**Data Structures and Algorithms – Project Documentation**

## GitHub Repository Link details and Credentials

<https://github.com/D3nan/Tafe_ICTPRG547_24S2T1>

Note: Collaboration invite sent to julie.ruiz@tafesa.edu.au

## Hashing Specifications

The hashing specification takes into account all the properties of the Student class to allow for all possible student enrollment variations which may cause some of the properties to be equal. For example:

* If a student is repeating the same program.
* If a student is enrolling in multiple programs at the same time.

Other improvements made to the override of the GetHashCode() include:

* Multiplication of a large prime number. This prevents the likelihood of the hashcode being close to one another, reducing the likelihood of collisions.
* Both StudentID and DateRegistered don’t allow for nulls as part of their type. However, the Program property is a string, which may let in null values. Therefore, the GetHashCode will check for this and set its hashcode to zero to prevent any NullException errors.

The chosen hash algorithm is SHA-256, which is a common method of encryption in cryptographic applications. This can also be used as a hashing technique to optimise efficient searching and sorting of items in a Collection.

The process of hashing converts data into an integer value to act as an index reference within a hash table. For search applications, this makes finding the data easier and quicker, as you will only be looking for this integer value instead of potentially larger data variables like a string with hundreds of characters. Hashing also helps with sorting by grouping data into "buckets" using hash values. Once grouped, each bucket can be sorted individually, which can be more efficient than sorting the entire dataset at once. This technique is useful for organising and processing large collections of data

## Searching

**What are searching algorithms used for:**

**Sequential Search:**

**Binary Search:**

## Sorting

List 3 sorting algorithms, document 1 advantage and 1 disadvantage of each.

|  |  |  |
| --- | --- | --- |
| **Name of Sorting Algorithm** | **Advantage** | **Disadvantage** |
|  |  |  |
|  |  |  |
|  |  |  |

## Testing Searching and Sorting

**Test Plan Template**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm**  **Tested** | **Expected output** | **Actual Output** | **Pass/Fail** | **Comments** |
| **Sequential Search** |  |  |  |  |
| **Binary Search** |  |  |  |  |
| **Sort asc** |  |  |  |  |
| **Sort desc** |  |  |  |  |

**NUnit testing Screen shots - Searching and Sorting Algorithms**

## Dynamic Data Structures

**Diagram and explanation of Single Linked List**

**Diagram and explanation of Doubly Linked List**

**Diagram and explanation of Balanced Binary Search Tree**

## Testing

**Test Plan Template**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Test Case** | **Expected output** | **Actual Output** | **Pass/Fail** | **Comments** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

(add more rows as required)

**Testing Single Linked Lists Screen Shots**

**Testing Doubly Linked Lists Screen Shots**

**Testing Binary Tree Screen shots**

## Debugging

**Debugging Screen Shots**

## Further Research

|  |  |
| --- | --- |
| **Language** | **Suitability**  Provide at least 2 reasons why each language would be suitable for Data Structure Utilities and MVC (GUI) Applications: |
| C++ |  |
| C# |  |
| VB.NET |  |

**4 advantages of using Agile techniques for large scale applications:**