TR
Technical Report

CAB301: Algorithms and Complexity

Technical Approach and Analysis of a Tool Library Problem

Angus Clayton Harold Macdonald

QUT
School of Computer Science

Queensland University of Technology
Garden's Point, Brisbane,
Queensland
Australia

Angus Clayton Harold Macdonald
10448888
Phone +61 417 675 508
n10448888@qut.edu.au
www.houdini.dev

Contents

Contents c

1 Introduction 1
1.1 The Report 1
1.2 The Tool Library Problem 1

2 Data Structure Design 3
2.1 Tools 3
2.2 Members 5

3 Algorithm Analysis 7

4 Test Plans and Results 11
4.1 Staff 11
4.2 Member 15

d

CHAPTER 1

Introduction

1.1 The Report

This report has been completed under the requirement for a technical analysis of the practical software solution to the Tool Library Problem. The purpose of this report and the software solution is to demonstrate knowledge of data structures and algorithms that have been presented throughout *CAB301:Algorithms and Complexity*. This report will outline the Tool Library Problem (including its purpose and requirements), data structure and algorithm designs found within the software solution, analysis (Theoretical) of algorithms used and software test plans and test results.

1.2 The Tool Library Problem

You are required to develop a software system for a non-for-profit library that will allow members to borrow a tool (a device used to carry out a particular function, mostly used for construction, repairing or cleaning). A tool will belong to a category, and of a type e.g. a tool within the "Gardening" category and a type of "Hand Tools". Tools are structured within nine (9) categories, and fifty (50) unique types (these can be found in Appendix A). The library can store multiples of the same tool, although each tool must have a unique name. When the library receives a new tool a staff member will enter the tool into the system, given its name and quantity, and is able to remove a tool or quantity of a tool from the system.

When the library acquires a new member of the public, a staff member will enter the new members details; including their first name, last name, mobile phone number and a password. A staff member can also find the contact number of a member by the members first and last name. A member will use their name and password to log into the software system to view the catalogue of tools available, borrow tools, return tools, view tools they are currently borrowing and also view which tools are most popular amongst all members.

Given this problem, you are required to develop a solution to store the data of unique tools *only* using an array or array of array's, which we will refer to as *Tool Collection*. Tools within the the Tool Collection may only store the name of the tool, the quantity, available quantity, number of times the tool has been borrowed and which members have borrowed such tool. Tool Collection stores its contents within *continuous memory*, allowing for offset data searches from the base element.

Due to Tool Collection's continuous memory, it may have better space efficiency

than other data structures, as elements have no requirement to store data addresses of neighbouring elements. Within the development of the solution for storing the information of members, you are required to store such information *only* using a Binary Search Tree. Members within the Binary Search Tree may only contain information regarding their first name, last name, contact number, password and names of the tools in which the member is borrowing. The Binary Search Tree will organise its data similarly to a *linked-list*, where each node of the tree will contain access to its left and right node counterparts (organised by a members last name, then first name; alphabetically).

These two data structure requirements are crucial to the software system structure and algorithm design, as manipulation and searching within these data structures require appropriate, complete and efficient algorithms. These data structures are further described and visually displayed within the following chapter.

CHAPTER 2

Data Structure Design

2.1 Tools

As per the requirement of tools to be stored within an array, the development of two array-type objects were used within the software solution for storing tools; a single one-dimensional array that stores Tool objects: *ToolCollection*. The other a two-dimensional jagged String array: *CatType*. *ToolCollection* is an unsorted array of size n in which has linear change with the addition or deletion of new Tool's. This structure was used to store the tools within one continuous memory structure, as the time-complexity of a resizing one-dimensional array is $O(n)$. This single dimension array in comparison to multi-dimensional arrays allows for minimisation of unused memory, the latter in the case of increasing the size would generate allocated memory in which goes unused, and the decreasing of size would require a check of n length to ensure the non-removal of data.

Using a data structure that includes a minimal amount of unused allocated data, and minimal searches and checks within its use allows for an overall faster and more efficient process. Figure 2.1 shows the structure of a Tool type object, and Figure 2.2 shows storage structure of these within the one-dimensional *ToolCollection*.

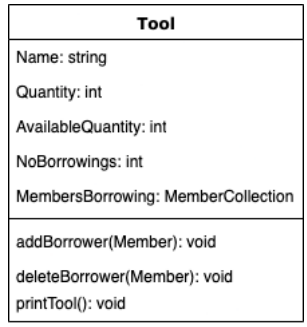


Figure 2.1: Tool Object.

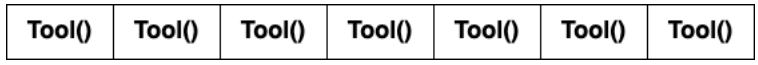


Figure 2.2: ToolCollection.

CatType is a two-dimensional jagged array that stores the names of tools that have been added to the system (Figure 2.3). The each cell within the jagged array represents a 'type' of tool, where passing over the array vertical filters throughout each 'Category', and passing over the array horizontally filters the 'Type' of tools within the category. As a Tool is added to the system, the user will select a number within range 1-9 for the category in which the tool belongs, and then another choice within range 1-5 or 1-6 (dependant on category choice).

		0	1	2	3	4	5
Gardening	0	Line Trimmers	Lawn Mowers	Hand Tools	Wheelbarrows	Garden Power Tools	
Flooring	1	Scrapers	Floor Lasers	Floor Levelling Tools	Floor Levelling Materials	Floor Hand Tools	Tiling Tools
Fencing	2	Hand Tools	Electric Fencing	Steel Fencing Tools	Power Tools	Fencing Accessories	
Measuring	3	Distance Tools	Laser Measurer	Measuring Jugs	Temperature & Humidity Tools	Levelling Tools	Markers
Cleaning	4	Draining	Car Cleaning	Vacuum	Pressure Cleaners	Pool Cleaning	Floor Cleaning
Painting	5	Sanding Tools	Brushes	Rollers	Paint Removal Tools	Paint Scrapers	Sprayers
Electronic	6	Voltage Tester	Oscilloscopes	Thermal Imaging	Data Test Tool	Insulation Testers	
Electricity	7	Test Equipment	Safety Equipment	Basic Hand Tools	Circuit Protection	Cable Tools	
Automotive	8	Jacks	Air Compressors	Battery Chargers	Socket Tools	Braking	Drivetrain

Figure 2.3: CatType Structure.

These two choices refer to an index location within the *CatType* object, in which the new Tool name is appended to the string within the index location, followed by a '~' character. When we wish to access the tools of a category and type, we then again use the users input as an index to return the string of the type. This string is then separated by the '~' character, returning us an array of tool names within that tool type. Figure 2.4 on page over demonstrates this structure in relation to a *ToolCollection* object.

	0	1	2	3	4	5
0	"Trimmer1000–LineTrim–"	**"LawnMower1~"**	Hand Tools	Wheelbarrows	Garden Power Tools	
1	Scrapers	Floor Lasers	Floor Levelling Tools	Floor Levelling Materials	Floor Hand Tools	Tiling Tools
2	Hand Tools	Electric Fencing	Steel Fencing Tools	Power Tools	Fencing Accessories	
3	Distance Tools	Laser Measurer	Measuring Jugs	Temperature & Humidity Tools	Levelling Tools	Markers
4	Draining	Car Cleaning	Vacuum	Pressure Cleaners	Pool Cleaning	Floor Cleaning
5	Sanding Tools	Brushes	Rollers	Paint Removal Tools	Paint Scrapers	**"Spray-o-matic~"**
6	Voltage Tester	Oscilloscopes	Thermal Imaging	Data Test Tool	Insulation Testers	
7	Test Equipment	Safety Equipment	Basic Hand Tools	Circuit Protection	Cable Tools	
8	**"AirJack~"**	Air Compressors	Battery Chargers	Socket Tools	Braking	Drivetrain

Tool()	**Tool()**	**Tool()**	**Tool()**	**Tool()**
Name: "Trimmer1000"	Name: "AirJack"	Name: "LawnMower1"	Name: "LineTrim"	Name: "Spray-o-matic"
Input Index: 0, 0	**Input Index**: 8, 0	**Input Index**: 0, 1	**Input Index**: 0, 0	**Input Index**: 5, 5

Figure 2.4: CatType in relation to ToolCollection.

2.2 Members

As per the requirement, members within the system must be stored within a Binary Search Tree. This tree is built by filtering the members by their last name, then first name alphabetically throughout the tree. Each *Node* within the tree contains a Member type object (Figure 2.5), and two further *Node* objects in which we refer to the *Left* and *Right* child; where the Left child contains a Member which is alphabetically before, and the Right child contains a Member which is alphabetically after.

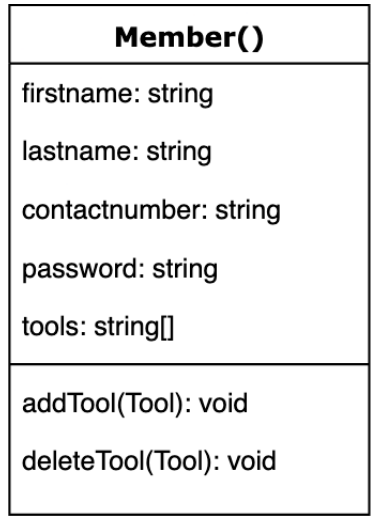


Figure 2.5: Member Object.

This data structure has a time-complexity of $O(\log_n)$; where n is the number of nodes that can be traversed within the tree, $n = \lceil n/2 \rceil$ after each node is evaluated. Figure 2.6 below shows an example of the structure storing Member objects by their last name, first name.

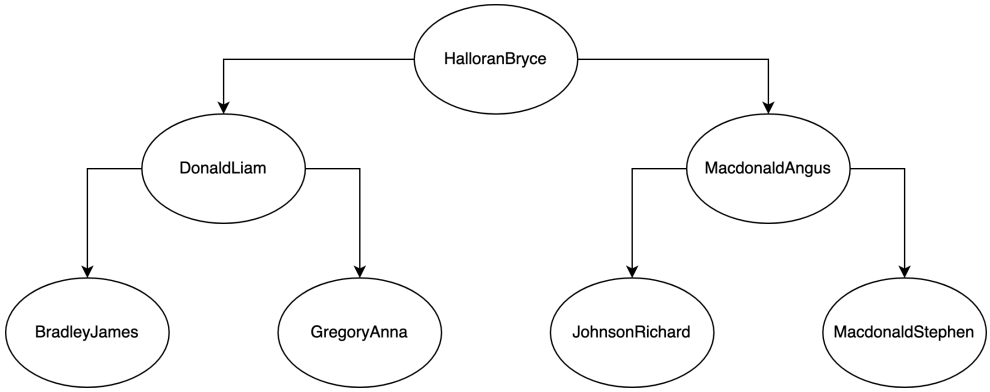


Figure 2.6: Binary Tree ordered alphabetically.

CHAPTER 3

Algorithm Analysis

Within the software system solution, there exists functionality for a member to view the top 3 most borrowed tools within the system. Each Tool object contains an internal variable that stores an integer value; initially 0, the value is increased by one each time a member borrows the tool. This internal variable, which we will refer to as *NoBorrows* has no opportunity to decrease throughout the system, and only increases through a member borrowing the tool. Our tool array needs to use a "sorting" algorithm to sort the array ascending, from there we can take the last 3 tools in descending order to find the top 3 most borrowed tools.

The decision to use a *Heap Sort* algorithm was chosen due to its efficiency, memory use and excellent time-complexity of $O(n \log(n))$. The following 2 algorithms have been implemented within the software solution to complete a *Heap Sort* on the *ToolCollection*, ordered by the *NoBorrows*.

Algorithm 1 HeapBottomUp(tools[1...n])

```
//Input: An array tools[1...n] Tool objects.
//Output: A heap tools[1...n] Tool objects.
 1: for i ← ⌊n/2⌋ downto 1 do
 2:     k ← i
 3:     v ← tools[i]
 4:     heaps ← false
 5:     while not heap and 2 * k ≤ n do
 6:         j ← 2 * k
 7:         if j < n then
 8:             if tools[j] < tools[j + 1] then
 9:                 j ← j + 1
10:             end if
11:         end if
12:         if v ≥ tools[j] then
13:             heap ← true
14:         else
15:             tools[k] ← tools[j]
16:             k ← j
17:         end if
18:     end while
19:     tools[k] ← v
20: end for
```

Algorithm 2 MaxKeyDelete(tools[1...n], size)

```
//Input: An array tools[1...n] Tool objects, An integer to refer to the size of the array.
//Output: An array tools[1...n] Tool objects.
 1: temp ← tools[0]
 2: tools[0] ← tools[size − 1]
 3: tools[size − 1] ← temp
 4: n ← size − 1
 5: v ← tools[0]
 6: heap ← false
 7: k ← 0
 8: while not heap and (2 ∗ k + 1) ≤ n − 1 do
 9:     j ← 2 ∗ k + 1
10:     if j < (n − 1) then
11:         if tools[j] < tools[j + 1] then
12:             j ← j + 1
13:         end if
14:     end if
15:     if v ≥ tools[j] then
16:         heap ← true
17:     else
18:         tools[k] ← tools[j]
19:         k ← j
20:     end if
21: end while
22: tools[k] = v
```

These two algorithms are executed to Heap Sort out tool array, where *HeapBottomUp* is executed once, then *MaxKeyDelete* is executed 3 times (to move the maximum value to the end of the array 3 times, as we are only interested in the top 3 maximum tools). The time-complexity of *MaxKeyDelete* is $\sum_{k=0}^{n-1} k = 2*k+1 = O(log(n))$ as the *while* loop has a step size that is exponential.

v represents the largest value within the function, once all other values are less than or equal to v, and the loop has iterated throughout the exponential increase of $j \leftarrow 2*k+1$ and $k \leftarrow j$, the function will exit. It can Due to the exponential value of the while loop, we know the time-complexity is $O(log(n))$ as all logarithmic functions are exponential.

HeapBottomUp contains a similar internal *while* loop, in which the k and j variables have an exponential growth between each other; ultimately has a time complexity of $O(log(n))$. This *while* loop is nested within a a *for* loop of "$i \leftarrow \lfloor n/2 \rfloor$**downto** 1".

As this *for* loop is iterable from the value of n, we understand that the scaling factor for the loop is n; thus a time-complexity of $O(n)$. Our internal loop of $O(log(n))$

is nested within this $O(n)$, so our complete time-complexity is $O(n) * O(log(n))$ or simply $O(n * log(n))$. As our entire function runs *HeapBottomUp* once, and *MaxKey-Delete* three times, we can determine that our overall time-complexity is that of the greater value, $O(n * log(n))$.

CHAPTER 4

Test Plans and Results

The following section involves a series of screenshots of the software solutions graphic user interface completing the required functionality of the Tool Library Problem.

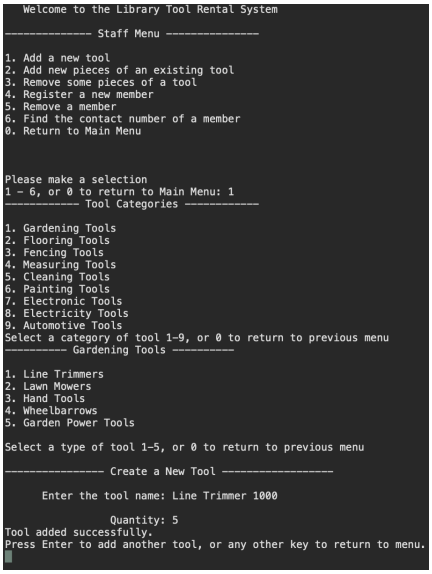
4.1 Staff

4.1.1 Log In

Staff member inputs the username ”staff” and password ”today123” they will be greeted with the staff menu.

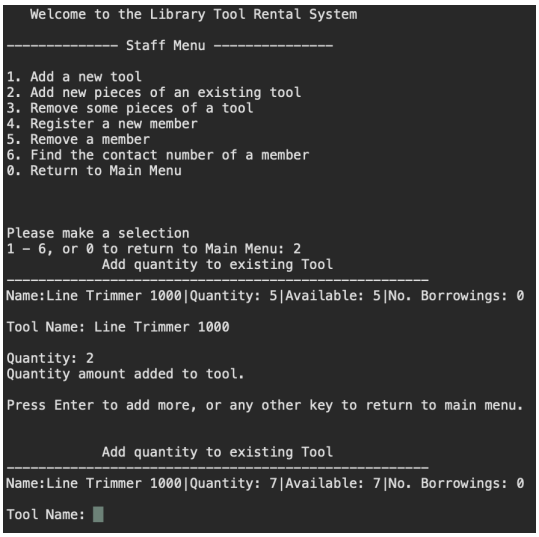
4.1.2 Add a Tool

Staff member selects the Add a Tool option, selects the category and tool type, enters a name and quantity and the tool is added to the system.



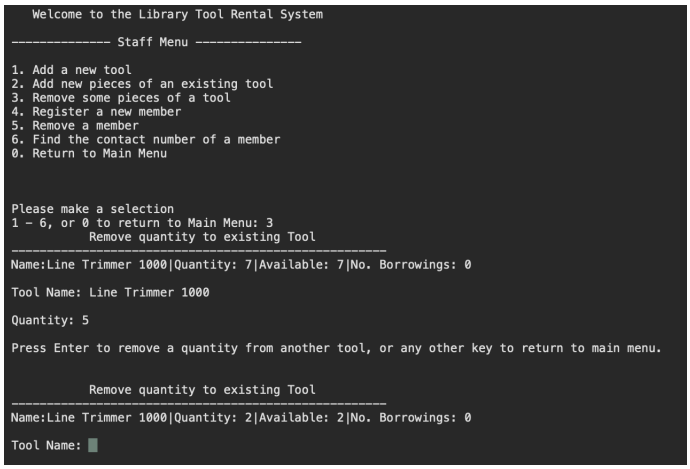
4.1.3 Add New Pieces to Existing Tool

Staff member enters the name of an existing tool, a quantity to increase and the amount is increased.



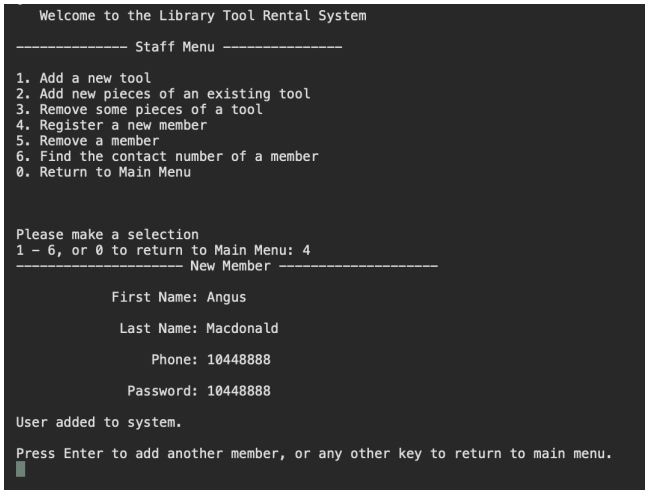
4.1.4 Remove Some Pieces From an Existing Tool

Staff member enters the name of an existing tool, a quantity to remove and the amount is removed.



4.1.5 Create a Member

Staff member enters the first name, last name, contact number and password for a new member of the library, the member is then added to the library.



4.1.6 Remove a Member

Staff member enters the first name, last name for an existing member of the library, the member is then removed from the library.

```
   Welcome to the Library Tool Rental System

-------------- Staff Menu ---------------

1. Add a new tool
2. Add new pieces of an existing tool
3. Remove some pieces of a tool
4. Register a new member
5. Remove a member
6. Find the contact number of a member
0. Return to Main Menu



Please make a selection
1 - 6, or 0 to return to Main Menu: 5
------------------ Remove Member --------------------

        First Name: Angus

         Last Name: Macdonald

This member has been deleted from the system.

Press Enter to remove another member, or any other key to return to main menu.
```

4.1.7 Contact Number of Member

Staff member enters the first name, last name for an existing member of the library, the contact number for the member is then displayed.

```
   Welcome to the Library Tool Rental System

-------------- Staff Menu ---------------

1. Add a new tool
2. Add new pieces of an existing tool
3. Remove some pieces of a tool
4. Register a new member
5. Remove a member
6. Find the contact number of a member
0. Return to Main Menu



Please make a selection
1 - 6, or 0 to return to Main Menu: 6
------- Contact Number of Member -------

Enter the member details

First name: Angus

Last Name: Macdonald

The members contact number: 10448888

Press Enter to search again,
or any other key to exit to main menu.
```

4.2 Member

4.2.1 Log In

Member inputs the first name, last name and password they will be greeted with the member menu.

```
   Welcome to the Library Tool Rental System

-------------- Member Login -------------

       First Name: Angus

       Last Name: Macdonald

       Password: ********

Logging in ...

   Welcome to the Library Tool Rental System

-------------- Member Menu --------------

1. Display all the tools of a tool type
2. Borrow a tool
3. Return a tool
4. List all the tools that I am renting
5. Display the top three(3) most frequently rented tools
0. Return to Main Menu

----------------------------------------

Please make a selection
1 - 5, or 0 to return to log out:
```

4.2.2 Display Tools of a Type

Member inputs the category and type of a tool, they will be greeted with all tools information of tools from that type.

```
   Welcome to the Library Tool Rental System

-------------- Member Menu --------------

1. Display all the tools of a tool type
2. Borrow a tool
3. Return a tool
4. List all the tools that I am renting
5. Display the top three(3) most frequently rented tools
0. Return to Main Menu

----------------------------------------

Please make a selection
1 - 5, or 0 to return to log out:

-------- Display Tools of a Type --------

1. Gardening Tools
2. Flooring Tools
3. Fencing Tools
4. Measuring Tools
5. Cleaning Tools
6. Painting Tools
7. Electronic Tools
8. Electricity Tools
9. Automotive Tools
Select a category between 1-9, or 0 to return to main menu.
1---------- Gardening Tools ----------

1. Line Trimmers
2. Lawn Mowers
3. Hand Tools
4. Wheelbarrows
5. Garden Power Tools

Select a type of tool 1-5, or 0 to return to previous menu
1
Name:Line Trimmer 1000|Quantity: 2|Available: 2|No. Borrowings: 0
Name:Line Trimmer 2000|Quantity: 1|Available: 1|No. Borrowings: 0

Press any key to return to main menu.
```

4.2.3 Borrow a Tool

Member inputs the category and type of a tool, they will be greeted with all tools information of tools from that type, from which they type the name of the tool they wish to borrow, which is then borrowed.

```
------------------ Borrow a Tool --------------------

1. Gardening Tools
2. Flooring Tools
3. Fencing Tools
4. Measuring Tools
5. Cleaning Tools
6. Painting Tools
7. Electronic Tools
8. Electricity Tools
9. Automotive Tools
Select a category between 1-9, or 0 to return to main menu.
1---------- Gardening Tools ----------

1. Line Trimmers
2. Lawn Mowers
3. Hand Tools
4. Wheelbarrows
5. Garden Power Tools

Select a type of tool 1-5, or 0 to return to previous menu
1

------------------ Borrow a Tool --------------------

Name:Line Trimmer 1000|Quantity: 2|Available: 2|No. Borrowings: 0
Name:Line Trimmer 2000|Quantity: 1|Available: 1|No. Borrowings: 0

Enter the name of the tool: Line Trimmer 1000

You are now borrowing the current tools:
Line Trimmer 1000
Press any key to return to main menu.
```

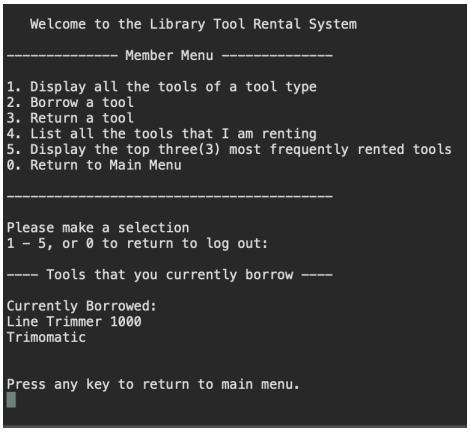
4.2.4 Return a Tool

Member inputs the name of a tool they are currently borrowing, the tool is then returned.

```
   Welcome to the Library Tool Rental System

-------------- Member Menu --------------

1. Display all the tools of a tool type
2. Borrow a tool
3. Return a tool
4. List all the tools that I am renting
5. Display the top three(3) most frequently rented tools
0. Return to Main Menu

-----------------------------------------

Please make a selection
1 - 5, or 0 to return to log out:

------- Return a Tool -------

Currently Borrowed: Line Trimmer 1000 | Trimomatic |  |
Enter the name of the tool: Trimomatic
Tool has been returned.

Press Enter to return another tool, or any key to return to menu
```

4.2.5 List All Tools Borrowing

Member can view all tools they are currently borrowing.



4.2.6 Display Top 3 Most Borrowed

Member can view the top 3 most borrowed tools within the system.