Assignment 2

Data Structure & Algorithm – 1649

**Nguyen Ngoc Quang Minh – GCD191322**

Mentor: Truong Nguyen Xuan Vinh

Class: GCD0807A

**P4 Implement a complex ADT and algorithm in an executable programming language to solve a well-**

**defined problem**

# Define problem & Idea to solve problem

The team now has to develop these kinds of collections for the system. They should design ADT / algorithms for these 2 structures and implement a demo version with message is a string of maximum 250 characters.

There will be a lot of ideas and ways to choose data structures to solve the problem. They will have implementation complexity as well as algorithmic complexity. In this assignment, I will clarify on those issues.

Choose a data

structure:

•

First

I will choose a data structure

including: ArrayList, Queue, and

Linked List to solve the problem.

•

Use ArrayList to store value when

user input from keyboard. Then

use the Queue interface to be

able to use the built

-

in methods.

•

Next step I will use Queue to

output the reversed characters

from the original output and save

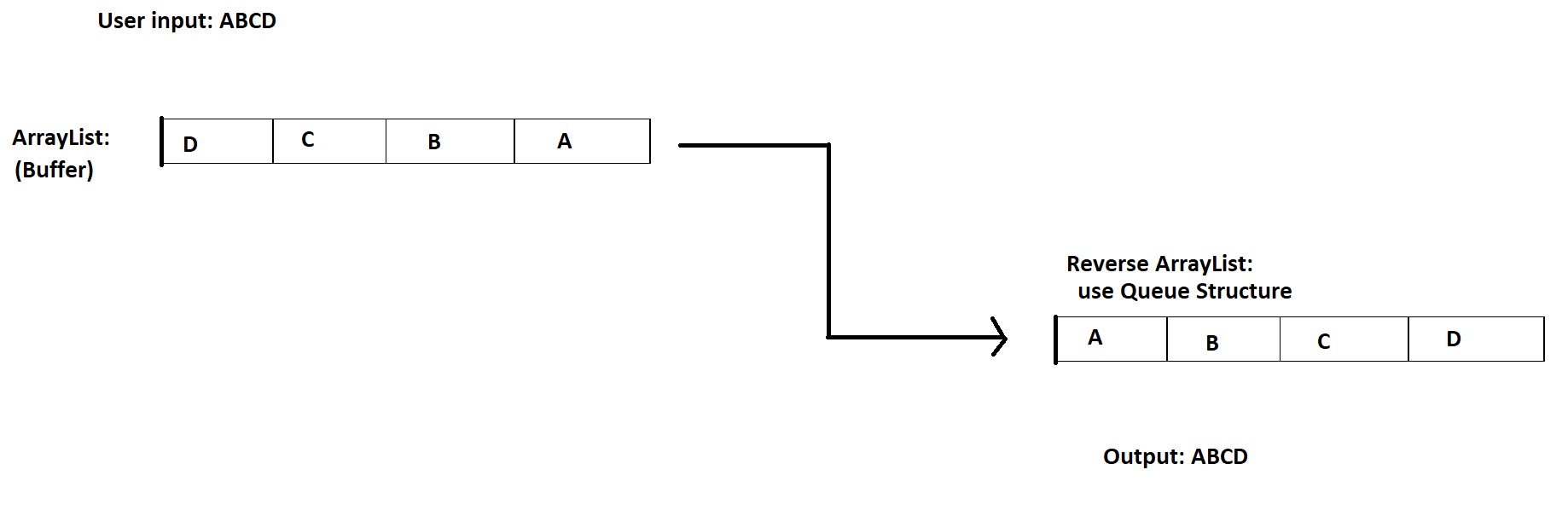
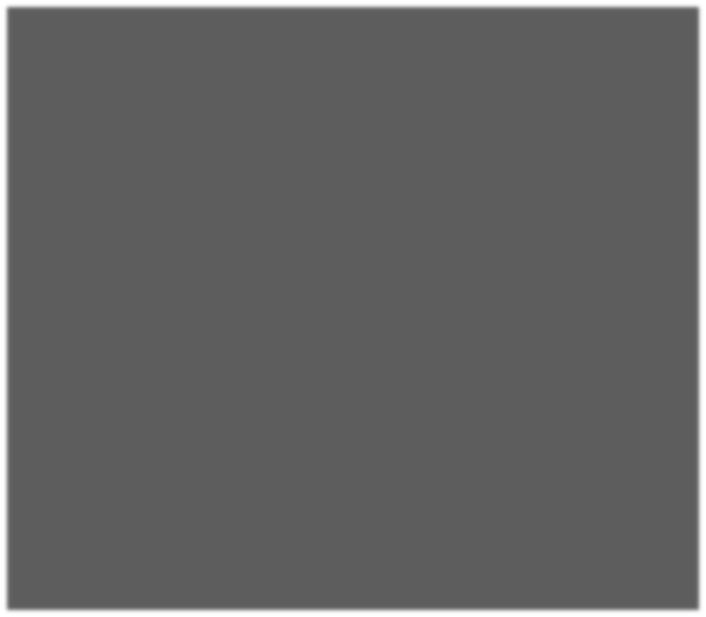
it to a linked list in a FIFO

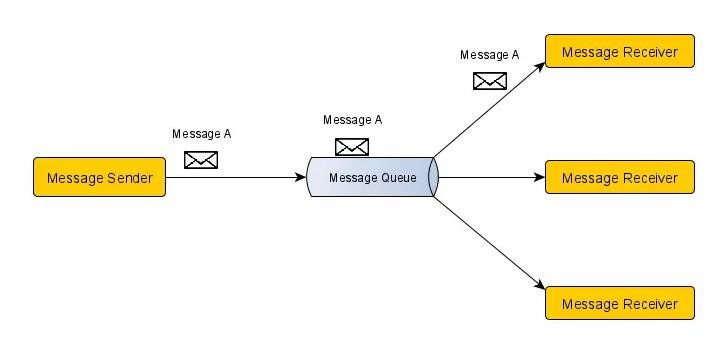
structure.

•

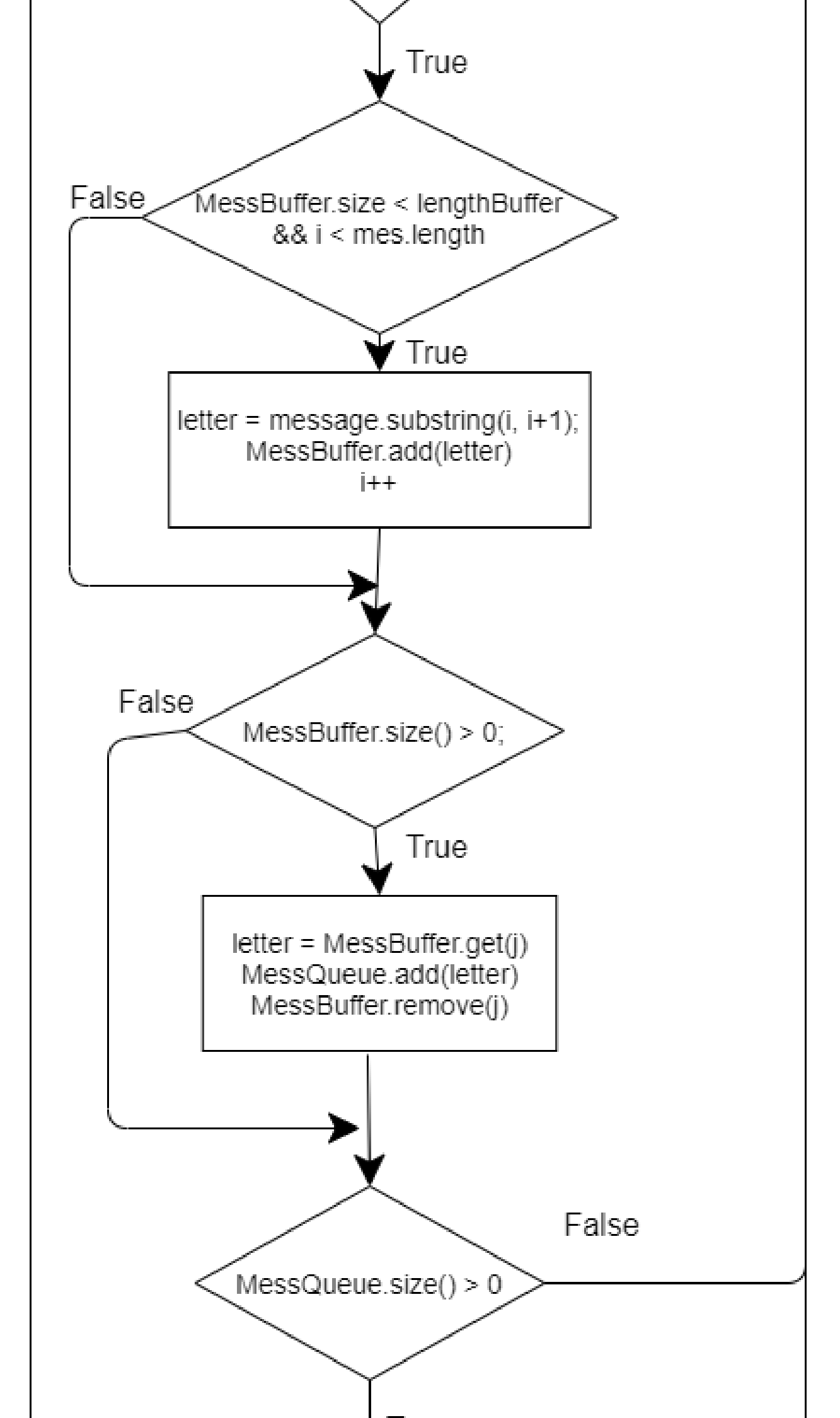
Finally, add each letter to a String

variable.



Sending message model:

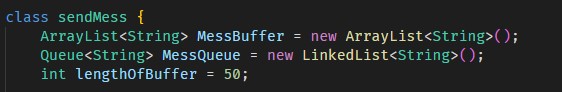
• Even though several message receivers are listening in the same message queue, the message delivered from the message sender to just one receiver in the point-topoint paradigm. Rather than message publisher and message consumer, the phrases message sender and message receiver are commonly used in the point-to-point approach. To demonstrate, I've drawn a basic diagram below.

Preview the middleware sending message by flowchart

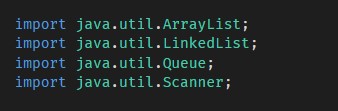
**P5** Implement error handling and report test results

# Object created in chatbox middleware & import usage for coding

**Object to create:**

* MessBuffer: to store the data user input from keyboard & it will be String data type. We use buffers to optimize memory, space, and execution time.
* MessQueue: to store the data after reverse from MessBuffer. Here I use the Queue interface to reusable methods which have already. It will be LinkedList<String> data type.
* LengthOfBuffer: to set Length of MessBuffer.

**Import usage for coding**:

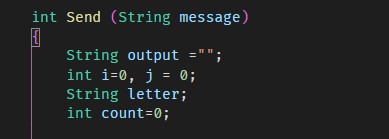
ArrayList, LinkedList, Queue: to reusable methods of us which have already. It will help my code shorter and easer to code.

Scanner: to store input user from keyboard.

# Method to send message

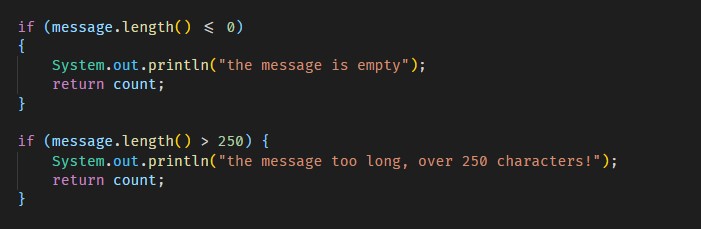
**a. Some variable usage in method**

* message: the argument to pass message values user input
* output: to store the message after reverse.
* letter: to store each letter we cut from MessQueue.
* count: to count numbers of buffer we used.



# b. Check input message

User can type over 250 or no type any characters. We must check it and give notification for user. Here I use if condition to check it.



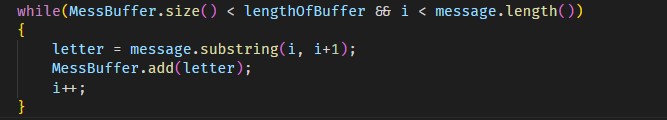
## c. Take each character and add it into ArrayList (MessBuffer)

The condition will check 2 things. First one is the MessBuffer will not bigger than LengthOfBuffer. Second one check the letter is not bigger than user input message.

To take each character. I use substring method. Use i variable. Because I just want to take only 1 letter. So begin indexis “i” and end indexis “i+1”.

Then, I add each character intoArrayList (MessBuffer) by add method.

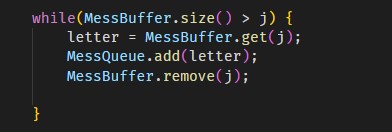
Finally,I will increase i to change index.



# d. Get all characters into Queue and converse it

If size of bigger than j, we will take each letter from ArrayList (Buffer) into the Queue. Then remove letter itself in the first index. The removing has meaning that after size of MessBuffer = 0. We will take another letter. Because it’s just store 50 characters.

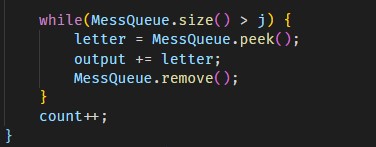
Example: in my code. I set LengthOfBuffer = 50. So, it can store only 50 characters. If you want to store more character. You must remove all data in ArrayList and reuse it.



# e. Store character into a String variable

We will store all character into a variable. First, we use while loop to check length of Queue when it smaller than 0. Because we will reduce index to 0 after add letter into “output” variable.

Count: to count numbers of buffer usage when send message.



# f. Main function

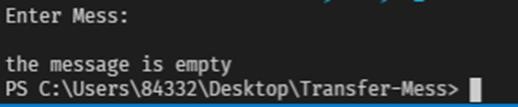
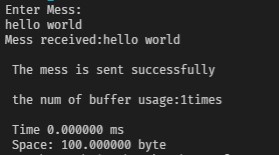
This main function use to run the program, calculate Time and Space when running. Here I use CurrentTimeMillis() to calculate. Because in my program, when I send message, I remove it and reuse

ArrayList. So, I just use 100 byte only.

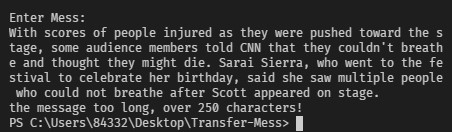
# Test case

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Serial**  **Number** | **Test case** | **Input data** | **Expected output** | **Actual output** |  | **Result** |
| **1** | Enter message with lower 50 characters | Input: “hello world” | Sending success  Length = 11  Message received: hello world  Buffer usage: 1 | Length = 11  Message received: hello world  Buffer usage: 1 | Pass |  |
| **2** | Sending empty message | Input: null | The message is empty | The message is empty | Pass |  |
| **3** | Input over 250 characters | Input: message over 250 characters | the message too long, over 250 characters! | the message too long, over 250 characters! | Pass |  |
| **4** | Input special character | Input: “/ @#$” | Sending success  Message received: / @#$ | Sending success  Message received: / @#$ | Pass |  |
| **5** | Input Vietnamese character | Input: “Xin chào, tôi đến từ Việt  Nam” | Sending success Message received: Xin chào, tôi đến từ Việt Nam | Sending success  Message received: Xin chào, tôi đến từ Việt Nam | Fail |  |

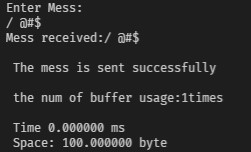
## Test case 1: Enter message with lower 50 characters Test case 2: Sending empty message



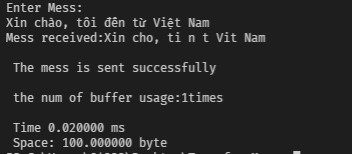
## Test case 3: Input over 250 characters



## Test case 4: Input special character



## Test case 5: Input Vietnamese character

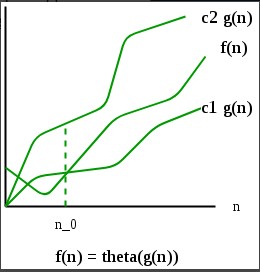


**P6** Discuss how asymptotic analysis can be used to assess the effectiveness of analgorithm

### 1. 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. 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.

**Explain Asymptotic Notation:**

* Best Case: The time required by the algorithm is the minimum of all. E.g., in sorting algorithm when all the given numbers in the input are alreadysorted.
* Average Case: The time required by an algorithm is the average time taken by all. E.g., In the given algorithm, when given number in the input is halfsorted.
* Worst Case: The time required by an algorithm is the maximum of all others. E.g., in sorting algorithm when all the given numbers in the input are given in reverse order as required.

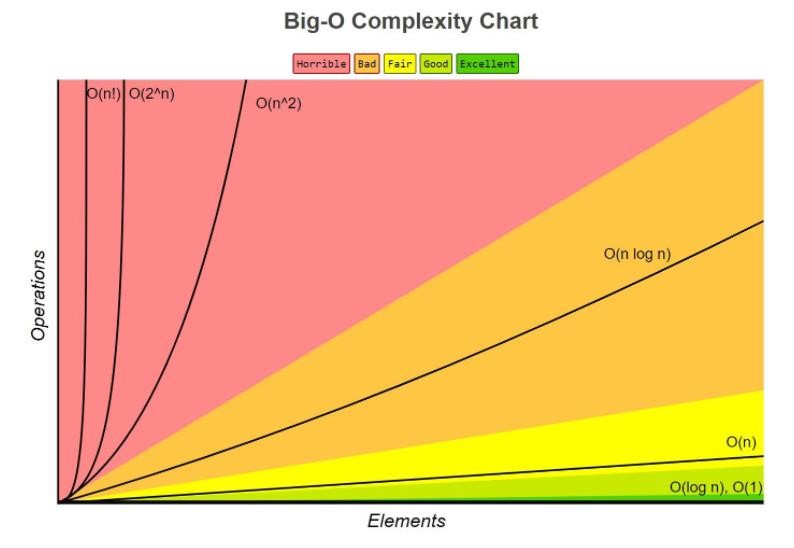
**Types of Asymptotic Notations:**

* Big Oh (O Notation): This notation gives an upper bound of an algorithm, that boundsthe function from above.
* Big Omega (Ω Notation): ): Big Omega represents the best case that can happen, which means that the complexity of this algorithm couldn’t be less than the value of Ω(n)
* Big Theta (Θ): Big theta represents the average that can be achieved between the worst and best case of the algorithm complexity, so Ω(n)≤Θ(n)≤O(n). The f(n) value will alwaysbetween c1g(n) and c2g(n) where c1, c2 constants

## 2. Big Oh

We must consider how long an algorithm will take to complete and how much space it will require to calculate while developing an application. And it's up to us to figure out how to cope with how quickly the algorithms' runtimes expand in relation to their input sizes so that the applicationcan be deployedeffectively.

The general step wise procedure for Big-O runtime analysis is as follows:



•

•

•

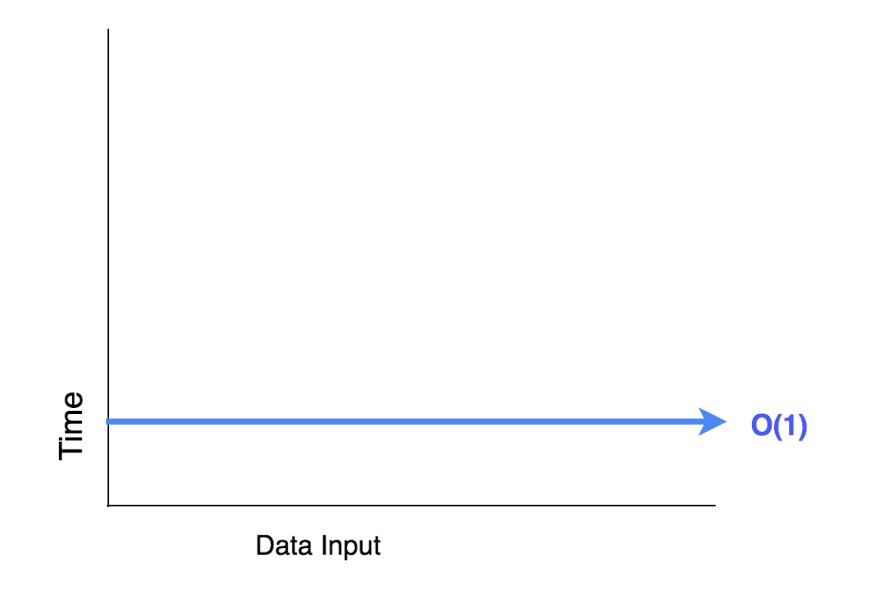
•

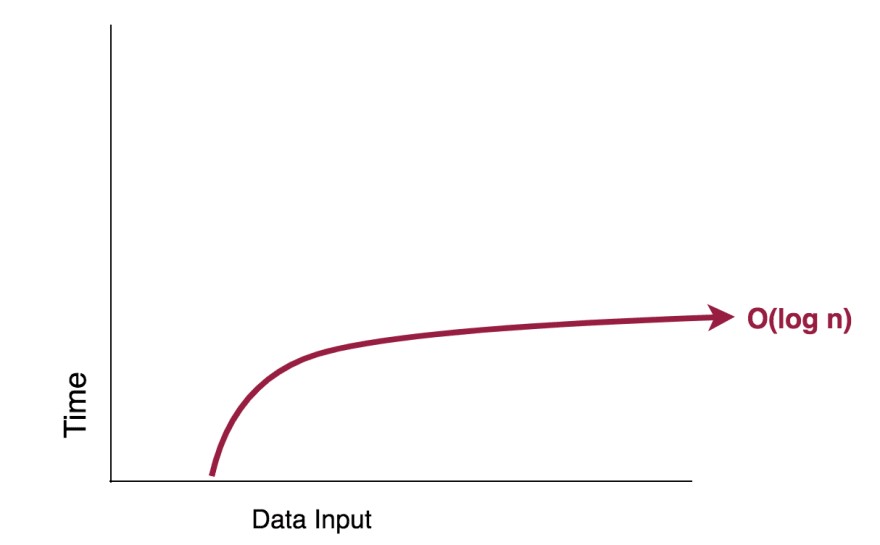
Figure out what the input is and what n represents.

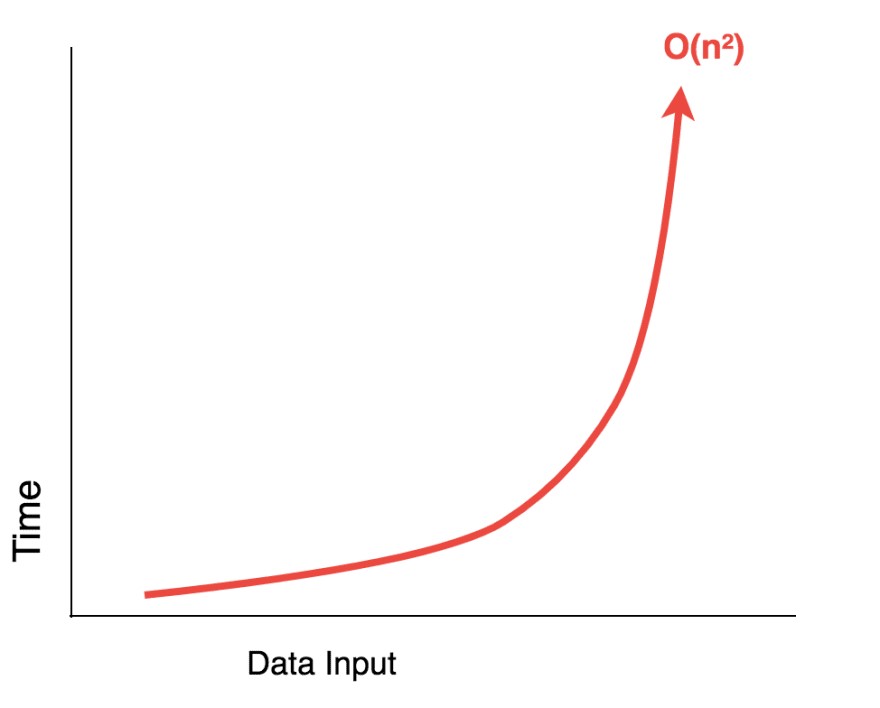
Express the maximum number of operations, the algorithm performs in terms of n.

Eliminate all excluding the highest order terms.

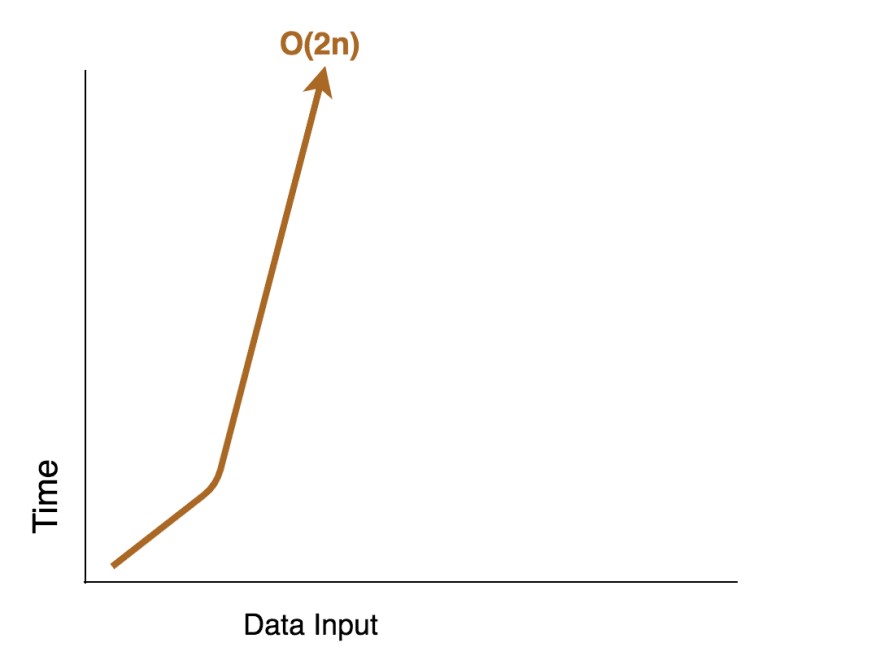
Remove all the constant factors.

**O(1) has the least complexity**: Often called “constant time”, if you can create an algorithm to solve the problem in O(1), you are probably at your best. In some scenarios, the complexity may go beyond O(1), then we can analyze them by finding its O(1/g(n)) counterpart.

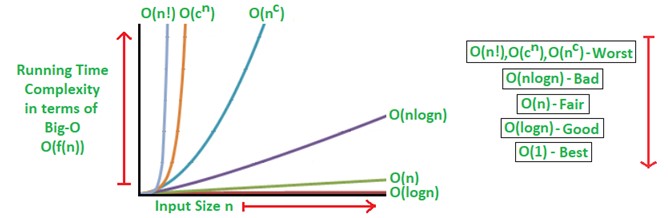
**O(log(n)) is more complex than O(1), but less complex than polynomials**: As complexity is often related to divide and conquer algorithms, O(log(n)) is generally a good complexity you can reach for sorting algorithms. O(log(n)) is less complex than O(√n), because the square root function can be considered a polynomial,where the exponent is 0.5

**Complexity of polynomials increases as the exponent increases:**For example, O(n⁵) is more complex than O(n⁴). Due to the simplicity of it, we actually went over quite many examples of polynomials in the previous sections.

**O(n²) - Quadratic Time:** Represents an algorithm whose performance is directly proportional to the squared size of the input data set. This is common with algorithms that involve nested iterations over the data set. As the input increases, the time to run the algorithm grows at the rate of its square.

**O(2n) - Quasilinear Time:** An algorithm is said to have an exponential time or O(2^n) if its runtime doubles with each addition to the input data set, starting off very shallow, then rising meteorically.

### 3. Some examples about Big-Oh complexity in real calculator



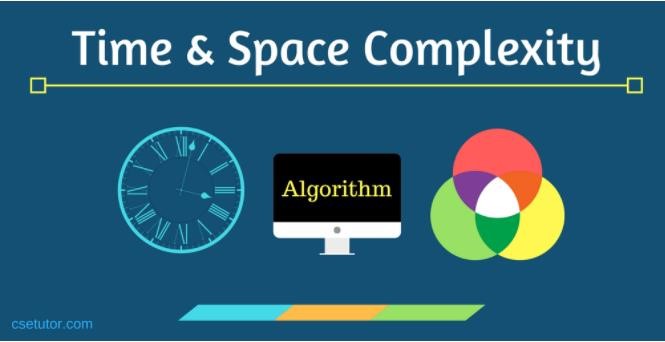
Algorithmic Examples of Runtime Analysis:

Some of the examples of all those types of algorithms (in worst-case scenarios) are mentioned below:

* Logarithmic algorithm – O(logn) – Binary Search.
* Linear algorithm – O(n) – Linear Search.
* Superlinearalgorithm – O(nlogn) – Heap Sort, Merge Sort.
* Polynomial algorithm – O(n^c) – Strassen’s Matrix Multiplication, Bubble Sort, Selection Sort, Insertion Sort, Bucket Sort.
* Exponential algorithm – O(c^n) – Tower of Hanoi.
* Factorial algorithm – O(n!) – Determinant Expansion by Minors, Brute force Search algorithm for Traveling Salesman Problem.

**P7** Determine two ways in which the efficiency of an algorithm can be measured, illustrating your answer with an example

### 1. Analyze performance

Algorithms are often quite different from one another, though the objective of these algorithms are the same. For example, we know that a set of numbers can be sorted using different algorithms. Number of comparisons performed by one algorithm may vary with others for the same input. Hence, time complexity of those algorithms may differ. At the same time, we need to calculate the memory space required by each algorithm.

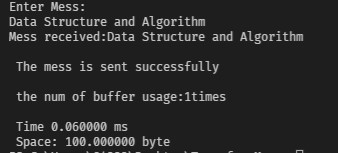
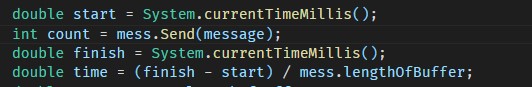
Performance analysis of an algorithm depends upon two factors i.e. amount of memory used and amount of compute time consumed on any CPU. Formally they are notified as complexities in terms of:

* **Space**: The amount of memory is evaluated by calculating the maximum amount of memory that the algorithm needs to use.
* **Time**: Time is evaluated by calculating the number of key operations (such as comparisons in sorting algorithms).

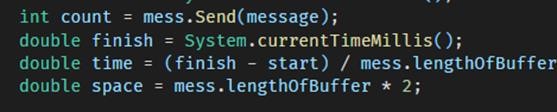
**Time Complexity:**

The time factor of an algorithm describes the amount of time an algorithm takes to run from start to finish. A function T(n) can be used to describe the request time, with T(n) being the number of steps.

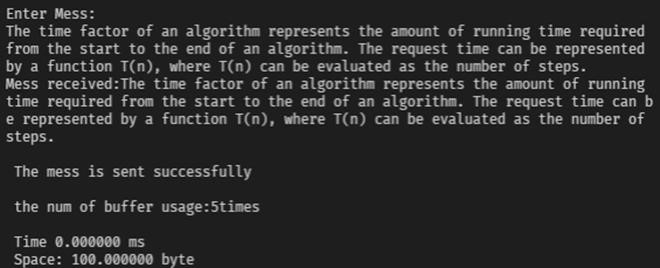
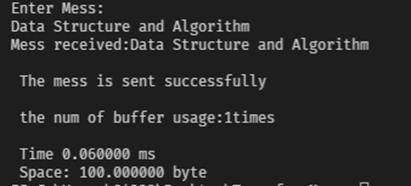
Example: Here I use currentTimeMillis() to calculate time when sending message. From beginning to ending.



**Space Complexity:**

Space complexity analysis estimates the amount of memory required by the algorithm to process input data. While processing the input data, the algorithm needs to store the transient temporary data structures in memory. The way the algorithm is designed affects the number, type, and size of these data structures.

Here I use my program to example about Space complexity when I am sending the message. Because in my program, when I send message, I remove it and reuse ArrayList. So, I just use 100 byte only. And 2 test give me the same result. And this image below is result:



**ASSIGNMENT 2**



|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | **BTEC Level 5 HND Diploma in Computing** | | |
| **Unit number and title** | Unit 19: Data Structures and Algorithms | | |
| **Submission date** |  | **Date Received 1st submission** |  |
| **Re-submission Date** |  | **Date Received 2nd submission** |  |
| **Student Name** | Nguyen Ngoc Quang Minh | **Student ID** | GCD191322 |
| **Class** | GCD0807A | **Assessor name** | Truong Nguyen Xuan Vinh |
| **Student declaration**  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. | | | |
|  | | **Student’s signature** | *QuangMinh* |

**Grading grid**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P  4 | P  5 | P  6 | P  7 | M4 | M5 | D3 | D4 |
|  |  |  |  |  |  |  |  |

1

|  |  |  |
| --- | --- | --- |
|  **Summative Feedback:**  **Resubmission Feedback:** | | |
| **Grade:** | **Assessor Signature:** | **Date:** |
| **Internal Verifier’s Comments:** | | |
| **IV Signature:** | | |

Contents

**P4** Implement a complex ADT and algorithm in an executable programming language to solve a well-defined

problem ........................................................................................................................................................... 5

1. **Define problem & Idea to solve problem** .......................................................................................... 5
2. **Sending message model:** .................................................................................................................... 6
3. **Preview the middleware sending message by flowchart** ................................................................ 7

**P5** Implement error handling and report test results ........................................................................................... 9

1. **Object created in chatbox middleware & import usage for coding**................................................. 9
2. **Method to send message** ....................................................................................................................... 9 **a.** **Some variable usage in method**.......................................................................................................... 9
   1. **Check input message** ....................................................................................................................... 10
   2. **Take each character and add it into ArrayList (MessBuffer)** ............................................................... 10
   3. **Get all characters into Queue and converse it** ................................................................................... 11
   4. **Store character into a String variable**................................................................................................ 11
   5. **Main function** .................................................................................................................................. 12
3. **Test case**............................................................................................................................................ 13

**P6** Discuss how asymptotic analysis can be used to assess the effectiveness of an algorithm ............................... 16

1. **Asymptotic analysis** ......................................................................................................................... 16
2. **Big Oh** ................................................................................................................................................ 17
3. **Some examples about Big-Oh complexity in real calculator** ......................................................... 20

**P7** Determine two ways in which the efficiency of an algorithm can be measured, illustrating your answer with an

example ......................................................................................................................................................... 21

**1.** **Analyze performance** ....................................................................................................................... 21

**Time Complexity:** .................................................................................................................................... 21 **Space Complexity:** ................................................................................................................................... 22

**Reference**....................................................................................................................................................... 24

Figure 1 Middleware Sending message ............................................................................................................... 5

Figure 2 Idea to build program ........................................................................................................................... 6 Figure 3 Sending message model ....................................................................................................................... 6

Figure 4 Object use to sending message ............................................................................................................. 9 Figure 5 Main function .................................................................................................................................... 12 Figure 6 Asymptotic Analysis............................................................................................................................ 16

Figure 7 Big-O Complexity................................................................................................................................ 17

**P4** Implement a complex ADT and algorithm in an executable programming language to solve a well-defined problem

#### 1. Define problem & Idea to solve problem

The team now has to develop these kinds of collections for the system. They should design ADT / algorithms for these 2 structures and implement a demo version with message is a string of maximum 250 characters.

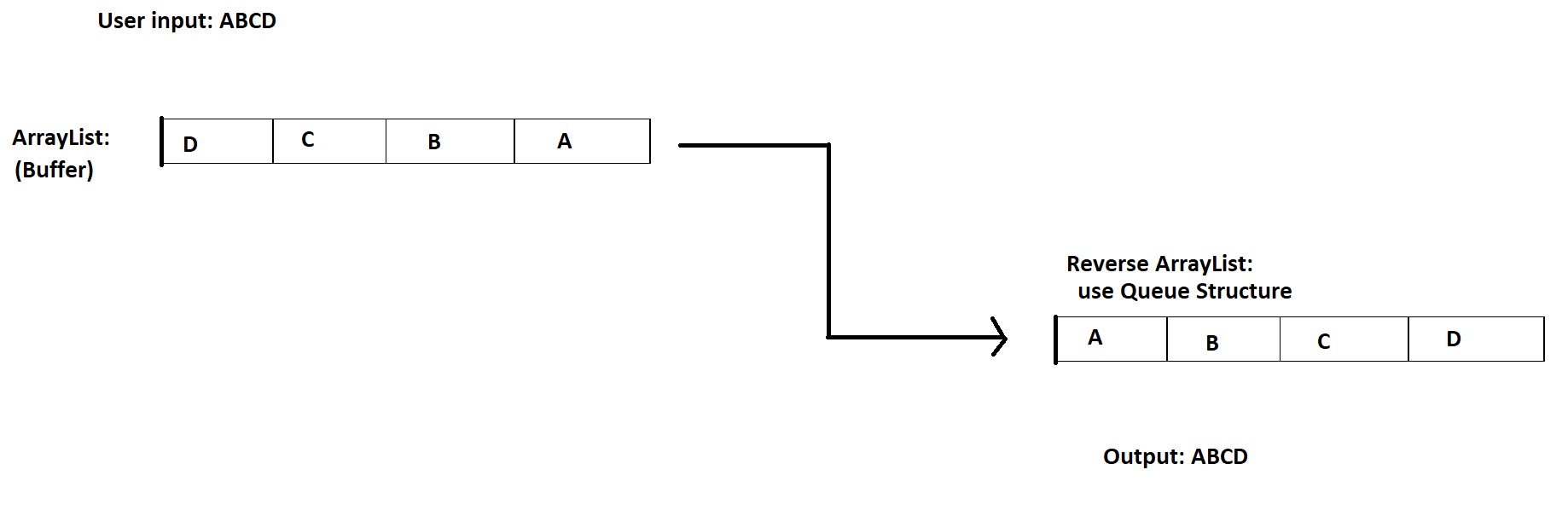


*Figure 1 Middleware Sending message*

There will be a lot of ideas and ways to choose data structures to solve the problem. They will have implementation complexity as well as algorithmic complexity. In this assignment, I will clarify on those issues.

**Choose a data structure:**

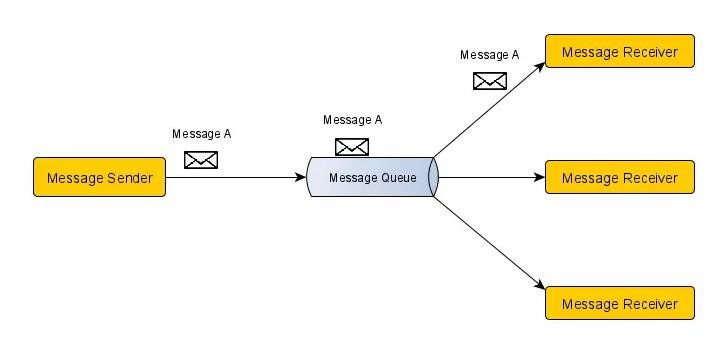
* First I will choose a data structure including: ArrayList, Queue, and Linked List to solve the problem.
* Use ArrayList to store value when user input from keyboard. Then use the Queue interface to be able to use the built-in methods.
* Next step I will use Queue to output the reversed characters from the original output and save it to a linked list in a FIFO structure.
* Finally, add each letter to a String variable.



*Figure 2 Idea to build program*

**2. Sending message model:**

Even though several message receivers are listening in the same message queue, the message delivered from the message sender to just one receiver in the point-to-point paradigm. Rather than message publisher and message consumer, the phrases message sender and message receiver are commonly used in the point-to-point approach. To demonstrate, I've drawn a basic diagram below.

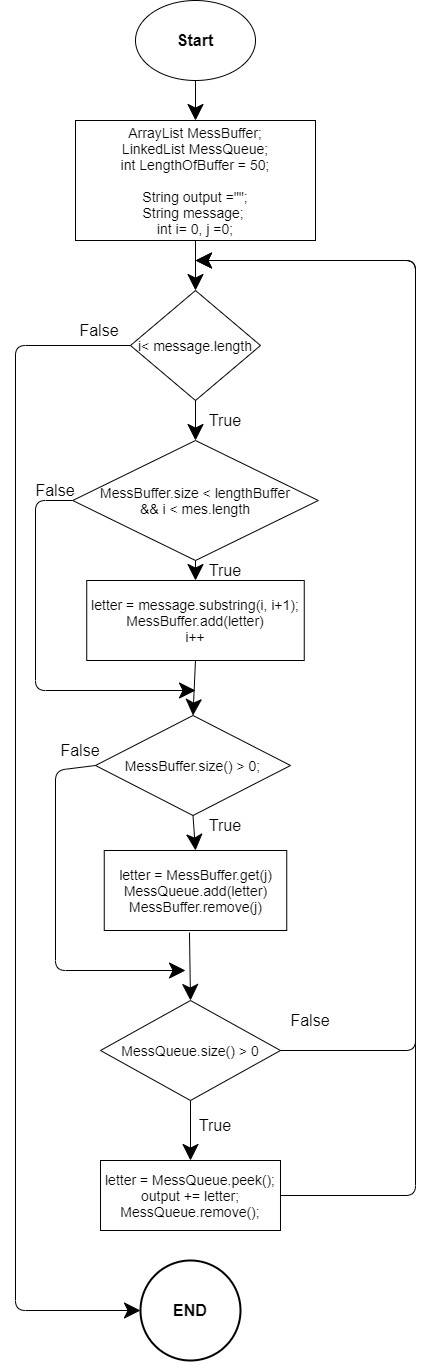


*Figure 3 Sending message model*

Different portions of the system can interact and process actions asynchronously by using a message queue. Message queues also provide a lightweight cache for temporarily storing messages, as well as endpoints for software components to connect to queues to send and receive messages.

Our software development team has worked on finishing the creation of a middleware solution to deliver a message based on the aforesaid concept. By implementing ADT and applying it to two data structures, a message system with a maximum character limit of 250 may be created.

#### 3. Preview the middleware sending message by flowchart

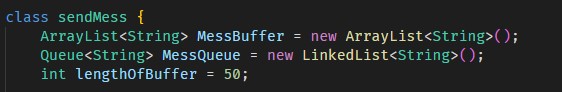


**P5** Implement error handling and report test results

**1. Object created in chatbox middleware & import usage for coding**

**Object to create:**

* MessBuffer: to store the data user input from keyboard & it will be String data type. We use buffers to optimize memory, space, and execution time.
* MessQueue: to store the data after reverse from MessBuffer. Here I use the Queue interface to reusable methods which have already. It will be LinkedList<String> data type.
* LengthOfBuffer: to set Length of MessBuffer.

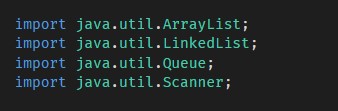


*Figure 4 Object use to sending message*

**Import usage for coding**:

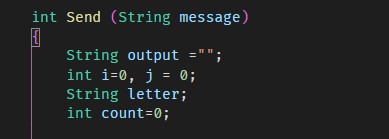
ArrayList, LinkedList, Queue: to reusable methods of us which have already. It will help my code shorter and easer to code.

Scanner: to store input user from keyboard.



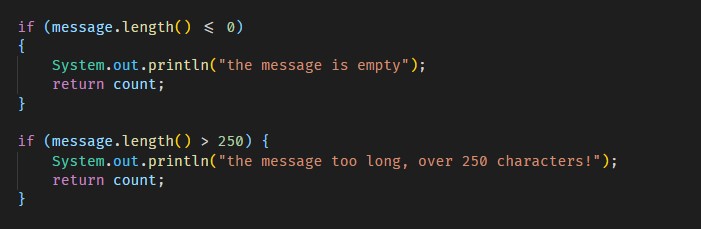
**2. Method to send message**

1. **Some variable usage in method** 
   * message: the argument to pass message values user input
   * output: to store the message after reverse.
   * letter: to store each letter we cut from MessQueue.
   * count: to count numbers of buffer we used.



1. **Check input message**

User can type over 250 or no type any characters. We must check it and give notification for user. Here I use if condition to check it.



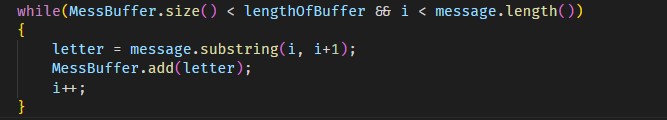
1. **Take each character and add it into ArrayList (MessBuffer)**

The condition will check 2 things. First one is the MessBuffer will not bigger than LengthOfBuffer. Second one check the letter is not bigger than user input message.

To take each character. I use substring method. Use i variable. Because I just want to take only 1 letter. So begin index is “i” and end index is “i+1”.

Then, I add each character into ArrayList (MessBuffer) by add method.

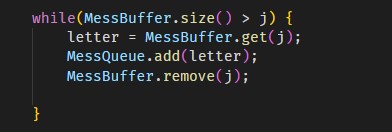
Finally, I will increase i to change index.



1. **Get all characters into Queue and converse it**

If size of bigger than j, we will take each letter from ArrayList (Buffer) into the Queue. Then remove letter itself in the first index. The removing has meaning that after size of MessBuffer = 0. We will take another letter. Because it’s just store 50 characters.

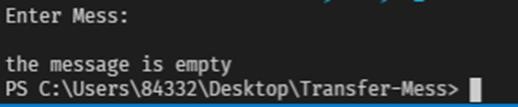
Example: in my code. I set LengthOfBuffer = 50. So, it can store only 50 characters. If you want to store more character. You must remove all data in ArrayList and reuse it.



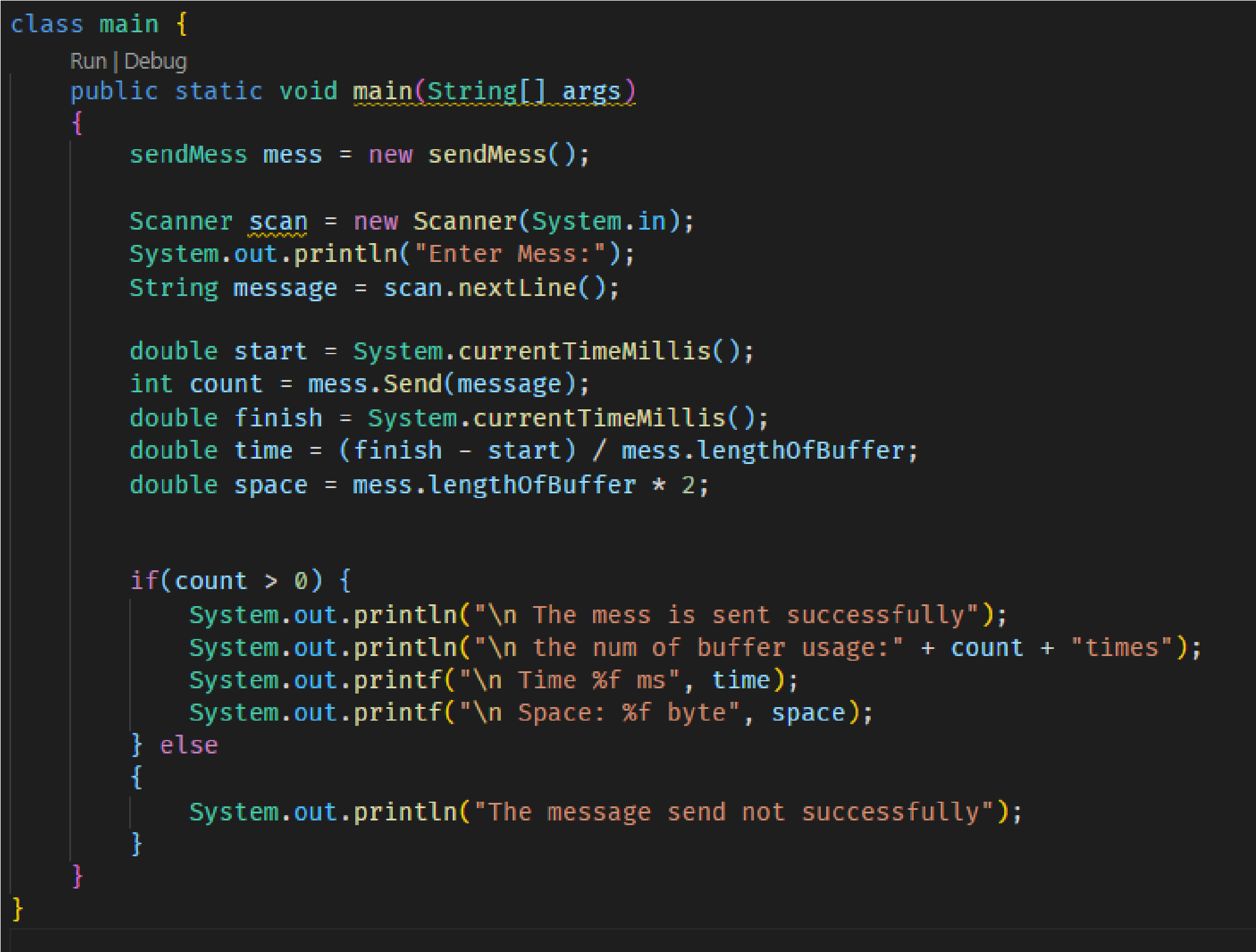
1. **Store character into a String variable**

We will store all character into a variable. First, we use while loop to check length of Queue when it smaller than 0. Because we will reduce index to 0 after add letter into “output” variable.

Count: to count numbers of buffer usage when send message.



1. **Main function**



*Figure 5 Main function*

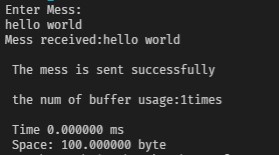
This main function use to run the program, calculate Time and Space when running. Here I use CurrentTimeMillis() to calculate. Because in my program, when I send message, I remove it and reuse ArrayList. So, I just use 100 byte only.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Serial Test case Number** | | **Input data** | **Expected output Actual output** | | **Result** |
| **1** | Enter message with lower 50 characters | Input: “hello world” | Sending success  Length = 11 Message received: hello world Buffer usage: 1 | Length = 11 Message received: hello world Buffer usage: 1 | Pass |
| **2** | Sending empty message | Input: null | The message is empty | The message is empty | Pass |
| **3** | Input over 250 characters | Input: message over 250 characters | the message too long, over 250 characters! | the message too long, over 250 characters! | Pass |
| **4** | Input special character | Input: “/  @#$” | Sending success Message received: / @#$ | Sending success  Message received: /  @#$ | Pass |
| **5** | Input Vietnamese character | Input: “Xin chào, tôi  đến từ Việt Nam” | Sending success Message received: Xin chào, tôi đến từ Việt Nam | Sending success Message received:  Xin chào, tôi đến từ  Việt Nam | Fail |

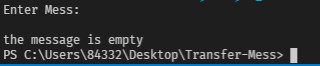
#### 3. Test case

**Image Result:**

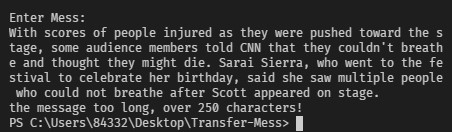
**Test case 1: Enter message with lower 50 characters**



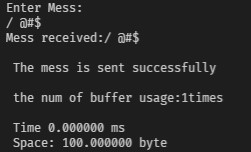
**Test case 2: Sending empty message**



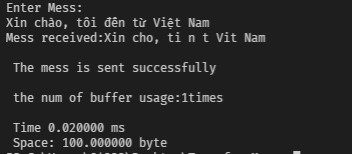
**Test case 3: Input over 250 characters**



**Test case 4: Input special character**



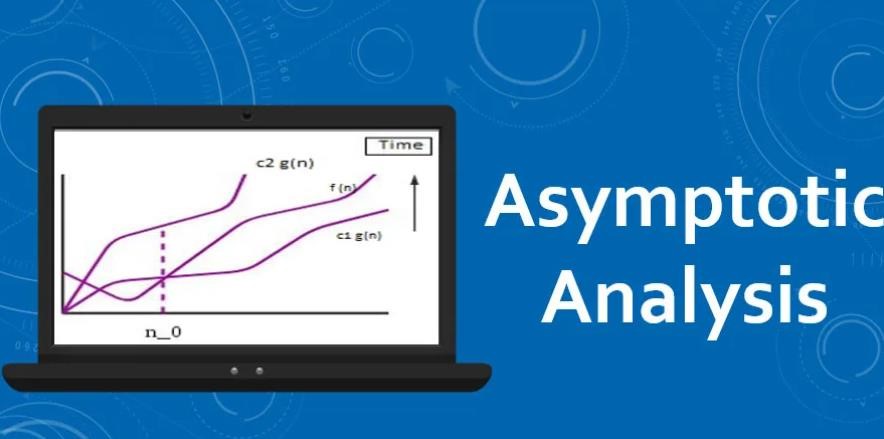
**Test case 5: Input Vietnamese character**



**P6** Discuss how asymptotic analysis can be used to assess the effectiveness of an algorithm

#### 1. 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. 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 6 Asymptotic Analysis*

This analysis needs a variable input to the algorithm, otherwise the work is assumed to require a constant amount time. All factors other than the input operation are considered constant.

**Explain Asymptotic Notation:**

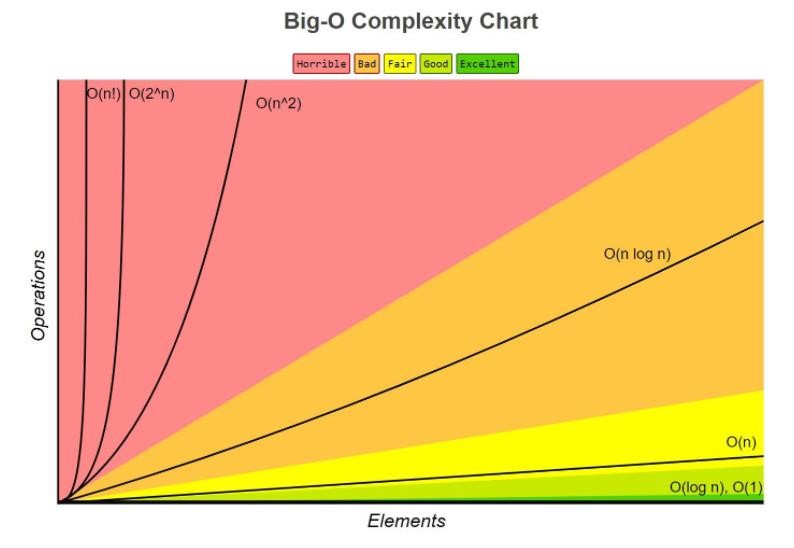
* Best Case: The time required by the algorithm is the minimum of all. E.g., in sorting algorithm when all the given numbers in the input are already sorted.
* Average Case: The time required by an algorithm is the average time taken by all. E.g., In the given algorithm, when given number in the input is half sorted.
* Worst Case: The time required by an algorithm is the maximum of all others. E.g., in sorting algorithm when all the given numbers in the input are given in reverse order as required.

**Types of Asymptotic Notations:**

* Big Oh (O Notation): This notation gives an upper bound of an algorithm, that bounds the function from above.
* Big Omega (Ω Notation): ): Big Omega represents the best case that can happen, which means that the complexity of this algorithm couldn’t be less than the value of Ω(n)
* Big Theta (Θ): Big theta represents the average that can be achieved between the worst and best case of the algorithm complexity, so Ω(n)≤Θ(n)≤O(n). The f(n) value will always between c1g(n) and c2g(n) where c1, c2 constants

#### 2. Big Oh

We must consider how long an algorithm will take to complete and how much space it will require to calculate while developing an application. And it's up to us to figure out how to cope with how quickly the algorithms' runtimes expand in relation to their input sizes so that the application can be deployed effectively.



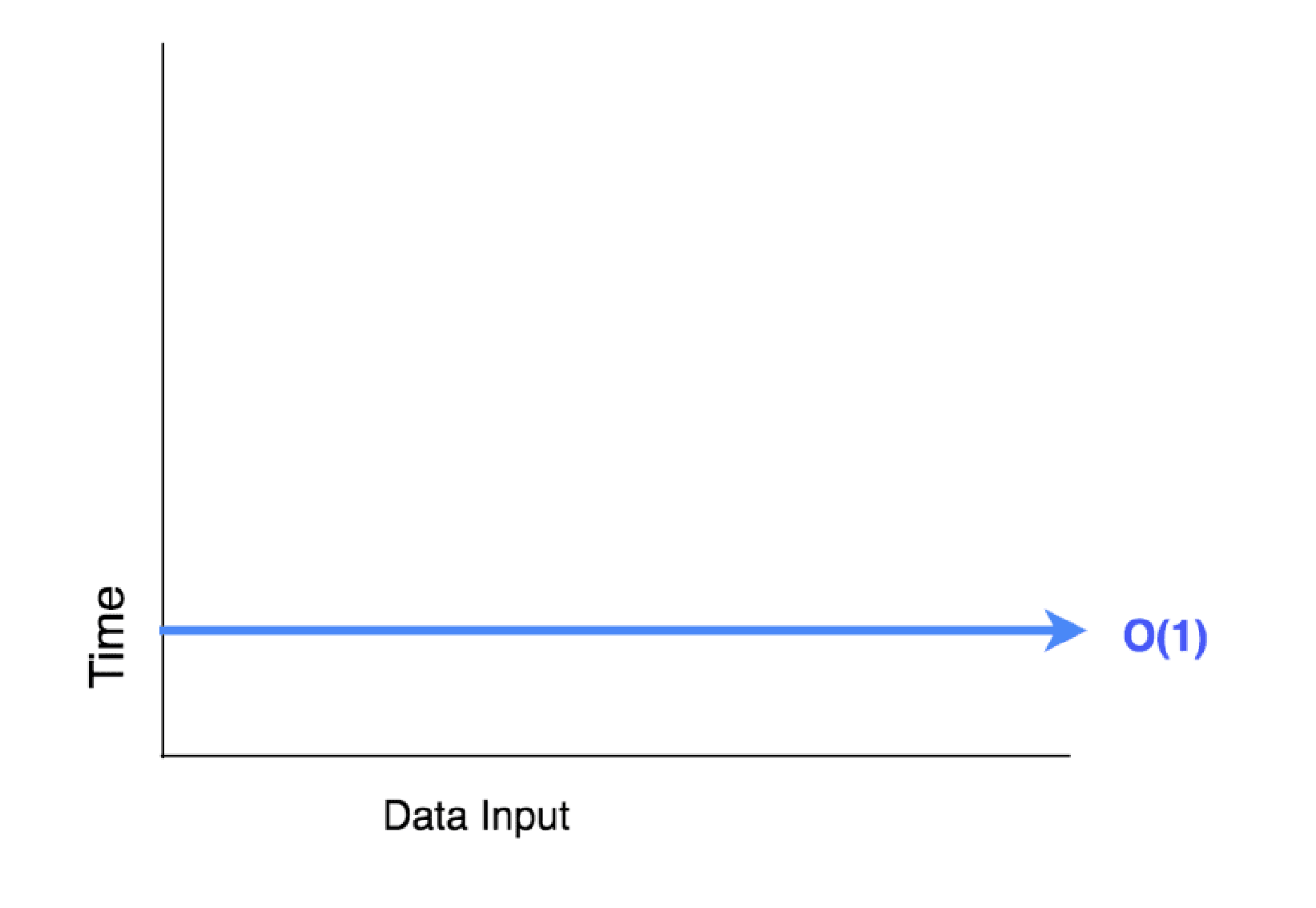
*Figure 7 Big-O Complexity*

The general step wise procedure for Big-O runtime analysis is as follows:

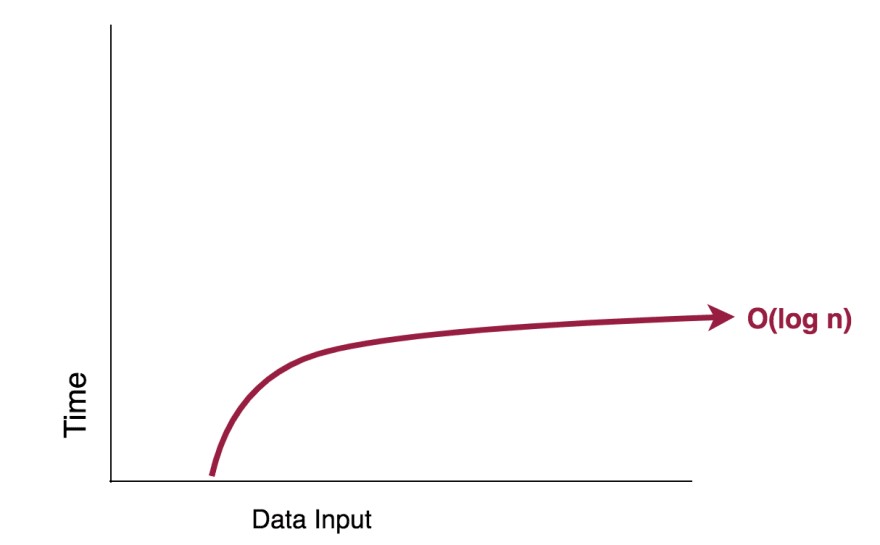
* Figure out what the input is and what n represents.
* Express the maximum number of operations, the algorithm performs in terms of n.
* Eliminate all excluding the highest order terms.
* Remove all the constant factors.

**Some common time complexities described in the Big-O notation:**

**O(1) has the least complexity**: Often called “constant time”, if you can create an algorithm to solve the problem in O(1), you are probably at your best. In some scenarios, the complexity may go beyond O(1), then we can analyze them by finding its O(1/g(n)) counterpart.

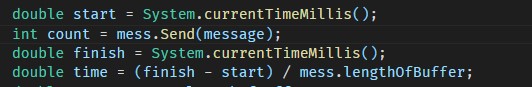


**O(log(n)) is more complex than O(1), but less complex than polynomials**: As complexity is often related to divide and conquer algorithms, O(log(n)) is generally a good complexity you can reach for sorting algorithms. O(log(n)) is less complex than O(√n), because the square root function can be considered a polynomial, where the exponent is 0.5

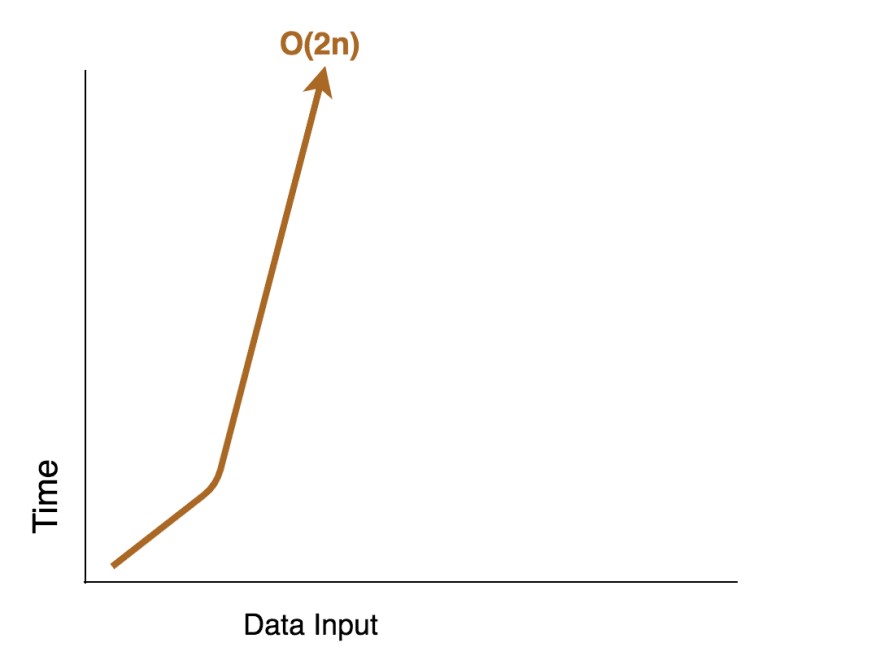


**Complexity of polynomials increases as the exponent increases:** For example, O(n⁵) is more complex than O(n⁴). Due to the simplicity of it, we actually went over quite many examples of polynomials in the previous sections.

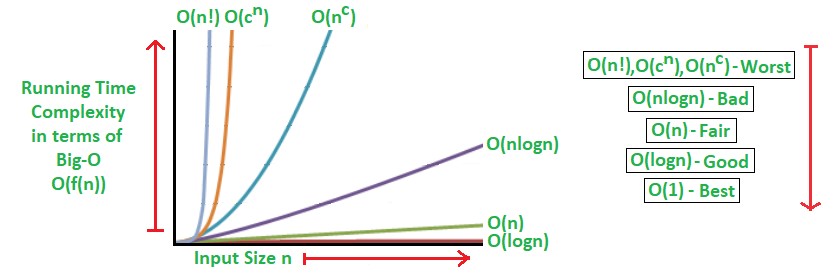
**O(n²) - Quadratic Time:** Represents an algorithm whose performance is directly proportional to the squared size of the input data set. This is common with algorithms that involve nested iterations over the data set. As the input increases, the time to run the algorithm grows at the rate of its square.



**O(2n) - Quasilinear Time:** An algorithm is said to have an exponential time or O(2^n) if its runtime doubles with each addition to the input data set, starting off very shallow, then rising meteorically.



#### 3. Some examples about Big-Oh complexity in real calculator



Algorithmic Examples of Runtime Analysis:

Some of the examples of all those types of algorithms (in worst-case scenarios) are mentioned below:

* Logarithmic algorithm – O(logn) – Binary Search.
* Linear algorithm – O(n) – Linear Search.
* Superlinear algorithm – O(nlogn) – Heap Sort, Merge Sort.
* Polynomial algorithm – O(n^c) – Strassen’s Matrix Multiplication, Bubble Sort, Selection Sort, Insertion Sort, Bucket Sort.
* Exponential algorithm – O(c^n) – Tower of Hanoi.
* Factorial algorithm – O(n!) – Determinant Expansion by Minors, Brute force Search algorithm for Traveling Salesman Problem.

**P7** Determine two ways in which the efficiency of an algorithm can be measured, illustrating your answer with an example

#### 1. Analyze performance

Algorithms are often quite different from one another, though the objective of these algorithms are the same. For example, we know that a set of numbers can be sorted using different algorithms. Number of comparisons performed by one algorithm may vary with others for the same input. Hence, time complexity of those algorithms may differ. At the same time, we need to calculate the memory space required by each algorithm.

Performance analysis of an algorithm depends upon two factors i.e. amount of memory used and amount of compute time consumed on any CPU. Formally they are notified as complexities in terms of:

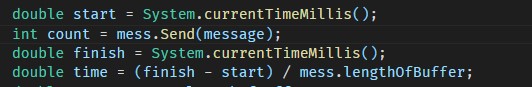
* **Space**: The amount of memory is evaluated by calculating the maximum amount of memory that the algorithm needs to use.
* **Time**: Time is evaluated by calculating the number of key operations (such as comparisons in sorting algorithms).

**Time Complexity:**

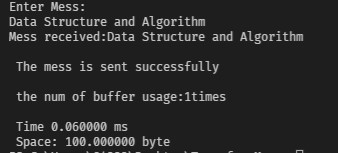
The time factor of an algorithm describes the amount of time an algorithm takes to run from start to finish.

A function T(n) can be used to describe the request time, with T(n) being the number of steps.

Example: Here I use currentTimeMillis() to calculate time when sending message. From beginning to ending.

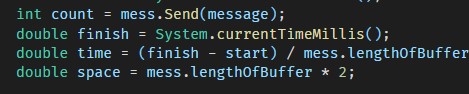


Here I use my program to example about Time complexity when I am sending the message. And this image below is result:

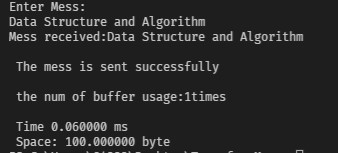


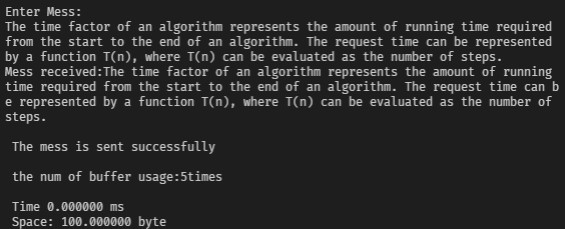
**Space Complexity:**

Space complexity analysis estimates the amount of memory required by the algorithm to process input data. While processing the input data, the algorithm needs to store the transient temporary data structures in memory. The way the algorithm is designed affects the number, type, and size of these data structures.



Here I use my program to example about Space complexity when I am sending the message. Because in my program, when I send message, I remove it and reuse ArrayList. So, I just use 100 byte only. And 2 test give me the same result. And this image below is result:





In addition, when evaluating the complexity of an algorithm, it is common to consider certain cases: bestcase, worst-case, and average-case. These cases may occur depending on the nature, or organization of the input data.

**Reference**

Chakraborty, A., n.d. [Online]

Available at: https://www.tutorialspoint.com/abstract-data-type-in-data-structures

Ramesh, A. (2019). examples.javacodegeeks.com. From https://examples.javacodegeeks.com/adtjavatutorial/

Wikipedia. (2020). Wikipedia. Retrieved 2021 from Wikipedia.vn

[Powered by TCPDF (www.tcpdf.org)](http://www.tcpdf.org/)