CAB301 Assignment

Algorithms and Complexity

Bill Wen
2020 Semester 1

Contents

1.0 Display Top 10 Movie Algorithm

See Appendix for the original code.

```
void topTenArray()
{
    IF BST is empty
        RETURN
    ELSE
        SET movieArray with getMovieArray()

    COMPUTE every movie in movieArray into alphabetical order using quickSortDescending()

    FOR all the movies in the movieArray up to ten movies
        PRINT movie
}

Movie[] getMovieArray()
{
    INIT movieArray
    FUNC perform recursiveIterate() on the root node
    RETURN movieArray
}


void recursiveIterate()
{
    SET node to movieArray with an index
    INCREMENT index

    IF the leftNode is empty
        THEN perform recursiveIterate() on the leftNode
    IF the rightNode is empty
        THEN perform recursiveIterate() on the rightNode
}
```

```
void quickSortDescending(titles, start, end)
{
    IF end is larger than start

        THEN FUNCTION SET variable pivot to partition(titles, start, end)

        THEN IF pivot is larger than 1
            THEN FUNCTION quickSortDescending(titles, start, pivot - 1)

        THEN IF pivote + 1 is smaller than end
            THEN FUNCTION quickSortDescending(titles, pivot +1, end)

    }
}

int partition(titles, start, end)
{
    FUNCTION perform getTimesBorrowed() to array titles[start] and assign to variable
pivot

    WHILE true
        WHILE titles[start].getTimesBorrowed() is larger than pivot
            INCREMENT start

        WHILE titles[end].getTimesBorrowed() is smaller than pivot
            DECREMENT end

        IF start is smaller than end
            THEN IF titles[start].getTimesBorrowed() is equal to
titles[end].getTimesBorrowed()
                THEN RETURN end

            SWAP title[start] with title[end]
        ELSE
            RETURN end
}
```

2.0 Algorithm Time Complexity

The entire method involves three algorithms, iterating through the binary search tree to return each movie, quick sort and iterating through each movie in the sorted array and printing it. The first and latter algorithms both have an time efficiency of *O(n)* which is of an inconsequential time taken. Thus will be omitted from the final time efficiency result.

2.1 Best Case

The best case scenario is when the pivot happens to be in the center of the list every time. Therefore, each sub-list will be of size $\log_2 n$ until the size reaches 1. This will result in the algorithm using *O(nlog n)* time.

$$O(nlogn)$$

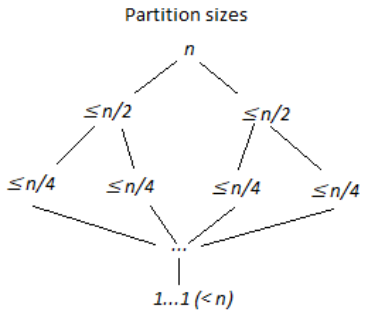

Figure 1 Best Case

2.2 Worst Case

The worst case scenario is when the pivot happens to be the largest or smallest element in the list. If this happens repeatedly the size of each partition will be n – 1 before a size of 1 is reached. This will result in $O(n^2)$ time.

$$O(n^2)$$

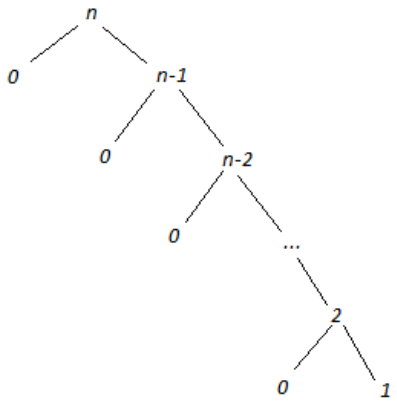

Figure 2 Worst Case

2.3 Average Case

In calculating the average case using percentiles, we can assume that if every pivot point occurs roughly in the middle between the 75th and 25th percentile it will split the list with at most an imbalance of 25% and 75%. Assuming such pivot points will be chosen every time, the list will only have to be partitioned at most $\log_{4/3} n$ iterations until reaching a size of 1 which will result in an efficiency of *O(nlog n)*.

$$O(nlogn)$$

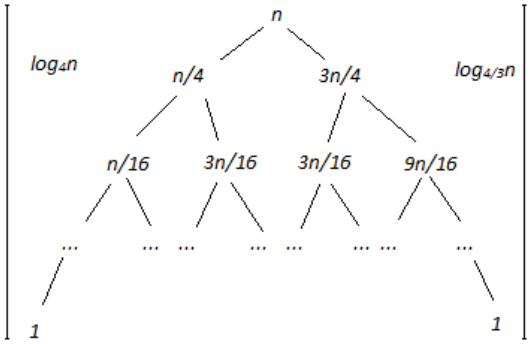

Figure 3 Average Case (Percentile)

3.0 Functional Testing Results

3.1 Main Menu

Test case: Choosing option not in range
Expected outcome: Error and prompting user to try again
Actual outcome: As expected

```
Welcome to the Community Library.
==========Main Menu==========
 1. Staff Login
 2. Member Login
 0. Exit
==============================


Please make a selection (1-2, or 0 to exit):
3
Error: Invalid Input


Please make a selection (1-2, or 0 to exit):
```

Test case: Staff login
Expected outcome: Staff login screen
Actual outcome: As expected

```
Please make a selection (1-2, or 0 to exit):
1

Enter Username:
```

Test case: Member login
Expected outcome: Member login screen
Actual outcome: As expected

```
Please make a selection (1-2, or 0 to exit):
2

Enter Username:
```

Test case: Exit
Expected outcome: Exits app
Actual outcome: As expected

```
Press any key to close this window . . .
```

3.2 Staff Menu

Test case: Log in with incorrect information
Expected outcome: Error and denied access
Actual outcome: As expected

```
Enter Username:
sf
Enter Password:
sf

Your logon information was incorrect.

Welcome to the Community Library.
===========Main Menu===========
 1. Staff Login
 2. Member Login
 0. Exit
================================


Please make a selection (1-2, or 0 to exit):
```

Test case: Log in with correct information
Expected outcome: Granted access
Actual outcome: As expected

```
Enter Username:
staff
Enter Password:
today123

==========Staff Menu==========
 1. Add a new movie DVD
 2. Remove a movie DVD
 3. Register a new Member
 4. Find a registered member's phone number
 0. Return to main menu
==============================


Please make a selection (1-4, or 0 to return to main menu):
```

Test case: Adding movie
Expected outcome: Prompts user to type movie information and adds it
Actual outcome: As expected

```
==========Staff Menu==========
 1. Add a new movie DVD
 2. Remove a movie DVD
 3. Register a new Member
 4. Find a registered member's phone number
 0. Return to main menu
==============================


Please make a selection (1-4, or 0 to return to main menu):
1
Enter the movie title:
Jaws
```

Test case: Adding existing movie
Expected outcome: Asks user if they want to add copies
Actual outcome: As expected

```
Enter the movie title:
Jaws
Enter the number of copies you would like to add:
```

Test case: Adding movie copies
Expected outcome: Asks user to input copies to add and then add I t
Actual outcome: As expected

```
Enter the movie title:
Jaws
Enter the number of copies you would like to add:
5
Jaws now has 6 copies available.
```

Test case: Removing non-existent movie
Expected outcome: Error
Actual outcome: As expected

```
Enter movie title:
Anaconda
Movie not found.
```

Test case: Removing movie
Expected outcome: Asks which movie and removes it
Actual outcome: As expected

```
Enter movie title:
Jaws
Movie has been removed.
```

Test case: Registering member
Expected outcome: Prompts user to enter member information and adds member
Actual outcome: As expected

```
Enter member's first name:
Mike
Enter member's last name:
Chen
Enter member's address:
Nice Place
Enter member's phone number:
000
Enter member's password (4 digits):
1234
Successfully registered. There are now 1 registered members.
```

Test case: Find member's phone number from name
Expected outcome: Return the correct phone number
Actual outcome: As expected

```
Enter member's first name:
Mike
Enter member's last name:
Chen
Mike Chen's phone number is: 000
```

Test case: Find non-existent member's phone number from name
Expected outcome: Error
Actual outcome: As expected

```
Enter member's first name:
Ben
Enter member's last name:
Smith
Member not found
```

Test case: Choose out of bounds option
Expected outcome: Error and prompts user to retry
Actual outcome: As expected

```
==========Staff Menu==========
 1. Add a new movie DVD
 2. Remove a movie DVD
 3. Register a new Member
 4. Find a registered member's phone number
 0. Return to main menu
==============================


Please make a selection (1-4, or 0 to return to main menu):
5
Error: Invalid Input


Please make a selection (1-4, or 0 to exit):
```

Test case: Return to main menu
Expected outcome: Returns to main menu
Actual outcome: As expected

```
Welcome to the Community Library.
===========Main Menu===========
 1. Staff Login
 2. Member Login
 0. Exit
===============================
```

3.3 Member Menu

Test case: Incorrect login
Expected outcome: Error and denied access
Actual outcome: As expected

```
Enter Username:
Ben
Enter Password:
Smith

Logon information incorrect.

Welcome to the Community Library.
==========Main Menu==========
 1. Staff Login
 2. Member Login
 0. Exit
==============================
```

Test case: Correct login
Expected outcome: Granted access
Actual outcome: As expected

```
Enter Username:
ChenMike
Enter Password:
1234

=========Member Menu=========
 1. Display all movies
 2. Borrow a movie DVD
 3. Return a movie DVD
 4. List current borrowed movie DVDs
 5. Display top 10 most popular movies
 0. Return to main menu
============================
```

Test case: Display all movies
Expected outcome: Prints all movies and their information
Actual outcome: As expected

```
Title: Jaws
Starring: Ben
Director: Jack
Genre: Drama
Classification: Mature (M15+)
Duration: 90
Release Date: 2020
Copies Available: 4
Times Borrowed: 2


Title: Finding Nemo
Starring: Jamie
Director: Bill
Genre: Thriller
Classification: Mature Accompanied (MA15+)
Duration: 30
Release Date: 2021
Copies Available: 1
Times Borrowed: 0
```

Test case: Borrow non-existent movie
Expected outcome: Error
Actual outcome: As expected

```
Enter movie title:
Jaws 2
Movie does not exist.
```

Test case: Borrow movie with no copies available
Expected outcome: Error
Actual outcome: As expected

```
Enter movie title:
Conjuring
Conjuring is unavailable.
```

Test case: Borrow movie
Expected outcome: Movie borrowed and print message telling user what they are borrowing
Actual outcome: As expected

```
Enter movie title:
Jaws
You have borrowed Jaws.
You are borrowing:
Jaws
```

Test case: Return non-existent movie
Expected outcome: Error
Actual outcome: As expected

```
Enter movie title:
Jaws 2
Jaws 2 not found.
```

Test case: Return movie
Expected outcome: Movie returned
Actual outcome: As expected

```
Enter movie title:
Jaws
Movie DVD returned.
```

Test case: List all borrowed DVDs with no DVDs borrowed
Expected outcome: Print message saying no movies borrowed
Actual outcome: As expected

```
Please make a selection (1-5, or 0 to return to main menu):
4
You have no borrowed movies.
```

Test case: List all borrowed DVDs
Expected outcome: Print message saying what movies user is borrowing
Actual outcome: As expected

```
You are borrowing:
Jaws, Shawshank Redemption
```

Test case: Display top 10 most popular movies
Expected outcome: Print message showing 10 most popular movies in order and times borrowed
Actual outcome: As expected

```
Ip Man borrowed 3 times.
Jaws borrowed 2 times.
Finding Dory borrowed 1 times.
AVP borrowed 2 times.
Shawshank Redemption borrowed 1 times.
Getting a 7 borrowed 1 times.
Finding Nemo borrowed 0 times.
The Godfather borrowed 0 times.
12 Angry Men borrowed 0 times.
Passion of Christ borrowed 0 times.
```

Test case: Choose out of bounds option
Expected outcome: Error and prompts user to retry
Actual outcome: As expected

```
=========Member Menu=========
 1. Display all movies
 2. Borrow a movie DVD
 3. Return a movie DVD
 4. List current borrowed movie DVDs
 5. Display top 10 most popular movies
 0. Return to main menu
============================


Please make a selection (1-5, or 0 to return to main menu):
6
Error: Invalid Input


Please make a selection (1-5, or 0 to exit):
```

Test case: Return to main menu
Expected outcome: Returns to main menu
Actual outcome: As expected

```
Welcome to the Community Library.
===========Main Menu===========
 1. Staff Login
 2. Member Login
 0. Exit
==============================
```

Appendix 4.0

```
void topTenArray() // Displays top 10 in terms of popularity
{
	if (root == null)
	{
		Console.WriteLine("There are no movies.");
		  return;
	}
	else
		movieArray = root.getMovieArray();

	quickSortDescending(movieArray, 0, movieArray.Length - 1);
	int moviesPrinted = 0;
	foreach (Movie movie in movieArray)
	{
		if (moviesPrinted >= 10)
		{
			return;
		}
		Console.WriteLine(movie.getTitle() + " borrowed " +
		movie.getTimesBorrowed() + " times.");
		moviesPrinted++;
	}
}

Movie[] getMovieArray() // Convert BST to array
{
	movieArray = new Movie[moviesInBST];
	index = 0;
	recursiveIterate();
	index = 0;
	return movieArray;
}

void recursiveIterate() // Function used by getMovieArray()
{
	movieArray[index] = data;
	index++;
	if (leftNode != null)
	{
		leftNode.recursiveIterate();
	}
	if (rightNode != null)
	{
		rightNode.recursiveIterate();
	}
}
```

```
void quickSortDescending(Movie[] titles, int start, int end) // Reversed quicksort
{
	if (start < end)
	{
		int pivot = partition(titles, start, end);
		if (pivot > 1)
		{
			quickSortDescending(titles, start, pivot - 1);
		}
		if (pivot + 1 < end)
		{
			quickSortDescending(titles, pivot + 1, end);
		}
	}

}

int partition(Movie[] titles, int start, int end) // Partioning function
{
	int pivot = titles[start].getTimesBorrowed();
	while (true)
	{
		while (titles[start].getTimesBorrowed() > pivot)
		{
			start++;
		}

		while (titles[end].getTimesBorrowed() < pivot)
		{
			end--;
		}

		if (start < end)
		{
			if (titles[start].getTimesBorrowed()==titles[end].getTimesBorrowed())
			{
			    return end;
			}

			Movie temp = titles[start];
			titles[start] = titles[end];
			titles[end] = temp;
		}
		else
			return end;
	}
}
```