


## ASSIGNMENT 2 FRONT SHEET

<b>Qualification</b>	<b>BTEC Level 5 HND Diploma in Business</b>		
<b>Unit number and title</b>	Unit 19: Data Structures and Algorithms		
<b>Submission date</b>	August 19 <sup>th</sup> , 2021	<b>Date Received 1<sup>st</sup> submission</b>	August 19 <sup>th</sup> , 2021
<b>Re-submission Date</b>		<b>Date Received 2<sup>nd</sup> submission</b>	
<b>Student Name</b>	NGUYEN HUU HOANG	<b>Student ID</b>	BDAF190022
<b>Class</b>	BH-AF-2005-2.3	<b>Assessor name</b>	NGO THI MAI LOAN
<b>Student declaration</b> I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.			
		<b>Student's signature</b>	

### Grading grid

P4	P5	P6	P7

☐ **Summative Feedback:**

☐ **Resubmission Feedback:**

**Grade:**

**Assessor Signature:**

**Date:**

**Signature & Date:**

## Contents

A. INTRODUCTION .....	6
B. CONTENT.....	6
LO1. IMPLEMENT COMPLEX DATA STRUCTURES AND ALGORITHMS .....	6
(P4) IMPLEMENT A COMPLEX ADT AND ALGORITHM IN AN EXECUTABLE PROGRAMMING LANGUAGE TO SOLVE A WELL -DEFINED PROBLEM. ....	6
1. Design ADT .....	6
2. Execute ADT .....	13
(P5) PERFORM ERROR HANDLING AND TEST RESULTS REPORT.....	15
1. Java Exceptions - Try...Catch .....	15
2. Implement error handling .....	16
3. Test case and test result.....	18
(P6) ASYMPTOTIC ANALYSIS TECHNIQUE IS USED TO EVALUATE THE EFFICIENCY OF THE ALGORITHM.....	20
1. What is Algorithm Analysis?.....	20
2. What is Asymptotic Analysis?.....	20
3. Asymptotic Notations .....	21
a. Big O notation .....	21
b. Omega Notation.....	22
c. Theta Notation .....	22
(P7) IDENTIFY TWO WAYS IN WHICH THE EFFICIENCY OF AN ALGORITHM CAN BE MEASURED, ILLUSTRATING YOUR ANSWER WITH AN EXAMPLE. ....	23
1. Algorithmic complexity .....	23
2. What is space complexity?.....	23
a. Definition.....	23
b. Example .....	24
3. What is time complexity? .....	25

a.	Why is Time Complexity Essential .....	26
b.	Definition.....	26
c.	Example .....	26
d.	General Rules for Estimation .....	27
C.	CONCLUSION.....	28
D.	REFERENCES .....	28

## List of Figures

Figure 1 Class Node.....	7
Figure 2 class MyQueue .....	8
Figure 3 class MyStack.....	9
Figure 4 Option 1 .....	11
Figure 5 Option 2 .....	11
Figure 6 Option 3 .....	12
Figure 7 Option 4 .....	12
Figure 8 Test option 1 .....	13
Figure 9 Test option 2 .....	13
Figure 10 Test option 3 .....	14
Figure 11 Test option 4 .....	14
Figure 12 Test option 4.1 .....	14
Figure 13 Try ... catch Syntax.....	15
Figure 14 Try ... catch Example.....	16
Figure 15 Check empty in MyQueue class .....	16
Figure 16 Check empty in MyStack class.....	17
Figure 17 Check exception in main class .....	17
Figure 18 Run program before use try ... catch .....	18
Figure 19 Result -Check exception in main class .....	18
Figure 20 Algorithm analysis.....	20
Figure 21 Asymptotic analysis.....	21

Figure 22 Big O notation .....	22
Figure 23 Omega Notation.....	22
Figure 24 Theta Notation .....	23
Figure 25 What is space complexity .....	24
Figure 26 Space complexity.....	25
Figure 27 sum of all elements in the array.....	25
Figure 28 Time complexity.....	26
Figure 29 A sequence of operations.....	26
Figure 32 Nested Loop.....	27

## List of Tables

Table 1 Information test.....	18
Table 2 Test case & result.....	19

## A. INTRODUCTION

Continued from Assignment 1, in this assignment 2, I was asked to design ADT / algorithms for these 2 structures and implement a demo version with a message in a string of a maximum of 250 characters. The program uses **ADT (STACK / Queue)** as a buffer during data transmission and reception. I need to write a report on the implementation of the 2 data structures and how to measure the efficiency of related algorithms. The report should also evaluate the use of ADT in design and development, including the complexity, the trade-off and the benefits.

## B. CONTENT

### LO1. IMPLEMENT COMPLEX DATA STRUCTURES AND ALGORITHMS

#### (P4) IMPLEMENT A COMPLEX ADT AND ALGORITHM IN AN EXECUTABLE PROGRAMMING LANGUAGE TO SOLVE A WELL -DEFINED PROBLEM.

##### 1. Design ADT

In this assignment, I have been tasked with designing and implementing a complex ADT. Specific tasks are as follows: I was asked to design ADT / algorithms for these 2 structures and implement a demo version with a message in a string of a maximum of 250 characters. The program uses Queue data structure and Stack data structure as a buffer during data transmission and reception (message).

Here is my blueprint: Enter 1 to enter the message, type "done" to end, if the message exceeds 250 characters, an error message will be reported., enter 2 to send the message, enter 3 to display the message, enter 4 to delete the message just entered, and 5 to exit the program.

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    MyStack ms = new MyStack();
    MyQueue mq = new MyQueue();

    try {
        while (true){
            System.out.println("ENTER YOUR OPTIONS");
            System.out.println("1. ENTERING THE MESSAGE");
            System.out.println("2. SENDING THE MESSAGE");
            System.out.println("3. DISPLAYING THE MESSAGE");
            System.out.println("4. DELETE MESSAGE JUST ENTERED");
            System.out.println("5. EXIT");
            System.out.println("-----");
```

I will use Linked list to design for Node class and MyQueue class. So all messages will be stored in class MyQueue.

```
1      package com.company;  
2  
3      public class Node {  
4          String data;  
5          Node next;  
6  
7          public Node(String data) {  
8              this.data = data;  
9              this.next = null;  
10         }  
11  
12     }
```

*Figure 1 Class Node*

```

3      public class MyQueue {
4
5          public Node head;
6          private Node tail;
7          public int size;
8
9          public MyQueue() {
10             head = null;
11             tail = null;
12             size = 0;
13         }
14         boolean isEmpty(){
15             return head == null;
16         }
17         void enqueue(Node node){
18             if (isEmpty()){
19                 head = tail = node;
20             }else{
21                 tail.next = node;
22                 tail = node;
23             }
24             size++;
25         }
26         Node dequeue() {
27             Node node = null;
28             if (isEmpty()){
29                 node = head;
30                 head = head.next;
31             }else{
32                 System.out.println("Empty Queue!");
33             }
34             return node;
35         }
36         void display(){
37             while (head != null){
38                 System.out.println(head.data);
39                 head = head.next;
40             }
41         }
42     }

```

Figure 2 class MyQueue



Next, I use the MyStack class to process messages, store received messages and display the messages on the screen.

```
3      public class MyStack {
4
5          Node top;
6
7          public MyStack(){
8              this.top = null;
9          }
10         boolean isEmpty() { return top == null; }
13     @ void push(Node node){
14         node.next = top;
15         top = node;
16     }
17     Node pop(){
18         Node x = top;
19         top = top.next;
20         return x;
21     }
22     void display(){
23         Node current=top;
24         while (current != null){
25             System.out.println(current.data);
26             current = current.next;
27         }
28     }
29 }
30
```

Figure 3 class MyStack

And finally, the main program, this is the class that will create an interface for you to input and output messages, below is the code of the program

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    MyStack ms = new MyStack();
    MyQueue mq = new MyQueue();

    try {
        while (true){
            System.out.println("ENTER YOUR OPTIONS");
            System.out.println("1. ENTERING THE MESSAGE");
            System.out.println("2. SENDING THE MESSAGE");
            System.out.println("3. DISPLAYING THE MESSAGE");
            System.out.println("4. DELETE MESSAGE JUST ENTERED");
            System.out.println("5. EXIT");
            System.out.println("-----");

            int choice = sc.nextInt();
            switch (choice){
                case 1:
                    System.out.println("Enter the message and type 'done' to exit:");
                    String mess = "";
                    while (!mess.equals("done")){
                        mess = sc.nextLine();
                        if(!mess.equals("done") && !mess.isEmpty() && mess.length() < 250){
                            mq.enqueue(new Node(mess));
                        }
                    }
                    break;
                case 2:
                    if (mq.isEmpty()){
                        System.out.println("No messages have been sent yet!\n");
                    }else{
                        while (!mq.isEmpty()){
                            ms.push(mq.dequeue());
                        }
                        System.out.println("Message has been sent!\n");
                    }
                    break;
                case 3:
                    System.out.println("The message received is :");
                    ms.display();
                    System.out.println();
                    break;
                case 4:
                    ms.pop();
                    System.out.println("A message has been removed!");
                    break;
                case 5:
                    System.exit(0);
                    break;
                default:
                    System.out.println("Wrong key, please re-enter!\n");
                    break;
            }
        }
    } catch (Exception ex){
        System.out.println("Enter the wrong key!");
    }
}
```

## 2. Activity description Flowchart

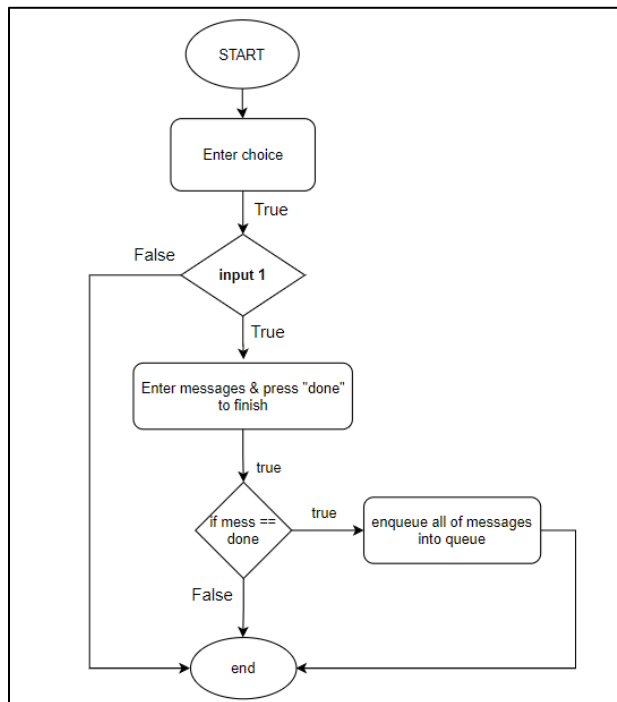


Figure 4 Option 1

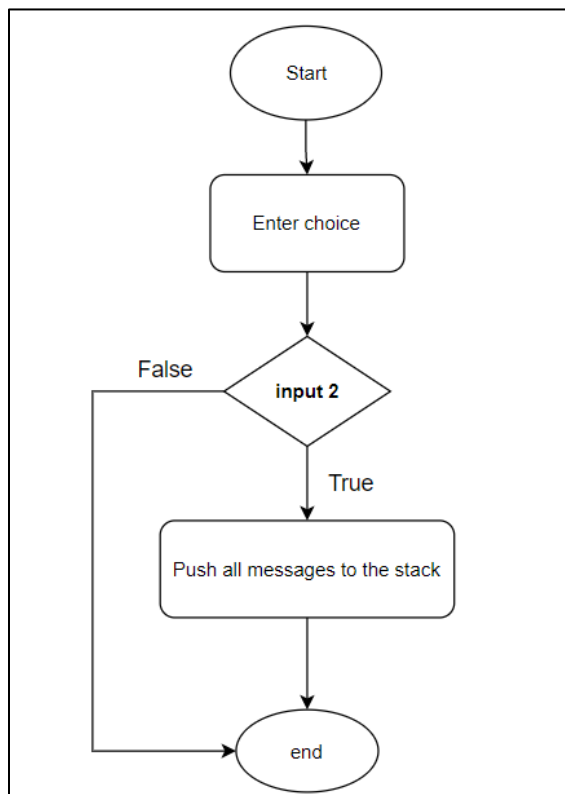
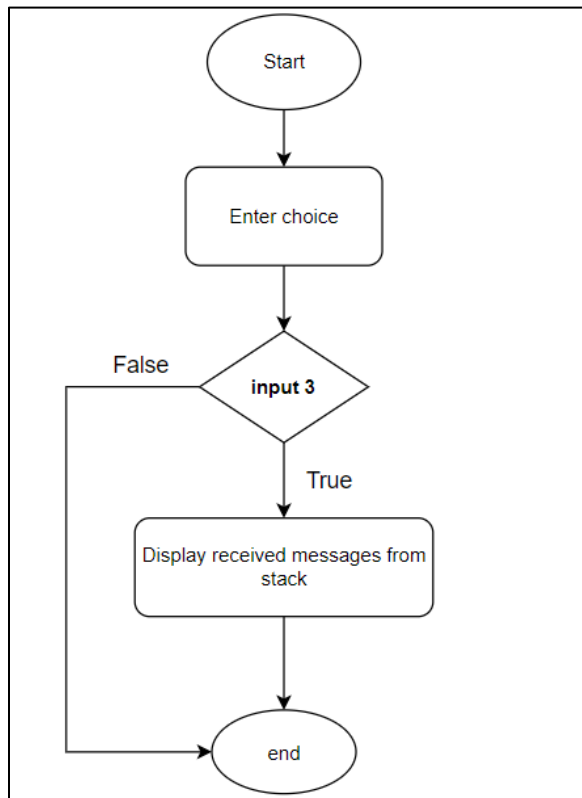
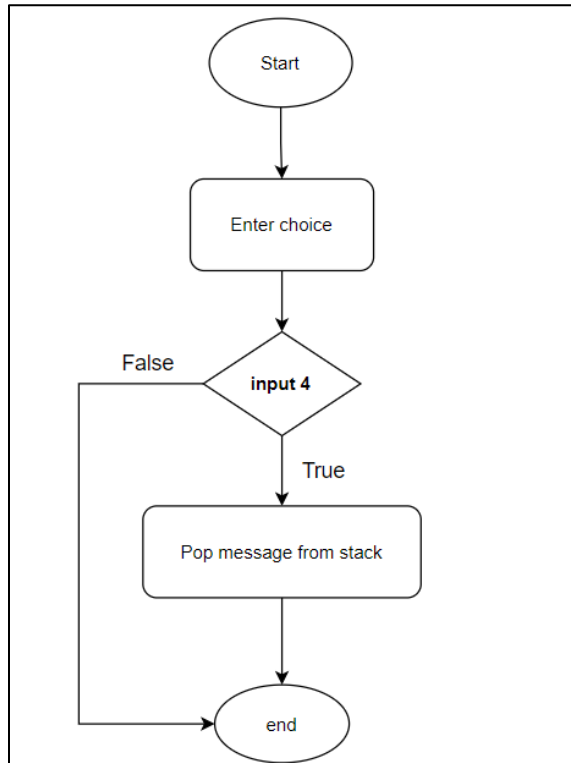


Figure 5 Option 2



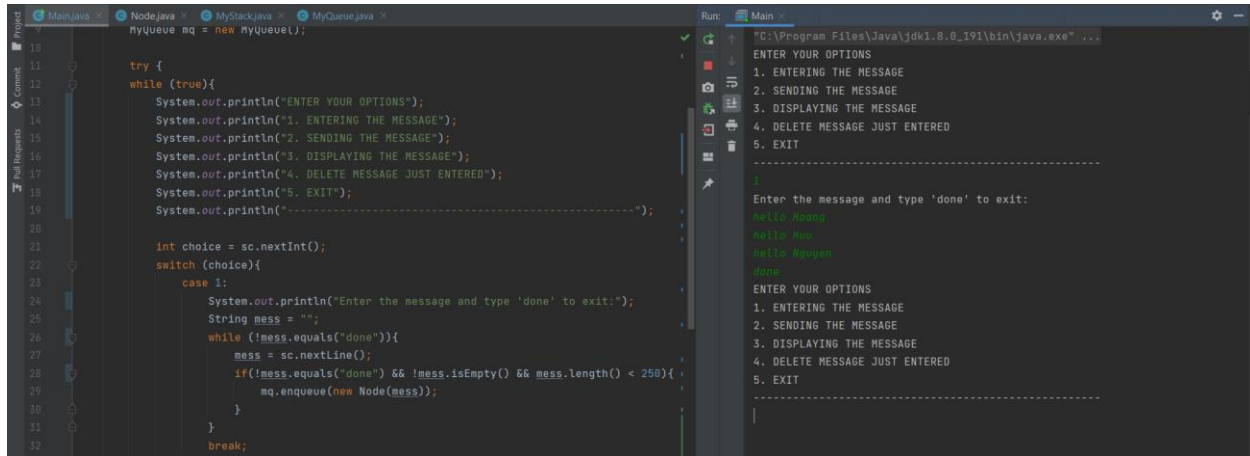
*Figure 6 Option 3*



*Figure 7 Option 4*

### 3. Execute ADT

#### ❖ Option 1: Enter the message



```

11 try {
12     while (true){
13         System.out.println("ENTER YOUR OPTIONS");
14         System.out.println("1. ENTERING THE MESSAGE");
15         System.out.println("2. SENDING THE MESSAGE");
16         System.out.println("3. DISPLAYING THE MESSAGE");
17         System.out.println("4. DELETE MESSAGE JUST ENTERED");
18         System.out.println("5. EXIT");
19         System.out.println("-----");
20
21         int choice = sc.nextInt();
22         switch (choice){
23             case 1:
24                 System.out.println("Enter the message and type 'done' to exit:");
25                 String mess = "";
26                 while (!mess.equals("done")){
27                     mess = sc.nextLine();
28                     if(!mess.equals("done") && !mess.isEmpty() && mess.length() < 250){
29                         mq.enqueue(new Node(mess));
30                     }
31                 }
32                 break;

```

```

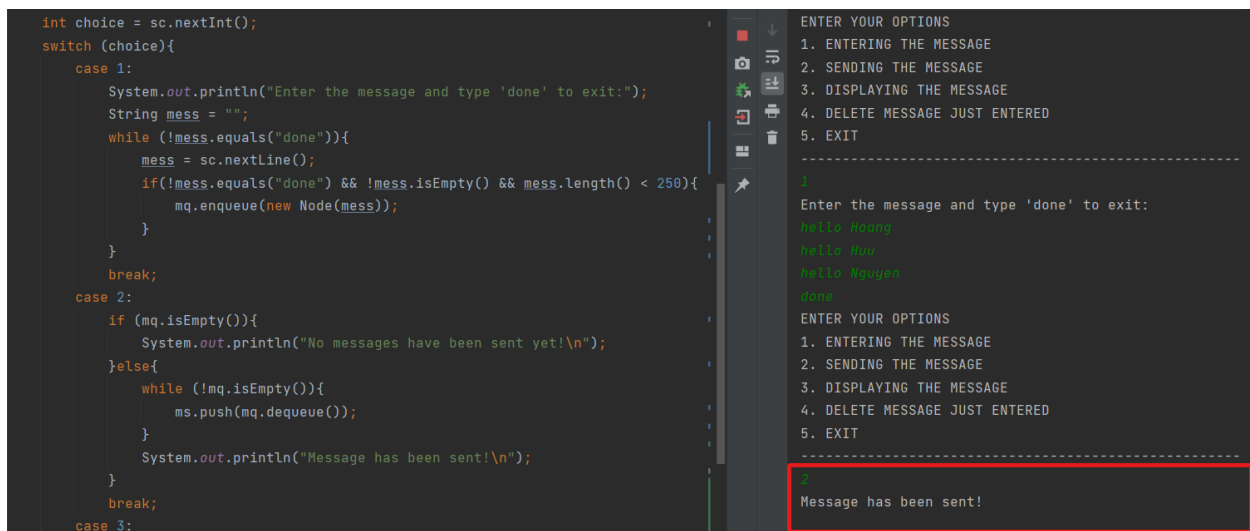
C:\Program Files\Java\jdk1.8.0_191\bin\java.exe ...
ENTER YOUR OPTIONS
1. ENTERING THE MESSAGE
2. SENDING THE MESSAGE
3. DISPLAYING THE MESSAGE
4. DELETE MESSAGE JUST ENTERED
5. EXIT
-----
Enter the message and type 'done' to exit:
hello Hoang
hello Huu
hello Nguyen
done
ENTER YOUR OPTIONS
1. ENTERING THE MESSAGE
2. SENDING THE MESSAGE
3. DISPLAYING THE MESSAGE
4. DELETE MESSAGE JUST ENTERED
5. EXIT
-----

```

Figure 8 Test option 1

The input I will type is hello Hoang, hello Huu, hello Nguyen, then I will type 'done' to stop typing the message.

#### ❖ Option 2: Send the message



```

int choice = sc.nextInt();
switch (choice){
    case 1:
        System.out.println("Enter the message and type 'done' to exit:");
        String mess = "";
        while (!mess.equals("done")){
            mess = sc.nextLine();
            if(!mess.equals("done") && !mess.isEmpty() && mess.length() < 250){
                mq.enqueue(new Node(mess));
            }
        }
        break;
    case 2:
        if (mq.isEmpty()){
            System.out.println("No messages have been sent yet!\n");
        }else{
            while (!mq.isEmpty()){
                ms.push(mq.dequeue());
            }
            System.out.println("Message has been sent!\n");
        }
        break;
    case 3:

```

```

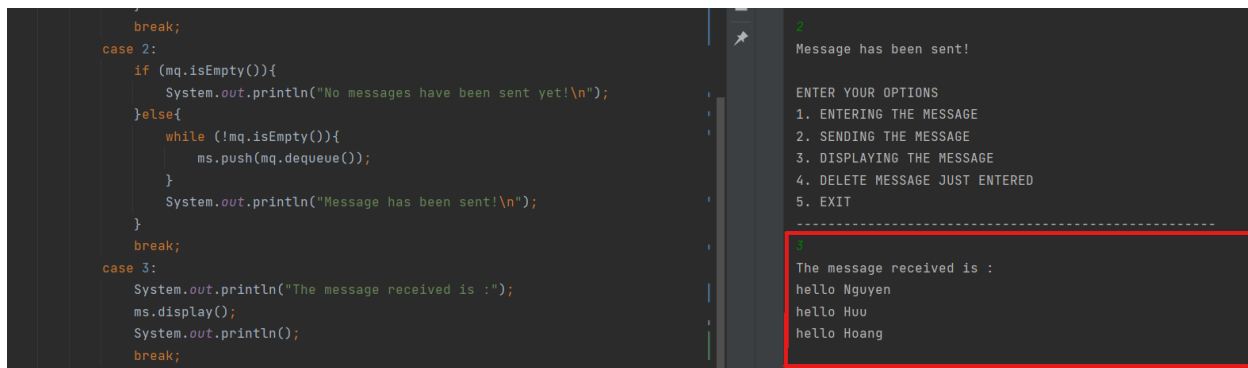
ENTER YOUR OPTIONS
1. ENTERING THE MESSAGE
2. SENDING THE MESSAGE
3. DISPLAYING THE MESSAGE
4. DELETE MESSAGE JUST ENTERED
5. EXIT
-----
Enter the message and type 'done' to exit:
hello Hoang
hello Huu
hello Nguyen
done
ENTER YOUR OPTIONS
1. ENTERING THE MESSAGE
2. SENDING THE MESSAGE
3. DISPLAYING THE MESSAGE
4. DELETE MESSAGE JUST ENTERED
5. EXIT
-----
Message has been sent!

```

Figure 9 Test option 2

After selecting Option 2, the message entered in option 1 has been sent and the information has been stored in Class MyQueue.

### ❖ Option 3: Display the message



```

break;
case 2:
    if (mq.isEmpty()){
        System.out.println("No messages have been sent yet!\n");
    }else{
        while (!mq.isEmpty()){
            ms.push(mq.dequeue());
        }
        System.out.println("Message has been sent!\n");
    }
    break;
case 3:
    System.out.println("The message received is :");
    ms.display();
    System.out.println();
    break;

```

```

Message has been sent!

ENTER YOUR OPTIONS
1. ENTERING THE MESSAGE
2. SENDING THE MESSAGE
3. DISPLAYING THE MESSAGE
4. DELETE MESSAGE JUST ENTERED
5. EXIT

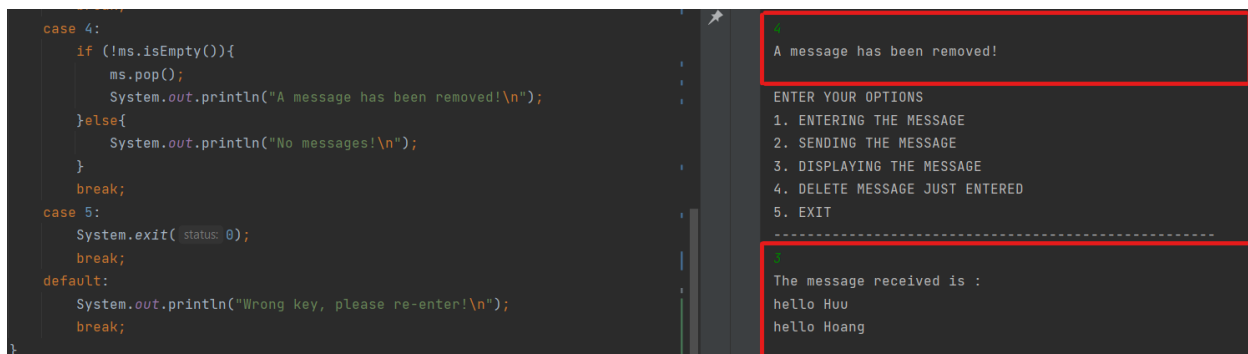
-----
The message received is :
hello Nguyen
hello Huu
hello Hoang

```

Figure 10 Test option 3

To explain this operation, when we put data into a Queue with a First In First Out structure and when we retrieve data out with a Stack, it follows a First In Last Out structure. So we will see the last message sent at the top.

### ❖ Option 4: Delete the message just entered



```

case 4:
    if (!ms.isEmpty()){
        ms.pop();
        System.out.println("A message has been removed!\n");
    }else{
        System.out.println("No messages!\n");
    }
    break;
case 5:
    System.exit( status: 0);
    break;
default:
    System.out.println("Wrong key, please re-enter!\n");
    break;
}

```

```

A message has been removed!

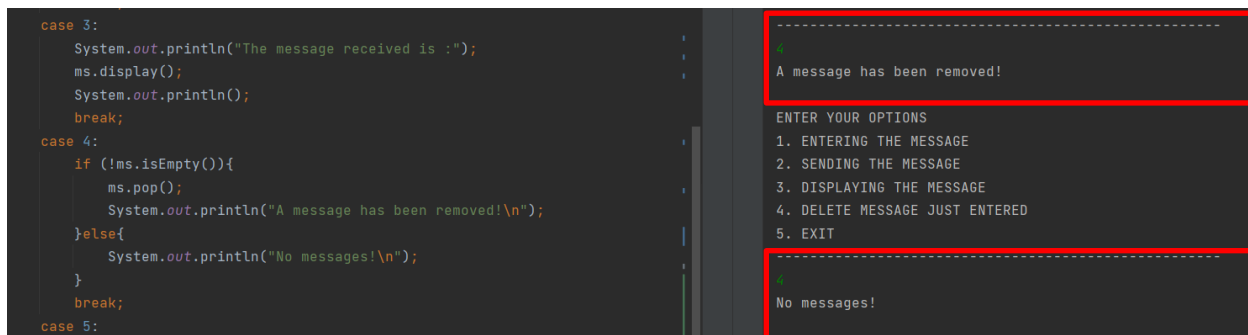
ENTER YOUR OPTIONS
1. ENTERING THE MESSAGE
2. SENDING THE MESSAGE
3. DISPLAYING THE MESSAGE
4. DELETE MESSAGE JUST ENTERED
5. EXIT

-----
The message received is :
hello Huu
hello Hoang

```

Figure 11 Test option 4

And this is the result after I delete it 2 more times, the program will say "No message" because no more messages are stored.



```

case 3:
    System.out.println("The message received is :");
    ms.display();
    System.out.println();
    break;
case 4:
    if (!ms.isEmpty()){
        ms.pop();
        System.out.println("A message has been removed!\n");
    }else{
        System.out.println("No messages!\n");
    }
    break;
case 5:

```

```

-----
A message has been removed!

ENTER YOUR OPTIONS
1. ENTERING THE MESSAGE
2. SENDING THE MESSAGE
3. DISPLAYING THE MESSAGE
4. DELETE MESSAGE JUST ENTERED
5. EXIT

-----
No messages!

```

Figure 12 Test option 4.1

## (P5) PERFORM ERROR HANDLING AND TEST RESULTS REPORT

### 1. Java Exceptions - Try...Catch

#### ❖ Definition

When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things. When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an exception (throw an error).

The try statement allows you to define a block of code to be tested for errors while it is being executed. The catch statement allows you to define a block of code to be executed if an error occurs in the try block.

#### ❖ Syntax of try ... catch

The **try** statement allows you to define a block of code to be tested for errors while it is being executed.

The **catch** statement allows you to define a block of code to be executed if an error occurs in the try block.

The try and catch keywords come in pairs:

#### Syntax

```
try {  
    // Block of code to try  
}  
catch(Exception e) {  
    // Block of code to handle errors  
}
```

Figure 13 Try ... catch Syntax

## ❖ Example

### Example

```
public class Main {  
    public static void main(String[] args) {  
        try {  
            int[] myNumbers = {1, 2, 3};  
            System.out.println(myNumbers[10]);  
        } catch (Exception e) {  
            System.out.println("Something went wrong.");  
        }  
    }  
}
```

The output will be:

```
Something went wrong.
```

Figure 14 Try ... catch Example

## 2. Implement error handling

When running the program, there are many different errors such as: errors by the writer, errors in input information, or unforeseen errors. When there is an error Java will stop and display the error information.

In the program I implement error management by try ... catch.

- Check empty in MyQueue class when designing ADT

```
Node dequeue() {  
    Node node = null;  
    try {  
        node = head;  
        head = head.next;  
    } catch (Exception e) {  
        System.out.println("Empty Queue!");  
    }  
    return node;  
}
```

Figure 15 Check empty in MyQueue class



- Check empty in MyStack class when designing ADT

```
Node pop(){
    Node x = top;
    try{
        top = top.next;
    }catch (Exception e){
        System.out.println("Empty stack!");
    }
    return x;
}
```

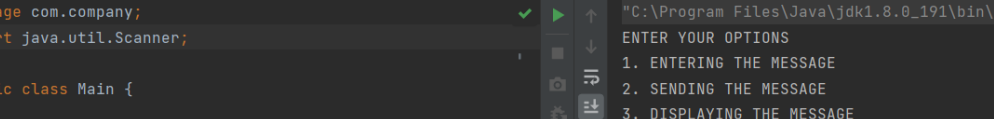
Figure 16 Check empty in MyStack class

- Check exception in main class when designing ADT

```
6 public static void main(String[] args) {
7     Scanner sc = new Scanner(System.in);
8     MyStack ms = new MyStack();
9     MyQueue mq = new MyQueue();
10
11     try {
12         while (true){...}
13     }catch (Exception ex){
14         System.out.println("Enter the wrong key!");
15     }
16 }
17 }
```

Figure 17 Check exception in main class

Why should I use try ... catch for this problem? Because when the user presses the keys to select the function, the user is only **allowed to select the numbers**, if press the char or string, the program will stop and cannot continue to run.



```
1 package com.company;
2 import java.util.Scanner;
3
4 public class Main {
5
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         MyStack ms = new MyStack();
9         MyQueue mq = new MyQueue();
10
11         try {
12             while (true){...}
13         }catch (Exception ex){
14             System.out.println("Enter the wrong key!");
15         }
16     }
17 }
```

Run: Main ×

"C:\Program Files\Java\jdk1.8.0\_191\bin\java.exe" ...

ENTER YOUR OPTIONS

1. ENTERING THE MESSAGE
2. SENDING THE MESSAGE
3. DISPLAYING THE MESSAGE
4. DELETE MESSAGE JUST ENTERED
5. EXIT

-----

abc

Enter the wrong key!

Process finished with exit code 0

### 3. Test case and test result

<b>Module test</b>	Input, output, delete, check empty
<b>Tester</b>	Nguyen Huu Hoang
<b>Create Date</b>	18/08/2021
<b>Test environment</b>	IDE IntelliJ

Table 2 Test case & result

TEST	WHAT IS BEING TESTED	TEST DATA USED	EXPECTED RESULTS	DATE	ACTUAL RESULT	NOTE
1	Option 1 - Enter message less than 250 word	String mess = "hello Hoang", "hello Huu", "hello Nguyen"	The program can transfer this message	08/18 <sup>th</sup> , 2021	The program treats this input as a text message and sends it	
2	Option 2 - Send typed messages	String mess	Messages will be processed and notified successfully	08/18 <sup>th</sup> , 2021	Messages have been processed and sent successfully	Repeat sending number still no change message content
3	Option 3 - Test message display	String mess	Previously processed messages will be displayed	08/18 <sup>th</sup> , 2021	The messages are all displayed successfully	Repeat display does not change the content of messages
4	Delete each newly typed message	String mess	The messages will be deleted one by one	08/18 <sup>th</sup> , 2021	Imported messages have been deleted	Messages will be deleted one by one until the message is empty
5	Exit the program		Running program will be stopped	08/18 <sup>th</sup> , 2021	The program stopped after selecting the key	
6	Use try ... catch to catch the exception		The program will stop and display an error when incorrect input	08/18 <sup>th</sup> , 2021	The program gives the error "Enter the wrong key" and stops immediately	Required when entering an option is data type <a href="#">int</a>

## (P6) ASYMPTOTIC ANALYSIS TECHNIQUE IS USED TO EVALUATE THE EFFICIENCY OF THE ALGORITHM

### 1. What is Algorithm Analysis?

- Algorithm analysis is the determination of the computational complexity of algorithms, which is the amount of time, storage, and/or other resources required to implement them.
- **Why analyzes an algorithm?** The simplest reason to analyze an algorithm is to explore its characteristics to assess its suitability for different applications or to compare it with other algorithms for the same application. Furthermore, analyzing an algorithm can help us understand it better and can suggest wise improvements.

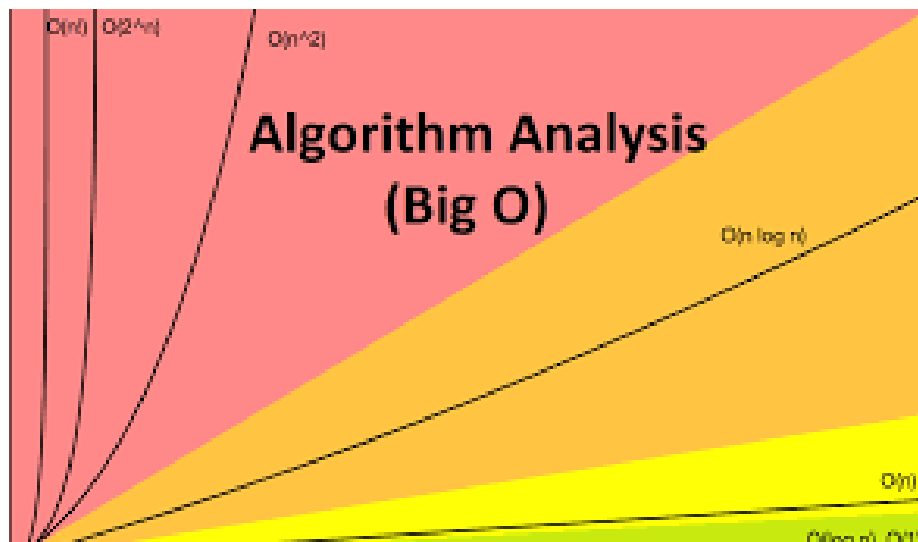


Figure 20 Algorithm analysis

### 2. What is Asymptotic Analysis?

Asymptotic analysis is the process of calculating the running time of an algorithm in mathematical units to find the program's limitations, or "run-time performance." The goal is to determine the best case, worst case and average case time required to execute a given task.

- **Best case:** is the minimum time required to execute the program.
- **Average case:** is the average time it takes to execute the program.
- **Worst case:** is the maximum time required to execute the program

While not a method of deep learning training, Asymptotic analysis is a crucial diagnostic tool for programmers to evaluate an algorithm's efficiency, rather than just its accuracy.

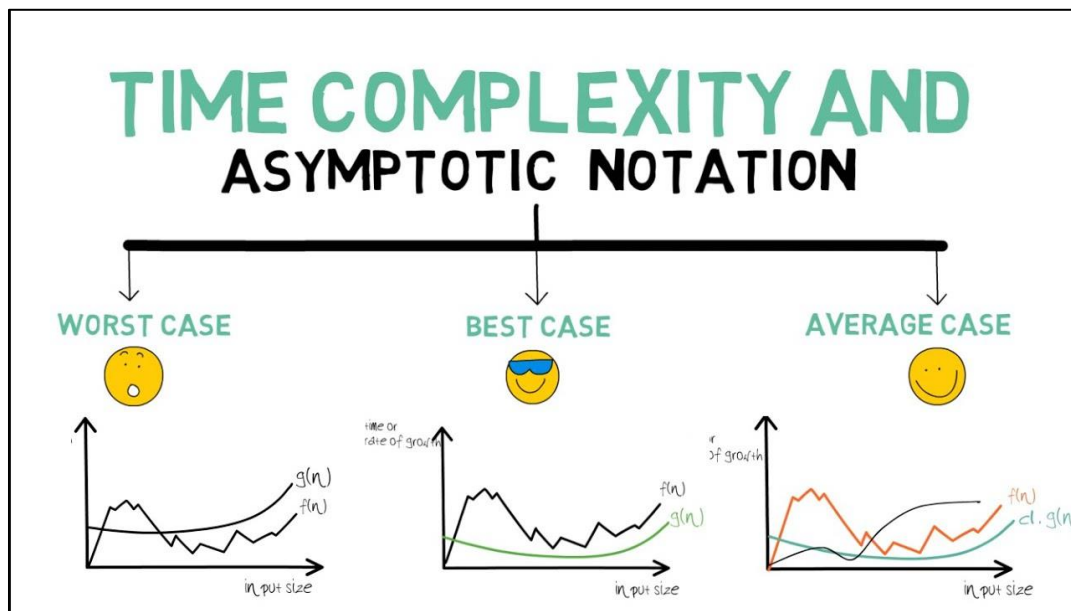


Figure 21 Asymptotic analysis

### 3. Asymptotic Notations

. While run-time performance can be calculated with many different functions, the limiting behavior of the algorithm is expressed graphically using the simple notation:

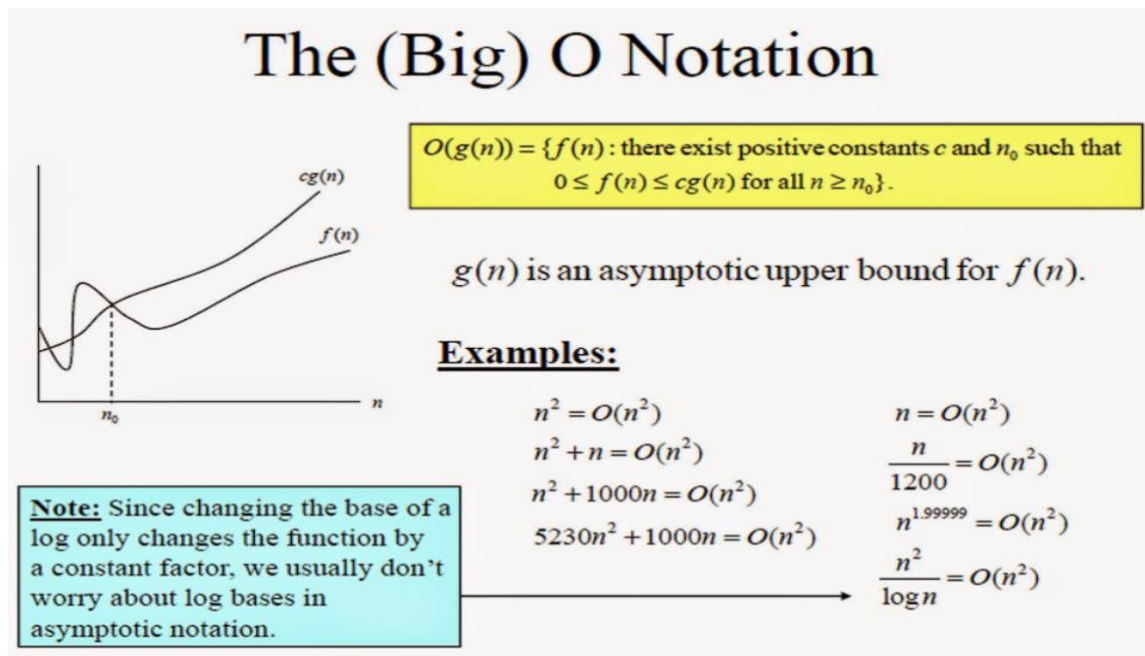
#### a. Big O notation

$O(n)$ : Is the upper bound of an algorithm's running time and measures the worst-case scenario of how long an algorithm can possibly take to complete a given operation.

We express complexity using big-O notation, User to measure running time. Basically, it tells you how fast a function grows or declines. Big O specifically describes the worst-case scenario and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm.

**For example**, when analyzing some algorithm, one might find that the time (or the number of steps) it takes to complete a problem of size  $n$  is given by  $T(n) = 4n^2 - 2n + 2$ .

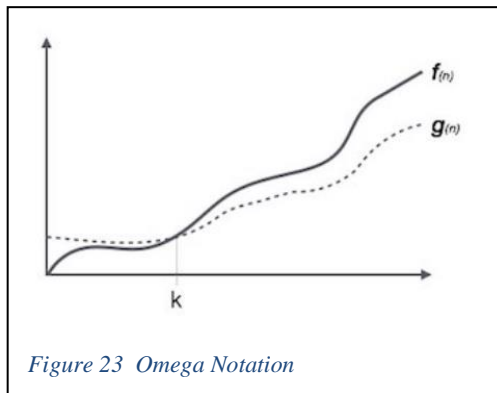
.If we ignore constants (which makes sense because those depend on the particular hardware the program is run on) and slower-growing terms, we could say “ $T(n)$  grows at the order of  $n^2$ ” and write:  $T(n) = O(n^2)$ .



**Form:**  $O(f(n)) = \{ g(n) : \text{there exists } c > 0 \text{ and } n_0 \text{ such that } f(n) \leq c \cdot g(n) \text{ for all } n > n_0. \}$

Figure 22 Big O notation

## b. Omega Notation



$\Omega(n)$ : Is the lower bound of an algorithm's running time and measures the best-case scenario of how long an algorithm can possibly take to complete a given operation.

$\Omega$  notation provides an asymptotic lower bound. It is useful for finding the Best time an Algorithm can take

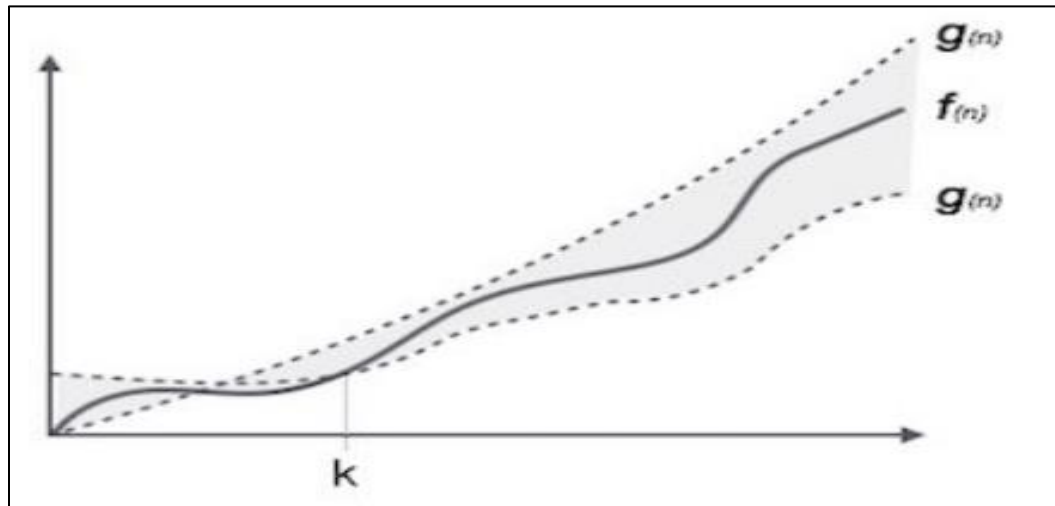
**Form:**  $\Omega(f(n)) \geq \{ g(n) : \text{there exists } c > 0 \text{ and } n_0 \text{ such that } g(n) \leq c \cdot f(n) \text{ for all } n > n_0. \}$

## c. Theta Notation

$\Theta(n)$ : Is charting both the upper and lower running time boundaries, with the average case scenario express as the average between each border.

*The notation describes asymptotic tight bounds*

A theoretical measure of the execution of an algorithm, usually the time or memory needed, given the problem size  $n$ , which is usually the number of items. Informally, saying some equation  $f(n) = \Theta(g(n))$  means it is within a constant multiple of  $g(n)$ . The equation is read, “ $f$  of  $n$  is theta  $g$  of  $n$ ”.



**Form:**  $\Theta(f(n)) = \{ g(n) \text{ if and only if } g(n) = O(f(n)) \text{ and } g(n) = \Omega(f(n)) \text{ for all } n > n_0. \}$

Figure 24 Theta Notation

**(P7) IDENTIFY TWO WAYS IN WHICH THE EFFICIENCY OF AN ALGORITHM CAN BE MEASURED, ILLUSTRATING YOUR ANSWER WITH AN EXAMPLE.**

### 1. Algorithmic complexity

The time it takes a computer to execute an algorithm depends not only on the algorithm itself, but also on the computer. To evaluate the efficiency of an algorithm, it is possible to consider the number of calculations that must be performed when implementing this algorithm. Usually, the number of calculations performed depends on the size of the problem, i.e., the size of the input. So, the algorithmic complexity is an input dependent function. However, in practical applications, we do not need to know the exact functions, but only need to know a good enough estimate of them.

To estimate the complexity of an algorithm, we often use the concept of **big O** and **Theta**, and 2 main ways to measure the efficiency of an algorithm: **Time complexity** and **space complexity**.

### 2. What is space complexity?

#### a. Definition

The space complexity of an algorithm or a computer program is the amount of memory space required to solve an instance of computational problem as a function of the size of input.

Similar to time complexity, space complexity is often expressed asymptotically in big O notation, such as  $O(n)$ ,  $O(n\log(n))$  ... where **n** is the input size in units of bits needed to represent the input

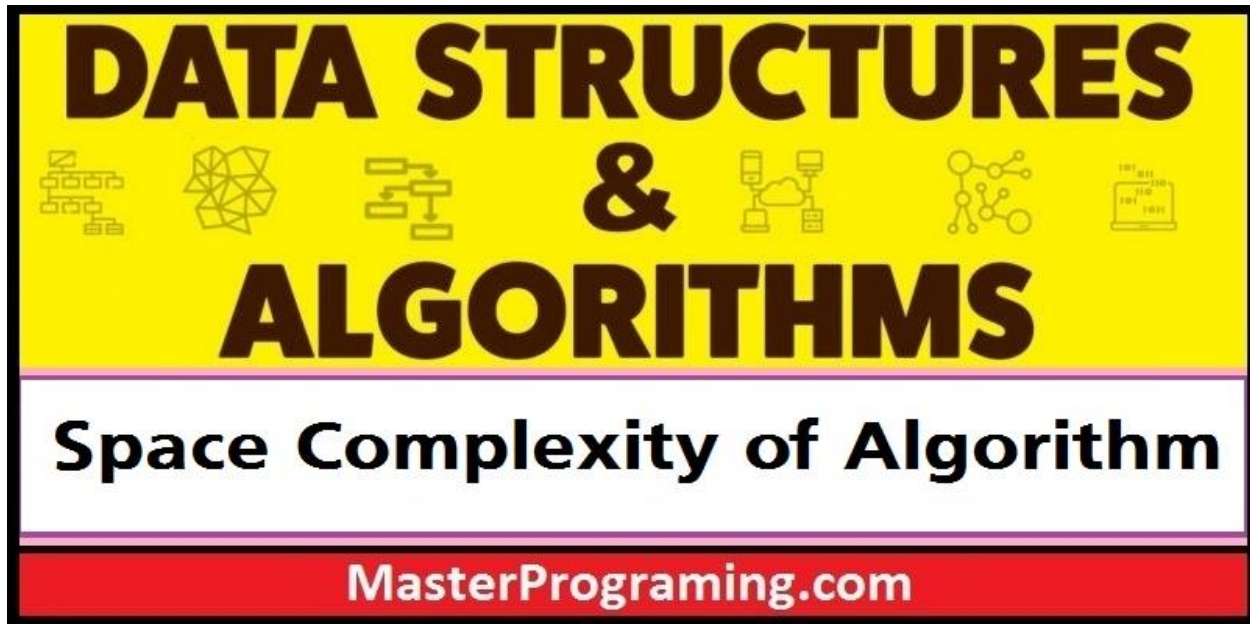


Figure 25 What is space complexity

❖ For any algorithm, memory is required for the following purposes:

- To store program instructions
- To store constant values
- To store variable values

Auxiliary space: is the temporary space (excluding the input size) allocated by your algorithm to solve problems with respect to input size

Space complexity includes both Auxiliary space and space used by the input

#### b. Example

❖ *Space complexity = Input size + auxiliary space*

We give input as data type int with the number of bytes is 4



```
static int spaceComp(int a, int b){ // a -> 4 bytes & b -> 4 bytes
    int sum;                        // sum -> 4 bytes
    sum = a + b;                    // auxiliary space (sum) -> 4 bytes
    return sum;                     // total = 16 bytes
}
```

Figure 26 Space complexity

❖ *Example: sum of all elements in the array*

With an array of data type int, we will have  $N * 4$  bytes

➔ So, Space complexity is  $O(n)$

```
static int sumOfArray(int[] arr){ // arr -> N*4 bytes
    int sum = 0;                  // sum -> 4 bytes
    for (int j : arr) {           // j -> 4 bytes
        sum += j;                 // auxiliary -> 4 bytes
    }
    return sum;                   // total = 4 * N + 12
}
```

Figure 27 sum of all elements in the array

### 3. What is time complexity?

Space complexity is sometimes ignored because the space used is minimal and/or obvious, but sometimes it becomes as important an issue as time.

### a. Why is Time Complexity Essential

By definition, the Space complexity of an algorithm quantifies the amount of space or memory taken by an algorithm to run as a function of the length of the input. While Time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the input. Now that we know why Time complexity is so significant, it is time to understand what is time complexity and how to evaluate it.

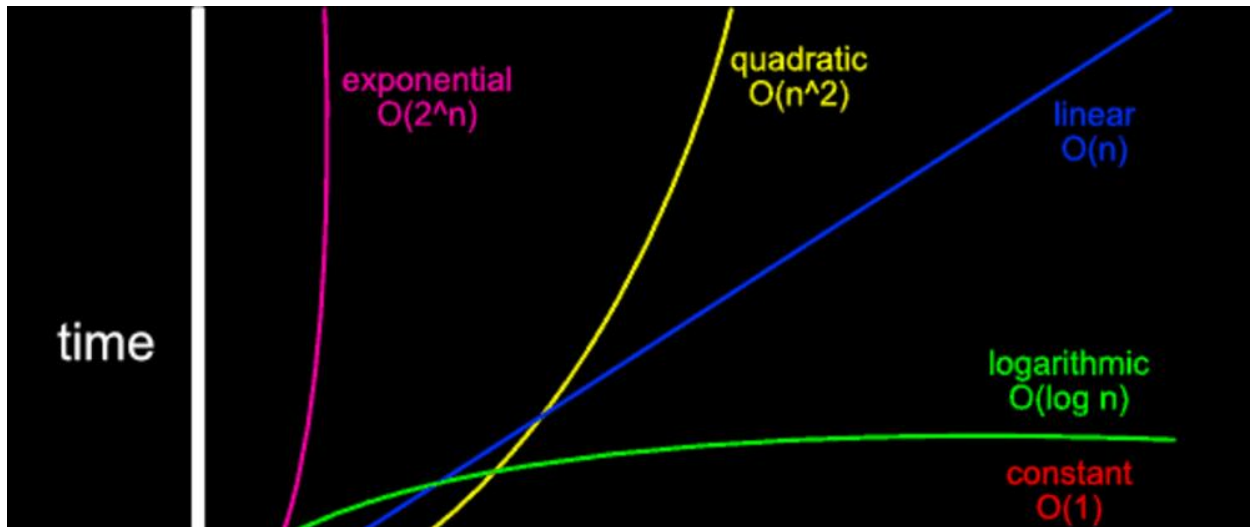


Figure 28 Time complexity

### b. Definition

Time complexity is the amount of time taken by an algorithm to run, as a function of the length of the input. It measures the time taken to execute each statement of code in an algorithm.

### c. Example

➔ So, time complexity is  $O(1)$

```
public static void main(String[] args) {

    int x = 10;           // 1 time
    int y = 10;           // 1 time
    int z = x+y;          // 1 time
    System.out.println("Result = " + z); // 1 time
}
```

Figure 29 A sequence of operations

- **Example:** Nested Loop

→ So, time complexity is  $O(n^2)$

```
int n = 10; // 1 time
int sum = 0; // 1 time
for (int i = 0; i < n; i++){ // n time
    for (int j = 0; j < n; j++){ // n time
        sum += i; // n * n time
        System.out.println("Result = " + sum); // 1 time
    }
}
```

*Figure 30 Nested Loop*

#### d. General Rules for Estimation

- 1) **Loops:** The running time of a loop is at most the running time of the statements inside of that loop times the number of iterations.
- 2) **Nested Loops:** Running time of a nested loop containing a statement in the inner the most loop is the running time of the statement multiplied by the product of the size of all loops.
- 3) **Consecutive Statements:** Just add the running times of those consecutive statements.
- 4) **If/Else:** Never more than the running time of the test plus the larger of running times of S1 and S2.

## C. CONCLUSION

After learning Data Structure & Algorithm and working with this Assignment, I have learned a lot. Not only did I get basic algorithm knowledge, but my teacher taught me how to put it into practice. Thanks to that, I can have a designs ideas Assignment.

However, due to limited knowledge and time, the DSA research report is not in-depth and has not really received high appreciation from subject teachers. The lack of practical experience also made me ignore many of the problems that exist in the system. During the next course of study at BTEC, I will continue to work hard to gain more knowledge and experience for myself.

## D. REFERENCES

- [1]. Vi.wikipedia.org. 2021. *Phân tích thuật toán – Wikipedia tiếng Việt*. [online] Available at: [https://vi.wikipedia.org/wiki/Ph%C3%A2n\\_t%C3%ADch\\_thu%E1%BA%ADt\\_to%C3%A1n](https://vi.wikipedia.org/wiki/Ph%C3%A2n_t%C3%ADch_thu%E1%BA%ADt_to%C3%A1n) [Accessed 18 August 2021].
- [2]. Hoang, N., 2021. *Sign in - Google Accounts*. [online] Classroom.google.com. Available at: <https://classroom.google.com/u/0/c/MjM1NTc2MzQ0NTkx> [Accessed 18 August 2021].
- [3]. GreatLearning Blog: Free Resources what Matters to shape your Career!. 2021. *Time Complexity: What is Time Complexity & Algorithms of it?*. [online] Available at: <https://www.mygreatlearning.com/blog/why-is-time-complexity-essential/> [Accessed 18 August 2021].