Part c we proved that $I$ and $\neg B \Rightarrow \text{Post}$, where $B$ is the loop guard $i \geq 0$. Finally, we argue that the loop terminates because $i$ starts at $n$, and is repeatedly decremented by one until $i < 0$. 