Problem 1 (Asymptotics)

Part 1(a)

We order the given functions as follows:

$$\frac{1}{n} \ll n^{1/\log n} \ll (\log n)^2 \ll (\log(n \cdot \log n))^2 \ll \log(n!) \ll n \log n$$

Explanations: 1. $\frac{1}{n} = O(n^{1/\log n})$: - $\frac{1}{n}$ decays to 0 as $n \to \infty$, while $n^{1/\log n}$ is constant and does not decay. Thus, $\frac{1}{n}$ grows slower.

2. $n^{1/\log n} = O((\log n)^2)$: - $n^{1/\log n} = e$, which is a constant. Any unbounded logarithmic function like $(\log n)^2$ grows faster than a constant and will increase past the constant eventually.

3. $(\log n)^2 = O((\log(n \cdot \log n))^2)$: - $(\log(n \cdot \log n))^2 = (\log n + \log\log n)^2$. The argument of the outer logarithm in each function is the only difference. Specifically, the argument of $\log(n \cdot \log n)$ is $n \cdot \log n$, which is greater than n. Since $n \cdot \log n$ grows faster than n, we have $(\log(n \cdot \log n))^2$ growing faster than $(\log n)^2$.

4. $(\log(n \cdot \log n))^2 = O(\log(n!))$: - $\log(n!) \sim n \log n - n$. The factorial argument of the log in each function will dominate the other, but both are still dominated by polynomial growth.

5. $\log(n!) = O(n \log n)$: All polynomial growth dominates logarithmic growth.

Part 2

Problem Statement: For some given positive-valued functions $f(n)$, $g(n)$, $h(n)$, with n being a positive integer:

—

(a) If $f(n) = O(g(n))$, then $f(n) = O(\log(g(n)))$ Answer: False

Counterexample: Let $g(n) = n$ and $f(n) = n$. Then $f(n) = O(g(n))$, because $f(n) \leq c \cdot g(n)$ for some constant $c > 0$.

However, $\log(g(n)) = \log(n)$. The conclusion that $f(n) = O(\log(g(n)))$ would imply $n = O(\log(n))$, which is incorrect because n grows faster than $\log(n)$. This invalidates the statement.

—

(b) If $f(n) = \Theta(g(n))$, then $f(n)^2 = \Theta(g(n)^3)$ Answer: False

If both functions are equal to n, then the statement incorrectly concludes that $n^2 = \Theta(n^3)$.

—

(c) If $f(n) = \Omega(n \cdot g(n))$ and $g(n) = \Omega(n \cdot h(n))$, then $f(n) = \Omega(n^2 \cdot h(n))$ Answer: True

Proof: The information given implies that f(n) has the same long term growth rate as g(n), and the same holds for g(n) and h(n). Transitively, this would mean that f(n) has the same growth rate as h(n) no matter the constant n, as constants are dropped when comparing long term growth rate. Thus the statement is true.

article amsmath

Problem 2 (Asymptotics)

Arrange the following functions in ascending order of growth rate. You do not need to formally prove anything, however, provide a single sentence reasoning for the placement of each adjacent pair in the ordering.

$$\begin{aligned} g_{01}(n) &= n^{\frac{101}{100}}, & g_{02}(n) &= n^{2^{n+1}}, \\ g_{03}(n) &= n(\log n)^3, & g_{04}(n) &= n^{\log\log n}, \\ g_{05}(n) &= \log(n^{2^n}), & g_{06}(n) &= n!, \\ g_{07}(n) &= 2^{\sqrt{\log n}}, & g_{08}(n) &= 2^{2^{n+1}}, \\ g_{09}(n) &= \log(n!), & g_{10}(n) &= \lfloor \log(n) \rfloor !, \\ g_{11}(n) &= 2^{\log\sqrt{n}}, & g_{12}(n) &= \sqrt{2}^{\log n}. \end{aligned}$$

$$\begin{aligned} g_{01}(n) &= n^{\frac{101}{100}}, & g_{02}(n) &= n^{2^{n+1}}, \\ g_{03}(n) &= n(\log n)^3, & g_{04}(n) &= n^{\log\log n}, \\ g_{05}(n) &= \log(n^{2^n}), & g_{06}(n) &= n!, \\ g_{07}(n) &= 2^{\sqrt{\log n}}, & g_{08}(n) &= 2^{2^{n+1}}, \\ g_{09}(n) &= \log(n!), & g_{10}(n) &= \lfloor \log(n) \rfloor !, \\ g_{11}(n) &= 2^{\log\sqrt{n}}, & g_{12}(n) &= \sqrt{2}^{\log n}. \end{aligned}$$

Hierarchy of Growth Rates

Based on the corrected computations, the functions are ordered from slowest to fastest growth as follows:

$$g_{11}(n), g_{12}(n) < g_{07}(n) < g_{05}(n) < g_{03}(n) < g_{01}(n) < g_{09}(n), g_{10}(n) < g_{04}(n) < g_{06}(n) < g_{02}(n) < g_{08}(n)$$

1. $g_{11}(n), g_{12}(n)$: Both simplify to $2^{\log\sqrt{n}}$, which is the smallest growth rate in this set.

2. $g_{12}(n) < g_{07}(n)$: Exponential growth in $g_{07}(n) = 2^{\sqrt{\log n}}$ dominates the slower exponential growth of $g_{12}(n)$.

3. $g_{07}(n) < g_{05}(n)$: The growth rate of $g_{05}(n) = \log(n^{2^n})$, which simplifies to $2^n \log n$, is faster than $g_{07}(n)$ due to the inclusion of 2^n.

4. $g_{05}(n) < g_{03}(n)$: Polynomial growth in $g_{03}(n) = n(\log n)^3$ dominates the logarithmic-exponential growth in $g_{05}(n)$.

5. $g_{03}(n) < g_{01}(n)$: Slightly superlinear growth in $g_{01}(n) = n^{101/100}$ outpaces the linear-logarithmic combination in $g_{03}(n)$.

6. $g_{01}(n) < g_{09}(n)$: Factorial growth in $g_{09}(n) = \log(n!)$ eventually dominates the polynomial growth in $g_{01}(n)$.

7. $g_{09}(n) < g_{10}(n)$: The factorial growth in $g_{10}(n) = \lfloor \log(n) \rfloor!$ outpaces the logarithmic-factorial growth in $g_{09}(n)$.

8. $g_{10}(n) < g_{04}(n)$: Exponential-logarithmic growth in $g_{04}(n) = n^{\log \log n}$ dominates the factorial terms in $g_{10}(n)$ for large n.

9. $g_{04}(n) < g_{06}(n)$: Pure factorial growth in $g_{06}(n) = n!$ eventually dominates $g_{04}(n)$.

10. $g_{06}(n) < g_{02}(n)$: Exponential growth in $g_{02}(n) = n^{2^{n+1}}$ outpaces the factorial growth of $g_{06}(n)$.

11. $g_{02}(n) < g_{08}(n)$: Double exponential growth in $g_{08}(n) = 2^{2^{n+1}}$ dominates the single exponential growth of $g_{02}(n)$.

Problem 3 (Mysteriousness)

You encounter the mysterious piece of pseudocode shown in **Algorithm 1**. Answer the following questions based on this pseudocode.

Algorithm 1: Mystery Function, $H(a, m)$

Input: Integers a, m.
Output: Unknown integer value dependent on inputs a and m.

```
H(a, m):
    if m = 0
        return a  1
    else
        c ← a  1
        for i from 1 to m
            c ← c · a
        return H(a, m  1) + c
```

Part (a)

- $H(a, 1) = a^2 - 1$, 1 recursive call $(H(a, 0))$.
- $H(a, 2) = a^3 - 1$, 2 recursive calls. $(H(a, 1))$, $(H(a, 0))$
- $H(a, 3) = a^4 - 1$, 3 recursive calls. $(H(a, 2))$, $(H(a, 1))$, $(H(a, 0))$

Part (b)

We claim that:

$$H(a, m) = a^{m+1} - 1$$

for $m \geq 0$.

Proof by Induction

Base Case:

$$H(a, 0) = a - 1.$$

Holds with formula:

$$a^{0+1} - 1 = a - 1.$$

Inductive Step: Assume that the formula holds for $m = k$

$$H(a, k) = a^{k+1} - 1.$$

Need to prove that the formula holds for $m = k + 1$:

$$H(a, k + 1) = a^{(k+1)+1} - 1 = a^{k+2} - 1.$$

From the pseudocode, for $m = k + 1$:

$$H(a, k + 1) = H(a, k) + c,$$

where $c = a - 1$ and is multiplied by a $k + 1$ times:

$$c = (a - 1) \cdot a^{k+1}.$$

Using substitution with the inductive hypothesis we can obtain: $H(a, k) = a^{k+1} - 1$, we get:

$$H(a, k + 1) = (a^{k+1} - 1) + a^{k+1}.$$

Simplify:

$$H(a, k + 1) = a^{k+2} - 1.$$

Thus, the formula holds for $m = k + 1$. Additionally, by induction the formula

$$H(a, m) = a^{m+1} - 1$$

holds for all $m \geq 0$.

Part (c)

The runtime of the function $H(a, m)$ is $\Theta(m^2)$.

This is because of the recursive calls that go m levels deep, then lead to the loop of m times per recursive call, which result in the function runtime of $H(a, m)$ being $\Theta(m^2)$.

Problem 4 (Understanding Recurrences)

Based on the pseudocode given in Algorithm 2, answer the following questions. For simplicity, you may assume that n is always a power of 2.

1. Write a recurrence $R(n)$ that describes the number of times `RecFunc(n)` prints "Hi". Your recurrence should include base cases.

2. Solve the recurrence for $R(n)$ in asymptotic terms. That is, your answer should be in the form $R(n) = \Theta(g(n))$ for some function $g(n)$ that is as simple as possible. You may solve your recurrence using any method you like as long as the relevant work is shown.

Algorithm 2: Recursive Function, `RecFunc(n)`

Input: Integer n.
Output: Nothing, but many statements are printed.

```
RecFunc(n):
    if n = 1
        print "Hi"
    else
        RecFunc(n/2)
        RecFunc(n/2)
        RecFunc(n/2)
        RecFunc(n/2)
        for i = 1, ..., 3n
            print "Hi"
```

Part (a)

The recurrence relation is given by:

$$R(n) = 4R\left(\frac{n}{2}\right) + 3n \quad \text{for } n > 1,$$

with the base case:

$$R(1) = 1.$$

Part (b)

We solve the recurrence:

$$R(n) = 4R\left(\frac{n}{2}\right) + 3n, \quad R(1) = 1.$$

Substitute $R\left(\frac{n}{2}\right)$:

$$R(n) = 4\left[4R\left(\frac{n}{4}\right) + 3\frac{n}{2}\right] + 3n = 16R\left(\frac{n}{4}\right) + 6n + 3n.$$

Substitute $R\left(\frac{n}{4}\right)$:

$$R(n) = 16\left[4R\left(\frac{n}{8}\right) + 3\frac{n}{4}\right] + 6n + 3n = 64R\left(\frac{n}{8}\right) + 12n + 6n + 3n.$$

After k steps, the recurrence becomes:

$$R(n) = 4^k R\left(\frac{n}{2^k}\right) + 3n\sum_{i=0}^{k-1} 2^i.$$

Base Case

When the recursive steps have reached the end represented by when $k = \log_2 n$ the argument of R reaches the base case:

$$R\left(\frac{n}{2^k}\right) = R(1) = 1.$$

Overall this results in:

$$R(n) = 4^{\log_2 n} R(1) + 3n \sum_{i=0}^{\log_2 n - 1} 2^i.$$

$4^{\log_2 n}$: Using the property $a^{\log_b c} = c^{\log_b a}$, we get:

$$4^{\log_2 n} = n^{\log_2 4} = n^2.$$

$\sum_{i=0}^{\log_2 n-1} 2^i$ is a geometric series with ratio 2:

$$\sum_{i=0}^{\log_2 n-1} 2^i = 2^{\log_2 n} - 1 = n - 1.$$

Substituting into $R(n)$, we get:

$$R(n) = n^2 + 3n(n-1).$$

$$R(n) = n^2 + 3n^2 - 3n = 4n^2 - 3n.$$

Asymptotic Complexity

Because of the rules of runtime, the constant will be dropped resulting in

$$R(n) = \Theta(n^2).$$

Problem 5 (Programming Assignment)

username: noahichen