Problem 1. Prove \( \log_2(n!) = \Theta(n \log n) \).

Solution. Let us prove first \( \log_2(n!) = O(n \log n) \):

\[
\log_2(n!) = \log_2(\prod_{i=1}^{n} i) \\
\leq \log_2 n^n \\
= n \log_2 n \\
= O(n \log n).
\]

Now we prove \( \log_2(n!) = \Omega(n \log n) \):

\[
\log_2(n!) = \log_2(\prod_{i=1}^{n} i) \\
\geq \log_2(\prod_{i=n/2}^{n} i) \\
\geq \log_2((n/2)^{n/2}) \\
= (n/2) \log_2(n/2) \\
= \Omega(n \log n).
\]

This completes the proof.

Problem 2. Let \( f(n) \) be a function of positive integer \( n \). We know:

\[
\begin{align*}
f(1) &= 1 \\
f(n) &\leq 2 + f(\lceil n/10 \rceil).
\end{align*}
\]

Prove \( f(n) = O(\log n) \). Recall that \( \lceil x \rceil \) is the ceiling operator that returns the smallest integer at least \( x \).

If necessary, you can use without a proof the fact that \( f(n) \) is monotone, namely, \( f(n_1) \leq f(n_2) \) for any \( n_1 < n_2 \).

Solution 1 (Expansion). Consider first \( n \) being a power of 10.

\[
\begin{align*}
f(n) &\leq 2 + f(n/10) \\
&\leq 2 + 2 + f(n/10^2) \\
&\leq 2 + 2 + 2 + f(n/10^3) \\
&\cdots \\
&\leq 2 \cdot \log_{10} n + f(1) \\
&= 2 \cdot \log_{10} n + 1 = O(\log n).
\end{align*}
\]

Now consider \( n \) that is not a power of 10. Let \( n' \) be the smallest power of 10 that is greater
than $n$. We have:

\[
\begin{align*}
f(n) & \leq f(n') \\
& \leq 2 \log_{10} n' + 1 \\
& \leq 2 \log_{10}(10n) + 1 \\
& \leq O(\log n).
\end{align*}
\]

**Solution 2 (Master Theorem).** Let $\alpha, \beta,$ and $\gamma$ be as defined in the Master Theorem (see the tutorial slides of Week 4). Thus, we have $\alpha = 1, \beta = 10,$ and $\gamma = 0.$ Since $\log_{10} 1 = 0 = \gamma,$ by the Master Theorem, we know that $f(n) = O(n^\gamma \log n) = O(\log n)$.

**Solution 3 (Substitution).** We aim to prove that, when $n \geq c_1$, $f(n) \leq c_2 \log_2 n$ for some constants $c_1, c_2$ to be determined later.

- For the base case, we need:
  \[
  f(c_1) \leq c_2 \log_2 c_1 \\
  \Rightarrow c_2 \geq \frac{f(c_1)}{c_1}.
  \]

- For the inductive case, fix an integer $k > c_1$. Assume that this is correct for all $c_1 \leq n < k$. Our goal is to find $c$ to make the claim hold also for $n = k$.

\[
\begin{align*}
f(n) & \leq 2 + f(\lceil n/10 \rceil) \\
& \leq 2 + c_2 \log_2 \lceil n/10 \rceil
\end{align*}
\]

We will consider only $n \geq c_1 \geq 3$ so that $\lceil n/10 \rceil \leq (n/10) + 1 \leq n/2$. With this, we continue the above derivation as follows:

\[
\begin{align*}
f(n) & \leq 2 + c_2 \log_2 (n/2) = 2 + c_2 \log_2 n - c_2.
\end{align*}
\]

To make the above at most $c_2 \log_2 n$, it suffices to set $c_2 \geq 2$.

To satisfy all the above, it suffices to set $c_1 = 3$, and $c_2 \geq \max\{2, \frac{1}{c_1} f(c_1)\} = \max\{2, \frac{1}{3} f(3)\}$.

**Problem 3.** Let $f(n)$ be a function of positive integer $n$. We know:

\[
\begin{align*}
f(1) &= 1 \\
f(n) &\leq 2 + f(\lceil 3n/10 \rceil).
\end{align*}
\]

Prove $f(n) = O(\log n)$. Recall that $\lceil x \rceil$ is the ceiling operator that returns the smallest integer at least $x$. 

---

2
Solution 1 (Expansion).

\[ f(n) \leq 2 + f(n_1) \quad \text{(define } n_1 = \lceil (3/10)n \rceil) \]
\[ f(n) \leq 2 + 2 + f(n_2) \quad \text{(define } n_2 = \lceil (3/10)n_1 \rceil) \]
\[ f(n) \leq 2 + 2 + 2 + f(n_3) \quad \text{(define } n_3 = \lceil (3/10)n_2 \rceil) \]
\[ \ldots \]
\[ f(n) \leq 2 + 2 + \ldots + 2 + f(n_h) \quad \text{(define } n_h = \lceil (3/10)n_{h-1} \rceil) \]
\[ = 2h + f(n_h). \]  

(1)

So it remains to analyze the value of \( h \) that makes \( n_h \) small enough. Note that we do not need to solve the precise value of \( h \); it suffices to prove an upper bound for \( h \). For this purpose, we reason as follows. First, notice that

\[ \lceil 3n/10 \rceil \leq (4n/10) \]  
when \( n \geq 10 \) (prove this yourself).

Let us set \( h \) to be the smallest integer such that \( n_h < 10 \) (this implies that \( n_{h-1} \geq 10 \) and \( n_h \geq (4/10)n_{h-1} \geq 4 \)). We have:

\[ n_1 \leq (4/10)n \]
\[ n_2 = \lceil (3/10)n_1 \rceil \leq (4/10)n_1 \leq (4/10)^2n \]
\[ n_3 \leq (4/10)^3n \]
\[ \ldots \]
\[ n_h \leq (4/10)^h n \]

Therefore, the value of \( h \) cannot exceed \( \log_{4/10} n \) (otherwise, \((4/10)^4 \cdot n < 1\), making \( n_h < 1\), which contradicts the fact that \( n_h \geq 4 \)). Plugging this into (1) gives:

\[ f(n) \leq 2 \log_{4/10} n + f(10) = O(\log n). \]  
(think: why?)

Solution 2 (Master Theorem). Let \( \alpha, \beta, \) and \( \gamma \) be as defined in the Master Theorem. Thus, we have \( \alpha = 1, \beta = 10/3, \) and \( \gamma = 0 \). Since \( \log_{10} \alpha = \log_{10} 1 = 0 = \gamma \), by the Master Theorem, we know that \( f(n) = O(n^\gamma \log n) = O(\log n) \).

Solution 3 (Substitution). We aim to prove that, when \( n \geq c_1 \), \( f(n) \leq c_2 \log_2 n \) for some constants \( c_1, c_2 \) to be determined later.

- For the base case, we need:
  \[ f(c_1) \leq c_2 \log_2 c_1 \]
  \[ \Rightarrow c_2 \geq \frac{f(c_1)}{c_1}. \]

- For the inductive case, fix an integer \( k > c_1 \). Assume that this is correct for all \( c_1 \leq n < k \). Our goal is to find \( c \) to make the claim hold also for \( n = k \).
\[
\begin{align*}
  f(n) & \leq 2 + f(\lceil 3n/10 \rceil) \\
  & \leq 2 + c_2 \log_2 \lceil n/10 \rceil
\end{align*}
\]

We will consider only \( n \geq c_1 \geq 5 \) so that \( \lceil 3n/10 \rceil \leq (3n/10) + 1 \leq n/2 \). With this, we continue the above derivation as follows:

\[
f(n) \leq 2 + c_2 \log_2(n/2) = 2 + c_2 \log_2 n - c_2.
\]

To make the above at most \( c_2 \log_2 n \), it suffices to set \( c_2 \geq 2 \).

To satisfy all the above, it suffices to set \( c_1 = 5 \), and \( c_2 \geq \max\{2, \frac{1}{c_1} f(c_1)\} = \max\{2, \frac{1}{5} f(5)\} \).

**Problem 4.** Let \( f(n) \) be a function of positive integer \( n \). We know:

\[
\begin{align*}
  f(1) &= 1 \\
  f(n) &\leq 2n + 4f(\lceil n/4 \rceil).
\end{align*}
\]

Prove \( f(n) = O(n \log n) \).

**Solution 1 (Expansion).** Consider first \( n \) being a power of 4.

\[
\begin{align*}
  f(n) &\leq 2n + 4f(n/4) \\
  &\leq 2n + 4(2n/4 + 4f(n/4^2)) \\
  &\leq 2n + 2n + 4^2 f(n/4^2) \\
  &= 2 \cdot 2n + 4^2 f(n/4^2) \\
  &\leq 2 \cdot 2n + 4^2 \cdot (2(n/4^2) + 4f(n/4^3)) \\
  &= 3 \cdot 2n + 4^3 f(n/4^3) \\
  &\vdots \\
  &= (\log_4 n) \cdot 2n + n \cdot f(1) \\
  &= (\log_4 n) \cdot 2n + n = O(n \log n).
\end{align*}
\]

Now consider that \( n \) is not a power of 4. Let \( n' \) be the smallest power of 4 that is greater than \( n \). This implies that \( n' < 4n \). We have:

\[
\begin{align*}
  f(n) &\leq f(n') \\
  &\leq (\log_4 n') \cdot 2n' + n' \\
  &< (\log_4(4n)) \cdot 8n + 4n = O(n \log n).
\end{align*}
\]

**Solution 2 (Master Theorem).** Let \( \alpha, \beta, \) and \( \gamma \) be as defined in the Master Theorem. Thus, we have \( \alpha = 4, \beta = 4, \) and \( \gamma = 1 \). Since \( \log_\beta \alpha = \log_4 4 = 1 = \gamma \), by the Master Theorem, we know that \( f(n) = O(n^\gamma \log n) = O(n \log n) \).

**Solution 3 (Substitution).** We aim to prove that, when \( n \geq c_1 \), \( f(n) \leq c_2 \log_2 n \) for some constants \( c_1, c_2 \) to be determined later.
• For the base case, we need:

\[ f(c_1) \leq c_2 \log_2 c_1 \]
\[ \Rightarrow c_2 \geq \frac{f(c_1)}{c_1}. \]

• For the inductive case, fix an integer \( k > c_1 \). Assume that this is correct for all \( c_1 \leq n < k \).

Our goal is to find \( c \) to make the claim hold also for \( n = k \).

\[ f(n) \leq 2n + 4c_2[n/4] \log_2[n/4] \]
\[ \leq 2n + 4c_2(n/4 + 1) \log_2(n/4 + 1). \]

We will consider only \( n \geq c_1 \geq 5 \) so that \( n/4 + 1 \leq n/2 \). With this, we continue the above derivation as follows:

\[ f(n) \leq 2n + 4c_2(n/4 + 1) \log_2(n/2) \]
\[ = 2n + (c_2n + 4c_2)(\log_2 n - 1) \]
\[ \leq 2n + (c_2n + 4c_2) \log_2 n - c_2n \]
\[ \leq 2n + c_2n \log_2 n + 4c_2 \log_2 n - c_2n \]

To make the above smaller than or equal to \( c_2n \log_2 n \), it suffices to make sure:

\[ 2n + 4c_2 \log_2 n \leq c_2 n \]

We will consider only \( n \geq c_1 \geq 2^8 \) so that \( \log_2 n \leq n/8 \). To make sure the above, it suffices to guarantee:

\[ 2n + 4c_2(n/8) \leq c_2 n \]
\[ \Leftrightarrow 2n + c_2n/2 \leq c_2 n \]
\[ \Leftrightarrow 2n \leq c_2n/2 \]
\[ \Leftrightarrow 4 \leq c_2. \]

To satisfy all the above, it suffices to set \( c_1 = 2^8 \), and \( c_2 \geq \max\{4, \frac{1}{c_1} f(c_1)\} = \max\{4, \frac{1}{2^8} f(2^8)\} \).

**Problem 5 (Bubble Sort).** Let us re-visit the sorting problem. Recall that, in this problem, we are given an array \( A \) of \( n \) integers, and need to re-arrange them in ascending order. Consider the following bubble sort algorithm:

1. If \( n = 1 \), nothing to sort; return.

2. Otherwise, do the following in ascending order of \( i \in [1, n - 1] \): if \( A[i] > A[i + 1] \), swap the integers in \( A[i] \) and \( A[i + 1] \).


Prove that the algorithm terminates in \( O(n^2) \) time.

As an example, support that \( A \) contains the sequence of integers \((10, 15, 8, 29, 13)\). After Step 2 has been executed once, array \( A \) becomes \((10, 8, 15, 13, 29)\).
**Solution 1.** Notice that Step 2 is executed $n - 1$ times in total. At its $j$-th ($1 \leq j \leq n - 1$) execution, it incurs at most $c \cdot j$ time for some constant $c > 0$. Hence, its worst-case time is no more than

$$
c \sum_{j=1}^{n-1} j = cn(n-1)/2 < cn^2 = O(n^2).
$$

**Solution 2.** Let $f(n)$ be the worst-case running time of bubble sort when the array has $n$ elements. From the base case (Step 1), we know:

$$
f(1) \leq c_1
$$

for some constant $c_1$. From the inductive case (Steps 2-3), we know:

$$
f(n) \leq c_2 n + f(n-1)
$$

for some constant $c_2$. Solving the recurrence (by the expansion method) gives $f(n) = O(n^2)$.

**Problem 6* (Modified Merge Sort).** Let us consider a variant of the merge sort algorithm for sorting an array $A$ of $n$ elements (we will use the notation $A[i..j]$ to represent the part of the array from $A[i]$ to $A[j]$):

- If $n = 1$ then return immediately.
- Otherwise set $k = \lceil n/3 \rceil$.
- Recursively sort $A[1..k]$ and $A[k+1..n]$, respectively.
- Merge $A[1..k]$ and $A[k+1..n]$ into one sorted array.

Prove that this algorithm runs in $O(n \log n)$ time.

**Solution.** Let $f(n)$ be the worst case time of the algorithm on an array of size $n$. We have: the following recurrence:

$$
f(1) \leq \alpha
$$

$$
f(n) \leq f(\lceil n/3 \rceil) + f(\lceil 2n/3 \rceil) + \beta \cdot n
$$

where $\alpha > 0$ and $\beta > 0$ are constants. Next we will prove that $f(n) = O(n \log n)$ using the substitution method. To simplify discussion, let us get rid of $\alpha$ by defining: $g(n) = f(n) - \alpha$. We thus have:

$$
g(1) \leq 0
$$

$$
g(n) \leq g(\lceil n/3 \rceil) + g(\lceil 2n/3 \rceil) + \alpha + \beta \cdot n
$$

$$
\leq g(\lceil n/3 \rceil) + g(\lceil 2n/3 \rceil) + (\alpha + \beta) \cdot n
$$

We will prove instead that $g(n) = O(n \log n)$ which will imply that $g(n) = O(n \log n)$.

To further simplify discussion, let us define $h(n) = \frac{1}{\alpha + \beta} \cdot g(n)$. Hence, we have

$$
h(1) \leq 0
$$

$$
h(n) \leq h(\lceil n/3 \rceil) + h(\lceil 2n/3 \rceil) + n
$$

(3)

(4)
We will prove that $h(n) = O(n \log n)$ which will imply that $g(n) = O(n \log n)$.

Assume that $h(n) \leq cn \log_2 n$ for some constant $c > 0$. It is easy to verify that this is true for $h(1), h(2), ..., h(32)$ as long as $c$ is greater than a certain constant, say, $\beta$.

Suppose that $h(n) \leq cn \log_2 n$ for all $n \leq k - 1$ and an arbitrary integer $k > 32$. Next, we will work out the condition for this to hold also on $n = k$ as well. From (4), we have:

$$h(k) \leq h(\lceil k/3 \rceil) + h(\lceil 2k/3 \rceil) + k$$

$$\leq c\lceil k/3 \rceil \log_2 \lceil k/3 \rceil + c\lceil 2k/3 \rceil \log_2 \lceil 2k/3 \rceil + k$$

$$\leq c(1 + k/3) \log_2 (1 + k/3) + c(1 + 2k/3) \log_2 (1 + 2k/3) + k$$  \hspace{1cm} (5)

For $k > 32$, it always holds that $1 + k/3 \leq k/2$ and $1 + 2k/3 \leq k$. Hence we have from (5):

$$h(k) \leq c(1 + k/3) \log_2 (k/2) + c(1 + 2k/3) \log_2 k + k$$

$$= c(1 + k/3) ((\log_2 k) - 1) + c(1 + 2k/3) \log_2 k + k$$

$$= ck \log_2 k + c \log_2 k - ck/3 + c \log_2 k + k$$

$$\leq ck \log_2 k + 2c \log_2 k - ck/3 + k$$

We want the above to be no greater than $ck \log_2 k$ for our argument to work. This is true as long as

$$2c \log_2 k - ck/3 + k \leq 0$$

$$\iff 2c \log_2 k \leq (c/3 - 1)k.$$

The above holds for any $k > 32$ as long as $c \geq 48$.

We can therefore set $c = \max\{48, \beta\}$, and assert that $h(n) \leq cn \log_2 n$. 

7