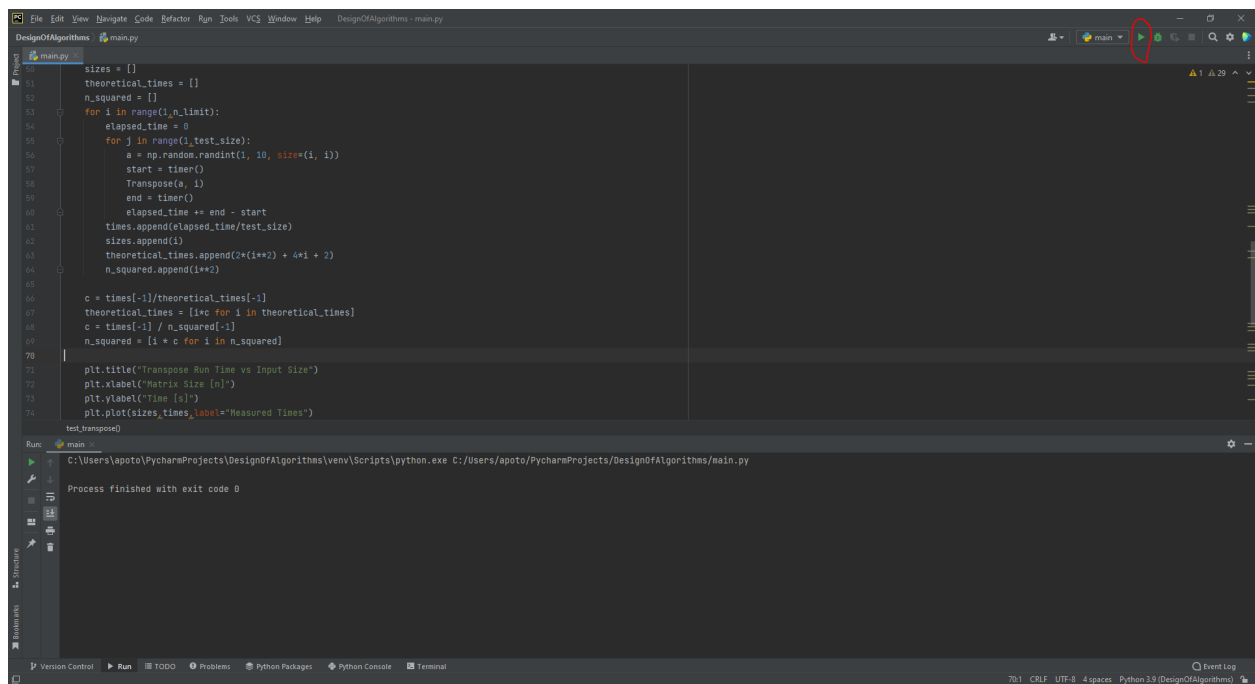


## Homework 1 Report

### Program

The program is written in python3. The program makes use of an external library used to generate plots called matplotlib. Matplotlib can be installed using `pip install matplotlib`. Matplotlib can also be installed using various package installers, such as apt-get, pycharm's library installer, or anaconda. To run the program, either use an IDE to open the script and run in the IDE, or use `python3 main.py` at a command line. Each input size is run 100 times and the arrays contain randomly generated numbers from 1 to 10 for each run. The program does not take arguments, and instead will make 2 graphs, one for the transpose function and one for the matrix multiplication function. I do not use linux or mac for development, so I was unable to figure out how to generate a typescript for this program. I was told to attach screenshots of how to run the program/ the program's output. If run from the IDE:



```
50 sizes = []
51 theoretical_times = []
52 n_squared = []
53 for i in range(1, n_limit):
54     elapsed_time = 0
55     for j in range(1, test_size):
56         a = np.random.randint(1, 10, size=(i, i))
57         start = timer()
58         Transpose(a, i)
59         end = timer()
60         elapsed_time += end - start
61     times.append(elapsed_time/test_size)
62     sizes.append(i)
63     theoretical_times.append(2*(i**2) + 4*i + 2)
64     n_squared.append(i**2)
65
66 c = times[-1]/theoretical_times[-1]
67 theoretical_times = [i*c for i in theoretical_times]
68 c = times[-1] / n_squared[-1]
69 n_squared = [i * c for i in n_squared]
70
71 plt.title("Transpose Run Time vs Input Size")
72 plt.xlabel("Matrix Size [n]")
73 plt.ylabel("Time [s]")
74 plt.plot(sizes, times, label="Measured Times")
75
76 testTranspose()

```

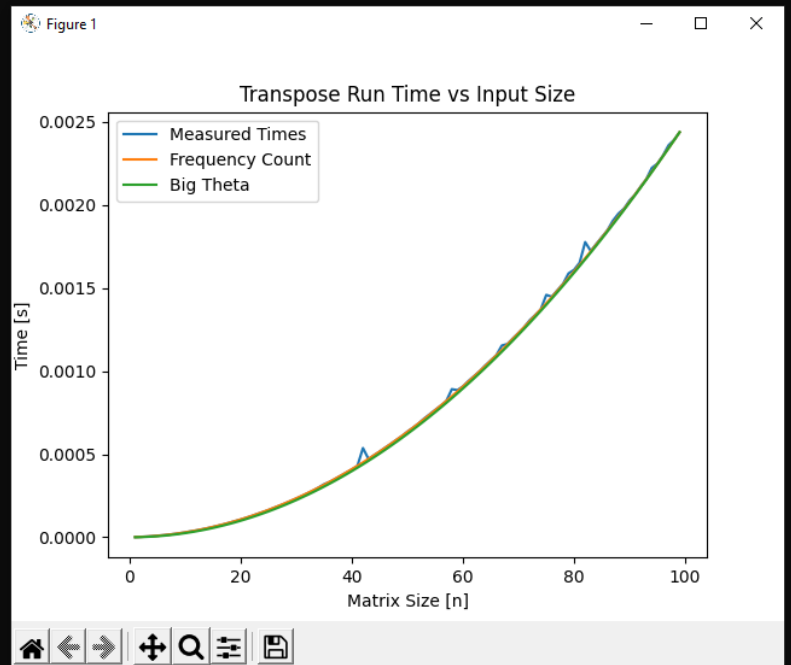
Run: main

C:\Users\apoto\PycharmProjects\DesignOfAlgorithms\venv\Scripts\python.exe C:\Users\apoto\PycharmProjects\DesignOfAlgorithms/main.py

Process finished with exit code 0

Alternatively, the program can be run from the command line using “python3 main.py”. The below also shows a sample output of the program.

```
C:\Users\apoto\PycharmProjects\DesignOfAlgorithms>python3 main.py
```



**As a final note, while I ran everything up to  $n=100$ , I would not recommend running the program with  $n\_limit = 100$  on the matrix multiplication as it will take upwards of 20 minutes to produce a plot.**

### Time Complexity

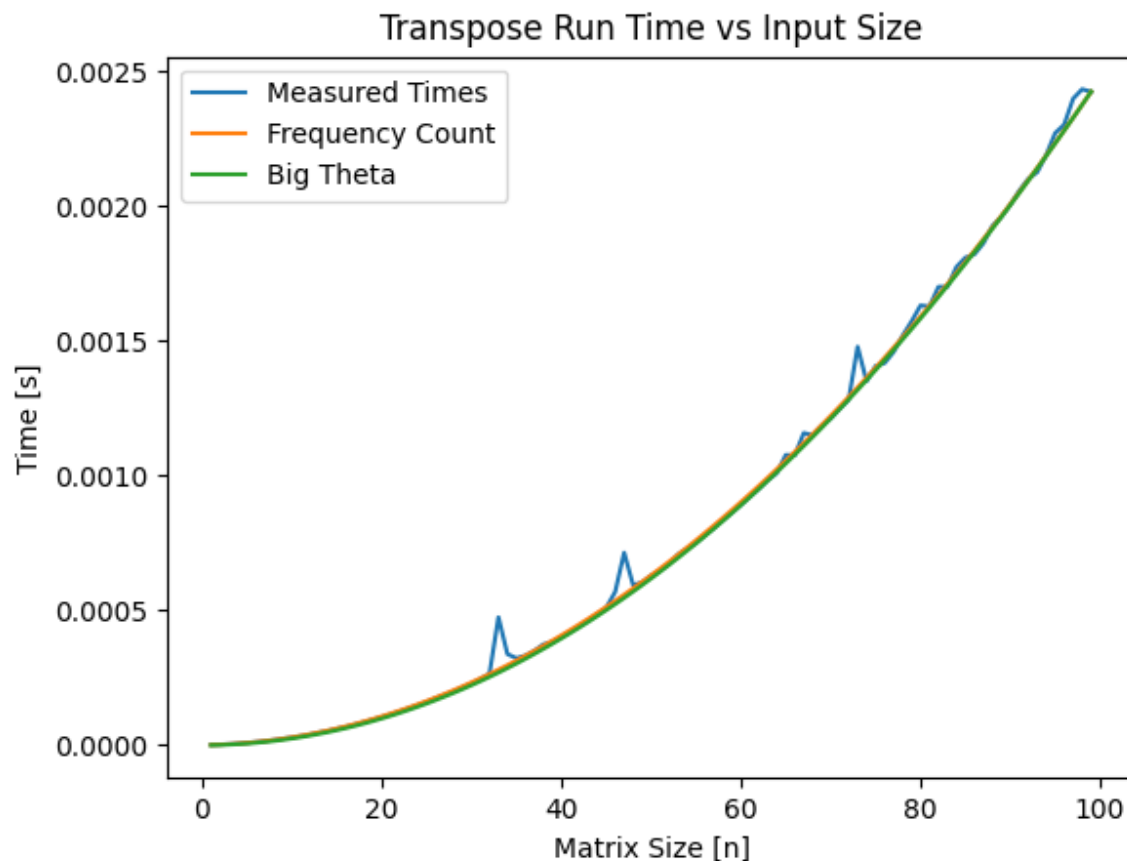
For better comparison, I did my best to get a frequency count for both algorithms, and plotted both the frequency count that I got, as well as the big theta of the frequency counts against the actual timings. The work done to get frequency count and the line by line big theta of each line is attached at the end.

Function	Big Theta	Frequency Count
Transpose	$\Theta(n^2)$	$2n^2 + 4n + 2$

Matrix Multiplication	$\Theta(n^3)$	$2n^3 + 3n^2 + 2n + 2$
-----------------------	---------------	------------------------

### Transpose

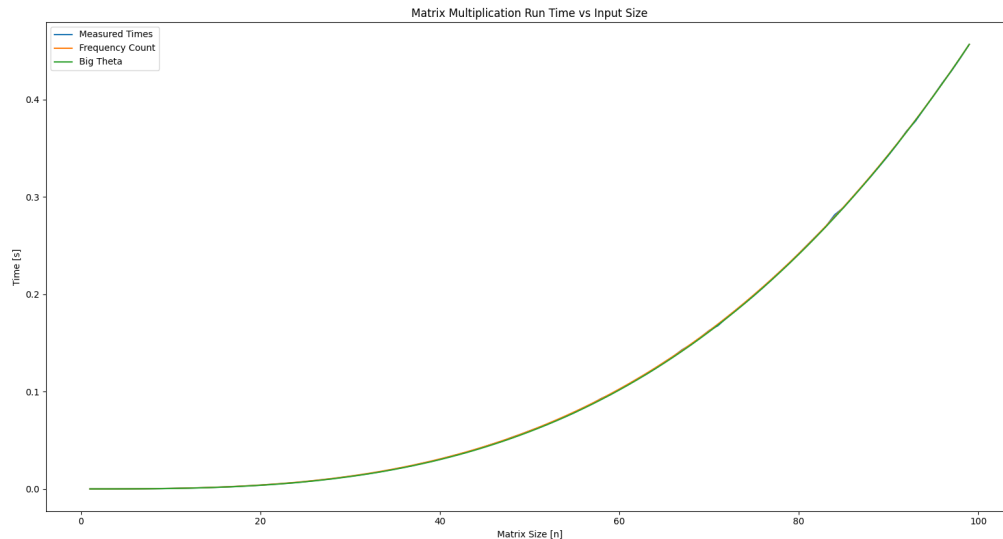
The time complexity of the transpose function was determined to be  $\Theta(n^2)$  which makes sense because traversing either triangle of a square matrix requires going through almost half of the entries in the matrix, and the total number of entries in the matrix follows  $n^2$  since a 1x1 has 1, a 2x2 has 4, and a 3x3 has 9 entries.



The time complexity is clearly a very close fit to the measured time except for a few spikes. The spikes are generated because computers are very complicated and they do many things in the background, any number of things, such as updates, downloads, or other applications could cause the computer to take longer on any given input size, creating the spikes. To alleviate these jumps, each input size could be run for much longer to get a smoother curve, however that does increase the time needed to generate the plot, and I thought that the above demonstrated how close the actual timings are to the predicted time complexity. Both the frequency count and the big theta curve are normalized to the last measured time value.

## Matrix Multiplication

The time complexity of the matrix multiplication was determined to be  $\Theta(n^3)$ , which makes sense because the size of the matrices grows following  $n^2$ , but the computation uses each entry in the matrix  $n$  times for a total of  $n^3$ .



Again, the time complexities are very close to the recorded times. For this algorithm, the lines are basically spot on since the time scale is larger, which means that the different background processes create less deviation on the final plot as the deviations are much smaller when compared to the run time of the algorithm.

## Conclusion

The theoretical time complexities of the algorithms seem to very closely align with the measured timings. This shows that being able to determine the time complexity of an algorithm can be a valuable tool to draw comparisons about the algorithms usage of time.

# Frequency Count Transpose

Python

```

1. def Transpose(a, n)
2.     for i in range(0, n):
3.         for j in range(i, n):
4.             t = a[i][j]
5.             a[i][j] = a[j][i]
6.             a[j][i] = t
7.     return a
    
```

Time complexity

$$\text{total frequency count} = n+1 + \sum_{i=0}^{n-1} (n-i+1) + 3 \sum_{i=0}^{n-1} (n-i) + 1$$

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

$$\sum_{m=1}^n n = \underbrace{n+n+n+\dots+n}_{n \times n = n^2} = n^2$$

$$\sum_{m=1}^n 1 = \underbrace{1+1+1+\dots+1}_{n \times 1 = n} = n$$

$$\sum_{m=1}^n m = \frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2}$$

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

$$\text{total frequency count} = n+1 + \frac{n^2}{2} + 2n - \frac{n}{2} - \frac{n}{2} + 3 \left( \frac{n^2}{2} + \frac{n}{2} - \frac{n^2}{2} - \frac{n}{2} \right) + 1$$

$$= \frac{n^2}{2} - \frac{n^2}{2} + \frac{3n^2}{2} - \frac{3n^2}{2} + n + 2n - \frac{n}{2} + 3n - \frac{3n}{2} + 1 + 1$$

$$= \frac{1}{2} 2n^2 + 4n + 2$$

This is the theoretical frequency count for the transpose function that I plot vs the actual timings

Ex n=3 To check work

2.	4	4
3.	4, 3, 2	9
4.	3, 2, 1	6
5.	3, 2, 1	6
6.	3, 2, 1	6
7.	1	1

$$= 32$$

$$2(3)^2 + 4(3) + 2$$

$$2 \cdot 9 + 12 + 2 = 32$$

Also gives

$\Theta(n^2)$

# Frequency Count Matrix Multiplication

```

1 def Mult(a, b, c, n):
2     for i in range(0, n): n+1  $\Theta(n)$ 
3         for j in range(0, n): n(n+1)  $\Theta(n)$ 
4             c[i][j] = 0 n * n  $\Theta(n^2)$ 
5             for k in range(0, n): n * n * (n+1)  $\Theta(n^3)$ 
6                 c[i][j] += a[i][k] * b[k][j] n * n * n  $\Theta(n^3)$ 
7     return c 1  $\Theta(1)$ 

```

Total  $\Theta(n^3)$

$$\text{total frequency count} = n+1 + n + n + n^2 + n^2 + n + n^3 + 1$$

$$= \boxed{2n^3 + 3n^2 + 2n + 2} \quad \text{confirms } \boxed{\Theta(n^3)}$$

Ex 3 To check work

2. 4	24
3. 4, 4, 4	12
4. 3, 3, 3	9
5. 4, 4, 4, 4, 4, 4, 4, 4	36
6. 3, 3, 3, 3, 3, 3, 3, 3	27
7. 1	1
	89

$$2(3^3) + 3(3^2) + 2(3) + 2$$

$$54 + 27 + 6 + 2 = 89$$



# CS 4310 | Homework #1 | Adrian Potok ①

2b) Prove by induction:

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

Basis  $n=1$ : Show LHS=RHS for  $n=1$

$$\text{LHS } \sum_{i=1}^1 i^2 = (1)^2 = 1$$

$$\text{RHS } \frac{n(n+1)(2n+1)}{6} \Big|_{n=1} = \frac{1(1+1)(2(1)+1)}{6} = \frac{(2)(3)}{6} = 1$$

Since LHS=RHS, the basis is established.

Induction Hypothesis: Assume the property holds for  $n=k$

$$\sum_{i=1}^k i^2 = \frac{k(k+1)(2k+1)}{6}; \quad k \geq 1$$

Induction Step: Show the property holds for  $n=k+1$

$$\sum_{i=1}^{k+1} i^2 = \frac{(k+1)(k+1+1)(2(k+1)+1)}{6} = \frac{(k^2+2k+k+2)(2k+3)}{6}$$

$$\begin{aligned} \sum_{i=1}^{k+1} i^2 &= \underbrace{1+4+9+\dots+k^2}_{\sum_{i=1}^k i^2} + (k+1)^2 = \frac{(k^2+3k+2)(2k+3)}{6} \\ &= \sum_{i=1}^k i^2 + (k+1)^2 = \frac{2k^3+3k^2+6k^2+9k+6}{6} \end{aligned}$$

$$= \frac{k(k+1)(2k+1)}{6} + \frac{6(k+1)^2}{6} \quad \text{RHS} = \frac{2k^3+9k^2+13k+6}{6}$$

$$= \frac{(k^2+k)(2k+1)}{6} + \frac{6(k^2+2k+1)}{6}$$

$$= \frac{2k^3+k^2+2k^2+k+6k^2+12k+6}{6}$$

$$\text{LHS} = \frac{2k^3+9k^2+13k+6}{6}$$

Because we have shown LHS=RHS, we establish that the property holds for  $n=k+1$  & prove the relation by induction.

# CS4310 / Homework #1 / Adnan Potok ②

8c Show the following relations hold

$$2n^2 2^n + n \log(n) = \Theta(n^2 2^n)$$

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{2n^2 2^n + n \log(n)}{n^2 2^n} &= \lim_{n \rightarrow \infty} \frac{2(n^2 2^n)}{n^2 2^n} + \frac{n \log(n)}{n^2 2^n} \\ &= 2 + \lim_{n \rightarrow \infty} \frac{n \log(n)}{n^2 2^n}, \text{ l'Hopital's} \\ &= 2 + \lim_{n \rightarrow \infty} \frac{\log(n) + (n)(1/n)}{(2n)(2^n) + n^2(2^n)(\log 2)}, \text{ l'Hopital's} \\ &= 2 + \lim_{n \rightarrow \infty} \frac{\log n + 1}{(2n)(2^n) + n^2(2^n) \log 2} \end{aligned}$$

$$= 2 + \lim_{n \rightarrow \infty} \frac{1/n}{2(2^n + n(2^n)(\log 2)) + \log 2(2n 2^n + n^2(2^n) \log 2)}$$

$$= 2 + \lim_{n \rightarrow \infty} \frac{0}{\infty} = 2 + 0 = 2$$

Because  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \text{constant} > 0 = 2$

$$\Rightarrow f(n) = \Theta(g(n))$$

or  $2n^2 2^n + n \log n = \Theta(n^2 2^n)$



# CS4310 | Homework #1 | Arden Patoke ③

8d  $\sum_{i=0}^n i^2 = \Theta(n^3)$

from 2b  
↓

$$\sum_{i=0}^n i^2 = \sum_{i=0}^0 i^2 + \sum_{i=1}^n i^2 = n(n+1)(2n+1), n \geq 1$$

$$\lim_{n \rightarrow \infty} \frac{2n^3 + 9n^2 + 13n + 6}{n^3} = \frac{2n^3 + 9n^2 + 13n + 6}{n^3}$$

$$= \lim_{n \rightarrow \infty} \frac{(2n^3 + 9n^2 + 13n + 6)}{n^3} \cdot \text{L'Hopital's Rule}$$

$$= \lim_{n \rightarrow \infty} \frac{(6n^2 + 18n + 13)}{3n^2} \cdot \text{L'Hopital's Rule}$$

$$= \lim_{n \rightarrow \infty} \frac{(12n + 18)}{6n} \cdot \text{L'Hopital's Rule}$$

$$\frac{12}{36} = \frac{1}{3}$$

Since  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \text{constant} > 0$

$$f(n) = \Theta(g(n))$$

or  $\sum_{i=0}^n i^2 = \Theta(n^3)$

# CS 4310 | Homework 1 | Adrian P.

$$8h \quad \frac{6n^3}{\log(n)+1} = O(n^3)$$

$$\lim_{n \rightarrow \infty} \frac{6n^3}{\log(n)+1} = \lim_{n \rightarrow \infty} \frac{6n^3}{\log(n)+1} \left( \frac{1}{n^3} \right) = \lim_{n \rightarrow \infty} \frac{6}{\log(n)+1} = \frac{6}{\log(\infty)+1} = 0$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0 \text{ which means } f(n) = O(g(n))$$

$$\text{or } \boxed{\frac{6n^3}{\log(n)+1} = O(n^3)}$$

# CS430 | Homework #1 | Anton B. Sch

So  $n^{1.001} + n \log(n) = \Theta(n^{1.001})$

$$\lim_{n \rightarrow \infty} \frac{n^{1.001} + n \log(n)}{n^{1.001}} = \lim_{n \rightarrow \infty} 1 + \frac{n \log(n)}{n^{1.001}}$$

L'Hopital's

$$= 1 + \lim_{n \rightarrow \infty} \frac{\log(n) + n(1/n)}{1.001 n^{0.001}} \quad \text{L'Hopital's}$$

$$= 1 + \lim_{n \rightarrow \infty} \frac{1/n}{(1.001)(0.001) n^{-0.999}}$$

$$= 1 + \lim_{n \rightarrow \infty} \frac{n^{0.999}}{(1.001)(0.001)} = 1 + \frac{1}{(1.001)(0.001)} \lim_{n \rightarrow \infty} n^{0.999}$$

$$= 1 + \frac{1}{(1.001)(0.001)} \frac{1}{(\infty)^{0.001}} = 1$$

$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 1 = \text{constant} > 0$  which means

$$f(n) = \Theta(g(n))$$

or  $n^{0.001} + n \log(n) = \Theta(n^{1.001})$



(54310) | Homework | Adrian Polak

$$8m) \quad 33n^3 + 4n^2 = \Omega(n^3)$$

$$\lim_{n \rightarrow \infty} \frac{33n^3 + 4n^2}{n^3} \quad \text{L'Hopital's}$$

$$= \lim_{n \rightarrow \infty} \frac{99n^2 + 8n}{3n^2} \quad \text{L'Hopital's}$$

$$= \lim_{n \rightarrow \infty} \frac{198n + 8}{6n}$$

$$= \lim_{n \rightarrow \infty} 33 + \frac{8}{6n} = 33 + \frac{8}{\infty} = 33$$

because  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 33 = \text{constant} > 0$   
then  $f(n) = \Theta(g(n))$

If  $f(n) = \Theta(g(n))$ , then  $f(n) = \Omega(g(n))$   
as well

or if  $33n^3 + 4n^2 = \Theta(n^3)$  which is shown

by  $\lim_{n \rightarrow \infty} \frac{33n^3 + 4n^2}{n^3} = 33 = \text{constant} > 0$

then it must also follow that  $\boxed{33n^3 + 4n^2 = \Omega(n^3)}$

# CS4310 | Homework #1 | Arthur B. Sore

(7)

9a Show the following is not true

$$10n^2 + 9 = O(n)$$

$$\lim_{n \rightarrow \infty} \frac{10n^2 + 9}{n} = \lim_{n \rightarrow \infty} 10n + \frac{9}{n} = 10(\infty) + \frac{9}{\infty} = \infty$$

Because  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$ ,  $f(n) = \Omega(g(n))$

$$\nexists f(n) = \Theta(g(n))$$

Which means

$$f(n) \neq O(g(n))$$

$$\text{or } 10n^2 + 9 = \Omega(n)$$

$$\text{as } f(n) = \Theta(g(n))$$

$$\nexists 10n^2 + 9 = O(n)$$

$$\text{iff } f(n) = O(g(n)) \nexists$$

$$f(n) = \Omega(g(n))$$

which means

$$\boxed{10n^2 + 9 \neq O(n)}$$



# CS4310 / Homework #1 / Adrian Potok

8

9d Show the following is not true

$$n^3 2^n + 6n^2 3^n = O(n^3 2^n)$$

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{n^3 2^n + 6n^2 3^n}{n^3 2^n} &= \lim_{n \rightarrow \infty} \frac{6n^2 3^n}{n^3 2^n} + \lim_{n \rightarrow \infty} \frac{n^3 2^n}{n^3 2^n} \\ &= \lim_{n \rightarrow \infty} \frac{6}{n} \left(\frac{3}{2}\right)^n + 1 \\ &= \lim_{n \rightarrow \infty} \frac{6 \ln(3/2) (3/2)^n}{1} = \lim_{n \rightarrow \infty} \frac{6 (3/2)^n \ln(3/2)}{1} \\ &= 1 + \infty = \infty \end{aligned}$$

Because  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$

$f(n) = \Omega(g(n)) \nmid f(n) = O(g(n))$

Because  $f(n) = \Theta(g(n))$  iff

$f(n) = \Omega(g(n)) \nmid f(n) = O(g(n))$   
 $\nmid f(n) = \Omega(g(n)), f(n) \neq O(g(n))$   
 else it would necessitate  $f(n) = \Theta(g(n))$

Or

Because  $n^3 2^n + 6n^2 3^n = \Omega(n^3 2^n)$

$\nmid n^3 2^n + 6n^2 3^n \neq O(n^3 2^n)$

then  $n^3 2^n + 6n^2 3^n \neq O(n^3 2^n)$