1 Asymptotic Notation (25 points)

For each of the following statements explain if it true or false and prove your answer. The base of log is 2 unless otherwise specified, and ln is log_e.

(a) \( n \tan n = O(2^n) \)

**Solution:** False. We prove this by contradiction. Suppose for contradiction that \( n \tan n = O(2^n) \). Then by definition, there are constant \( c, n_0 > 0 \) such that \( n \tan n \leq c \cdot 2^n \) for all \( n > n_0 \). Let \( n' \) be the an odd integer multiple \( \pi/2 \) that is larger than \( n_0 \), i.e., \( n' = c'(\pi/2) \) for an odd integer \( c' \) and \( n' > n_0 \). Then by the definition of the tangent function, we know that the limit of \( n \tan n \) as \( n \) approaches \( n' \) from below is \( \infty \), so for every \( x \in \mathbb{R} \) there is a small enough \( \epsilon_x > 0 \) such that \((n' - \epsilon_x)(\tan(n' - \epsilon_x)) > x \). By using \( \epsilon = \epsilon_c 2^n \) (or any smaller value of \( \epsilon \) which is still greater than 0), we get that \((n' - \epsilon) \tan(n' - \epsilon) > c \cdot 2^n \), which (since \( n' > n_0 \)) gives a contradiction. Hence \( n \tan n \) is not \( O(2^n) \).

(b) \( e^n = O(2^n) \)

**Solution:** False. For contradiction, assume that there are some constants \( c, n_0 > 0 \) such that \( e^n \leq c \cdot 2^n \) for all \( n > n_0 \). This is equivalent to \((e/2)^n \leq c \) for all \( n > n_0 \). By taking logarithms of both sides, we get that \( n \log(e/2) \leq \log c \) for all \( n > n_0 \), and hence \( n \leq \frac{\log c}{\log(e/2)} \) for all \( n > n_0 \). This is a contradiction, since \( \frac{\log c}{\log(e/2)} \) is a constant and so for large enough \( n \) we know that \( n > \frac{\log c}{\log(e/2)} \). Hence \( e^n \) is not \( O(2^n) \).

(c) \( n \log n = O(n^2) \)

**Solution:** True. Set \( c = 1 \) and \( n_0 = 1 \). Since \( \log n \leq n \) for all \( n \geq 1 \), we know that \( n \log n \leq n^2 = cn^2 \) for all \( n \geq 1 = n_0 \).

(d) \( \frac{n}{\log^2 n} = \Omega(n^{0.9}) \)
Solution: True. To show this, set \( c = 1 \). Suppose that \( \frac{n}{\log n} < n^{0.9} \). Then \( n^{0.05} < \log n \). But this is only true for \( 2.05136 < n < 1.4 \times 10^{43} \) (using Wolfram Alpha). So if we set \( n_0 = 1.4 \times 10^{43} \), we have that \( \frac{n}{\log n} \geq n^{0.9} \) for all \( n > n_0 \), as claimed.

2 Recurrences (25 pts)

Solve the following recurrences, giving your answer in \( \Theta \) notation (so prove both an upper bound and a lower bound). For each of them you may assume \( T(x) = 1 \) for \( x \leq 5 \). Show your work.

(a) \( T(n) = 6T(n/5) + n^2 \)

Solution: We can use the Master Theorem, which implies that \( T(n) = \Theta(n^2) \).

(b) \( T(n) = 3T(n - 3) \)

Solution: We can use the unrolling method:

\[
T(n) = 3T(n - 3) = 3^2T(n - 6) = 3^3T(n - 9) = \ldots
\]

This is at least \( 3^{\frac{n}{3}} - 3T(1) = 3^{\frac{n}{3}} - 3 \), and is at most \( 3^{\frac{n}{3} + 1} \). Hence \( T(n) = \Theta(3^{n/3}) \).

(c) \( T(n) = n^{1/6}T(n^{5/6}) + n \)

Solution: Draw out the recursion tree. A simple induction implies that each level has a total contribution of \( O(n) \). To bound the number of levels, note that the number of levels is the smallest \( x \) such that \( n^{(5/6)^x} \leq 5 \). By taking logs, we get that \( (5/6)^x \log n \leq \log 5 \), and hence \( (5/6)^x \leq (\log 5)/(\log n) \). This is equivalent to \( (6/5)^x \geq \log_5 n \). Again taking logs, we get that \( x \log(6/5) \geq \log \log_5 n \), and hence \( x \geq (\log \log_5 n)/(\log(6/5)) \). So the number of levels is \( \Theta(\log \log n) \), and hence \( T(n) = n \log \log n \).

3 Basic Proofs (25 pts)

(a) There are currently 131 students registered for the class. Prove that there are at least 3 students who have birthdays in the same week (assuming that there are exactly 52 weeks in each year).

Solution: This is basically the pigeonhole principle. Suppose for contradiction that that in every week there are at most 2 students who have birthdays in that week. Then the number of students is at most \( 52 \cdot 2 = 104 \), which contradicts the fact that there are 131 students. Hence there is at least one week which contains at least three birthdays.

(b) Prove by induction that \( \sum_{i=1}^{n} i^2 = \frac{1}{6} n(n + 1)(2n + 1) \)
Solution: For the base case \( n = 1 \), we have that \( \sum_{i=1}^{n} i^2 = \sum_{i=1}^{1} i^2 = 1 = \frac{1}{6}(1 \cdot 2 \cdot 3) = \frac{1}{6}n(n + 1)(2n + 1) \) as required.

For the inductive step, suppose that \( \sum_{i=1}^{n-1} i^2 = \frac{1}{6}(n-1)n(2(n-1) + 1) = \frac{1}{6}(n-1)n(2n-1) \). Then

\[
\sum_{i=1}^{n} i^2 = n^2 + \sum_{i=1}^{n-1} i^2 = n^2 + \frac{1}{6}(n-1)n(2n-1) = \frac{1}{6}(2n^3 - 3n^2 + n) = \frac{1}{6}(n(n + 1)(2n + 1))
\]

as required.

(c) Consider a polynomial \( P(x) = \sum_{k=0}^{n} a_k x^k \), and consider the following algorithm:

\[
y = 0; \\
\text{for } i = n \text{ down to } 0 \text{ do} \\
\mid y = a_i + (x \cdot y); \\
\text{end} \\
\text{return } y;
\]

Prove that this algorithm correctly computes \( P(x) \) when called on input \( x \).

Hint: Think of an appropriate “loop invariant” / “induction hypothesis” for a proof by induction.

Solution: We will prove by induction that at the end of each iteration of the for loop, \( y = \sum_{k=0}^{n-i} a_{k+i} x^k \) (where \( i \) is the value of \( i \) in the loop). If we can prove this then we are finished, since it implies that when \( i = 0 \), at the end of the loop we have that \( y = \sum_{k=0}^{n} a_k x^k \), and this is the value that is finally returned.

So to prove that at the end of each iteration \( y = \sum_{k=0}^{n-i} a_{k+i} x^k \), first consider the base case of \( i = n \). This is the very first iteration of the for loop, so at the end of the iteration we have that \( y = a_n + (x \cdot 0) = a_n a_n x^0 = \sum_{k=0}^{n-i} a_{k+i} x^k \) as claimed.

Now we do the inductive step. Suppose that the claim is true at the end of some iteration \( i + 1 \), and consider the next iteration \( i \) (note that the indices are going down since the for loop counts down). Then

\[
a_i + (x \cdot y) = a_i + x \cdot \sum_{k=0}^{n-(i+1)} a_{k+(i+1)} x^k = a_i + \sum_{k=0}^{n-i-1} a_{k+1+i} x^{k+1} \\
= a_i + \sum_{k=1}^{n-i} a_{k+i} x^k = \sum_{k=0}^{n-i} a_{k+1} x^k
\]
4 Mistakes and Insertion Sort (25 pts)

Given an array \([a_0, a_1, \ldots, a_{n-1}]\), a *mistake* is a pair \((i, j)\) such that \(i < j\) but \(a_i > a_j\). For example, in the array \([5, 3, 2, 10]\) there are three mistakes \(((0, 1), (0, 2), (1, 2))\). Note that the array has no mistakes if and only if it is sorted, so the number of mistakes can be thought of as a measure of how well-sorted an array is. For this problem, assume that all elements in an array are distinct.

(a) What is the expected number of mistakes in a random array? More formally, consider a random permutation \(\pi\) of \(n\) distinct elements \(a_0, \ldots, a_{n-1}\): what is the expected number of mistakes in the resulting array?

**Solution:** Let \(X_{ij}\) be an indicator random variable for the event that \((i, j)\) is a mistake. Since the permutation is random and all elements are distinct, we know that \(\Pr[X_{ij} = 1] = 1/2\) and \(\Pr[X_{ij} = 0] = 1/2\) and hence \(E[X_{ij}] = 1/2\). By linearity of expectations, the total expected number of mistakes is

\[
E \left[ \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} X_{ij} \right] = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} = \frac{n(n-1)}{4}
\]

(b) Recall the insertion sort algorithm:

\[
\begin{align*}
& \text{for } i = 1 \text{ to } n-1 \text{ do} \\
& \qquad j = i; \\
& \qquad \text{while } j > 0 \text{ and } A[j-1] > A[j] \text{ do} \\
& \qquad \qquad \text{Swap } A[j] \text{ and } A[j-1]; \\
& \qquad \qquad j = j - 1; \\
& \text{end}
\end{align*}
\]

Suppose that our array has \(d\) mistakes. Prove a lower bound on the running time of insertion sort in terms of \(d\) (asymptotic notation OK).

**Solution:** Each iteration of the while loop decreases the number of mistakes by exactly 1. Hence the running time must be at least \(\Omega(d)\).

(c) Prove an upper bound on the running time of insertion sort in terms of \(n\) and \(d\) (asymptotic notation OK).

**Solution:** Each iteration of the while loop decreases the number of mistakes by exactly 1, and hence the while loop is executed precisely \(d\) times. Each iteration of the while loop takes only \(O(1)\) time, and hence the total time spent inside the while loop is \(O(d)\). There is an additional \(O(n)\) time spend outside of the while loop, and hence the total running time is \(O(n + d)\).