

# IbraFSG™ 8 - Week 12; Algorithms and Efficiency

Ibrahim Chehab

UTM RGASC

April 2, 2024

# Table of Contents

## 1 Introduction

- FSG HouseKeeping
- Welcome back to IbraFSGs™
- A Recap of the UltraSheet™

## 2 Algorithms and Efficiency

- Key Terms

## 3 Practice Problems

- Practice Problem I: The Tightest Bound
- Practice Problem II: FindMiiComplexity

## 4 Conclusion

- A Final Challenge...
- Goodbye, for now...

- ① **Reminder that a new FSG Session has been added:**
  - **When?** *Fridays, 12:00-1:00PM*
  - **Where?** *IB210*

- ① **Reminder that a new FSG Session has been added:**
  - **When?** *Fridays, 12:00-1:00PM*
  - **Where?** *IB210*
- ② **Assignment 2 Deadline Coming Up!** Assignment 2 is due today at 5:00PM! Don't forget to submit it on MarkUs!

- ① **Reminder that a new FSG Session has been added:**
  - **When?** *Fridays, 12:00-1:00PM*
  - **Where?** *IB210*
- ② **Assignment 2 Deadline Coming Up!** Assignment 2 is due today at 5:00PM! Don't forget to submit it on MarkUs!
- ③ **New Study Session Drop Ins!**

Feeling Stressed about exams? Don't know where to start? Drop by the MN3260 for a study skills session!

  - **When:** April 3<sup>rd</sup> 1-2PM
  - **Where:** MN3260

- ① **Reminder that a new FSG Session has been added:**
  - **When?** *Fridays, 12:00-1:00PM*
  - **Where?** *IB210*
- ② **Assignment 2 Deadline Coming Up!** Assignment 2 is due today at 5:00PM! Don't forget to submit it on MarkUs!
- ③ **New Study Session Drop Ins!**

Feeling Stressed about exams? Don't know where to start? Drop by the MN3260 for a study skills session!

  - **When:** April 3<sup>rd</sup> 1-2PM
  - **Where:** MN3260
- ④ **Join the UTM CS Discord Server!** <https://discord.gg/utmcs>
  - **Note:** The server is not affiliated with the RGASC, CSC148H5, or the University of Toronto Mississauga

- ① **Reminder that a new FSG Session has been added:**
  - **When?** *Fridays, 12:00-1:00PM*
  - **Where?** *IB210*
- ② **Assignment 2 Deadline Coming Up!** Assignment 2 is due today at 5:00PM! Don't forget to submit it on MarkUs!
- ③ **New Study Session Drop Ins!**

Feeling Stressed about exams? Don't know where to start? Drop by the MN3260 for a study skills session!

  - **When:** April 3<sup>rd</sup> 1-2PM
  - **Where:** MN3260
- ④ **Join the UTM CS Discord Server!** <https://discord.gg/utmcs>
  - **Note:** The server is not affiliated with the RGASC, CSC148H5, or the University of Toronto Mississauga
- ⑤ **MEGA FSG Coming Soon!** Stay tuned for a MEGA FSG session to help you prepare for the final exam!

# Welcome back to IbraFSGs™

- Welcome back to IbraFSGs™! Hello to new people and welcome back to tenured members.
- This week we will be going over algorithms and complexity analysis
- Today's session will act as a prelude to *CSC236*
  - *CSC236* is the course where you will learn about the theory of computation
  - The entire course has a backbone of *induction* and *recursion*
  - Towards the end of the course, you will learn about DFAs, NFAs, and Turing Machines (i.e: a DFA is a quintuple  $(Q, \Sigma, \delta, q_0, F)$ )



# Welcome back to IbraFSGs™

- Welcome back to IbraFSGs™! Hello to new people and welcome back to tenured members.
- This week we will be going over algorithms and complexity analysis
- Today's session will act as a prelude to *CSC236*
  - *CSC236* is the course where you will learn about the theory of computation
  - The entire course has a backbone of *induction* and *recursion*
  - Towards the end of the course, you will learn about DFAs, NFAs, and Turing Machines (i.e: a DFA is a quintuple  $(Q, \Sigma, \delta, q_0, F)$ )
- <https://www.bigocheatsheet.com/>

# Welcome back to IbraFSGs™

- Welcome back to IbraFSGs™! Hello to new people and welcome back to tenured members.
- This week we will be going over algorithms and complexity analysis
- Today's session will act as a prelude to *CSC236*
  - *CSC236* is the course where you will learn about the theory of computation
  - The entire course has a backbone of *induction* and *recursion*
  - Towards the end of the course, you will learn about DFAs, NFAs, and Turing Machines (i.e: a DFA is a quintuple  $(Q, \Sigma, \delta, q_0, F)$ )
- <https://www.bigocheatsheet.com/>
- **Fun Fact:** In *CSC263*, you will use something called the *Master Theorem* to analyze the complexity of divide and conquer algorithms like *MergeSort* and *QuickSort*

## Bonus! The Master Theorem: A Comprehensive Overview

**NOTE: THIS IS COMPLETELY OUT OF SCOPE FOR CSC148! JUST HERE FOR FUN FACT!!** The Master Theorem provides a method for solving recurrence relations of the form:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where,

- $a \geq 1$  and  $b > 1$  are constants,
- $f(n)$  is an asymptotically positive function.

**The theorem is divided into three cases:**

- ❶ **Case 1:** If  $f(n) \in O(n^{\log_b a - \varepsilon})$  for some  $\varepsilon > 0$ , then  $T(n) \in \Theta(n^{\log_b a})$ .
- ❷ **Case 2:** If  $f(n) \in \Theta(n^{\log_b a} \log^k n)$  for some  $k \geq 0$ , then  $T(n) \in \Theta(n^{\log_b a} \log^{k+1} n)$ .
- ❸ **Case 3:** If  $f(n) \in \Omega(n^{\log_b a + \varepsilon})$  for some  $\varepsilon > 0$ , and if  $af\left(\frac{n}{b}\right) \leq cf(n)$  for some  $c < 1$  and sufficiently large  $n$ , then  $T(n) \in \Theta(f(n))$ .

*Tl;Dr - CS isn't all about coding, there's a lot of math involved too!*

*Tldrdr - CS gets real hard real fast*

# A Recap of the UltraSheet™

- An *UltraSheet™* is a "cheat sheet" that you compile for **yourself** to review course materials
  - Sharing UltraSheets™ is **counter-productive** and **will not help you learn the material**
  - However, reviewing content in a group and simultaneously updating your UltraSheets™ is a good idea
- It acts like your own personalized textbook chapter
  - It allows you to **regurgitate all the course information in a contiguous, organized manner** and helps you **find gaps in your knowledge**
  - You should **not** be copying the textbook or lecture slides verbatim; You should be **summarizing** the content in your own words while tying in examples and analogies
- UltraSheets™ help with type 1 and 2 questions
- In case you didn't notice yet, Petersen *loves Type 1 and Type 2* questions - Use this information how you will ;P

**Required Key Terms:** The following key terms are required for this week's content. You should be able to define and explain these terms in your UltraSheets™:

- **Big-O Notation ( $\mathcal{O}(n)$ )**

- Upper Bound
- A function  $f(n)$  is  $\mathcal{O}(g(n))$  if there exists a constant  $c > 0$  and  $n_0 > 0$  such that  $f(n) \leq c \cdot g(n)$  for all  $n \geq n_0$

# Key Terms

**Required Key Terms:** The following key terms are required for this week's content. You should be able to define and explain these terms in your UltraSheets™:

- **Big-O Notation ( $\mathcal{O}(n)$ )**

- Upper Bound
- A function  $f(n)$  is  $\mathcal{O}(g(n))$  if there exists a constant  $c > 0$  and  $n_0 > 0$  such that  $f(n) \leq c \cdot g(n)$  for all  $n \geq n_0$

- **Big-Theta Notation ( $\Theta(n)$ )**

- Tight Bound
- A function  $f(n)$  is  $\Theta(g(n))$  if and only if  $f(n)$  is  $\mathcal{O}(g(n))$  and  $f(n)$  is  $\Omega(g(n))$
- In other words;  $f(n)$  is  $\Theta(g(n))$  if and only if there exist constants  $c_1 > 0$ ,  $c_2 > 0$ , and  $n_0 > 0$  such that  $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$  for all  $n \geq n_0$

# Key Terms (Cont'd)

**Suggested Key Terms:** The following key terms are *recommended* for further studying this week's content. Most are out of the scope of the course and are **not** required.

- **Big-Omega Notation ( $\Omega(n)$ )**

- Lower Bound
- A function  $f(n)$  is  $\Omega(g(n))$  if there exists a constant  $c > 0$  and  $n_0 > 0$  such that  $f(n) \geq c \cdot g(n)$  for all  $n \geq n_0$

- **The Master Theorem**

- A method for solving recurrence relations of the form  $T(n) = aT\left(\frac{n}{b}\right) + f(n)$
- Allows for finding time complexity of divide and conquer algorithms

- **Induction (Weak, Strong, Structural)**

- Recall this from MAT102
- Used to formally prove the correctness and efficiency-class of algorithms

Aaaaand.... that's it for today!

## Example of CSC236 Problem (1/3):

**Question 1:** Show that if  $f \in \mathcal{O}(g)$  and  $g \in \Theta(h)$ , then  $f + g \in \Theta(h)$ .

### Proof.

Given  $f \in \mathcal{O}(g)$ , there exists  $c_1, n_0 > 0$  such that  $f(n) \leq c_1 g(n)$  for all  $n \geq n_0$ . For  $g \in \Theta(h)$ , there are constants  $c_2, c_3, n_1 > 0$  making  $c_2 h(n) \leq g(n) \leq c_3 h(n)$  for all  $n \geq n_1$ .

We aim to find  $c_4, c_5, n_2 > 0$  satisfying  $c_4 h(n) \leq f(n) + g(n) \leq c_5 h(n)$  for all  $n \geq n_2$ . Combining our initial conditions, we get:

$$\begin{aligned} 0 &\leq f(n) \leq c_1 g(n) \\ c_2 h(n) &\leq g(n) \leq c_3 h(n) \end{aligned}$$





## Example of CSC236 Problem (2/3):

Continuation of Proof:

**Proof (continued).**

Combining the given, for  $n \geq \max(n_0, n_1)$ :

$$c_2 h(n) \leq f(n) + g(n) \leq c_1 g(n) + c_3 h(n)$$

## Example of CSC236 Problem (2/3):

### Continuation of Proof:

#### Proof (continued).

Combining the given, for  $n \geq \max(n_0, n_1)$ :

$$c_2 h(n) \leq f(n) + g(n) \leq c_1 g(n) + c_3 h(n)$$

From the definition of  $\Theta(n)$ , we know that  $g(n)$  is upper-bounded by  $c_3 \cdot h(n)$  for all  $n \geq n_1$ . Hence, we can simplify the above inequality to:

$$\implies c_2 \cdot h(n) \leq f(n) + g(n) \leq c_1 \cdot (c_3 \cdot h(n)) + c_3 \cdot h(n) \quad \text{for all } n \geq \max(n_0, n_1)$$

## Example of CSC236 Problem (2/3):

### Continuation of Proof:

#### Proof (continued).

Combining the given, for  $n \geq \max(n_0, n_1)$ :

$$c_2 h(n) \leq f(n) + g(n) \leq c_1 g(n) + c_3 h(n)$$

From the definition of  $\Theta(n)$ , we know that  $g(n)$  is upper-bounded by  $c_3 \cdot h(n)$  for all  $n \geq n_1$ . Hence, we can simplify the above inequality to:

$$\implies c_2 \cdot h(n) \leq f(n) + g(n) \leq c_1 \cdot (c_3 \cdot h(n)) + c_3 \cdot h(n) \quad \text{for all } n \geq \max(n_0, n_1)$$

Factoring out  $h(n)$  from the RHS, we get:

$$\implies c_2 \cdot h(n) \leq f(n) + g(n) \leq (c_1 \cdot c_3 + 1) \cdot h(n) \quad \text{for all } n \geq \max(n_0, n_1)$$



## Example of CSC236 Problem (3/3):

### Finishing off the Proof:

#### Proof (continued).

$$\implies c_2 \cdot h(n) \leq f(n) + g(n) \leq (c_1 \cdot c_3 + 1) \cdot h(n) \quad \text{for all } n \geq \max(n_0, n_1)$$

Take  $c_4 = c_2$  and  $c_5 = c_1 \cdot c_3 + 1$ . We have shown that  $f(n) + g(n)$  is upper-bounded by  $(c_1 \cdot c_3 + 1) \cdot h(n)$  for all  $n \geq \max(n_0, n_1)$ . We can also show that  $f(n) + g(n)$  is lower-bounded by  $c_2 \cdot h(n)$  for all  $n \geq \max(n_0, n_1)$ .

Hence, we have shown that  $f(n) + g(n) \in \Theta(h(n))$ , thus completing the proof. □

*Note: This proof is a bit more advanced than what you would see in CSC148. It's meant to give you a taste of what you'll see in CSC236.*

*Note 2: This proof has been heavily condensed for time purposes. The actual proof would be more detailed and rigorous.*

## Jamboard Link



<https://tinyurl.com/ibrafsg0403>

*0403 for April 3<sup>rd</sup>*

**Note:** Jamboard isn't ideal because of its limitations, hence I'm working on my own alternative. Stay tuned!

## Practice Problem I: The Tightest Bound

Peace and d.aki are arguing over the tightest upper-bound for the following function. Peace argues that it's  $\mathcal{O}(n^2)$  while d.aki argues that it's  $\mathcal{O}(n^3)$ . Given the following function, who is correct? Justify your answer.

```
def mystery(n: int):  
    L = []  
    for i in range(n):  
        for j in range(min(50, n)):  
            L.insert(0, j)
```

*Hint: Recall the definition of  $\mathcal{O}$  notation*

*Hint 2: Consider modeling a piecewise function  $T(n)$  dictating the number of steps taken by the function for arbitrary input  $n$*

## Practice Problem 1: The Tightest Bound

Peace and d.aki are arguing over the tightest upper-bound for the following function. Peace argues that it's  $\mathcal{O}(n^2)$  while d.aki argues that it's  $\mathcal{O}(n^3)$ . Given the following function, who is correct? Justify your answer.

```
def mystery(n: int):  
    L = []  
    for i in range(n):  
        for j in range(min(50, n)):  
            L.insert(0, j)
```

*Hint: Recall the definition of  $\mathcal{O}$  notation*

*Hint 2: Consider modeling a piecewise function  $T(n)$  dictating the number of steps taken by the function for arbitrary input  $n$*

Recall: For a function  $f$  to be an element of  $\mathcal{O}(g)$ , there must exist constants  $c > 0$  and  $n_0 > 0$  such that  $f(n) \leq c \cdot g(n)$  for all  $n \geq n_0$

## Practice Problem I Debrief

What if we changed the min to max? Abdulkader thinks it's still  $\mathcal{O}(n^2)$ , but drew thinks it's  $\mathcal{O}(n^3)$ . Who is correct. Justify your answer.

```
def mystery2(n: int):  
    L = []  
    for i in range(n):  
        for j in range(max(50, n)):  
            L.insert(0, j)
```

*Hint: Modify your piecewise function  $T(n)$  to account for the change in the function, then compare the two functions.*



## Practice Problem II: FindMiiComplexity

What is the worst-case asymptotic complexity of the following function? Justify your answer.

```
def foo(x: int):  
    n = x  
    for i in range(n):  
        if n <= 50:  
            for j in range(n):  
                x += 1  
        if n <= 100:  
            for j in range(n):  
                if n <= 75:  
                    for k in range(n):  
                        x += 1
```

*Hint: Split each part of the function into its own piecewise function  $T(n)$ , then combine them to find the overall complexity.*

*Hint 2: This problem is probably harder than anything you'll ever get in CSC148*

## Practice Problem II: FindMiiComplexity

What is the worst-case asymptotic complexity of the following function? Justify your answer.

```
def foo(x: int):  
    n = x  
    for i in range(n):  
        if n <= 50:  
            for j in range(n):  
                x += 1  
        if n <= 100:  
            for j in range(n):  
                if n <= 75:  
                    for k in range(n):  
                        x += 1
```

*Hint: Split each part of the function into its own piecewise function  $T(n)$ , then combine them to find the overall complexity.*

*Hint 2: This problem is probably harder than anything you'll ever get in CSC148*

**Bonus (time permitting):** Change one line of this function to make it  $\mathcal{O}(n)$

## A Final Challenge...

The following question is definitely a *CSC236*-level problem, and you will NOT be tested on it in *CSC148* - I REPEAT: **YOU WILL NOT BE TESTED ON THIS IN CSC148**. However, it's a fun problem to think about and will give you a taste of what you'll see in *CSC236*.

Thank you for coming!



**Figure 1:** Image courtesy of Looney Tunes™ and Warner Bros.™

# Thank you for coming!



**Figure 2:** Ibrahim right now because FSGs are over

Genuinely, thank you for coming to my FSGs. If anything, I hope I made CSC148 easier for you. This has truly been an amazing experience for me, and I hope to see you all in the future (potentially as a TA :eyes:) Feel free to reach out to me on Discord or on the UTM CS Discord server if you have any questions or need help with anything.

**In all seriousness, keep an eye out for our MEGA FSG coming soon....**