CS-3410 **Algorithms** **Spring 2026**

Homework 1 **Brandon Aikman**

1. Problem 1-1 (p15)

Table 1: Comparison of Running Times

$f(n)$	**1 second**	**1 hour**
$\log n$	$2^{1,000,000}$	$2^{3,600,000,000}$
$\sqrt{n}$	$1 \cdot 10^{12}$	$1.296 \cdot 10^{19}$
n	$1 \cdot 10^{6}$	$3.6 \cdot 10^{9}$
$n \log n$	62,746	$1.334 \cdot 10^{8}$
n^2	1,000	60,000
n^3	100	1,532
2^n	20	31
$n!$	9	12

2. Exercise 2.3-4 (p44)

   Prove that when $n \geq 2$ is an exact power of 2, the solution of the recurrence

$$T(n) = \begin{cases} 2 & n = 2, \\ 2T(n/2) + n & n > 2 \end{cases}$$

is $T(n) = n \log n$. Show: $T(n) \leq cn \log n$

Assume: $T(n/2) \leq cn/2 \log n/2$

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &\leq 2(cn/2 \log n/2) + n \\ &= cn \log n/2 + n \\ &= cn \log n - cn \log 2 + n \\ &= cn \log n + (1 - c)n \\ &\leq cn \log n \text{ for } c \geq 1 \end{aligned}$$

Base Case:

$T(2) = 2$

$T(4) = 2(2) + 4 = 8 \equiv 4 \log 4 = 8$

Thus, we see that for any $n \geq 2$ where n is an exact power of 2, the recurrence is $T(n) = n \log n$.

3. Problem 2-3 (p46)

   (a) $\Theta(n)$

   (b)
```
naiveHorner(A,n,x)
  p = 0
  for i = 0 to n
    item = A[i]
    for j = 1 downto i
      item *= x
    p += item
  return p
```
   The running time of `naiveHorner` is $\Theta(n^2)$, and is beat by `Horner` asymptotically.

   (c) Show that, at termination, $p = \sum_{k=0}^{n} A[k] \cdot x^k$.
   At the start of each iteration of the `for` loop,

$$p = \sum_{k=0}^{n-(i+1)} A[k+i+1] \cdot x^k.$$

   Initialization:
   We start by showing that the loop invariant holds before the first loop iteration, when $i = n$. This yields the summation from $k = 0$ to -1, which is empty. Therefore, $p = 0$ initially, which shows that the loop invariant holds prior to the first iteration of the loop.

   Maintenance:
   Next, we see that the `for` loop works by adding subsequent terms from A, multiplied by higher order terms of x. When $i = n-1$, $p = A[n]$. Then, when $i = n-2$, $p = A[n-1] + A[n]x$. Thus, we see that decrementing i for the next iteration of the `for` loop then preserves the loop invariant.

   Termination:
   The loop variable i starts at n and decreases by 1 in each iteration. Once $i = 0$, the loop terminates. Substituing $i = 0$ in the summation, we clearly see that $p = A[0] + A[1]x + \ldots + A[n-1]x^{n-1} + A[n]x^n$. Hence, the algorithm correctly implements Horner's rule to evaluate p.

4. Exercise 3.2-2 (p62)

The statement *"The running time of algorithm A is at least $O(n^2)$"* **is meaningless, because $O(n^2)$ defines an upper bound on the asymptotic behavior of the algorithm, so it will never have a time complexity greater than $O(n^2)$. Asserting that something will be 'at least' the highest possible value it can hold means nothing.**

5. Exercise 3.2-6 (p63)
   Prove that $o(g(n)) \cap \omega(g(n))$ is the empty set.

   o-notation denotes an asymptotically loose upper bound, formally defined as the set

   $o(g(n)) = \{f(n):$ for any positive constant $c > 0,$ there exists a constant n_0 such that $0 \leq f(n) < cg(n)$ for all $n \geq n_0.\}$

   Additionally, ω-notation denotes an asymptotically loose lower bound, formally defined as the set

   $\omega(g(n)) = \{f(n):$ for any positive constant $c > 0,$ there exists a constant n_0 such that $0 \leq cg(n) < f(n)$ for all $n \geq n_0.\}$

   So, we clearly see that the intersection of these two sets is the empty set, because $f(n)$ cannot be both exclusively less than $cg(n)$ and exclusively greater than $cg(n)$ simultaneously.

6. Using the substitution method, show that the solution of $T(n) = T(\lceil n/2 \rceil) + 1$ is $O(\log n)$.
   Show: $T(n)$ is $O(\log n) \rightarrow T(n) \leq c \log n$

$$\forall n, \lceil n/2 \rceil \leq \frac{n+1}{2}$$
$$\log(n+1) < \log n + \tfrac{1}{2} \text{ for } n > 1$$

   Assume: $T(\frac{n+1}{2}) \leq c \log \frac{n+1}{2}$

$$\begin{aligned}
T(n) &= T(\lceil n/2 \rceil) + 1 \\
&\leq T(\tfrac{n+1}{2}) + 1 \\
&\leq c \log \tfrac{n+1}{2} + c \\
&= c \log \tfrac{n+1}{2} + c \log 2 \\
&= c \log(n+1) \\
&= c \log n + \tfrac{c}{2} \text{ for } c \geq 2
\end{aligned}$$

   Picking $c = \max\{T(2), T(3)\}$ yields $T(2) \leq c < (\log 2 + 1/2)c$ and $T(3) \leq c < (\log 3 + 1/2)c$, establishing the inductive hypothesis for the base cases.

   Thus, we have $T(n) \leq c \log n + \frac{c}{2}$ for all $n \geq 2$, which implies that the solution to the recurrence is $T(n) = O(\log n)$.

7. Exercise 4.5-1 (a, b, d, e) (p106

   (a) $T(n) = 2T(n/4) + 1$
   Choosing $\epsilon = 1 > 0$, $f(n) = O(n^{\log_4 2 - 1})$.
   So, $T(n) = \Theta(\sqrt{n})$.

   (b) $T(n) = 2T(n/4) + \sqrt{n}$
   Because $f(n) = \sqrt{n} = \Theta(\sqrt{n})$, $T(n) = \Theta(\sqrt{n} \log n)$.

   (c) N/A

   (d) $T(n) = 2T(n/4) + n$
   Choosing $\epsilon = 2$, $f(n) = \Omega(n)$.
   Additionally, for $c = 1/2$ and $\forall n$, $cf(n) \geq af(n/b) \equiv n/2 \geq n/2$.
   So, $T(n) = \Theta(n)$.

   (e) $T(n) = 2T(n/4) + n^2$
   Choosing $\epsilon = 12$, $f(n) = \Omega(n^2)$.
   Additionally, for $c = 1/2$ and $\forall n$, $cf(n) \geq af(n/b) \equiv n^2/2 \geq n^2/2$.
   So, $T(n) = \Theta(n^2)$.

8. Exercise 4.5-2 (p106)

   Given that Strassen's algorithm follows $T(n) = 7T(n/2) + \Theta(n^2)$ and that Caesar's algorithm follows $T(n) = aT(n/4) + \Theta(n^2)$, we must find a such that $n^{log_4 a} < n^{\log_2 7}$. This is true for $a \leq 48$, so $a = 48$ is the largest integer for which his algorithm could run faster than Strassen's.

9. Exercise 5.2-1 (p133)
   The probability of hiring exactly one time is $\frac{1}{n}$ (this occurs on the best-case scenario of a forward-sorted list). The probability of hiring exactly n times is $\frac{1}{n!}$ (this occurs on the worst-case scenario of a reverse-sorted list).

10. Exercise 5.2-2 (p133)
    The probability of hiring exactly twice is the Harmonic sum over n: $\frac{H_{n-1}}{n}$. This evaluates to $\frac{\ln n + \gamma}{n}$, which we can rewrite to $\Theta(\frac{\log n}{n})$.