

# Problem Set 4 Solution

Hyungmo Gu

April 8, 2020

## Question 1

- a. **Statement:**  $\forall f, g : \mathbb{N} \rightarrow \mathbb{R}^+, b \in \mathbb{R}^+, (g(n) \in \Theta(f(n))) \wedge (n_0 \in \mathbb{N}, n \geq n_0 \Rightarrow f(n) \geq b \wedge g(n) \geq b) \wedge (b > 1) \Rightarrow \log_b(g(n)) \in \Theta(\log_b(f(n)))$

**Statement Expanded:**  $\forall f, g : \mathbb{N} \rightarrow \mathbb{R}^+, b \in \mathbb{R}^+, \left( \exists c_1, c_2, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \right) \wedge \left( \exists n_1 \in \mathbb{N}, n \geq n_1 \Rightarrow f(n) \geq b \wedge g(n) \geq b \right) \wedge (b > 1) \Rightarrow \left( \exists d_1, d_2, n_2 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_2 \Rightarrow d_1 \cdot \log_b(g(n)) \leq \log_b(f(n)) \leq d_2 \cdot \log_b(g(n)) \right)$

*Proof.* Let  $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$ , and  $b \in \mathbb{R}^+$ . Assume  $c_1 = 1$ ,  $c_2 = b$ , and  $n_0 = 1$ , and  $n \in \mathbb{N}$  such that  $n \geq n_0$  and  $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ . Assume  $f(n)$  and  $g(n)$  are eventually  $\geq b$ . Assume  $b > 1$ . Let  $d_1 = 1$ ,  $d_2 = 2$ , and  $n_2 = n_0$ . Assume  $n \geq n_2$ .

We need to show  $d_1 \cdot \log_b g(n) \leq \log_b f(n) \leq d_2 \cdot \log_b g(n)$ .

We will do so in two parts. One for  $(d_1 \cdot \log_b g(n) \leq \log_b f(n))$  and the other for  $(\log_b f(n) \leq d_2 \cdot \log_b g(n))$ .

**Part 1**  $(d_1 \cdot \log_b g(n) \leq \log_b f(n))$ :

The assumption tell us

$$c_1 \cdot g(n) \leq f(n) \tag{1}$$

Then, it follows from the fact  $\forall x, y \in \mathbb{R}^+, x \geq y \Leftrightarrow \log x \geq \log y$

$$\log(c_1 \cdot g(n)) \leq \log(f(n)) \tag{2}$$

Then, using the fact  $b > 1$ , we can calculate

$$\frac{\log(c_1 \cdot g(n))}{\log b} \leq \frac{\log(f(n))}{\log b} \quad (3)$$

$$\frac{\log(c_1) + \log(g(n))}{\log b} \leq \frac{\log(f(n))}{\log b} \quad (4)$$

Then,

$$\frac{\log(g(n))}{\log b} \leq \frac{\log(f(n))}{\log b} \quad (5)$$

by the fact  $c_1 = 1$  and  $\log c_1 = 0$ .

Then, since  $\frac{\log f(x)}{\log b} = \log_b f(x)$ ,

$$\log_b(g(n)) \leq \log_b(f(n)) \quad (6)$$

Then, because we know  $d_1 = 1$ , we can conclude

$$\log_b(g(n)) \leq d_1 \cdot \log_b(f(n)) \quad (7)$$

**Part 2** ( $\log_b f(n) \leq d_2 \cdot \log_b g(n)$ ):

The assumption tells us

$$f(n) \leq c_2 \cdot g(n) \quad (8)$$

Then, it follows from the fact  $\forall x, y \in \mathbb{R}^+, x \geq y \Leftrightarrow \log x \geq \log y$

$$\log(f(n)) \leq \log(c_2 \cdot g(n)) \quad (9)$$

Then, using the fact  $b > 1$ , we can calculate

$$\frac{\log(f(n))}{\log b} \leq \frac{\log(c_2 \cdot g(n))}{\log b} \quad (10)$$

$$\frac{\log(f(n))}{\log b} \leq \frac{\log(c_2) + \log(g(n))}{\log b} \quad (11)$$

Then, since  $c_2 = b$ ,

$$\frac{\log(f(n))}{\log b} \leq \frac{\log(b) + \log(g(n))}{\log b} \quad (12)$$

Then, using the fact  $g(n)$  is eventually  $\geq b$ , we can write

$$\frac{\log(f(n))}{\log b} \leq \frac{\log(g(n)) + \log(g(n))}{\log b} \quad (13)$$

$$\frac{\log(f(n))}{\log b} \leq \frac{2 \cdot \log(g(n))}{\log b} \quad (14)$$

Then, since  $\frac{\log f(x)}{\log b} = \log_b f(x)$ ,

$$\log_b(f(n)) \leq 2 \cdot \log_b(g(n)) \quad (15)$$

Then, because we know  $d_2 = 2$ , we can conclude

$$\log_b(f(n)) \leq d_2 \cdot \log_b(g(n)) \quad (16)$$

□

**Notes:**

- $\forall x, y \in \mathbb{R}^+, x \geq y \Leftrightarrow \log x \geq \log y$
- $\exists c_1, c_2, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$
- **Definition of Eventually:**  $\exists n_0 \in \mathbb{N}, n \geq n_0 \Rightarrow P$ , where  $P : \mathbb{N} \rightarrow \{\text{True}, \text{False}\}$

b. *Proof.* Let  $k \in \mathbb{N}$ .

First, we will analyze the cost of loop 2 over iteration of loop 1.

The code tells us loop 2 starts at  $j_k = 1$  with  $j_k$  increasing by a factor of 3 per iteration until  $j_k \geq i$ .

Using these facts, we can calculate that the terminating condition occurs when

$$3^k \geq i \tag{1}$$

$$k \geq \log_3 i \tag{2}$$

Because we know the number of iterations is the smallest value of  $k$  satisfying the above inequality, we can conclude loop 2 has

$$\lceil \log_3 i \rceil \tag{3}$$

iterations.

Next, we need to determine the total number of iterations of loop 2 over all iterations of loop 1.

The code tells us loop 1 starts at  $i = 1$  and ends at  $i = n$  with each  $i$  increasing by 1 per iteration.

Using these facts, we can conclude loop 2 has total of

$$\lceil \log_3 1 \rceil + \lceil \log_3 2 \rceil + \cdots + \lceil \log_3 n \rceil = \sum_{i=1}^n \lceil \log_3 i \rceil \tag{4}$$

iterations. □

c. After scratching head and looking at solution many times, I realized that there are many things I do not yet understand, and it's the best to write what I have and learn from the solution. Here is my best attempt :).

*Proof.* Let  $n \in \mathbb{N}$ .

The previous answer tells us the exact cost of the algorithm is

$$\sum_{i=1}^n \lceil \log_3 i \rceil \quad (1)$$

Then, it follows by changing the variable  $i$  to  $i' = \log_3 i$  we can write

$$\sum_{i'=0}^{\lceil \log_3 n \rceil} i' \quad (2)$$

Then, because we know  $\sum_{i=0}^n i = \frac{n(n+1)}{2}$ , we can conclude

$$\sum_{i'=0}^{\lceil \log_3 n \rceil} i' = \frac{(\lceil \log_3 n \rceil)(\lceil \log_3 n \rceil + 1)}{2} \quad (3)$$

$$= \frac{\lceil \log_3 n \rceil^2 + \lceil \log_3 n \rceil}{2} \quad (4)$$

Then, we can conclude the runtime of the algorithm is  $\Theta(\log_3^2 n)$ . □

### Correct Solution:

We need to determine  $\Theta$  of the algorithm.

We will prove that the  $\Theta$  of the algorithm is  $\Theta(n \log n)$ .

The answer to previous question tells us the total exact cost of the algorithm is

$$\sum_{i=1}^n \lceil \log_3 i \rceil \quad (5)$$

Then, by using fact 1  $\forall x \in \mathbb{R}, x \leq \lceil x \rceil \leq x + 1$ , we can calculate

$$\sum_{i=1}^n \log_3 i \leq \sum_{i=1}^n \lceil \log_3 i \rceil \leq \sum_{i=1}^n (\log_3 i + 1) \quad (6)$$

$$\sum_{i=1}^n \log_3 i \leq \sum_{i=1}^n \lceil \log_3 i \rceil \leq \left( \sum_{i=1}^n \log_3 i + \sum_{i=1}^n 1 \right) \quad (7)$$

$$\sum_{i=1}^n \log_3 i \leq \sum_{i=1}^n \lceil \log_3 i \rceil \leq \sum_{i=1}^n \log_3 i + n \quad (8)$$

Then,

$$\log_3 \left( \prod_{i=1}^n i \right) \leq \sum_{i=1}^n \lceil \log_3 i \rceil \leq \log_3 \left( \prod_{i=1}^n i \right) + n \quad (9)$$

$$\log_3(n!) \leq \sum_{i=1}^n \lceil \log_3 i \rceil \leq \log_3(n!) + n \quad (10)$$

by the fact  $\forall a, b \in \mathbb{R}^+, \log(a) + \log(b) = \log(ab)$ .

Then,

$$\frac{\ln n!}{\ln 3} \leq \sum_{i=1}^n \lceil \log_3 i \rceil \leq \frac{\ln(n!)}{\ln 3} + n \quad (11)$$

by changing the base to  $e$  using the formula  $\log_3 n! = \frac{\log_e n!}{\log_e 3} = \frac{\ln n!}{\ln 3}$ .

Now, the fact 2 tells us  $n! \in \Theta(e^{n \ln n - n + \frac{1}{2} \ln n})$ .

Because we know from fact 3 that  $n \ln n - n + \frac{1}{2} \ln n$  is eventually  $\geq 1$ , we can conclude  $e^{n \ln n - n + \frac{1}{2} \ln n}$  is eventually  $\geq e$ .

Since  $n!$  is also eventually  $\geq e$ , by using solution to problem 1.a with  $g(n) = n!$  and  $f(n) = e^{n \ln n - n + \frac{1}{2} \ln n}$  and  $b = e$ , we can write

$$\ln(n!) \in \Theta(\ln(e^{n \ln n - n + \frac{1}{2} \ln n})) \quad (12)$$

$$\ln(n!) \in \Theta(n \ln n - n + \frac{1}{2} \ln n) \quad (13)$$

Then,

$$\ln(n!) \in \Theta(n \ln n) \quad (14)$$

by the fact  $n \ln n - n + \frac{1}{2} \ln n \in \Theta(n \ln n)$ .

So, since the algorithm runs at least  $\frac{\ln n!}{\ln 3}$ , we can conclude it has asymptotic lower bound of  $\Omega(n \ln n)$ , and since the algorithm runs at most  $\frac{\ln n!}{\ln 3} + n$ , we can conclude it has upper bound running time of  $\mathcal{O}(n \ln n)$ .

Since the value of  $\Omega$  and  $\mathcal{O}$  are the same, we can conclude the algorithm has running time of  $\Theta(n \ln n)$  or  $\Theta(n \log n)$ .

#### Notes:

- In a main flow of proof, when there is a huge interruption like showing  $\ln(n!) \in \Theta(n \ln n)$ , how can a sentence be started to tell the audience we are working on another major idea?
- When an interruption in proof has been occurred for another major part of a proof, how can a sentence be started to combine parts together?
- How can a sentence be written to say condition  $x_1$ ,  $x_2$ , and  $x_3$  are satisfied, so a statement  $y$  can be used to an equation or an idea?

## Question 2

a. We need to evaluate tight asymptotic upper bound.

We will prove that the tight asymptotic upper bound of the algorithm is  $\mathcal{O}(n^2)$ .

First, we need to analyze the number of iterations of loop 2 per iteration of loop 1.

The code tells us loop 2 starts at  $j = 0$  and ends at most  $j = i - 1$  with  $j$  increasing by 1 per iteration.

Then, using these facts, we can conclude loop 2 has at most

$$\left\lceil \frac{i - 1 - 0 + 1}{1} \right\rceil = i \quad (1)$$

iterations.

Next, we need to determine the total number of iterations of loop 2 over all iterations of loop 1.

The code tells us that loop 1 starts at  $i = n$  and ends at most  $i = 0$  with  $i$  decreasing by 1 per iteration.

Because we know each iteration of loop 1 takes  $i$  iterations by loop 2, using these facts, we can conclude the total number of iterations of loop 2 is at most

$$n + (n - 1) + (n - 2) + \cdots + 0 = \sum_{i=1}^n \quad (2)$$

$$= \frac{n(n + 1)}{2} \quad (3)$$

iterations, or  $\mathcal{O}(n^2)$ .

### **Correct Solution:**

We need to evaluate tight asymptotic upper bound.

We will prove that the tight asymptotic upper bound of the algorithm is  $\mathcal{O}(n^2)$ .

First, we need to analyze the cost of loop 2.

The code tells us loop 2 starts at  $j = 0$  and ends at most  $j = i - 1$  with  $j$  increasing by 1 per iteration.

Then, since each iteration of loop 2 takes a constant step (1 step), using these facts, we can conclude the cost of loop 2 is at most

$$1 \cdot (i - 1 - 0 + 1) = i \quad (1)$$

steps.



Next, we need to determine cost of loop 1.

The code tells us that loop 1 starts at  $i = n$  and ends at most  $i = 0$  with  $i$  decreasing by 1 per iteration.

Because we know each iteration of loop 1 takes  $i + 1$  steps (where  $i$  is from loop 2 and 1 from line 8), using these facts, we can conclude the total cost of loop 1 is at most

$$(n + 1) + n + (n - 1) + (n - 2) + \cdots + 1 = \sum_{i=0}^n (i + 1) \quad (2)$$

$$= \sum_{i=0}^n i + \sum_{i=0}^n 1 \quad (3)$$

$$= \sum_{i=0}^n i + (n + 1) \quad (4)$$

$$= \frac{n(n + 1)}{2} + (n + 1) \quad (5)$$

$$= \frac{(n + 1)(n + 2)}{2} \quad (6)$$

steps.

Finally, adding the cost of line 6, we can conclude the algorithm has total cost of  $\frac{(n+1)(n+2)}{2} + 1$  steps, which is  $\mathcal{O}(n^2)$ .

#### Notes:

- Noticed professor writes proof that gets to a point (i.e. ... where each iteration takes  $i + 1$  **steps**), and provides more detailed explanation in brackets (i.e. ... where each iteration takes  $i + 1$  steps (**Adding the cost of loop 2 and 1 step for other constant time operations**)).
- Noticed professor uses 'finally' when proof has reached the final step that leads to its conclusion.

b. Let  $n, k \in \mathbb{N}$ , and  $list = [0, 0, \dots, 0, 1, 0, \dots, 0]$  where 1 is at  $\lceil \frac{n}{2} \rceil$  position.

We will prove that the tight asymptotic lower bound running time of this algorithm is  $\Omega(n^2)$ .

First, we need to evaluate the cost of loop 2.

The code tells us loop 2 starts at  $j_k = 0$ , and  $j_k$  will increase by 1 until  $j_k \geq \lceil \frac{n}{2} \rceil + 1$  (where  $+1$  is because of loop 2 stopping at  $j_k = \lceil \frac{n}{2} \rceil$  by the if condition on line 10).

Using the fact  $j_k = k + 1$ , we can calculate that loop 2 stops when

$$k + 1 \geq \lceil \frac{n}{2} \rceil + 1 \quad (1)$$

$$k \geq \lceil \frac{n}{2} \rceil \quad (2)$$

Since we are looking for the smallest value of  $k$  (because the smallest value of  $k$  translates to number of iterations), we can conclude the loop has

$$\lceil \frac{n}{2} \rceil \quad (3)$$

iterations.

Since each iteration takes a constant time (1 step), the cost of loop 2 is

$$\lceil \frac{n}{2} \rceil \cdot 1 = \lceil \frac{n}{2} \rceil \quad (4)$$

steps.

Next, we need to evaluate the cost of loop 1.

The code tell us loop 1 will start at  $i_k = n$ , and  $i_k$  will decrease by 1 per iteration until  $i_k \leq \lceil \frac{n}{2} \rceil$ .

Using the fact  $i_k = k - 1$ , we can write loop 1 stops when

$$k - 1 \leq \lceil \frac{n}{2} \rceil \quad (5)$$

$$k \leq \lceil \frac{n}{2} \rceil + 1 \quad (6)$$

Since we are looking for the largest value of  $k$  (because the largest value of  $k$  translates to number of iterations), we can conclude loop 1 has

$$\lceil \frac{n}{2} \rceil + 1 \quad (7)$$

iterations.

Since each costs  $\lceil \frac{n}{2} \rceil + 1$  steps, we can conclude loop 1 has cost of

$$\left(\left\lceil \frac{n}{2} \right\rceil + 1\right) \left(\left\lceil \frac{n}{2} \right\rceil + 1\right) = \left\lceil \frac{n}{2} \right\rceil^2 + 2 \cdot \left\lceil \frac{n}{2} \right\rceil + 1 \quad (8)$$

steps.

Finally, by adding the cost of line 6 (1 step), the total running time of this algorithm is

$$\left\lceil \frac{n}{2} \right\rceil^2 + 2 \cdot \left\lceil \frac{n}{2} \right\rceil + 2 \quad (9)$$

steps, which is  $\Omega(n^2)$

### **Correct Solution:**

Let  $n, k \in \mathbb{N}$ , and  $list = [0, 0, \dots, 0, 1, 0, \dots, 0]$  where 1 is at  $\lceil \frac{n}{2} \rceil$  position.

We will prove that the tight asymptotic lower bound running time of this algorithm is  $\Omega(n^2)$ .

First, we need to evaluate the cost of loop 2.

The code tells us loop 2 starts at  $j_k = 0$ , and  $j_k$  will increase by 1 until  $j_k \geq \lceil \frac{n}{2} \rceil + 1$  (where +1 is because of loop 2 stopping at  $j_k = \lceil \frac{n}{2} \rceil$  by the if condition on line 10).

Using the fact  $j_k = k$ , we can calculate that loop 2 stops when

$$k \geq \left\lceil \frac{n}{2} \right\rceil + 1 \quad (1)$$

Since we are looking for the smallest value of  $k$  (because the smallest value of  $k$  translates to number of iterations), we can conclude the loop has

$$\left\lceil \frac{n}{2} \right\rceil + 1 \quad (2)$$

iterations.

Since each iteration takes a constant time (1 step), the cost of loop 2 is

$$\left(\left\lceil \frac{n}{2} \right\rceil + 1\right) \cdot 1 = \left\lceil \frac{n}{2} \right\rceil + 1 \quad (3)$$

steps.

Next, we need to evaluate the cost of loop 1.

The code tell us loop 1 will start at  $i_k = n$ , and  $i_k$  will decrease by 1 per iteration until  $i_k \leq \left\lceil \frac{n}{2} \right\rceil$ .

Using the fact  $i_k = n - k$ , we can write loop 1 stops when

$$n - k \leq \left\lceil \frac{n}{2} \right\rceil \quad (4)$$

$$-k \leq \left\lceil \frac{n}{2} \right\rceil - n \quad (5)$$

$$k \geq n - \left\lceil \frac{n}{2} \right\rceil \quad (6)$$

Since we are looking for the largest value of  $k$  (because the largest value of  $k$  translates to number of iterations), we can conclude loop 1 has

$$n - \left\lceil \frac{n}{2} \right\rceil \quad (7)$$

iterations.

Since each iteration costs  $\left\lceil \frac{n}{2} \right\rceil + 2$  steps (where  $\left\lceil \frac{n}{2} \right\rceil + 1$  is the cost of loop 2 and  $+1$  is the cost of line 14), we can conclude loop 1 has cost of

$$\left(\left\lceil \frac{n}{2} \right\rceil + 2\right) \left(n - \left\lceil \frac{n}{2} \right\rceil\right) \quad (8)$$

steps.

Finally, since the loop takes  $\lceil \frac{n}{2} \rceil + 1$  extra steps (where  $\lceil \frac{n}{2} \rceil$  is the cost of traveling from  $j = 0$  until  $j = \lceil \frac{n}{2} \rceil$  and  $+1$  is the cost of line 14) before coming to a full stop, the total running time is at least

$$\left(\left\lceil \frac{n}{2} \right\rceil + 2\right) \left(n - \left\lceil \frac{n}{2} \right\rceil\right) + \left\lceil \frac{n}{2} \right\rceil + 1 \quad (9)$$

steps, which is  $\Omega(n^2)$

#### Notes:

- Noticed there is no room for errors. (most of mark deductions are from not being careful with the analysis).
- Realized I need to take time to verify and re-verify steps using examples at a very fine level (i.e at this step this happens ... at this step this happens) until conclusion.
- Noticed professor uses  $i_k = n - k$  when going backward starting from  $n$ . And for the inequality,  $i_k \leq$  is used as opposed to the normal  $i_k \geq$ .

c. Let  $k, n \in \mathbb{N}$ , and  $list = [a_1, a_2, \dots, a_n]$  where  $a_1, \dots, a_n$  are even numbers.

We want to prove the best-case lower bound running time of this algorithm is  $\Omega(n)$ .

First, we need to analyze the cost of loop 2.

Given the iteration count  $k$ , the code tells us, the loop starts at  $j_k = 0$  and increases by 1 per iteration, and so we know  $j_k = k$ .

Because we know loop 2 runs until  $j_k \geq i$ , we can conclude loop 2 stops when

$$k \geq i \quad (1)$$

Since we are looking for the smallest value of  $k$  (because it represents the number of iterations), we can conclude loop 2 has  $i$  iterations.

Because we know each iteration of loop 2 costs a constant time (1 step), we can conclude loop 2 has cost of at least

$$k \cdot 1 = k \quad (2)$$

steps.

Now, we need to evaluate the cost of loop 1.

The code tells us loop 1 starts at  $i = n$  and ends at  $i = n$  due to the truthy condition of line 14.

Using these facts, we can conclude loop 1 has

$$\lceil n - n + 1 \rceil = 1 \tag{3}$$

iteration.

Because we know each iteration of loop 1 costs  $i + 2$  steps (where  $i$  is from the cost of loop 2, and  $+2$  are from the cost of line 8 and line 16), we can conclude loop 1 has cost of at least

$$(i + 2) \cdot 1 = i + 2 \tag{4}$$

steps.

Finally, because we know  $i = n$ , the total running time is at least  $n + 2$ , which is  $\Omega(n)$ .

## Question 3

## Question 4