Kaleb Kim

9/21/24

CSDS 310 Assignment 2

Note: Arrays are zero-indexed.

Problem 1

a) $\log^k n = o(n^\varepsilon)$, proven by the Limit Asymptotic Theorem:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\log^k n}{n^\varepsilon} = \lim_{n\to\infty} \left(\frac{\log n}{n^{\varepsilon/k}}\right)^k \text{, applying L'Hopital's :}$$

$$\overset{\star}{=} \lim_{n\to\infty} \left(\frac{1/n}{\varepsilon n^{\varepsilon/k-1}}\right)^k$$

$$= \left(\frac{0}{\varepsilon(\infty)^{\varepsilon/k-1}}\right)^k$$

$$= 0$$

This means that $g(n)$ is the asymptotic upper bound of $f(n)$. As this result does not depend on constants k, ε, this is a strict bound—which is also part of the Limit Asymptotic Theorem.

b) $n^k = o(c^n)$, proven by the Limit Asymptotic Theorem again:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^k}{c^n}$$

$$\overset{\star}{=} \lim_{n\to\infty} \frac{kn^{k-1}}{c^n}$$

$$\overset{\star}{=} \lim_{n\to\infty} \frac{k(k-1)n^{k-2}}{c^n}$$

As the denominator remains fixed, this pattern will continue until:

$$\lim_{n\to\infty} \frac{k(k-1)(...)(2)(1)n^1}{c^n} \overset{\star}{=} \lim_{n\to\infty} \frac{k(k-1)(...)(2)}{c^n}$$

$$= 0$$

By the same reason as part a), this result proves that $g(n)$ is the asymptotic strict upper bound of $f(n)$.

c) There is no asymptotic relation between the two. If we were to prove this through the closest definition of Big-O notation, stating there exists some constant $a > 0$ that satisfies the following:

$$0 \le f(n) \le ag(n)$$

$$0 \le \left(\sqrt{n}\right) \le a\left(n^{\sin(n)}\right)$$

$$0 \le n^{\frac{1}{2}} \le an^{\sin(n)}$$

$$0 \le \ln\left(n^{\frac{1}{2}}\right) \le \ln\left(an^{\sin(n)}\right)$$

$$0 \le \frac{1}{2} \le a\sin(n)$$

Regardless of what we choose for a, $\sin(n)$ can be 0 for sufficiently large inputs for n, making the inequality false. If we were to try other notation definitions, there will still be no bound.

Problem 2

a) True. By the definition of Big O, with $a \in \mathbb{R}$ such that $a > 0$:

$$0 \le f(n) \le ag(n)$$
$$0 \le f(n) + c \le af(n)$$
$$0 \le 1 \le \frac{af(n)}{f(n) + c}$$

Using a limit to evaluate a sufficiently large n:

$$\lim_{n \to \infty} \frac{af(n)}{f(n) + c} \overset{\star}{=} \lim_{n \to \infty} \frac{af'(n)}{f'(n)}$$
$$= \lim_{n \to \infty} a$$
$$= a$$

Since there exists a such that $a \ge 1$ to make the inequality will hold true, this proves the original statement.

b) False. For $f(n) = 2^n$, by the definition of Theta Notation $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$, with c_1, c_2 are positive and n is sufficiently large. We have $g(n) = f(2n) = 2^{(2n)} = (2^n)^2$.
Substituting:

$$0 \le c_1(2^n)^2 \le 2^n \le c_2(2^n)^2$$
$$0 \le c_1 2^n \le 1 \le c_2 2^n$$

No value of c_1, c_2 can fix this inequality the way I can fix her for any sufficiently large n. Thus, the statement is false.

c) True. Let $g(n) = f(2n) = (2n)^c = 2^c n^c$. By the definition of Big O, with $a \in \mathbb{R}$ such that $a > 0$:

$$0 \le f(n) \le ag(n)$$
$$0 \le n^c \le a(2^c n^c)$$
$$0 \le 1 \le a \cdot 2^c$$

Suppose $a = 1$ and given that $c > 0$, for any sufficiently large value of n the inequality holds true. Thus, there exists a constant which holds the statement true.

Problem 3

a) Converting $T(n)$ to Master Theorem form, $T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^d)$. This falls under Case 2, thus $T(n) = \Theta(n \log n)$

b) yeah

Problem 4

Pseudocode:

```
1  procedure MERGE(A, n):
2  |  if n_A == 1:
3  |  |  return A
4  |  A' ← Array of ⌈n/2⌉ elements
```

```
5  | for 0 ≤ i < ⌊n/2⌋:
6  |    | A'[i] ← SUBMERGE(A[2i], A[2i+1])
7  | if n is odd:
8  |    | A'[n/2] ← A[n − 1] // Put last element in A'
9  | return MERGE(A')
```

The merging of two subarrays, cleverly named:

```
 1  procedure SUBMERGE(A, B):
 2  | i ← 0
 3  | j ← 0
 4  | C ← Array of n_A + n_B elements
 5  | while i + j < len(A) + len(B):
 6  |    | if i == len(A):
 7  |    |    | C[i+j] ← B[j]
 8  |    |    | j ← j + 1
 9  |    | else if j == len(B) or A[i] < B[j]:
10  |    |    | C[i+j] ← A[i]
11  |    |    | i ← i + 1
12  |    | else:
13  |    |    | C[i+j] ← B[j]
14  |    |    | j ← j + 1
15  | endwhile
16  | return C
```

Proof:

Runtime analysis: We can apply the Master Theorem since we have a recursive case in which the merging is linear,