Midterm 2 Version 2 Solution

Hyungmo Gu

April 5, 2020

Question 1

a.

$$\begin{aligned}
100 \div 3 &= 33, \text{Remainder } \mathbf{1}\\
33 \div 3 &= 11, \text{Remainder } \mathbf{0}\\
11 \div 3 &= 3, \text{Remainder } \mathbf{2}\\
3 \div 3 &= 1, \text{Remainder } \mathbf{0}\\
1 \div 3 &= 0, \text{Remainder } \mathbf{1}
\end{aligned}$$

It follows from above that the ternary representation of 100 is $(10201)_3$.

Attempt 2:

$$\begin{aligned}
100 + (-1 \cdot 3^4) &= 100 - 81 = 19\\
19 + (-1 \cdot 3^3) &= 19 - 27 = -8\\
-8 + (+1 \cdot 3^2) &= -8 + 9 = 1\\
1 + (0 \cdot 3^1) &= 1 + 0 = 1\\
1 + (-1 \cdot 3^0) &= 1 - 1 = 0
\end{aligned}$$

So by flipping the signs, and reading from top to bottom, we can conclude the balanced ternary representation of 100 is $(11T101)_{bt}$

Notes:

- Balanced ternary representation expresses a decimal using 1, 0 and -1
- **T** represents negative sign in balanced ternary representation.
- Is my way of calculating balanced ternary representation correct? My approach was 'which sign should be used given 3^n so the calculation stops at 3^0?'

b. The largest number expressible by an n-digit binary representation is

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

<u>Correct Solution:</u>

$$\sum_{i=0}^{n-1} 2^i = \frac{1-2^{n-1+1}}{1-2} = 2^n - 1 \tag{1}$$

Notes:

- Noticed professor simplified solution using geometric series
- Geometric series with finite sum

$$\sum_{i=0}^{n} r^k = \frac{1-r^{n+1}}{1-r}, \text{ where } |r| > 1 \tag{2}$$

c.

$f(n) \in \mathcal{O}(n)$	True	$g(n) \in \Omega(n)$	False	$f(n) \in \Omega(g(n))$	True
$f(n) \in \Theta(g(n))$	False	$g(n) \in \Theta(\log_3 n)$	False	$f(n) + g(n) \in \Theta(f(n))$	True

Notes:

- Learned $\sqrt{n}$ rises faster than $\log n$.
- Learned if $g(n) \in \Theta(f(n))$ is true then $f(n) + g(n) \in \Theta(f(n))$ is true.

d.

k	0	1	2	3
$i \cdot i \cdot i$	$2 = 2^{3^0}$	$2^3 = 2^{3^1}$	$2^9 = 2^{3^2}$	$2^27 = 2^{3^3}$

We can deduce from above that $i_k = 2^{3^k}$

e. $\lceil \log_3(\log_2(n) - 1) \rceil$

Correct Solution:

We want to find the smallest value of k satisfying $2 \cdot i_k \geq n$, and the value is

$$\lceil \log_3(\log_2(n) - 1) \rceil$$

Question 2

- **Statement:** $\forall n \in \mathbb{N}, n \geq 2 \Rightarrow \prod_{i=1}^{n} \frac{2^i-1}{2^i} \geq \frac{1}{2n}$

Proof. Let $n \in \mathbb{N}$. Assume $n \geq 2$.

We will prove the statement using induction on n.

Base Case (n = 2):

Let $n = 2$.

We want to show $\prod_{i=1}^{2} \frac{2^i-1}{2^i} \geq \frac{1}{2 \cdot (2)}$

Starting from $\prod_{i=1}^{2} \frac{2^i-1}{2^i}$, we can conclude

$$\begin{aligned} \prod_{i=1}^{2} \frac{2^i - 1}{2^i} = \left(\frac{1}{2}\right) \cdot \left(\frac{3}{4}\right) &= \frac{3}{8} && (1) \\ &\geq \frac{2}{8} && (2) \\ &\geq \frac{1}{4} && (3) \end{aligned}$$

Inductive Case:

Let $n \in \mathbb{N}$. Assume $\prod_{i=1}^{n} \frac{2^i-1}{2^i} \geq \frac{1}{2n}$.

We want to show $\prod_{i=1}^{n+1} \frac{2^i-1}{2^i} \geq \frac{1}{2(n+1)}$.

Starting from $\frac{1}{2(n+1)}$, because we know $n \geq 1$, we can conclude

$$\frac{1}{2(n+1)} \leq \frac{1}{2\cdot(n+n)} \tag{4}$$

$$= \frac{1}{2\cdot 2n} \tag{5}$$

Then, using inductive hypothesis $\prod_{i=1}^{n} \frac{2^i-1}{2^i} \geq \frac{1}{2n}$, we can conclude that

$$\frac{1}{2\cdot(n+1)} \leq \prod_{i=1}^{n} \frac{2^i-1}{2^i} \cdot \frac{1}{2} \tag{6}$$

$$= \prod_{i=1}^{n} \frac{2^i-1}{2^i} \cdot \left(1-\frac{1}{2}\right) \tag{7}$$

$$< \prod_{i=1}^{n} \frac{2^i-1}{2^i} \cdot \left(1-\frac{1}{2^{n+1}}\right) \tag{8}$$

$$= \prod_{i=1}^{n} \frac{2^i-1}{2^i} \cdot \left(\frac{2^{n+1}}{2^{n+1}}-\frac{1}{2^{n+1}}\right) \tag{9}$$

$$= \prod_{i=1}^{n} \frac{2^i-1}{2^i} \cdot \left(\frac{2^{n+1}-1}{2^{n+1}}\right) \tag{10}$$

$$= \prod_{i=1}^{n+1} \frac{2^i-1}{2^i} \tag{11}$$

□

Notes:

– Solved the inductive part of the problem by starting from the left hand side and calculating to the right as far as I can, and then repeating the same procedure from the right to the left until there were few missing steps left in between.

Question 3

- **Statement:** $\exists a \in \mathbb{R}^+, an^2+1 \in \Theta(n^4)$

  Negation of Statement: $\forall a, c_1, c_2, n_0 \in \mathbb{R}^+,\ \exists n \in \mathbb{N},\ (n \geq n_0) \wedge \Big((c_1 n^4 > an^2+1) \vee (an^2+1 > c_2 n^4)\Big)$

Proof. Let $a, c_1, n_0 \in \mathbb{R}^+$, and $n = \left\lceil max(n_0, \sqrt[4]{\frac{2}{c_1}}, \sqrt{\frac{2a}{c_1}}) \right\rceil + 1$.

We will disprove the statement by showing $(n \geq n_0)$ and $(c_1 n^4 > an^2 + 1)$.

Part 1 ($n \geq n_0$):

Because we know $\left\lceil max(n_0, \sqrt[4]{\frac{2}{c_1}}, \sqrt{\frac{2a}{c_1}}) \right\rceil \geq n_0$, we can conclude

$$n = \left\lceil max(n_0, \sqrt[4]{\frac{2}{c_1}}, \sqrt{\frac{2a}{c_1}}) \right\rceil + 1 > n_0 \tag{1}$$

Part 2 (Showing $c_1 n^4 > an^2 + 1$):

We need to show $c_1 n^4 > an^2 + 1$.

We will do so by showing $\frac{c_1 n^4}{2} > an^2$ and $\frac{c_1 n^4}{2} > 1$, and then combining them.

For the first inequality, using the fact that $\sqrt{\frac{2a}{c_1}} < \left\lceil max(n_0, \sqrt[4]{\frac{2}{c_1}}, \sqrt{\frac{2a}{c_1}}) \right\rceil + 1 = n$, we can calculate

$$\begin{align}
\sqrt{\frac{2a}{c_1}} &< n \tag{2}\\
\frac{2a}{c_1} &< n^2 \tag{3}\\
2a &< c_1 n^2 \tag{4}\\
a &< \frac{c_1 n^2}{2} \tag{5}\\
an &< \frac{c_1 n^3}{2} \tag{6}
\end{align}$$

For the second inequality, using the fact that $\sqrt[4]{\frac{2}{c_1}} < \left\lceil max(n_0, \sqrt[4]{\frac{2}{c_1}}, \sqrt{\frac{2a}{c_1}}) \right\rceil + 1 = n$, we can calculate

$$\begin{align}
\sqrt[4]{\frac{2}{c_1}} &< n \tag{7}\\
\frac{2}{c_1} &< n^4 \tag{8}\\
1 &< \frac{c_1 \cdot n^4}{2} \tag{9}
\end{align}$$

Then, by combining the two results together,

$$1 + an^2 < \frac{c_1 n^4}{2} + \frac{c_1 n^4}{2} \tag{10}$$
$$< c_1 n^4 \tag{11}$$

$\square$

Question 4

a. Let $n \in \mathbb{N}$.

We need to evaluate the total number of iterations.

We will do so by calculating the number of iterations in loop 2, and then calculating the number of iterations in loop 1.

For loop 2, the code tells us it starts at $j = 0$ and ends at $j = i^2 - 1$, and j increases by 2 per iteration.

Using this fact, we can conclude that loop 2 has

$$\left\lceil \frac{i^2 - 1 - 0 + 1}{2} \right\rceil \tag{1}$$

iterations.

Furthermore, since each iteration takes 1 step, we can conclude loop 2 has total of

$$\left\lceil \frac{i^2}{2} \right\rceil \cdot 1 = \left\lceil \frac{i^2}{2} \right\rceil \tag{2}$$

iterations.

For loop 1, the code tells us that it starts at $i = n$ and ends at $i = 1$ with i decreasing by 1 per iteration.

Because we know each iteration takes $\left\lceil \frac{i^2}{2} \right\rceil$ steps, we can conclude the loop 1 has

$$\left\lceil \frac{n^2}{2} \right\rceil + \left\lceil \frac{(n-1)^2}{2} \right\rceil + \cdots + \left\lceil \frac{1^2}{2} \right\rceil = \sum_{i=1}^{n} \left\lceil \frac{i^2}{2} \right\rceil \tag{3}$$

iterations.

Since the signs of ceilings and floors can be ignored, $\sum_{i=1}^{n} \left\lceil \frac{i^2}{2} \right\rceil$ can be simplified to

$$\sum_{i=1}^{n} \frac{i^2}{2} = \frac{1}{2} \sum_{i=1}^{n} i^2 \tag{4}$$

Then, since $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$,

$$\begin{aligned} \frac{1}{2} \sum_{i=1}^{n} i^2 &= \frac{1}{2} \cdot \frac{n(n+1)(2n+1)}{6} \qquad (5) \\ &= \frac{n(n+1)(2n+1)}{12} \qquad (6) \end{aligned}$$

b. *Proof.* **<u>Part 1 (Determining upper bound worst-case running time):</u>**

Let $n \in \mathbb{N}$. Assume the algorithm runs on worst possible case.

We will evaluate the upper bound worst-case running time by calculating the total cost of the algorithm, and then the big-oh.

First, we will evaluate the cost of loop 2.

The code tells us loop 2 starts at $j = i + 1$ and ends at $j = n - 1$, and j increases incrementally.

Using the fact, we can calculate loop 2 has at most

$$\left\lceil \frac{n - 1 - (i + 1) + 1}{2} \right\rceil = n - i - 1 \tag{1}$$

iterations.

Because we know each iteration in loop 2 costs a single step, we can conclude loop 2 has cost of at most

$$(n - i - 1) \cdot 1 = n - i - 1 \tag{2}$$

steps.

Now, we will calculate the cost of loop 1.

Because we know the if condition **if lst[i] % 2 == 0** will be satisfied at most n times from $i = 0$ to $i = n - 1$, and because we know each iteration costs $(n - i - 1)$ steps, we can conclude the loop has cost of at most

$$\begin{aligned}
\sum_{i=0}^{n-1}(n-i-1) &= \sum_{i=0}^{n-1}[(n-1)-i] && (3)\\
&= \sum_{i=0}^{n-1}(n-1) - \sum_{i=0}^{n-1} i && (4)\\
&= n \cdot (n-1) - \frac{n \cdot (n-1)}{2} && (5)\\
&= \frac{n \cdot (n-1)}{2} && (6)
\end{aligned}$$

steps.

Then, it follows from above the loop has upper bound running time of $\mathcal{O}(n^2)$.

Part 2 (Evaluating Lower Bound Running Time):

Let $n \in \mathbb{N}$, and $lst = [2, 3, 4, 5, \ldots, n]$.

We need to calculate the total cost of the algorithm given the input group, and the omega.

Since we know the line **list[j] = list[j] + 1** will turn all i^{th} value to an even number, we can conclude the if condition **if list[i] % 2 == 0** and it's inner loop will run n many times.

Since, this is equivalent to finding the total cost of algorithm for big-oh, the algorithm has total cost of

$$\frac{n \cdot (n-1)}{2} \qquad (7)$$

Then, it follows from above the algorithm has lower bound running time of $\Omega(n^2)$.

Then, since the value in $\mathcal{O}$ and Ω are equal, $\Theta(n^2)$ is true. $\square$