IFT-2125 assignment #1

Tarik Hireche | 202 301 89

04. March 2025

1 Asymptotic notation (25 points)

Question 1

We are asked to prove or refute the following propositions:

1. $\lfloor \frac{n}{2} \rfloor \in O(\sqrt{n})$
2. $n\sqrt{n}\log(n!) \in O(n^3 \log(n))$

Let's start with the first proposition, $\lfloor \frac{n}{2} \rfloor \in O(\sqrt{n})$:

Intuitively, this proposition is false because when n is very large $\lfloor \frac{n}{10} \rfloor \approx \frac{n}{10}$.

Let's prove this rigorously. Suppose that $\lfloor \frac{n}{10} \rfloor \in O(\sqrt{n})$, then by the definition of the big O notation:

$$(\exists c \in \mathbb{R}^{\geq 0}), (\exists n_0 \in \mathbb{N}) : \forall n \geq n_0, \left\lfloor \frac{n}{10} \right\rfloor \leq c.\sqrt{n} \tag{1.1}$$

Also, we can see that:

$$\frac{n}{10} - 1 \leq \left\lfloor \frac{n}{10} \right\rfloor \leq \frac{n}{10} \ (*) \tag{1.2}$$

So $\frac{n}{2} - 1$ consitutes a lower bound for $\lfloor \frac{n}{10} \rfloor$ and $\frac{n}{2}$ is an upper bound for $\lfloor \frac{n}{10} \rfloor$.
That means that $\lfloor \frac{n}{10} \rfloor$ is at most $\frac{n}{10}$. We are going to use that fact.

Great, now let's setup the equation and do a bit of mental gymnastics. We have that:

$$\left\lfloor \frac{n}{10} \right\rfloor \leq c.\sqrt{n} \tag{1.3}$$

$$\left\lfloor \frac{n}{10} \right\rfloor \leq c.n^{\frac{1}{2}} \tag{1.4}$$

We saw earlier that $\lfloor \frac{n}{2} \rfloor \geq \frac{n}{2} - 1$, so using (*) this would imply:

$$\frac{n}{10} - 1 \leq c.n^{\frac{1}{2}} \tag{1.5}$$

For very large n , the term -1 becomes negligible. So we have:

$$\frac{n}{10} \leq c.n^{\frac{1}{2}} \tag{1.6}$$

$$\frac{1}{10}.n \leq c.n^{\frac{1}{2}} \tag{1.7}$$

$$\frac{1}{n^{\frac{1}{2}}}.\left(\frac{1}{10}.n\right) \leq \frac{1}{n^{\frac{1}{2}}}.\left(c.n^{\frac{1}{2}}\right) \tag{1.8}$$

$$\frac{1}{10}.n^{1-\frac{1}{2}} \leq c.n^{\frac{1}{2}-\frac{1}{2}} \tag{1.9}$$

$$\frac{1}{10}.\sqrt{n} \leq c \tag{1.10}$$

$$\sqrt{n} \leq 10c \tag{1.11}$$

$$n \leq (10c)^2 \tag{1.12}$$

Let's set $(10c)^2 = d$, that means that:

$$n \leq d, \text{où } d \in \mathbb{N} \tag{1.13}$$

$$\rightarrow\leftarrow \text{ Contradiction.} \tag{1.14}$$

By supposing that $\lfloor \frac{n}{2} \rfloor \in O(\sqrt{n})$, we have stumbled upon a contradiction. Why? Well it is because n cannot have an upper bound because by definition, the condition must hold for all $n \geq n_0$, the threshold. Therefore:

$$\left\lfloor \frac{n}{2} \right\rfloor \notin O(\sqrt{n})$$

2. $n\sqrt{n}\log(n!) \in O(n^3 \log(n))$

   After giving some thoughts about this, I want to explore an interesting property of the big O notation to demonstrate the proposition. I am going to use the following property viewed in class and available in the chapter #3 titled *Asymptotic Notation p.10* :

$$f_1(n) \in O(g_1(n)) \wedge f_2(n) \in O(g_2(n)) \rightarrow f_1(n).f_2(n) \in O(g_1(n).g_2(n))$$

$$\text{Let } f_1(n) = n\sqrt{n},\ f_2(n) = \log(n!),\ g_1(n) = n^3,\ g_2(n) = \log(n)$$

   Okay, we have a bunch of work to do now.

   I) Let's show that $f_1(n) \in O(g_1(n))$.

   If $n\sqrt{n} \in O(n^3)$, then by the definition:

$$(\exists c \in \mathbb{R}^{\geq 0}), (\exists n_0 \in \mathbb{N}) : \forall n \geq n_0, n\sqrt{n} \leq c.n^3 \tag{1.1}$$

$$n\sqrt{n} \leq c.n^3 \tag{1.2}$$

$$\frac{1}{n}.(n\sqrt{n}) \leq \frac{1}{n}.(c.n^3) \tag{1.3}$$

$$n^{\frac{1}{2}} \leq c.n^2 \tag{1.4}$$

$$1 \leq c.n^{2-\frac{1}{2}} \tag{1.5}$$

$$\frac{1}{c} \leq n^{\frac{3}{2}} \tag{1.6}$$

We raise both sides of inequality (1.6) to the power of $\frac{2}{3}$ to isolate n. This operation leaves us with:

$$n \geq \frac{1}{c^{\frac{2}{3}}} \tag{1.7}$$

We have found a threshold $n_0 = \frac{1}{c^{\frac{2}{3}}}$ from which $n^3 \geq n\sqrt{n}$ with $c = 1$. With this, we can conclude that:

$$n\sqrt{n} \in O(n^3) \tag{1.8}$$

II) Let's show that $f_2(n) \in O(g_2(n))$.

If $\log(n!) \in O(\log(n))$, then by the definition:

$$(\exists c \in \mathbb{R}^{\geq 0}), (\exists n_{0'} \in \mathbb{N}) : \forall n \geq n_{0'}, \log(n!) \leq c'. \log(n) \tag{1.9}$$

$\log(n!)$ is somewhat abstract to visualize isn't it? Let's find a way to make it clearer to understand. First of all, we can observe that

$$\log(n!) = \log[n \times (n-1) \times (n-2) \times ... \times 3 \times 2 \times 1] \tag{1.10}$$

$$= \log(n) + \log(n-1) + ... + \log(3) + \log(2) + \log(1) \tag{1.11}$$

We have seen a similar exercice in class that splits this sum in half and I am going to use the same idea to find a lower bound for $\log(n!)$.

Let's call this lower bound function $f_{lb}(n)$. If $f_{lb}(n)$ is a lower bound for $\log(n!)$ that means:

$$f_{lb}(n) \leq \log(n!) \tag{1.12}$$

Which also means that $f_{lb}(n)$ has to also be in in $O(\log(n))$ if $\log(n!)$ is.

Let's find $f_{lb}(n)$:

$$\log(n!) = \log(n) + \log(n-1) + ... + \log(3) + \log(2) + \log(1) \tag{1.13}$$

$$= \sum_{i=1}^{n} \log(i) \tag{1.14}$$

Let's split the sum in half, the first half S_{n_1} from [1,n/2] and the second half S_{n_2} from $[\frac{n}{2} + 1, n]$:

$$S_{n_1} = \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \log(i), \ S_{n_2} = \sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^{n} \log(i) \tag{1.15}$$

Of course this makes it obvious that:

$$\log(n!) = S_{n_1} + S_{n_2} \tag{1.16}$$

Every term in S_{n_2} is at least $\log(\frac{n}{2})$ as the sum starts from $i = \frac{n}{2} + 1$ and since we split our sum in half then there are $\frac{n}{2}$ terms in the second sum; $|\{S_{n_2}\}| = n - (\frac{n}{2} + 1) + 1 = \frac{n}{2}$.

$$\text{We can therefore use } \frac{n}{2} \times \log\left(\frac{n}{2}\right) \text{ as a lower bound for} \log(n!) : \tag{1.17}$$

$$\frac{n}{2}. \log\left(\frac{n}{2}\right) \leq \log(n!) \tag{1.18}$$

if $\frac{n}{2}. \log(\frac{n}{2})$ is a lower bound for $\log(n!)$ then it must be in $O(\log(n))$ as well. Let's play around with:

$$\frac{n}{2}. \log\left(\frac{n}{2}\right) \leq c'. \log(n) \tag{1.19}$$

$$\frac{\frac{n}{2}. \log(\frac{n}{2})}{\log(n)} \leq c'. \frac{\cancel{\log(n)}}{\cancel{\log(n)}} \tag{1.20}$$

$$\frac{n}{2} \cdot \frac{\log(\frac{n}{2})}{\log(n)} \leq c' \tag{1.21}$$

For large n, $\frac{\log(\frac{n}{2})}{\log(n)} \to 1$ as the coefficient $\frac{1}{2}$ becomes negligible, therefore the equation becomes:

$$\frac{n}{2} \leq c' \tag{1.22}$$

$$n \leq 2c' \tag{1.23}$$

$$\rightarrow\leftarrow \text{ Contradiction} \tag{1.24}$$

Even if we did not clean up the expression, the contradiction had already risen up (at the *line (1.20)*). We had on the left side an expression that is a lower bound for $\log(n!)$ bounded by a constant c' $\in \mathbb{R}$, it was already an obvious contradiction.

Since have once again stumbled upon a contradiction and clearly showed that n is bounded by a constant, we therefore proved that:

$$\log(n!) \notin O(\log(n))$$

Now since $n\sqrt{n}$ is at most $O(n^3)$, the entire product $n\sqrt{n}\log(n!)$ will grow faster than all of the functions in the set $O(n^3 \log(n))$ because:

1. $n\sqrt{n}$ is bounded by n^3
2. But $\log(n!)$ grows much faster than $\log(n)$
3. Therefore the product $n\sqrt{n}\log(n!)$ cannot be bounded by in $n^3\log(n)$.

In conclusion, this shows that:

$$n\sqrt{n}\log(n!) \notin O(n^3\log(n))$$

Question 2

Using the limit rule, we are asked to determine the relative order (O, Ω or Θ) of the following functions:

1. $f(n) = \frac{n}{\sqrt[3]{n}}, g(n) = \ln\sqrt{n}$
2. $f(n) = 2^{bn}, g(n) = 3^n$ where $b \geq \mathbb{N}^{\geq 2}$

The limit rule states that:

$$\lim_{n\to+\infty} \frac{f(n)}{g(n)} = \begin{cases} 0 \to f(n) \in O(g(n)) \\ k \in \mathbb{R}^+ \to f(n) \in O(g(n)), g(n) \in O(f(n)) \\ +\infty \to f(n) \notin O(g(n)), g(n) \in O(f(n))) \end{cases}$$

Let's start with the question 1:

$$\lim_{n\to+\infty} \frac{\frac{n}{\sqrt[3]{n}}}{\ln\sqrt{n}} \tag{1.1}$$

$$(*)\frac{n}{\sqrt[3]{n}} = n^{1-\frac{1}{3}} = n^{\frac{2}{3}} \to \lim_{n\to+\infty} \frac{n^{\frac{2}{3}}}{\ln\sqrt{n}} \tag{1.2}$$

$$= \lim_{n\to+\infty} \frac{n^{\frac{2}{3}}}{\frac{1}{2}\ln n} \overset{\text{RH}}{=} \lim_{n\to+\infty} \frac{\frac{2}{3}n^{-\frac{1}{3}}}{\frac{1}{2n}} \tag{1.3}$$

$$= \lim_{n\to+\infty} \frac{2}{3n^{\frac{1}{3}}} \times 2n = \lim_{n\to+\infty} \frac{4n}{3n^{\frac{1}{3}}} = \lim_{n\to+\infty} \frac{4n^{1-\frac{1}{3}}}{3} = \lim_{n\to+\infty} \frac{4}{3} n^{\frac{2}{3}} \tag{1.4}$$

$$= \frac{4}{3} \lim_{n\to+\infty} n^{\frac{2}{3}} = +\infty \tag{1.5}$$

$$\text{Now since: } \lim_{n\to+\infty} \frac{n^{\frac{2}{3}}}{\frac{1}{2}\ln n} = +\infty$$

Therefore, by the limit rule:

$$\ln(\sqrt{n}) \in O\left(\frac{n}{\sqrt[3]{n}}\right)$$

Continuing with question 2:

$$\lim_{n\to+\infty} \frac{2^{bn}}{3^n}, \text{where } b \geq \mathbb{N}^{\geq 2} \tag{1.6}$$

Notice here that because $b \geq 2 \in \mathbb{N}$:

$$2^{bn} \geq 3^n, \forall n \geq 1 \tag{1.7}$$

The term $\left(\frac{2^b}{3}\right) > 1$, $\forall b \geq 2 \in \mathbb{N}$, so:

$$\lim_{n\to+\infty} \frac{2^{bn}}{3^n} = \lim_{n\to+\infty} \left(\frac{2^b}{3}\right)^n = +\infty \tag{1.8}$$

Therefore, for any $b \geq 2 \in \mathbb{N}$, by the limit rule:

$$3^n \in O(2^{bn})$$

Question 3

We are asked to show that $f(n) = 2n^2 - \sin(n)$ is an eventually non decreasing function (E.N.D).

For it to be E.N.D, there needs to be number $k_0 \in \mathbb{N}^*$, a threshold such that:

$$\forall n \geq k_0 \rightarrow f(n+1) \geq f(n)$$

That means that from that number k_0:

$$f(n+1) - f(n) \geq 0, \forall n \geq k_0$$

$$\underbrace{2(n+1)^2 - \sin(n+1)}_{\text{f(n+1)}} - \underbrace{[2n^2 - \sin(n)]}_{\text{f(n)}} \tag{1.1}$$

$$\underbrace{2(n^2 + 2n + 1) - \sin(n+1)}_{\text{f(n+1)}} - \underbrace{[2n^2 - \sin(n)]}_{\text{f(n)}} \tag{1.2}$$

$$\cancel{2n^2} + 4n + 2 - \sin(n+1) - \cancel{2n^2} + \sin(n) \tag{1.3}$$

That means that the following expression has to be greater than zero:

$$4n + 2 + \sin(n) - \sin(n+1) \geq 0 \tag{1.4}$$

$$\text{First, we can see that: } 4n + 2 \geq 2 \tag{1.5}$$

Also, the sine function is bounded:

$$-1 \leq \sin(n) \leq 1 \tag{1.6}$$

$$-1 \leq \sin(n+1) \leq 1 \tag{1.7}$$

Combining both sine terms:

$$-2 \leq \sin(n+1) - \sin(n) \leq 2 \tag{1.8}$$

Adding the remaining term $4n + 2$ to complete the operation:

$$4n \leq 4n + 2 + \sin(n+1) - \sin(n) \leq 4n + 4 \tag{1.9}$$

Therefore:

$$f(n+1) - f(n) = 4n + 2 + \sin(n+1) - \sin(n) \geq 4n \geq 4 > 0 \quad \forall n \geq 1 \tag{1.10}$$

Since we have been able to find a strictly positive lower boundary for $f(n+1) - f(n)$, we have shown that $f(n) = 2n^2 - \sin(n)$ is E.N.D.

2 Recurrences

Question 1

We are asked to solve the following reccurence by finding its caracteristic polynomial, its roots with their multiplicity.

$$t_n = 4t_{n-1} - 4t_{n-2} + 4(n+1)4^n \tag{2.1}$$

We multiply by 4:

$$4 \times (t_n = 4t_{n-1} - 4t_{n-2} + 4(n+1)4^n \tag{2.2}$$

$$4t_n = 16t_{n-1} - 16t_{n-2} + 16(n+1)4^n \ \blacktriangle \tag{2.3}$$

Setting $n = n - 1$ in the original form :

$$t_{n-1} = 4t_{n-2} - 4t_{n-3} + 4n4^{n-1} \tag{2.4}$$

We multiply by 4 again:

$$4t_{n-1} = 16t_{n-2} - 16t_{n-3} + 16n4^{n-1} \ \blacksquare \tag{2.5}$$

Let's perform the following operation: $(\blacktriangle) - (\blacksquare)$

$$4t_n = 16t_{n-1} - 16t_{n-2} + 16(n+1)4^n \tag{2.6}$$

$$-$$

$$4t_{n-1} = 16t_{n-2} - 16t_{n-3} + 16n4^{n-1}$$

$$=$$

$$4t_n - 4t_{n-1} = 16t_{n-1} - 16t_{n-2} + 16(n+1)4^n - \left(16t_{n-2} - 16t_{n-3} + 16n4^{n-1}\right) \tag{2.7}$$

$$4t_n - 4t_{n-1} = 16t_{n-1} - 16t_{n-2} + 16(n+1)4^n - 16t_{n-2} + 16t_{n-3} - 16n4^{n-1} \tag{2.8}$$

$$4t_n - 4t_{n-1} - 16t_{n-1} + 32t_{n-2} - 16t_{n-3} = 16n.4^n + 16.4^n - 16n\frac{4^n}{4} \tag{2.9}$$

$$4t_n - 4t_{n-1} - 16t_{n-1} + 32t_{n-2} - 16t_{n-3} = 16\left[n4^n - n\frac{4^n}{4} + 4^n\right] \tag{2.10}$$

$$4t_n - 4t_{n-1} - 16t_{n-1} + 32t_{n-2} - 16t_{n-3} = 16n4^n - 16n\frac{4^n}{4} + 16.4^n \tag{2.11}$$

$$4t_n - 4t_{n-1} - 16t_{n-1} + 32t_{n-2} - 16t_{n-3} = 16n4^n - 4n4^n + 16.4^n \tag{2.12}$$

$$4t_n - 20t_{n-1} + 32t_{n-2} - 16t_{n-3} = 12n4^n + 16.4^n \tag{2.13}$$

We end up with this linear recurrence. The non-homogeneous part are the terms:

$$12n4^n + 16.4^n \tag{2.14}$$

The caracteristic polynomial of the reccurence (2.17) is:

$$4x^3 - 20x^2 + 32x - 16 \tag{2.15}$$

Great, let's find the solutions for the homogeneous part first, meaning all solutions that satisfy $P(x) = 0$.

$$4x^3 - 20x^2 + 32x - 16 = 0 \text{ mutiply by } 1/4 \tag{2.16}$$

$$P(x) = x^3 - 5x^2 + 8x - 4 = 0 \tag{2.17}$$

I like to use ruffini's method to perform division. Although, for that we need to know a root of the polynomial. Let's try $x = 1$:

$$1^3 - 5^2 + 8 - 4 = 0 \tag{2.18}$$

$$24 + 4 \neq 0 \rightarrow x = 1 \text{ is not a root}$$

Let's try $x = 2$:

$$2^3 - 5.2^2 + 8.2 - 4 = 0 \tag{2.19}$$

$$8 - 4 - 4 = 0 \rightarrow x = 2 \text{ is a root}$$

Great, now we know that $(x - 2)$ is a factor of P(x), we can perform the division using Ruffini's method. We put the coefficients of the divident P(x) in the first line, and the divisor on the left, in this case 2. We then follow Ruffini's method:

$$\begin{array}{c|cccc} 2 & 1 & -5 & 8 & -4 \\ + & \downarrow & 2 & -6 & 4 \\ \hline & 1 & -3 & 2 & 0 \end{array}$$

$$\text{We end up with: } x^2 - 3x + 2 \tag{2.20}$$

$$\triangle = b^2 - 4ac = (-3)^2 - 4(1)(2) = 9 - 8 = 1 \tag{2.21}$$

$$r_{1,2} = \frac{-b \pm \sqrt{\triangle}}{2a}$$

$$r_1 = 2, r_2 = 1 \tag{2.22}$$

$$\text{Therefore, } P(x) = (x - 2)(x - 2)(x - 1) \tag{2.23}$$

$$x = 2 \ \text{ with multiplicity of 2} \tag{2.24}$$

$$x = 1 \text{ with a multiplicty of } 1 \tag{2.25}$$

So the general solution t_n^{hom} for the homogeneous part is:

$$t_n^{\text{hom}} = (c_1 + c_2 n)2^n \tag{2.26}$$

Now for the non-homogenenous part, because 4^n does not correspond to a root of P(x), it means that 4 is not a solution of P(x).

The non-homogeneous term has $(n+1)$ in front, which is a polynomial of degree 1 and because $x = 4$ is not a root, we do not need to multiply by extra powers of n (to make it independant). Therefore, we are looking for a solution in the form:

$$t_n^{\text{P\_sol}} = (An + B)4^n \tag{2.27}$$

Let's plug $t_n^{\text{P\_sol}}, t_{n-1}^{\text{P\_sol}}$ and $t_{n-2}^{\text{P\_sol}}$ into our original recurrence: $t_n - 4t_{n-1} + 4t_{n-2} = 4(n+1)4^n$

$$t_n^{\text{P\_sol}} = (An + B)4^n \tag{2.28}$$

$$t_{n-1}^{\text{P\_sol}} = [A(n-1) + B]4^{n-1} \tag{2.29}$$

$$t_{n-2}^{\text{P\_sol}} = [A(n-2) + B]4^{n-2} \tag{2.30}$$

Now let's compute each term of the recurrence:

$$t_n = (An + B)4^n \tag{2.31}$$

$$-4t_{n-1} = -[A(n-1) + B]4^n \tag{2.32}$$

$$4t_{n-2} = [A(n-2) + B]4^{n-1} \tag{2.33}$$

Combining all terms:

$$t_n - 4t_{n-1} + 4t_{n-2} = 4(n+1)4^n \tag{2.34}$$

$$= \tag{2.35}$$

$$(An + B)4^n - [A(n-1) + B]4^n + [A(n-2) + B]4^{n-1} \tag{2.36}$$

$$4^n(An + B - A(n-1) - B) + [A(n-2) + B]4^{n-1} \tag{2.37}$$

$$4^n(\cancel{An} + \cancel{B} - \cancel{An} + A \cancel{-B}) + [A(n-2) + B]4^{n-1} \tag{2.38}$$

$$= A.4^n + [A(n-2) + B]4^{n-1} \tag{2.39}$$

$$= 4^{n-1}[4A + A(n-2) + B] \tag{2.40}$$

$$= 4^{n-1}[4A + An - 2A + B] \tag{2.41}$$

$$= 4^{n-1}[An + 2A + B] \tag{2.42}$$

After replacing every term, we end up with:

$$4^{n-1}[An + 2A + B] = (n+1)4^{n+1} \tag{2.43}$$

$$4^n \frac{1}{4}[An + 2A + B] = 4(n+1)4^n \tag{2.44}$$

$$\left[4^n \frac{1}{4}[An + 2A + B] = 4(n+1)4^n\right] \times \frac{1}{4^n} \tag{2.45}$$

$$\frac{1}{4}[An + 2A + B] = 4(n+1) \tag{2.46}$$

$$An + 2A + B = 16(n+1) \tag{2.47}$$

$$\underbrace{An}_{\text{deg=1}} + \underbrace{(2A+B)}_{\text{deg=0}} = \underbrace{16n}_{\text{deg=1}} + \underbrace{16}_{\text{deg=0}} \tag{2.48}$$

We have the following system of equation:

$$An = 16n \tag{2.49}$$

$$2A + B = 16 \tag{2.50}$$

$$An = 16n \to A = 16 \tag{2.51}$$

$$2 \times \underbrace{16}_{\text{A}} + B = 16 \tag{2.52}$$

$$32 + B = 16 \tag{2.53}$$

$$B = -16 \tag{2.54}$$

We have found a valid particular solution:

$$t_n^{\text{P\_sol}} = \left(\underbrace{16n}_{\text{An}} - \underbrace{16}_{\text{B}}\right)4^n \tag{2.55}$$

Combining $t_n^{\text{hom}} + t^{\text{P\_sol}}$ to get t_n :

$$t_n = (c_1 + c_2 n)2^n + 16(n-1)4^n \tag{2.56}$$

Question 2

For this question, we are asked to analyze the complexity of a little python program that is recursive. I only took the portion of the code that is relevant for solving this problem - which are only the parts of the code where the method <<t.forward(length)>> is called.

```python
def tree(length, level):
  global count
  if level > 0 :
    t.forward(length)
  tree(length/alpha, level - 1)
  tree(length/alpha, level - 1)
  tree(length/alpha, level - 1)

tree(200,4)
```

Let's start with the base case, for the level $n = 0$, <<if level > 0: t.forward(length)>> is not called, that means $T(0) = 0$. Furthermore, we can see that for a level $n > 0$, we are going to call <<t.forward(length)>> exactly once and then call recursively our tree method to draw a tree that is one unit smaller than the last level 3 times until we reach the base case where $n = 0$ - where <<t.forward(length)>> is not called at all. Formalizing this mathematically, we get:

$$t(n) = \begin{cases} 0 \text{ if } n = 0 \\ 3t(n-1) + 1 \text{ if } n > 1 \end{cases} \tag{2.1}$$

$$\text{For a level } k \geq 1 :$$

$$t(k) = 1 + 3t(k-1) = 1 + 3 \times [1 + 3t(k-2)] = 1 + 3 + 3^2 \times t(k-3) \tag{2.2}$$

$$= 1 + 3 + 3^2[1 + 3t(k-4)] \tag{2.3}$$

$$1 + 3^2 + 3^3 + ... + 3^{k-1} + 3^k \times t(0) \tag{2.4}$$

$$= 3^{k-1} + 3^k \times 0 = 3^{k-1} \tag{2.5}$$

$$\sum_{k=1}^{n-1} 3^k = \frac{r^n - 1}{r - 1} = \frac{3^n - 1}{3 - 1} = \frac{3^n - 1}{2} \tag{2.6}$$

$$\Theta\left(\frac{3^n - 1}{2}\right) = \Theta\left(\max\left(\frac{3^n - 1}{2}\right)\right) \tag{2.7}$$

$$= \Theta(3^n) \tag{2.8}$$

To conclude, the complexity of the program when we are strictly looking at the calls of the <<t.forward(length)>> method is in $\Theta(3^n)$.

Question 3

We are tasked to give the complexity of some recurrences using the master theorem.

Note: I will use the notation $\log_2(n) = \lg(n)$.

The master theorem:

$$\text{Let } n_0 \geq 1 \in \mathbb{N}, \ell \geq 1, b \geq 2 \in \mathbb{N}, k \geq 0 \in \mathbb{N}, c \in \mathbb{R}^+ \tag{2.9}$$

$$T(n) = \ell T\left(\frac{n}{b}\right) + cn^k, \forall n \geq n_0 \tag{2.10}$$

where $\frac{n}{n_0}$ is a power of b and $T(n)$ is E.N.D. We have:

$$T(n) \in \begin{cases} \Theta(n^k) \text{ if } \ell < b^k \\ \Theta(n^k \log_b n) \text{ if } \ell = b^k \\ \Theta(n^{\log_b \ell}) \text{ if } \ell > b^k \end{cases} \tag{2.11}$$

1. $t(n) = t(\frac{n}{2}) + 4n$:
$$\ell = 1, b = 2, k = 1, 1 < 2^1 : \text{since } \ell < b^k \to t(n) \in \Theta(n) \tag{2.12}$$

2. $t(n) = 2t(\frac{n}{2}) + 2n$
$$\ell = 2, b = 2, k = 1, 2 = 2^1 : \text{since } \ell = b^k \to t(n) \in \Theta(n \lg(n)) \tag{2.13}$$

3. $t(n) = 3t(\frac{n}{3}) + 3n^3$
$$\ell = 3, b = 3, k = 3,\ 3 < 3^3 : \text{since } \ell < b^k \to t(n) \in \Theta(n^3) \tag{2.14}$$

4. $t(n) = 4t(\frac{n}{3}) + 2n$
$$\ell = 4, b = 3, k = 1, 4 > 3^1 : \text{since } \ell > b^k \to t(n) \in \Theta\left(n^{\log_3(4)}\right) \approx \Theta(n^{1.2618\ldots}) \tag{2.15}$$

5. $t(n) = 4t(\frac{n}{2}) + 2n^2$
$$\ell = 4, b = 2, k = 2, 4 = 2^2 : \text{since } \ell = b^k \to t(n) \in \Theta(n^2 \lg(n)) \tag{2.16}$$

3 Double pointers - Code Easy: Report

We are given an ascending sorted list of n exam scores. We are tasked to find the number of distinct pairs of scores that sum up to the median of this list. I have implemented a solution that uses the double pointers technique.

Complexity Analysis

3.0.1 Double pointers technique's time complexity

We use two pointers:

- Left pointer $\rightarrow$ pointing at the first element of the integer list.
- Right pointer $\rightarrow$ pointing at the last element of our integer list.

The idea is that they both start at opposite ends of the list. They move toward each other at most once per element and since each movement processes an element only once, the traversal of the whole list is in $O(n)$.

3.0.2 The list of numbers that is given as an input

Since the list that is given is already sorted, there is less work to do. Accessing an element by its index takes $O(1)$.

3.0.3 Finding the median

Since the list is already sorted, the median is either the middle element if the list is odd or the average of the two middle elements if the list is even. All of theses operations are constant.

3.0.4 The set data structure used to store pairs

On average, insertions and lookup in a set are in $O(1)$.

Combining all of theses time complexity, our program is $O(n)$.