**Data Structures and Algorithms – Project Documentation**

## GitHub Repository Link details and Credentials

https://github.com/DavidIkey2/Assignment1.git

## Hashing Specifications

I have chosen the Student ID from my Student class to hash, as the ID is unique and best for searching. (Students may have an identical first/ last name)

Hashing can be used to optimize efficient searching and sorting or items in a collection by mapping items to unique hash values in a hash table. CityHash is often considered the best hashing method if your application involves high-performance computing or processing large volumes of data, because is has been optimized for speed and efficiency on modern CPUs. It provides low collision rates and handles both small and large data efficiently for non-cryptographic purposes. CityHash works by processing input data in chunks of varying sizes, combining them using bitwise operations, multiplication and shifts to produce the final hash value.

## Searching

**What are searching algorithms used for:**

Searching algorithms are methods used to find specific data within a collection of data. Searching algorithms can be applied to various data structures, such as arrays, lists and databases.

**Sequential Search:**

Sequential or Linear Search is the simplest form of search, checking each element one by one until the desired element is found or the list ends. Sequential search has the advantage of working on both sorted and unsorted data. (Based on tests we did in python linear searches are fine for under one million items, taking less than 0.1 seconds)

**Binary Search:**

Binary Search works by repeatedly dividing the search interval in half, comparing the target value to the middle element. It searches the left half if it is less than the middle element and the right half if it is greater. Obviously, a disadvantage of binary search is that it requires the data to be sorted. (Based on tests we did in python binary searches were far more efficient for over one million items)

## Sorting

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

|  |  |  |
| --- | --- | --- |
| **Name of Sorting Algorithm** | **Advantage** | **Disadvantage** |
| **Bubble Sort** | **Simple to understand and implement** | **Very inefficient for large datasets** |
| **Selection Sort** | **Involves minimal data movement, useful where memory writing needs to be limited** | **Limited practical use due to poor efficiency** |
| **Merge Sort** | **Best for large datasets where stability and consistent performance is required** | **Requires additional space proportional to the input size** |

## Testing Searching and Sorting

**Test Plan Template**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm**  **Tested** | **Expected output** | **Actual Output** | **Pass/Fail** | **Comments** |
| **Sequential Search In Array** | Not -1 | 0 | Pass | Found Student 111222333 |
| **Sequential Search Not In Array** | -1 | -1 | Pass | Could not find Student 111222666 |
| **Binary Search in Array** | Not -1 | 3 | Pass | Found Student 111222333 |
| **Binary Search Not In Array** | -1 | -1 | Pass | Could not find Student 111222666 |
| **Sort asc** | s9, s7, s8, s0, s1, s2, s3, s4, s5, s6 | s9, s7, s8, s0, s1, s2, s3, s4, s5, s6 | Pass | Correctly Sorted |
| **Sort desc** | 6, s5, s4, s3, s2, s1, s0, s8, s7, s9 | 6, s5, s4, s3, s2, s1, s0, s8, s7, s9 | Pass | Correctly Sorted |

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

A screenshot of a computer

Description automatically generated

## 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:**