CS427 Fall Assignment #1
Contents: Chapter 1-2.2
Textbook, Pg.8 Ex.10 (a), Pg.18 Ex.9, Pg.59 Ex.3

Rules:

- Each pseudocode must specify its input and output.
- Necessary comments should be included.
- When analyzing complexity, specific reasons must be provided.
- For proof questions, a specific derivation process is required. Please refer to the example in the slide.

1. Euclid's algorithm, as presented in Euclid's treatise, uses subtractions rather than integer divisions. Write pseudocode for this version of Euclid's algorithm. (10 pts)

```
// Time Complexity O(n) – While loop can run up to n iterations if a and b are large and nieghboring.
// Space Complexity O(1) – no additional frames or functions called

EuclidsAlg(int a, int b) {
        // Input: two positive integers

        // O(n)
        While(a != b) {
                // The GCD of two numbers does not change if you replace the larger number with the difference of the two
                //O(1)
                If(a > b) {
                        a = a – b;
                } else {
                        b = b – a;
                }
        }
        // Output: Returns GCD (a) when a and b are equal
        Return a;

}
```

2. Consider the following algorithm for finding the distance between the two closest elements in an array of numbers. (10 pts)

ALGORITHM ***MinDistance*****(A[0..n − 1])**

//Input: Array A[0..n − 1] of numbers

//Output: Minimum distance between two of its elements

dmin **← ∞**
for i ← 0 to n − 1 do

for j ←0 to n−1 do
if i≠j and |A[i]−A[j]| < *dmin*

dmin **← |A[i]−A[j]|**

return *dmin*

Make as many improvements as you can in this algorithmic solution to the problem. If you need to, you may change the algorithm altogether; if not, improve the implementation given.

// Time Complexity O(n log n) – Using Arrays.Sort() using dual pivot quicksort with O(n) time to compare each element with pivots and O(log n) to divide the array into constant-fraction pieces
// Space Complexity: O(1) - with primitive ints in Arrays.Sort() create negligible space overhead

ALGORITHM ***MinDistance*****(A[0..n − 1])**

//Input: Array A[0..n − 1] of numbers

//Output: Minimum distance between two of its elements

Int dmin = +∞

// Exit if array is less than 2 numbers
//O(1)
If(A.length < 2) {
return dmin;

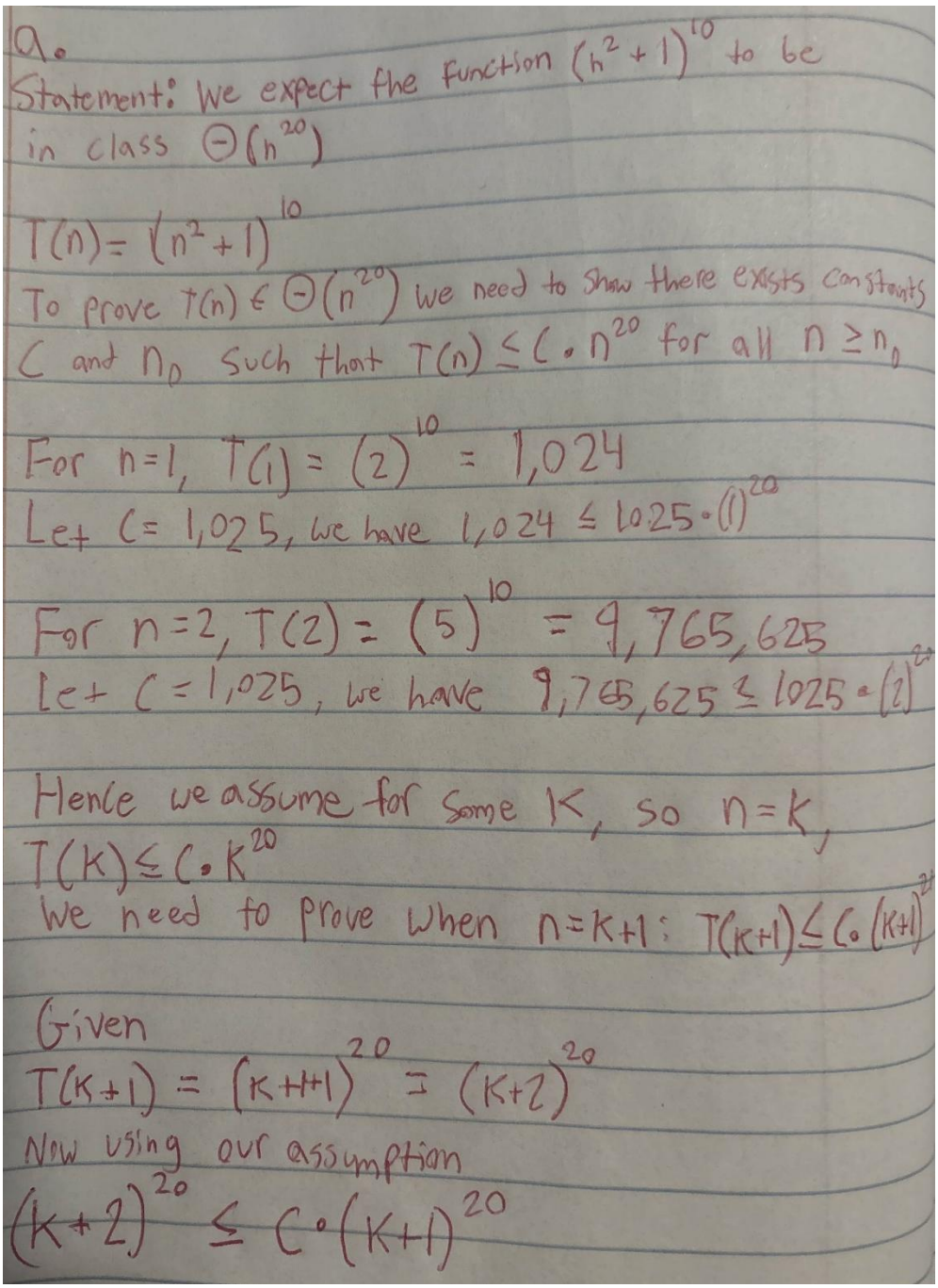
```
        }

        //Sort numbers O(n log n)
        Arrays.Sort(A);

        //O(n)
        For(int i = 1; i < A.length; i++) {
                // If neighboring numbers are the same – we return 0
                If (A[i] == A[i-1]) {
                        Return 0;
                }

                if(A[i] – A[i-1] < dmin) {
                        dmin = A[i] – A[i-1];
                }
        }
return dmin;
}
```

3. For each of the following functions, indicate the class $\Theta(g(n))$ the function belongs to. (Use the simplest g(n) possible in your answers.) Prove your assertions. (75 pts)

Note: Please follow the assignment format of the slides.

For each question, you need to provide **Statement**, the **Prove** of mathematical derivation, and **Conclusion**.

a. $(n^2+1)^{10}$



a.
Statement: We expect the function $(n^2+1)^{10}$ to be in class $\Theta(n^{20})$

$T(n) = (n^2+1)^{10}$

To prove $T(n) \in \Theta(n^{20})$ we need to show there exists constants C and n_0 such that $T(n) \leq C \cdot n^{20}$ for all $n \geq n_0$

For $n=1$, $T(1) = (2)^{10} = 1,024$

Let $C = 1,025$, we have $1,024 \leq 1025 \cdot (1)^{20}$

For $n=2$, $T(2) = (5)^{10} = 9,765,625$

Let $C = 1,025$, we have $9,765,625 \leq 1025 \cdot (2)^{20}$

Hence we assume for some K, so $n=k$,

$T(K) \leq C \cdot K^{20}$

We need to prove when $n=K+1$: $T(K+1) \leq C \cdot (K+1)^{20}$

Given

$T(K+1) = (K+1+1)^{20} = (K+2)^{20}$

Now using our assumption

$(K+2)^{20} \leq C \cdot (K+1)^{20}$

Left side: $(k+2)^{20} = 8$

Right side: $1025(k+1)^{20}$

For $c \geq 1025$ the above statement holds true, so as n becomes large, the exponential term will dominate.

Thus by PMI, $T(n) \in \Theta(n^{20})$

b. $\sqrt{10n^2 + 7n + 3}$

b. statement: We expect the function $\sqrt{10n^2+7n+3}$ to be in class $\Theta(n)$.

Pf: $T(n) = \sqrt{10n^2+7n+3}$

To prove $T(n) \in \Theta(n)$ we need to show there exists constants C and n_0 such that $T(n) \leq C \cdot n$ for all $n \geq n_0$

For $n=1$, $T(1) = \sqrt{10(1)^2+7(1)+3} = \sqrt{20}$

Let $C = \sqrt{20}+1$, we have $\sqrt{20} \leq \sqrt{20}+1 \cdot 1$

For $n=2$, $T(2) = \sqrt{10(2)^2+7(2)+3} = \sqrt{57}$

Let $C = \sqrt{20}+1$, we have $\sqrt{57} \leq (\sqrt{20}+1)2$

Hence we assume for some K, so $n=k$

$T(k) \leq C \cdot K$

We need to prove when $n=K+1$, $T(K+1) \leq C(K+1)$

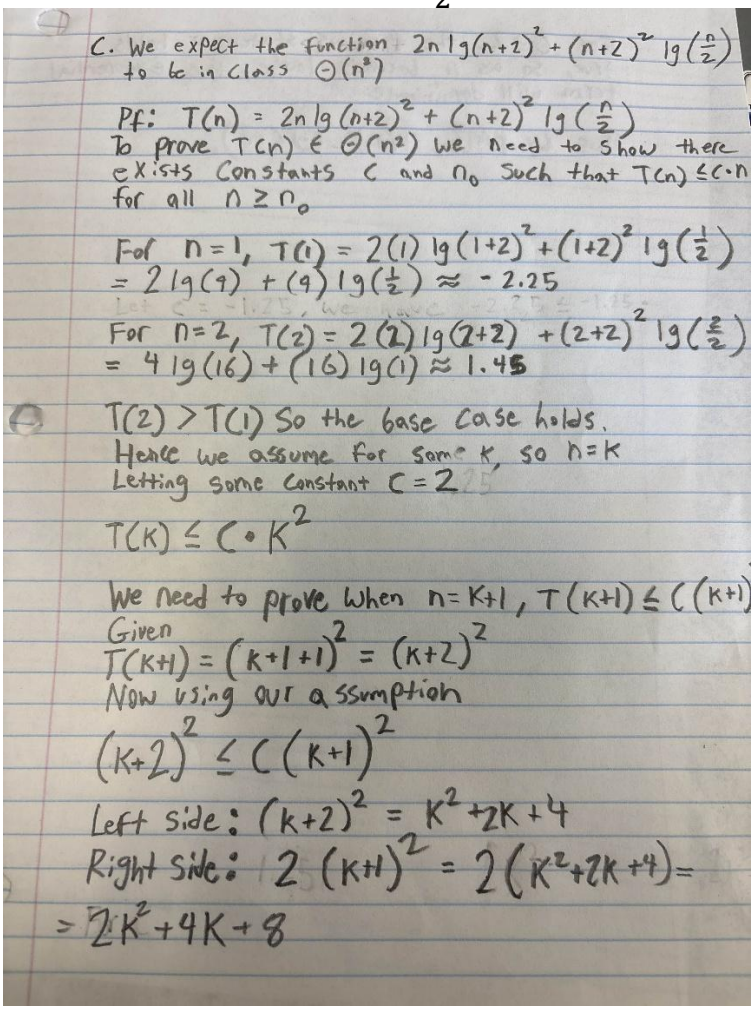
Given

$T(K+1) \leq C(K+1)$

This holds true for any C greater than 1.

Thus by PMI, $T(n) \in \Theta(n)$.

c. $2n \lg(n+2)^2 + (n+2)^2 \lg\frac{n}{2}$



C. We expect the function $2n\lg(n+2)^2 + (n+2)^2\lg\left(\frac{n}{2}\right)$ to be in class $\Theta(n^3)$

Pf: $T(n) = 2n\lg(n+2)^2 + (n+2)^2\lg\left(\frac{n}{2}\right)$
To prove $T(n) \in \Theta(n^2)$ we need to show there exists constants C and n_0 such that $T(n) \leq C \cdot n$ for all $n \geq n_0$

For $n=1$, $T(1) = 2(1)\lg(1+2)^2 + (1+2)^2\lg\left(\frac{1}{2}\right)$
$= 2\lg(9) + (9)\lg\left(\frac{1}{2}\right) \approx -2.25$

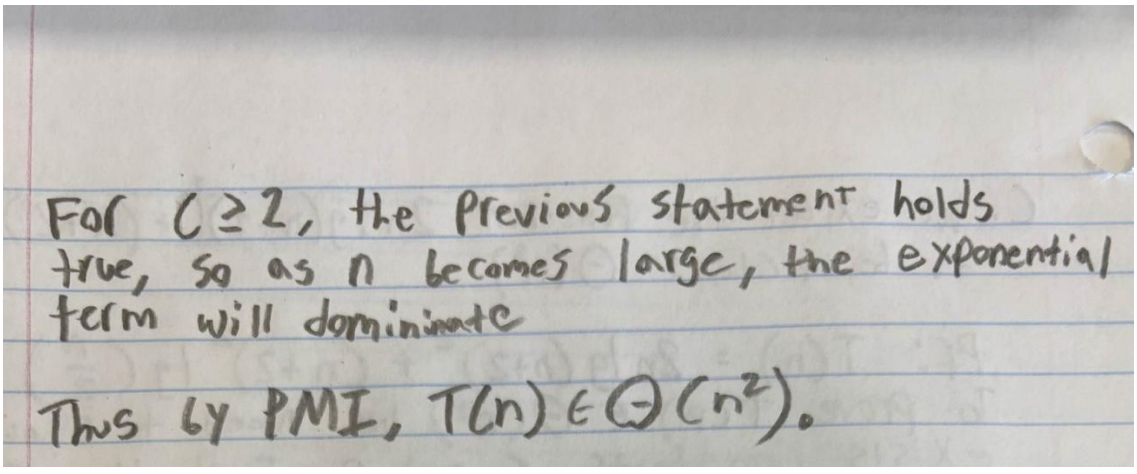
For $n=2$, $T(2) = 2(2)\lg(2+2) + (2+2)^2\lg\left(\frac{2}{2}\right)$
$= 4\lg(16) + (16)\lg(1) \approx 1.45$

$T(2) > T(1)$ So the base case holds.
Hence we assume for some K, so $n=K$
Letting some constant $C=2$

$T(K) \leq C \cdot K^2$

We need to prove when $n=K+1$, $T(K+1) \leq C((K+1)$
Given
$T(K+1) = (K+1+1)^2 = (K+2)^2$
Now using our assumption
$(K+2)^2 \leq C((K+1)^2$

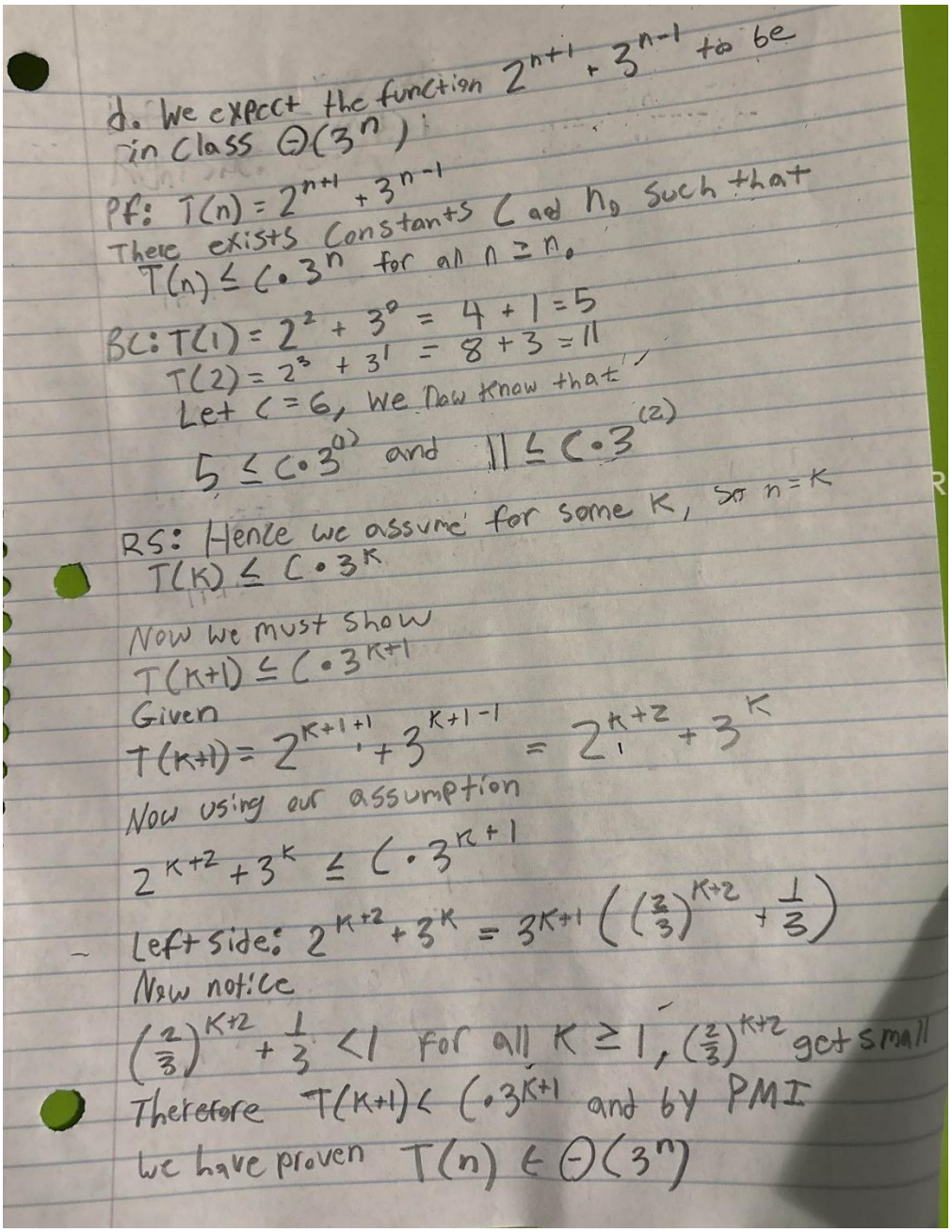
Left side: $(k+2)^2 = K^2+2K+4$
Right side: $2(K+1)^2 = 2(K^2+2K+4) =$
$= 2K^2+4K+8$



For $C \geq 2$, the previous statement holds true, so as n becomes large, the exponential term will dominate

Thus by PMI, $T(n) \in \Theta(n^2)$.

d. $2^{n+1} + 3^{n-1}$



d. We expect the function $2^{n+1} + 3^{n-1}$ to be in class $\Theta(3^n)$

Pf: $T(n) = 2^{n+1} + 3^{n-1}$

There exists constants C and n_0 such that $T(n) \leq C \cdot 3^n$ for all $n \geq n_0$.

BC: $T(1) = 2^2 + 3^0 = 4 + 1 = 5$

$T(2) = 2^3 + 3^1 = 8 + 3 = 11$

Let $C = 6$, we now know that

$5 \leq C \cdot 3^{(1)}$ and $11 \leq C \cdot 3^{(2)}$

RS: Hence we assume for some K, so $n = K$

$T(K) \leq C \cdot 3^K$

Now we must show

$T(K+1) \leq C \cdot 3^{K+1}$

Given

$T(K+1) = 2^{K+1+1} + 3^{K+1-1} = 2^{K+2} + 3^K$

Now using our assumption

$2^{K+2} + 3^K \leq C \cdot 3^{K+1}$

Left side: $2^{K+2} + 3^K = 3^{K+1}\left(\left(\frac{2}{3}\right)^{K+2} + \frac{1}{3}\right)$

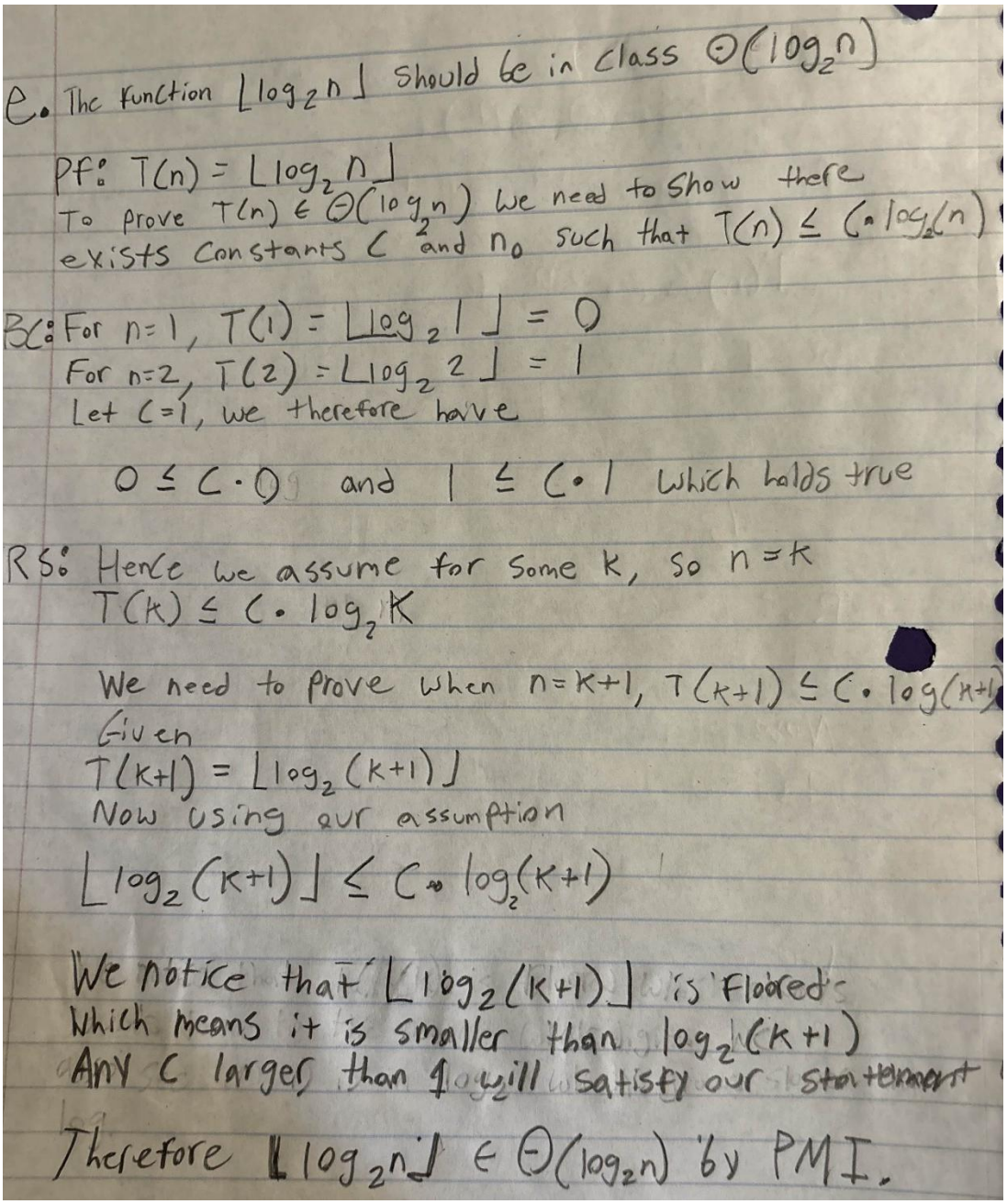
Now notice

$\left(\frac{2}{3}\right)^{K+2} + \frac{1}{3} < 1$ for all $K \geq 1$, $\left(\frac{2}{3}\right)^{K+2}$ gets small

Therefore $T(K+1) < C \cdot 3^{K+1}$ and by PMI

we have proven $T(n) \in \Theta(3^n)$

e. $\lfloor \log_2 n \rfloor$



e. The function $\lfloor \log_2 n \rfloor$ should be in class $\Theta(\log_2 n)$

Pf: $T(n) = \lfloor \log_2 n \rfloor$
To prove $T(n) \in \Theta(\log_2 n)$ we need to show there exists constants C and n_0 such that $T(n) \leq C \cdot \log_2(n)$

BC: For $n=1$, $T(1) = \lfloor \log_2 1 \rfloor = 0$
For $n=2$, $T(2) = \lfloor \log_2 2 \rfloor = 1$
Let $C=1$, we therefore have

$0 \leq C \cdot 0$ and $1 \leq C \cdot 1$ which holds true

RS: Hence we assume for some k, so $n=k$
$T(k) \leq C \cdot \log_2 k$

We need to prove when $n=k+1$, $T(k+1) \leq C \cdot \log(k+1)$
Given
$T(k+1) = \lfloor \log_2 (k+1) \rfloor$
Now using our assumption
$\lfloor \log_2 (k+1) \rfloor \leq C \cdot \log_2 (k+1)$

We notice that $\lfloor \log_2 (k+1) \rfloor$ is floored
which means it is smaller than $\log_2 (k+1)$
Any C larger than 1 will satisfy our statement

Therefore $\lfloor \log_2 n \rfloor \in \Theta(\log_2 n)$ by PMI.