CSCI2100: Regular Exercise Set 3

Prepared by Yufei Tao

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

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

$$\begin{aligned}\log_2(n!) &= \log_2(\Pi_{i=1}^{n} i) \\ &\le \log_2 n^n \\ &= n \log_2 n \\ &= O(n \log n).\end{aligned}$$

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

$$\begin{aligned}\log_2(n!) &= \log_2(\Pi_{i=1}^{n} i) \\ &\ge \log_2(\Pi_{i=n/2}^{n} i) \\ &\ge \log_2(n/2)^{n/2} \\ &= (n/2)\log_2(n/2) \\ &= \Omega(n \log n).\end{aligned}$$

This completes the proof.

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

$$\begin{aligned}f(1) &= 1 \\ f(n) &\le 2 + f(\lceil n/10 \rceil).\end{aligned}$$

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, $f(n_1) \le f(n_2)$ for any $n_1 < n_2$.

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

$$\begin{aligned}f(n) &\le 2 + f(n/10) \\ &\le 2 + 2 + f(n/10^2) \\ &\le 2 + 2 + 2 + f(n/10^3) \\ &\ldots \\ &\le 2 \cdot \log_{10} n + f(1) \\ &= 2 \cdot \log_{10} n + 1 = O(\log n).\end{aligned}$$

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:

$$\begin{aligned}
f(n) &\leq f(n') \\
&\leq 2\log_{10} n' + 1 \\
&\leq 2\log_{10}(10n) + 1 \\
&\leq O(\log n).
\end{aligned}$$

Solution 2 (Master Theorem). Let α, β, and γ 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 $\log_\beta \alpha = \log_{10} 1 = 0 = \gamma$, 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:

$$\begin{aligned}
f(c_1) &\leq c_2 \log_2 c_1 \\
\Rightarrow c_2 &\geq \frac{f(c_1)}{c_1}.
\end{aligned}$$

- 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 $n = k$.

$$\begin{aligned}
f(n) &\leq 2 + f(\lceil n/10 \rceil) \\
&\leq 2 + c_2 \log_2 \lceil n/10 \rceil
\end{aligned}$$

  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:

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

  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 $c_1 = 3$, 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:

$$\begin{aligned}
f(1) &= 1 \\
f(n) &\leq 2 + f(\lceil 3n/10 \rceil).
\end{aligned}$$

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

Solution 1 (Expansion).

$$
\begin{aligned}
f(n) &\leq 2 + f(n_1) \quad (\text{define } n_1 = \lceil (3/10)n \rceil) \\
f(n) &\leq 2 + 2 + f(n_2) \quad (\text{define } n_2 = \lceil (3/10)n_1 \rceil) \\
f(n) &\leq 2 + 2 + 2 + f(n_3) \quad (\text{define } n_3 = \lceil (3/10)n_2 \rceil) \\
&\ldots \\
f(n) &\leq \underbrace{2 + 2 + \ldots + 2}_{h \text{ terms}} + f(n_h) \quad (\text{define } n_h = \lceil (3/10)n_{h-1} \rceil) \\
&= 2h + f(n_h). \qquad (1)
\end{aligned}
$$

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

$$\lceil 3n/10 \rceil \leq (4n/10) \qquad (2)$$

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:

$$
\begin{aligned}
n_1 &\leq (4/10)n \\
n_2 &= \lceil (3/10)n_1 \rceil \leq (4/10)n_1 \leq (4/10)^2 n \\
n_3 &\leq (4/10)^3 n \\
&\ldots \\
n_h &\leq (4/10)^h n
\end{aligned}
$$

Therefore, the value of h cannot exceed $\log_{\frac{10}{4}} n$ (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:

$$f(n) \leq 2\log_{\frac{10}{4}} n + f(10) = O(\log n). \quad (\text{think: why?})$$

Solution 2 (Master Theorem). Let α, β, and γ be as defined in the Master Theorem. Thus, we have $\alpha = 1, \beta = 10/3$, and $\gamma = 0$. Since $\log_\beta \alpha = \log_{10/3} 1 = 0 = \gamma$, 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:

$$
\begin{aligned}
f(c_1) &\leq c_2 \log_2 c_1 \\
\Rightarrow c_2 &\geq \frac{f(c_1)}{c_1}.
\end{aligned}
$$

- 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 $n = k$.

$$\begin{aligned} f(n) &\le 2 + f(\lceil 3n/10 \rceil) \\ &\le 2 + c_2 \log_2 \lceil n/10 \rceil \end{aligned}$$

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

$$f(n) \le 2 + c_2 \log_2(n/2) = 2 + c_2 \log_2 n - c_2.$$

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

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

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

$$\begin{aligned} f(1) &= 1 \\ f(n) &\le 2n + 4f(\lceil n/4 \rceil). \end{aligned}$$

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

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

$$\begin{aligned} f(n) &\le 2n + 4f(n/4) \\ &\le 2n + 4(2n/4 + 4f(n/4^2)) \\ &\le 2n + 2n + 4^2 f(n/4^2) \\ &= 2 \cdot 2n + 4^2 f(n/4^2) \\ &\le 2 \cdot 2n + 4^2 \cdot (2(n/4^2) + 4f(n/4^3)) \\ &= 3 \cdot 2n + 4^3 f(n/4^3) \\ &... \\ &= (\log_4 n) \cdot 2n + n \cdot f(1) \\ &= (\log_4 n) \cdot 2n + n = O(n \log n). \end{aligned}$$

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:

$$\begin{aligned} f(n) &\le f(n') \\ &\le (\log_4 n') \cdot 2n' + n' \\ &< (\log_4 (4n)) \cdot 8n + 4n = O(n \log n). \end{aligned}$$

Solution 2 (Master Theorem). Let α, β, and γ 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 $n \ge c_1$, $f(n) \le c_2 \log_2 n$ for some constants c_1, c_2 to be determined later.

- For the base case, we need:

$$\begin{aligned} f(c_1) &\leq c_2 \log_2 c_1 \\ \Rightarrow c_2 &\geq \frac{f(c_1)}{c_1}. \end{aligned}$$

- 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 $n = k$.

$$\begin{aligned} f(n) &\leq 2n + 4c_2 \lceil n/4 \rceil \log_2 \lceil n/4 \rceil \\ &\leq 2n + 4c_2(n/4+1)\log_2(n/4+1). \end{aligned}$$

  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:

$$\begin{aligned} f(n) &\leq 2n + 4c_2(n/4+1)\log_2(n/2) \\ &= 2n + (c_2 n + 4c_2)(\log_2 n - 1) \\ &\leq 2n + (c_2 n + 4c_2)\log_2 n - c_2 n \\ &\leq 2n + c_2 n \log_2 n + 4c_2 \log_2 n - c_2 n \end{aligned}$$

  To make the above smaller than or equal to $c_2 n \log_2 n$, 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:

$$\begin{aligned} 2n + 4c_2(n/8) &\leq c_2 n \\ \Leftrightarrow 2n + c_2 n/2 &\leq c_2 n \\ \Leftrightarrow 2n &\leq c_2 n/2 \\ \Leftrightarrow 4 &\leq c_2. \end{aligned}$$

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 $A[i+1]$.
3. Recur in the part of the array from $A[1]$ to $A[n-1]$.

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 ($1 \le j \le n-1$) execution, it incurs at most $c \cdot j$ time for some constant $c > 0$. Hence, its worst-case time is no more than

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

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:

$$f(1) \quad \le \quad c_1$$

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

$$f(n) \quad \le \quad c_2 n + f(n-1)$$

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:

$$\begin{aligned} f(1) &\le \alpha \\ f(n) &\le f(\lceil n/3 \rceil) + f(\lceil 2n/3 \rceil) + \beta \cdot n \end{aligned}$$

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 α by defining: $g(n) = f(n) - \alpha$. We thus have:

$$\begin{aligned} g(1) &\le 0 \\ g(n) &\le g(\lceil n/3 \rceil) + g(\lceil 2n/3 \rceil) + \alpha + \beta \cdot n \\ &\le g(\lceil n/3 \rceil) + g(\lceil 2n/3 \rceil) + (\alpha + \beta) \cdot n \end{aligned}$$

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

$$h(1) \quad \le \quad 0 \tag{3}$$

$$h(n) \quad \le \quad h(\lceil n/3 \rceil) + h(\lceil 2n/3 \rceil) + n \tag{4}$$

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, β.

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:

$$\begin{aligned} h(k) &\leq h(\lceil k/3 \rceil) + h(\lceil 2k/3 \rceil) + k \\ &\leq c\lceil k/3 \rceil \log_2 \lceil k/3 \rceil + c\lceil 2k/3 \rceil \log_2 \lceil 2k/3 \rceil + k \\ &\leq c(1 + k/3) \log_2 (1 + k/3) + c(1 + 2k/3) \log_2 (1 + 2k/3) + k \end{aligned} \tag{5}$$

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

$$\begin{aligned} h(k) &\leq c(1 + k/3) \log_2 (k/2) + c(1 + 2k/3) \log_2 k + k \\ &= c(1 + k/3)((\log_2 k) - 1) + c(1 + 2k/3) \log_2 k + k \\ &= ck \log_2 k + c \log_2 k - c - ck/3 + c \log_2 k + k \\ &\leq ck \log_2 k + 2c \log_2 k - ck/3 + k \end{aligned}$$

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

$$\begin{aligned} 2c \log_2 k - ck/3 + k &\leq 0 \\ \Leftrightarrow \qquad 2c \log_2 k &\leq (c/3 - 1)k. \end{aligned}$$

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

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