­­CAB301

Group 96

Queensland University of Technology

Phase 3

Due 11:59 pm, Wednesday, 8th June 2022

# Introduction

This project aimed to develop a software application for a community library to manage its movie DVDs. By utilizing a variety of data structures and algorithms we developed an application that can efficiently store, manage and manipulate data relating to library members, staff, and movies. Our application allows staff members to register new movies, add and remove DVDs of existing movies, and add and remove members from the system. Member and renting details can also be displayed for staff users. Additionally, it allows registered members to view information about the library’s DVDs, borrow and return DVDs, and list the movies they’re current borrowing. Lastly, one of the key components of our app required the design and implementation of an algorithm that returns the top three most frequently borrowed movies by the members in descending order of their frequency. The computational requirements of this algorithm were explored using empirical time efficiency analysis and counting the number of basic operations executed. Lastly, the software was developed using Microsoft Visual Studio and is a C# Console Application.

This report outlines the design and analysis of the aforementioned algorithm, including exploration of the computational requirements, and demonstrates the Software Test Plan and Test Results for our application.

# Part B

### Design an algorithm and analyse the time efficiency of the algorithm using the empirical algorithm analysis technique introduced in Topic 2 for the following computational problem:

* Given an array where the elements are randomly stored, find the three largest elements in the array.

The parameter that characterizes the size of input is the number of elements, , in an array, .

**ALGORITHM**

// Given an array , of length , returns the three

// largest values in , in descending order.

// If , is returned for the smallest,

// of the three largest numbers.

**for**  **to** **do**

**if**

**else if**

**else if**

**return**

**Algorithm’s Basic Operations with a Counter**

**ALGORITHM**

// Given an array , of length , returns the three

// largest values in , in descending order.

// If , is returned for the smallest,

// of the three largest numbers.

Counter

Counter Counter // Assigning

**for**  **to** **do**

**if**

Counter Counter // 3 Assignments, 1 Comparison

**else if**

Counter Counter // 2 Assignments, 2 Comparisons

**else if**

Counter Counter // 1 Assignment, 3 Comparisons

**else**

Counter Counter // 3 Comparisons

Counter Counter Return Statement

**return**

### Empirical Time Efficiency Analysis

The algorithm was tested with arrays of size 0 to 1,000,000 in increments of 40,000. Each size was tested 10 times and the average number of basic operations executed was recorded. The arrays contained randomly stored integers between -500,000 and 500,000 inclusive.

The average ratio of and is (refer to Figure 2 in the Appendix). Since this value is almost exactly 2, this provides significant evidence to suggest that algorithm belongs to .

Chart, scatter chart

Description automatically generated

Figure : Number of basic operations executed vs. array size

As can be seen, the algorithm does indeed belong to as there exists a linear relationship between the number of elements in an array, and the number of basic operations executed.

# Software Test Plan and Test Results

Design a test plan for each of the application’s functions. In each of the test plans, please include the test scenarios/cases, actual test data for each of the test scenarios/cases. Provide screenshots of the tests and test results.

### Display the Top 3 Movies

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Movies in Library | No. Times Borrowed | Result | Screenshot |
| No movies | N/A | N/A | Success | Figure 2 |
| 1 Movie | Calvination | 1 | Success | Figure 3 |
| 2 Movies | Calvination  Calvination II | 2  1 | Success | Figure 4 |
| 4 Movies | Calvination  Calvination II  CJ’s New Beginning  The Engineer CJ | 2  1  3  0 | Success | Figure 5 |
| 10 Movies | Calvination  Calvination II  CJ’s New Beginning  The Engineer CJ  MXB344  MXB341  Any Marvel Movie  CAB420  CAB301  Any DC Movie | 2  1  3  0  5  3  15  8  12  0 | Success | Figure 6, 7 |

# Appendix

|  |  |  |
| --- | --- | --- |
| Array Size | Avg. No. Basic Ops. | **Ratio** |
| 40000 | 120104.8 |  |
| 80000 | 240098.4 | 1.999074 |
| 120000 | 360094 |  |
| 160000 | 480101.6 | 1.999603 |
| 200000 | 600112 |  |
| 240000 | 720102.8 | 1.999763 |
| 280000 | 840095.2 |  |
| 320000 | 960095.6 | 1.999776 |
| 360000 | 1080102 |  |
| 400000 | 1200098 | 1.99979 |
| 440000 | 1320104 |  |
| 480000 | 1440105 | 1.999861 |
| 520000 | 1560098 |  |
| 560000 | 1680102 | 1.999894 |
| 600000 | 1800103 |  |
| 640000 | 1920094 | 1.999898 |
| 680000 | 2040096 |  |
| 720000 | 2160099 | 1.999903 |
| 760000 | 2280105 |  |
| 800000 | 2400101 | 1.999921 |
| 840000 | 2520096 |  |
| 880000 | 2640104 | 1.99992 |
| 920000 | 2760097 |  |
| 960000 | 2880101 | 1.999924 |
| 1000000 | 3000102 |  |
|  |  | **Average Ratio:** 1.9998 |

Figure 2: Ratio between and

Text

Description automatically generated

Figure : Top 3 Movies (No movies in collection)

Text

Description automatically generated

Figure : Top 3 Movies (1 movie in collection)

Text

Description automatically generated

Figure : Top 3 Movies (2 movies in collection)

Text

Description automatically generated

Figure : Top 3 Movies (4 movies in collection)

Text

Description automatically generated

Figure : Top 3 Movies (10 movies in collection)

Table

Description automatically generated

Figure : An example of how the data for Testing was setup

Movies were created with names and number of borrowings seen in the table. These movies were added to a MovieCollection called collection, and this was passed to the ThreeLargest function.