Logo, company name

Description automatically generated

**School of Computing, Electrical and Applied Technology**

ISCG6426

Data Structures & Algorithms

# Assignment

**Semester 2, 2021**

**Due Date: Week 13 class**

**Total Marks:** 40

**Course Weighting:** 40%

**Learning outcomes covered in this assignment**

1. Apply object-oriented design and implementation techniques.
2. Interpret the trade-offs and issues involved in the design, implementation, and application of various data structures with respect to a given problem.
3. Explain the purpose and answer questions about data structures and design patterns that illustrate strengths and weaknesses with respect to resource consumption.
4. Assess the impact of data structures on algorithms.
5. Analyse the scalability of data structures and algorithms in terms of both space and time complexity.

**Cover Sheet**

|  |  |
| --- | --- |
| First Name | Sabrian |
| Last Name | Jufenlindo |
| Student ID | 1548517 |
| Chosen Data Structure(s) or Algorithm(s) | Deque |

By submitting files and/or work to the approved Moodle submission link for this assessment, I declare that all work has been performed by myself unless explicitly declared. Any code not created by me has been cited adequately. I accept that failure to comply with the Unitec Guidelines of Appropriate Student Conduct will result in enforcement of the relevant consequences.

Signed: Sabrian Jufenlindo Date: 31/10/2021

Table of Contents

[Assignment 1](#_Toc86502635)

[Introduction 3](#_Toc86502636)

[Purpose 3](#_Toc86502637)

[Strength 4](#_Toc86502638)

[Weakness 4](#_Toc86502639)

[Real -World Example 4](#_Toc86502640)

[Deque Structure 5](#_Toc86502641)

[Implementation 10](#_Toc86502642)

[References 18](#_Toc86502643)

## Introduction

A double-ended queue is a data structure that generalizes a queue. Deque has two ends, front and rear, where the items remain positioned in the data structure. Deque is different from queue as it has an unrestrictive nature of adding and removing data from the collection. Items can be added from the collection front or the rear. Likewise, item can also be removed from its front or rear.

By having this capabilities, this data structure is not restricted by FIFO(First in first out) rule as in a stack or LIFO(Last in first out) rule as in queue.

Timeline

Description automatically generated

## Purpose

Deque is invented so that it can be used as stack and queue. Therefore, it can perform both redo and undo operations.

In computing system, a deque is really useful for multiprocessor scheduling. Assuming we have 2 processors and each processor has a process to execute. Each processor has been assigned with a process or a job, and each process contains multiple threats. Each process maintains a deque that consists of threads that are ready to be executed. The processor executes a process, and if a process creates a child process then that process will be inserted at the front of the deque of the parent process. Suppose the processor P2 has completed the execution of all its threads then it “steals” the thread from the rear end of the processor P1 and adds to the front end of the processor P2. The processor P2 will take the thread from the front end; therefore, the deletion takes from both the ends, i.e., front and rear end. This is known as the **A-steal algorithm** for scheduling.

Diagram

Description automatically generated

## Strength

Deque has several strength compared to other data structure :

* Deque uses less overall memory than an array for the same number of values.
* Deque automatically frees allocated memory when its size drops lows enough.
* The time complexity of all deque of operations is O(1).

## Weakness

Deque has several weakness as follows:

* Concurrent access is not allowed
* Iterating random indexes in deque has a complexity of O(n) as it needs to iterate each element until it gets to the element of specified index.

## Real -World Example

Some real-world example :

* Deque can be used to store a web browser’s history. Recently visited URLs are added to the front of the deque and the URL at the back of the deque is removed after some specified number of insertions at the front.
* Deque can be used to store a software application’s list of undo operations.
* As mentioned in the purpose section, deque is also commonly used in a multi-threading processor.

## Deque Structure

Each element in the deque is represented using “Node” class. This class is constructed as inner class in the Deque class. Each node holds a link to its previous and next node. Within each node, there is specific data that is stored within it. Upon constructing the node class, its previous link and next link is set to null. Code representation :

Text

Description automatically generated

Deque class will hold 3 variables : the front node, the rear node and the size of the deque. Upon constructing the deque, its front and rear will be null where as its size will be 0. Code implementation:

Text

Description automatically generated

Deque will have 6 main method as follows:

* isEmpty() method

This method will be checking whether the deque is empty or not. It is done by checking if the size is equal to 0.

Graphical user interface, text

Description automatically generated

* pushFront(T data)

This method is used to add an element to the front of the deque. It creates a new node which consist of the newly passed data. If the front is equal to null (deque is empty) then it will assign front and rear of the deque to the new node.

If the front is not null, the new node will be the front of the deque. New node next link will be the previous front. The previous front “previous link” will be the new node.

Text

Description automatically generated

* pushRear(T data)

This method is used to add an element to the rear of the deque. It creates a new node which consist of the newly passed data. If the rear is equal to null(deque is empty) then it will assign front and rear of the deque to the new node.

If the rear is not null, the new node will be the rear of the deque. New node previous link will be the previous rear. The previous rear “next link” will be the new node.

Text

Description automatically generated

* popFront()

This method is used to remove the element from the front of the deque and returns the data. Before executing the entire method, it will check whether the deque is empty, if it is empty, an exception will be thrown.

If it is not empty, we will create a temporary node and assign its reference to the current front. Then, we swap the front node of the deque to next link of the current front.

If after assigning the new value to the front and the front is now null (deque is empty), we will assign null to the rear node of the deque as well. Else, if the front is not null (deque is not empty), we assign the previous link of the front to null. After this has been completed, we reduce the size of the deque and return the data that is hold in the temporary node.

Text

Description automatically generated

* popRear()

This method is used to remove the element from the rear of the deque and returns the data. Before executing the entire method, it will check whether the deque is empty, if it is empty, an exception will be thrown.

If it is not empty, we will create a temporary node and assign its reference to the current rear. Then, we swap the rear node of the deque to previous link of the current rear.

If after assigning the new value to the rear and the rear is now null (deque is empty), we will assign null to the front of the deque as well. Else, if the rear is not null (deque is not empty), we assign the rear next link to null. After this has been completed, we reduce the size of the deque and return the data that is hold in the temporary node.

Text

Description automatically generated

* Clear method

This method is used to empty the deque. This is achieved by using while loop, iterating each element in the deque and set it to null.

Text

Description automatically generated

## Implementation

This project is created using SFML with the help of TGUI (SFML library) to implement some widget to the application.

To visualize deque, I have decided to use the ball class (extends CircleShape class) where each ball is connected to each other that represent the link of each object in the deque.

User can click “Left-Click” to push an element to the front of the deque and “Right-Click” to push an element to the rear of the deque.

To remove an element from the front, user can press “Left-Arrow” and to remove an element from the back, user can press “Right-Arrow”. To quit the application, user need to press “Escape” button.

User can also move the ball within the boundary by clicking the “Middle-Scroll” button. It will highlight the ball and allow the ball to be moved around.

All operation that user do is displayed on the screen.

I have also differentiate ball based on its colour, “Green” ball represents the top of the deque and “Blue” ball represents the bottom of the deque. All ball within the deque has its colour grading from dark to light depending on its position in the deque.

For implementation sake and to ease user to notice the colour grading, the deque in this application has a total capacity of 42. Colour that is needed for the colour grading is stored in “rgbArray” of the application. Colour is automatically updated when the application update the circle.

Each ball is also labelled with the value that is contained within the ball. This is implemented when the ball is drawn. I have also implemented a read-only text box which display the value of the ball in the deque.

The application also has the ability to clear the deque, sort the deque and search an element inside the deque. All of them can be triggered by pressing a button on the window. When an element is found, the ball will be highlighted.

Below is the code snipped and screenshots of the application :

* Implementing color of the ball

Text

Description automatically generated

Text

Description automatically generated

Diagram

Description automatically generated with low confidence

* Labelling a ball based on its value

Text

Description automatically generated

Diagram

Description automatically generated with medium confidence

* Listing all the value of the deque in a read-only text box

Text

Description automatically generated

A picture containing diagram

Description automatically generated

* Moving ball within its boundary

Diagram

Description automatically generated

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

* Clear deque

Text

Description automatically generated

* Sort the deque

Text

Description automatically generated

* To search an element in the deque

Text

Description automatically generated

Text

Description automatically generated

# References

JavaTpoint. (n.d.). *Deque*. Retrieved from JavaTpoint: https://www.javatpoint.com/ds-deque

M, S. (2021, October 15). *Learn Deque in Data Structure From Scratch*. Retrieved from simplilearn: https://www.simplilearn.com/tutorials/data-structure-tutorial/dequeue-in-data-structure

Micmap. (n.d.). *The Deque class*. Retrieved from Micmap: http://micmap.org/php-by-example/manual/en/class.ds-deque.html

Pythonds. (n.d.). *The Deque Abstract Data Type*. Retrieved from Pythonds: https://runestone.academy/runestone/books/published/pythonds/BasicDS/TheDequeAbstractDataType.html

Wikipedia. (2021, August 21). *Double-ended queue*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Double-ended\_queue