CAB301: Algorithms and Complexity – Assignment 2

Ryan Indrananda - n10852565

Queensland University of Technology

Table of Contents

1. NoDVDs Algorithm Design

The provided MovieCollection Abstract Data Type (ADT) outlines a method, named NoDVDs, in order to correctly calculate the total number of DVDs currently in the movie collection. To implement this, we populate the movie collection as a binary search tree (BST) then calculate the total number of DVDs using in-order traversal. This algorithm essentially works in the following way (Tang, 2023):

1. Traverse the left subtree of the root node in in-order.
2. Visit the root node.
3. Traverse the right subtree of the root node in in-order.

The NoDVDs method keeps track of a count of DVDs (initially zero), then, through the use of a private helper method named InOrderTraverseCountDVDs, traverses through the movie collection as above and increments the DVD count by the total copies of the movie at the current node it is visiting, until all the movies (nodes) currently in the collection is accounted for. It returns the count of DVDs as an integer number. The pseudocode of this algorithm is:

```
ALGORITHM    NoDVDs()
    // Calculate the total number of DVDs in this movie collection
    // Output: An integer value of the total DVDs in the collection
    dvdCount ← 0
    return InOrderTraverseCountDVDs(root)

ALGORITHM    InOrderTraverseCountDVDs(root)
    // Given the root node of a BST, returns
    // Output: An integer value of the total DVDs in the collection
    if root ≠ null
        InOrderTraverseCountDVDs(root.LChild)
        dvdCount = dvdCount + root.Movie.TotalCopies
        InOrderTraverseCountDVDs(root.RChild)
    return dvdCount
```

2. NoDVDs Algorithm Empirical Analysis

We may then empirically analyse the NoDVDs algorithm to determine a hypothesis regarding the algorithm's efficiency class (Tang, 2023). To empirically analyse this, we will experimentally run the method over an increasing number of movies within the movie collection BST. The efficiency metric we will keep in mind of in this analysis is the execution time of the algorithm's implementation in relation to the increasing size of the movie collection and the total number of DVDs. The basic operation of this algorithm is the incrementation of *dvdCount* by a movie's total DVD copies done for every movie in the collection, or for every node in the BST. This analysis was done using the C# programming language.

We will analyse the NoDVDs algorithm and run it on collections with 100,000, 200,000, 400,000, 800,000, 1,600,000, 3,200,000, 6,400,000, and 12,800,000 unique movies each implemented using the following foreach loop. In each loop, the number of times the basic operation is done corresponds to the number of movies in the collection per loop. A stopwatch will be implemented, starting and stopping before and after NoDVDs is executed in every loop.

```
// Empirically analysing NoDVDs method.
Random random = new Random();
Stopwatch stopwatch = new Stopwatch();

int[] numDVDs = new int[] { 100000, 200000, 400000, 800000, 1600000, 3200000, 6400000, 12800000 };

foreach (int n in numDVDs)
{
    Console.WriteLine("Empirically analysing " + n + " movies' DVD count using NoDVDs(): ");

    stopwatch.Reset();
    stopwatch.Start();
    for (int i = 0; i < n; i++)
    {
        movieCollection.Insert(new Movie(generateMovieTitle(), (MovieGenre)random.Next(1, 5), (MovieClassification)random.Next(1, 4), random.Next(0, 200), random.Next(0, 200)));
    }
    stopwatch.Stop();

    Console.WriteLine("The time to add " + n + " movies to the collection is " + stopwatch.ElapsedMilliseconds + " milliseconds.");

    stopwatch.Reset();
    stopwatch.Start();
    movieCollection.NoDVDs();
    stopwatch.Stop();

    Console.WriteLine("The time to count the DVD count for " + n + " movies is " + stopwatch.ElapsedMilliseconds + " milliseconds.");
    Console.WriteLine("There are " + movieCollection.Number + " movies in the collection.");
    Console.WriteLine("There are " + movieCollection.NoDVDs() + " total DVDs in the collection.\n");

    movieCollection.Clear();

    Console.WriteLine("-------------------------------------------------------");
}
```

Movie titles are randomly generated using the following *generateMovieTitle* helper method within Program.cs. This method generates and returns a random 50-letter string:

```
// Helper method to generate movie titles to populate the collection for empirical analysis of NoDVDs().
// Adapted from https://stackoverflow.com/questions/1344221/how-can-i-generate-random-alphanumeric-strings.
2 references
static string generateMovieTitle()
{
    string letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
    var stringChars = new char[50];
    var random = new Random();

    for (int i = 0; i < stringChars.Length; i++)
    {
        stringChars[i] = letters[random.Next(letters.Length)];
    }

    string movieTitle = new(stringChars);
    return movieTitle;
}
```

The foreach loop above also will print out the number of movies in the collection in each loop to ensure the correct number of movies are tested and run through per loop, as the Insert method implementation does not allow for duplicate movies (based on their titles) to be inserted into the same movie collection. A sample of a randomly generated movie:

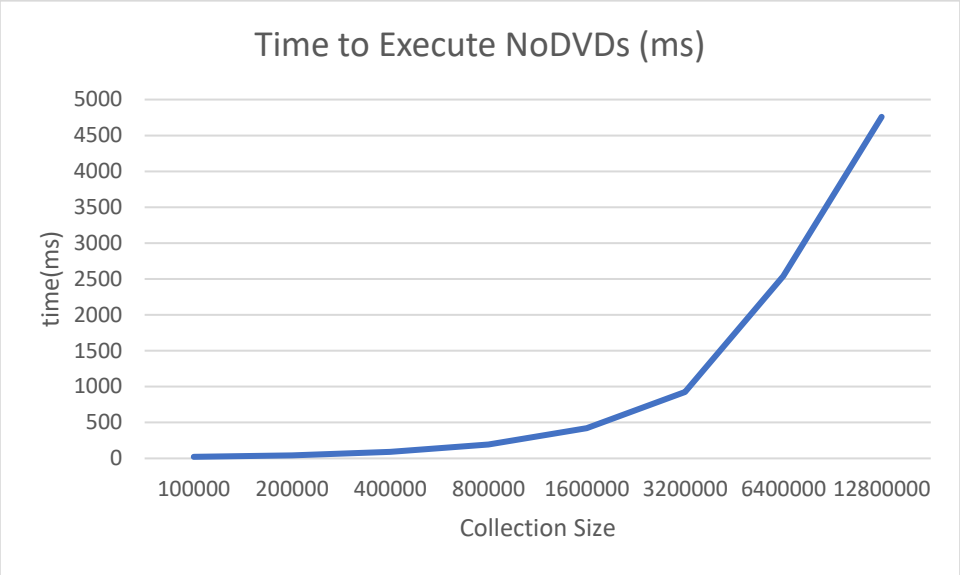
```
------------------------------------------------------
Sample randomly generated movie.

Title: UGYXZGslCYRhnEnnvwivtaJEpzGdmZanwuaEvfUUpKtHeLblrf, Genre: Action, Classification: M, Duration: 166, Available Copies: 85
------------------------------------------------------
```

The results of the foreach loop are the following:

```
Empirically analysing 100000 movies' DVD count using NoDVDs():
The time to add 100000 movies to the collection is 3783 milliseconds.
The time to count the DVD count for 100000 movies is 22 milliseconds.
There are 100000 movies in the collection.
There are 9965061 total DVDs in the collection.

------------------------------------------------------
Empirically analysing 200000 movies' DVD count using NoDVDs():
The time to add 200000 movies to the collection is 7134 milliseconds.
The time to count the DVD count for 200000 movies is 44 milliseconds.
There are 200000 movies in the collection.
There are 19876273 total DVDs in the collection.

------------------------------------------------------
Empirically analysing 400000 movies' DVD count using NoDVDs():
The time to add 400000 movies to the collection is 15375 milliseconds.
The time to count the DVD count for 400000 movies is 93 milliseconds.
There are 400000 movies in the collection.
There are 39800403 total DVDs in the collection.

------------------------------------------------------
Empirically analysing 800000 movies' DVD count using NoDVDs():
The time to add 800000 movies to the collection is 33850 milliseconds.
The time to count the DVD count for 800000 movies is 193 milliseconds.
There are 800000 movies in the collection.
There are 79545013 total DVDs in the collection.

------------------------------------------------------
Empirically analysing 1600000 movies' DVD count using NoDVDs():
The time to add 1600000 movies to the collection is 69116 milliseconds.
The time to count the DVD count for 1600000 movies is 422 milliseconds.
There are 1600000 movies in the collection.
There are 159132101 total DVDs in the collection.

------------------------------------------------------
Empirically analysing 3200000 movies' DVD count using NoDVDs():
The time to add 3200000 movies to the collection is 119093 milliseconds.
The time to count the DVD count for 3200000 movies is 924 milliseconds.
There are 3200000 movies in the collection.
There are 318283934 total DVDs in the collection.

------------------------------------------------------
Empirically analysing 6400000 movies' DVD count using NoDVDs():
The time to add 6400000 movies to the collection is 272164 milliseconds.
The time to count the DVD count for 6400000 movies is 2541 milliseconds.
There are 6400000 movies in the collection.
There are 636787096 total DVDs in the collection.

------------------------------------------------------
Empirically analysing 12800000 movies' DVD count using NoDVDs():
The time to add 12800000 movies to the collection is 541597 milliseconds.
The time to count the DVD count for 12800000 movies is 4759 milliseconds.
There are 12800000 movies in the collection.
There are 1273738057 total DVDs in the collection.

------------------------------------------------------
```

We will tabulate and chart the above results for better viewing and analysis:

size	100000	200000	400000	800000	1600000	3200000	6400000	12800000
time(ms)	22	44	93	193	422	924	2541	4759



We will now calculate the ratios of $t(2n)$ and $t(n)$ within the data.

$$\frac{t(200000)}{t(100000)} = \frac{44}{22} = 2.000, \frac{t(400000)}{t(200000)} = \frac{93}{44} = 2.114, \frac{t(800000)}{t(400000)} = \frac{193}{93} = 2.075,$$

$$\frac{t(1600000)}{t(800000)} = \frac{422}{193} = 2.187, \frac{t(3200000)}{t(1600000)} = \frac{924}{422} = 2.190, \frac{t(6400000)}{t(3200000)} = \frac{2541}{924} = 2.75,$$

$$\frac{t(12800000)}{t(6400000)} = \frac{4759}{2541} = 1.873$$

From these results, we can see that the ratios of $t(2n)$ and $t(n)$ tend to be greater than 2, therefore the algorithm is less efficient than $O(n)$. The ratios also tend to be less than 4, therefore the algorithm is more efficient than $O(n^2)$. Therefore, we can deduce that the time efficiency class of the algorithm is between $O(n)$ and $O(n^2)$. From the graph above, we can confirm that the time efficiency class of the NoDVDs algorithm is $O(n \log n)$, which is between $O(n)$ and $O(n^2)$. The final ratio between NoDVDs execution times for collections of 12800000 and 6400000 movies is less than 2, however this can potentially be attributed to random variance and future executions of NoDVDs against the two collections may yield different results. Potentially calculating averages over a number of NoDVDs executions instead of one execution would yield a consistent $2 < \frac{t(2n)}{t(n)} < 4$ result.

3. Algorithm Testing

a. Testing Plan and Data

All testing will be done manually through the creation of a Program.cs file and Main method. The file begins by instantiating the movie collection BST.

```
// Initialise a movie collection binary search tree (BST)
IMovieCollection movieCollection = new MovieCollection();
```

We will test to see if each implemented method as described within the Movie and MovieCollection ADTs behave as intended.

Under the Movie ADT, which is used to define the fields that make up a movie object (their title, genre, classification, duration, available copies, and total copies), we find methods to implement to correctly display a movie object, ToString, and to compare the title of a movie object against another title, CompareTo. ToString must return a string which contains details of any movie object passed through it, namely their title, genre, classification, duration, and number of available copies. CompareTo has multiple possible integer value returns: -1 if the movie's title is less than another movie's title by dictionary order, 0 if both movies' titles are the same, and 1 if greater than. Both these methods do not have a pre-condition.

Under the MovieCollection ADT, representing a collection of movies, we find various methods by which we may interact with said collection. The singular invariant of this ADT states that there should be no duplicate movies in the collection, and Number of movies in the collection must be greater than or equal to 0. The IsEmpty bool method is used to check whether the movie collection is empty or not, returning 'true' if it is and 'false' otherwise. However the method runs, we must ensure that the movie collection and the number of movies within it remain unchanged. Next, the Insert method allows the user to insert a movie object into the movie collection. As long as the movie does not previously exist within the collection, the method adds the movie into the collection and increments the Number of movies. If the movie previously exists, then the method returns false and all aspects of the collection remain the same. The Delete method allows the user to delete a movie object from the movie collection. If the movie does exist in the collection, then it removes the movie object, decrements the Number by one, and returns true. Otherwise, it returns false and all aspects of the collection remain the same. The Search method, called with a movie's title, returns the reference of the movie object if the movie does exist in the collection, otherwise it returns null and all aspects of the collection remain the same. The NoDVDs method, as explored above, counts the number

of total DVDs currently in the movie collection. The method returns an integer value of the number of total DVDs, and the collection remains unchanged. The ToArray method converts the movie collection into an array stored in descending dictionary order by their titles. This, similar to the NoDVDs method, was implemented using in-order traversal. Finally, the Clear method must remove all the movies in the collection and set Number to 0. All methods do not have a pre-condition.

The testing plan to test each method as per the descriptions and conditions above is as follows:

Test	**Precondition**	**Postcondition**	**Input Data**	**Expected Output**
1. Using IsEmpty method on empty collection.	Given an empty collection,	Returns True, if collection empty. Collection unchanged, new Number = old Number.	Initially empty movie collection.	Write a message if the IsEmpty method returns True. Collection unchanged, new Number = old Number.
2. Insert method.	Given a valid movie object,	Add movie to collection and new Number = old Number + 1 and **return true**, if not previously in collection.	Valid movie objects.	Returns True for every movie object inserted, new Number = old Number + number of movies inserted.
3. IsEmpty method on a populated movie collection.	Given a populated collection,	Returns False. Collection unchanged, new Number = old Number.	Populated movie collection.	Return False. Collection unchanged, new Number = old Number.
4. ToArray method.	Given a movie collection with movie objects in it,	Return array of movies stored in dictionary order by their titles. Collection remains unchanged, new	Populated movie collection.	Array of movies sorted in dictionary order by their titles. We will use a printArray helper

		Number = old Number.		method to check this. This will be discussed later in the report.
5. Insert existing movies into the BST.	Given movie objects we to already exist in the collection,	Movies not added to the collection. New Number = old Number and **return false**.	Existing movies.	Return False. Movies not added as they already exist. Collection and Number unchanged.
6. Search for movies in the collection.	Given existing movies and their titles,	Return reference of the movie object searched for.	Existing movies' title	Reference of movie object. Collection and Number unchanged.
7. Search for movies that do not exist in the collection.	Given non-existent movie titles,	Returns null	Non-existent movie title	Write a message that movie does not exist if Search method returns null. Collection and Number unchanged.
8. Use NoDVDs method to count total DVDs in the collection.	Given movie collection,	Return integer value of total DVDs in collection.	Populated movie collection.	Write message showing Integer value of total DVDs.
9. Delete method to delete movies in collection.	Given existing movie object,	Delete movie from collection. Return True. New Number = old Number – 1.	Existing movie object.	Movie deleted from collection. Return True. Number = old Number – 1.

10. Delete movies that do not exist in the collection.	Given non-existent movie object,	No movies deleted, return False, new Number = old Number.	Non-existent movie object,	No movies deleted, method returns False, collection and Number unchanged.
11. ToArray method to convert collection to array.	Given movie collection,	Return array of movies in the collection in descending dictionary order.	Populated movie collection	Array of movies in descending dictionary order. Use printArray helper method to print the array.
12. Clear method to clear the collection.	Given movie collection,	Deletes all movies from the collection, Number = 0.	Populated movie collection	All movies deleted, Number = 0.
13. NoDVDs when movie collection is empty.	Given movie collection is empty,	Number of total DVDs should return 0.	Empty movie colleciton	Zero DVDs in the collection.
14. Testing the CompareTo method.	Given a string and a movie's title to compare,	Returns 0 if movie title are the same, -1 if dictionary order of the string is less than the movie's title, 1 if greater than.	Same string, lesser string ("AAA") in terms of dictionary order, greater string ("ZZZ") in terms of dictionary order	0 for same title, -1 for lesser title, 1 for greater title.
15. Testing the ToString method.	Given movie object,	Returns movie title, genre, classification, duration, and available copies in that order.	Movie object	Movie title, genre, classification, duration, available copies in that order.

The tests will done on a 64-bit Windows 11 Home version 22H2 build 22621.1555 operating system laptop, with an AMD Ryzen 7 4800HS 2.9 GHz CPU, NVIDIA GeForce GTX 1660 Ti Max-Q dGPU, AMD Radeon(TM) iGPU, and 16GB of RAM. It is connected via Ethernet. All C# code implementations were done and will be ran in Microsoft Visual Studio Community 2022 (64-bit) version 17.4.4 with .NET Framework version 4.8.09032.

b. Testing Results

Test 1: Using IsEmpty method on empty collection.

```
// Test 1: Check if empty initially using IsEmpty()
Console.WriteLine("Test 1: Check if the BST is empty initially using the IsEmpty method.\n");
if (movieCollection.IsEmpty() == true)
{
    Console.WriteLine("The movie collection is empty.\nThere are " + movieCollection.Number + " movies in the collection.");
}

Console.WriteLine("-------------------------------------------------------");
```

Result:

```
Test 1: Check if the BST is empty initially using the IsEmpty method.

The movie collection is empty.
There are 0 movies in the collection.
-------------------------------------------------------
```

IsEmpty method behaves as intended. The output is as the expected output, as IsEmpty() returns True and writes a message saying the movie collection is empty. Number of movies in the collection stay the same.

Test 2: Insert method to add movies to the BST.

```
// Test 2: Use Insert method to insert some Movies into the BST. Use ToArray to
Console.WriteLine("Test 2: Use Insert method to insert some Movies into the BST.\n");
Console.WriteLine(movieCollection.Insert(new Movie("Shiva Baby", MovieGenre.Comedy, MovieClassification.M, 78, 12)));
Console.WriteLine(movieCollection.Insert(new Movie("Aftersun", MovieGenre.Drama, MovieClassification.M, 101, 7)));
Console.WriteLine(movieCollection.Insert(new Movie("The Good, the Bad and the Ugly", MovieGenre.Western, MovieClassification.M15Plus, 161, 18)));
Console.WriteLine(movieCollection.Insert(new Movie("Come and See", MovieGenre.History, MovieClassification.M, 142, 24)));
Console.WriteLine(movieCollection.Insert(new Movie("Fallen Angels", MovieGenre.Action, MovieClassification.M, 98, 13)));
Console.WriteLine(movieCollection.Insert(new Movie("Spider-Man: Into the Spider-Verse", MovieGenre.Action, MovieClassification.PG, 117, 3)));

// Check if the 'count' of movies in the collection is correct.
if (movieCollection.IsEmpty() != true)
{
    Console.WriteLine("\nThe movie collection is not empty.\nThere are " + movieCollection.Number + " movies in the collection");
}

Console.WriteLine("-------------------------------------------------------");
```

Result:

```
Test 2: Use Insert method to insert some Movies into the BST.

True
True
True
True
True
True

The movie collection is not empty.
There are 6 movies in the collection
--------------------------------------------------------
```

Insert method works as intended. It returns true for the six movie objects inserted into the movie collection. The number correctly reflects how many movies were inserted to the collection.

Test 3: IsEmpty method on a populated movie collection.

```
// Test 3: IsEmpty method on a populated movie collection.
Console.WriteLine("Test 3: IsEmpty method on a populated movie collection.\n");
if (movieCollection.IsEmpty() == false)
{
    Console.WriteLine("The movie collection is not empty.\nThere are " + movieCollection.Number + " movies in the collection");
}

Console.WriteLine("--------------------------------------------------------");
```

Result:

```
Test 3: IsEmpty method on a populated movie collection.

The movie collection is not empty.
There are 6 movies in the collection
--------------------------------------------------------
```

IsEmpty behaves as intended, returning false and prints a message saying collection is not empty.

Test 4: ToArray method.

A helper method ‘printArray’ was implemented to help test the ToArray method. This method is as follows:

```
// Helper method to print the movies array.
5 references
static void printArray(IMovie[] A)
{
    for (int i = 0; i < A.Length; i++)
        Console.Write(A[i].ToString());
}
```

```
// Test 4: Use ToArray and the printArray helper method to show the movies in the BST.
Console.WriteLine("Test 4: Use ToArray and the printArray helper method to show the movies in the BST.\n");

IMovie[] movies = movieCollection.ToArray();
printArray(movies);

Console.WriteLine("-------------------------------------------------------");
```

Result:

```
Test 4: Use ToArray and the printArray helper method to show the movies in the BST.

Title: Aftersun, Genre: Drama, Classification: M, Duration: 101, Available Copies: 7
Title: Come and See, Genre: History, Classification: M, Duration: 142, Available Copies: 24
Title: Fallen Angels, Genre: Action, Classification: M, Duration: 98, Available Copies: 13
Title: Shiva Baby, Genre: Comedy, Classification: M, Duration: 78, Available Copies: 12
Title: Spider-Man: Into the Spider-Verse, Genre: Action, Classification: PG, Duration: 117, Available Copies: 3
Title: The Good, the Bad and the Ugly, Genre: Western, Classification: M15Plus, Duration: 161, Available Copies: 18
-------------------------------------------------------
```

ToArray works as intended. A complete array of the movies sorted in descending dictionary order of their titles is returned.

Test 5: Inserting existing movies into the BST.

```
// Test 5: Insert existing movies into the BST.
Console.WriteLine("Test 5: Insert existing movies into the BST.\n");

if (movieCollection.Insert(new Movie("Aftersun", MovieGenre.Drama, MovieClassification.M, 101, 7)) == false)
    Console.WriteLine("This movie already exists in the collection.");
if (movieCollection.Insert(new Movie("Spider-Man: Into the Spider-Verse", MovieGenre.Action, MovieClassification.PG, 117, 3)) == false)
    Console.WriteLine("This movie already exists in the collection.");

Console.WriteLine();

movies = movieCollection.ToArray();
printArray(movies);

Console.WriteLine("\nThere are " + movieCollection.Number + " movies in the collection.");

Console.WriteLine("-------------------------------------------------------");
```

Result:

```
Test 5: Insert existing movies into the BST.

This movie already exists in the collection.
This movie already exists in the collection.

Title: Aftersun, Genre: Drama, Classification: M, Duration: 101, Available Copies: 7
Title: Come and See, Genre: History, Classification: M, Duration: 142, Available Copies: 24
Title: Fallen Angels, Genre: Action, Classification: M, Duration: 98, Available Copies: 13
Title: Shiva Baby, Genre: Comedy, Classification: M, Duration: 78, Available Copies: 12
Title: Spider-Man: Into the Spider-Verse, Genre: Action, Classification: PG, Duration: 117, Available Copies: 3
Title: The Good, the Bad and the Ugly, Genre: Western, Classification: M15Plus, Duration: 161, Available Copies: 18

There are 6 movies in the collection.
--------------------------------------------------------
```

Insert works as intended when trying to add pre-existing movies to the collection. The method returns false and writes a message to the user stating the movie already exists in the collection. The movies are not added as evident by using ToArray and printArray. The existing movie collection and the number of movies in it are unchanged.

Test 6: Search for movies in the collection.

```
// Test 6: Use Search method to search for movies in the BST. Count should not change as per the post-condition.
Console.WriteLine("Test 6: Use Search method to search for movies in the BST. Count should not change as per the post-condition.\n");
Console.Write(movieCollection.Search("Fallen Angels").ToString());
Console.Write(movieCollection.Search("Aftersun").ToString());
Console.WriteLine();

movies = movieCollection.ToArray();
printArray(movies);
Console.WriteLine("There are " + movieCollection.Number + " movies in the collection.");
```

Result:

```
Test 6: Use Search method to search for movies in the BST. Count should not change as per the post-condition.

Title: Fallen Angels, Genre: Action, Classification: M, Duration: 98, Available Copies: 13
Title: Aftersun, Genre: Drama, Classification: M, Duration: 101, Available Copies: 7

Title: Aftersun, Genre: Drama, Classification: M, Duration: 101, Available Copies: 7
Title: Come and See, Genre: History, Classification: M, Duration: 142, Available Copies: 24
Title: Fallen Angels, Genre: Action, Classification: M, Duration: 98, Available Copies: 13
Title: Shiva Baby, Genre: Comedy, Classification: M, Duration: 78, Available Copies: 12
Title: Spider-Man: Into the Spider-Verse, Genre: Action, Classification: PG, Duration: 117, Available Copies: 3
Title: The Good, the Bad and the Ugly, Genre: Western, Classification: M15Plus, Duration: 161, Available Copies: 18
There are 6 movies in the collection.
--------------------------------------------------------
```

Search method works as intended. It correctly returns the reference of the object (with the help of ToString). The movie collection and number of movies in it remain unchanged.

Test 7: Search for movies that do not exist in the collection.

```
// Test 7: Search movies that do not exist. Should return a null value as per the post-condition. Count should not change.
Console.WriteLine("Test 7: Search movies that do not exist. Should return a null value as per the post-condition. Count should not change.\n");
if (movieCollection.Search("Pride and Prejudice") == null)
    Console.WriteLine("Pride and Prejudice does not exist in the collection. Search method returns a null value.\n");

movies = movieCollection.ToArray();
printArray(movies);
Console.WriteLine("There are " + movieCollection.Number + " movies in the collection.");

Console.WriteLine("------------------------------------------------------------");
```

Result:

```
Test 7: Search movies that do not exist. Should return a null value as per the post-condition. Count should not change.

Pride and Prejudice does not exist in the collection. Search method returns a null value.

Title: Aftersun, Genre: Drama, Classification: M, Duration: 101, Available Copies: 7
Title: Come and See, Genre: History, Classification: M, Duration: 142, Available Copies: 24
Title: Fallen Angels, Genre: Action, Classification: M, Duration: 98, Available Copies: 13
Title: Shiva Baby, Genre: Comedy, Classification: M, Duration: 78, Available Copies: 12
Title: Spider-Man: Into the Spider-Verse, Genre: Action, Classification: PG, Duration: 117, Available Copies: 3
Title: The Good, the Bad and the Ugly, Genre: Western, Classification: M15Plus, Duration: 161, Available Copies: 18
There are 6 movies in the collection.
------------------------------------------------------------
```

Search behaves as intended when searching for non-existent movies. It returns a null value, and the test prints a message telling the user that the movie does not exist in the collection. Collection and Number remain unchanged.

Test 8: Use NoDVDs method to count total DVDs in the collection.

```
// Test 8: Use NoDVDs method to count how many total DVDs are available in the collection.
// Movie collection (using ToArray implementation to check) and count remain unchanged.
Console.WriteLine("Test 8: Use NoDVDs method to count how many total DVDs are available in the collection. " +
    "Movie collection (using ToArray implementation to check) and count remain unchanged.\n");
Console.WriteLine("There are " + movieCollection.NoDVDs() + " total DVDs in the collection.\n");

movies = movieCollection.ToArray();
printArray(movies);

Console.WriteLine("\nThere are " + movieCollection.Number + " movies in the collection.");

Console.WriteLine("------------------------------------------------------------");
```

Result:

```
Test 8: Use NoDVDs method to count how many total DVDs are available in the collection. Movie collection (using ToArray
implementation to check) and count remain unchanged.

There are 77 total DVDs in the collection.

Title: Aftersun, Genre: Drama, Classification: M, Duration: 101, Available Copies: 7
Title: Come and See, Genre: History, Classification: M, Duration: 142, Available Copies: 24
Title: Fallen Angels, Genre: Action, Classification: M, Duration: 98, Available Copies: 13
Title: Shiva Baby, Genre: Comedy, Classification: M, Duration: 78, Available Copies: 12
Title: Spider-Man: Into the Spider-Verse, Genre: Action, Classification: PG, Duration: 117, Available Copies: 3
Title: The Good, the Bad and the Ugly, Genre: Western, Classification: M15Plus, Duration: 161, Available Copies: 18

There are 6 movies in the collection.
------------------------------------------------------------
```

NoDVDs behaves as intended and correctly counts and returns the amount of DVDs in the collection. Manually counting: $7 + 24 + 13 + 12 + 3 + 18 = 77$. Collection and Number remain unchanged.

Test 9: Delete method to delete movies in the collection.

```
/* Test 9: Use Delete method to delete movies in the BST, then use NoDVDs and ToArray methods and count to check if the correct
movies are deleted. Delete method should return true. Count should reduce by however many movies are deleted. */
Console.WriteLine("Test 9: Use Delete method to delete movies in the BST, then use NoDVDs method to check if the correct\n" +
    "movies are deleted. Movie and DVD count should reduce by however many movies are deleted.\n");
Console.WriteLine(movieCollection.Delete(new Movie("Come and See", MovieGenre.History, MovieClassification.M, 142, 24)));
Console.WriteLine(movieCollection.Delete(new Movie("Shiva Baby", MovieGenre.Comedy, MovieClassification.M, 78, 12)));
Console.WriteLine();

movies = movieCollection.ToArray();
printArray(movies);

Console.WriteLine();
Console.WriteLine("There are " + movieCollection.Number + " movies in the collection.");
Console.WriteLine("There are " + movieCollection.NoDVDs() + " total DVDs in the collection.");

Console.WriteLine("------------------------------------------------------------");
```

Result:

```
Test 9: Use Delete method to delete movies in the BST, then use NoDVDs method to check if the correct
movies are deleted. Movie and DVD count should reduce by however many movies are deleted.

True
True

Title: Aftersun, Genre: Drama, Classification: M, Duration: 101, Available Copies: 7
Title: Fallen Angels, Genre: Action, Classification: M, Duration: 98, Available Copies: 13
Title: Spider-Man: Into the Spider-Verse, Genre: Action, Classification: PG, Duration: 117, Available Copies: 3
Title: The Good, the Bad and the Ugly, Genre: Western, Classification: M15Plus, Duration: 161, Available Copies: 18

There are 4 movies in the collection.
There are 41 total DVDs in the collection.
------------------------------------------------------------
```

Delete method works as intended. The method returns true when successfully deleting the movie objects passed into it. The correct movies are deleted as shown by the ToArray and printArray methods. Number is correctly decremented by the two movies deleted. NoDVDs also reflects the DVDs deleted.

Test 10: Deleting movies that do not exist in the collection.

```
// Test 10: Delete movies that do not exist. Should return false. Count should not change.
Console.WriteLine("Test 10: Delete movies that do not exist. Count should not change.\n");
Console.WriteLine(movieCollection.Delete(new Movie("Pride and Prejudice", MovieGenre.Drama, MovieClassification.G, 127, 5)));
Console.WriteLine();

movies = movieCollection.ToArray();
printArray(movies);
Console.WriteLine("There are " + movieCollection.Number + " movies in the collection.");
Console.WriteLine("There are " + movieCollection.NoDVDs() + " total DVDs in the collection.");

Console.WriteLine("-------------------------------------------------------");
```

Result:

```
Test 10: Delete movies that do not exist. Count should not change.

False

Title: Aftersun, Genre: Drama, Classification: M, Duration: 101, Available Copies: 7
Title: Fallen Angels, Genre: Action, Classification: M, Duration: 98, Available Copies: 13
Title: Spider-Man: Into the Spider-Verse, Genre: Action, Classification: PG, Duration: 117, Available Copies: 3
Title: The Good, the Bad and the Ugly, Genre: Western, Classification: M15Plus, Duration: 161, Available Copies: 18
There are 4 movies in the collection.
There are 41 total DVDs in the collection.
-------------------------------------------------------
```

Delete works as intended. No movies are deleted and the method returns false when deleting a movie that does not exist in the collection. The movie collection and number remain unchanged.

Test 11: ToArray method to convert collection to array.

```
// Test 11: Use ToArray method to convert movie collection to array. Use printArray helper method to print the array.
Console.WriteLine("Test 11: Use ToArray method to convert movie collection to array. Use printArray helper method to print the array.\n");
movies = movieCollection.ToArray();
printArray(movies);

Console.WriteLine("There are " + movieCollection.Number + " movies in the collection.");

Console.WriteLine("-------------------------------------------------------");
```

Result:

```
Test 11: Use ToArray method to convert movie collection to array. Use printArray helper method to print the array.

Title: Aftersun, Genre: Drama, Classification: M, Duration: 101, Available Copies: 7
Title: Fallen Angels, Genre: Action, Classification: M, Duration: 98, Available Copies: 13
Title: Spider-Man: Into the Spider-Verse, Genre: Action, Classification: PG, Duration: 117, Available Copies: 3
Title: The Good, the Bad and the Ugly, Genre: Western, Classification: M15Plus, Duration: 161, Available Copies: 18
There are 4 movies in the collection.
-------------------------------------------------------
```

This confirms that the ToArray method works as intended. It correctly returns the movies array in descending dictionary order. Number remains unchanged.

Test 12: Clear method.

```
// Test 12: Use Clear method to clear the binary search tree.
Console.WriteLine("Test 12: Use Clear method to clear the binary search tree.\n");
movieCollection.Clear();

movies = movieCollection.ToArray();
printArray(movies);

Console.WriteLine("\nThere are " + movieCollection.Number + " movies in the collection.");

Console.WriteLine();

Console.WriteLine("------------------------------------------------------------");
```

Result:

```
Test 12: Use Clear method to clear the binary search tree.


There are 0 movies in the collection.

------------------------------------------------------------
```

The clear method works as intended. It clears all the movies in the collection, as evident by using ToArray and printArray printing nothing. Number of the collection is zero.

Test 13: NoDVDs when movie collection is empty.

```
// Test 13: Use NoDVDs when the movie collection is empty.
Console.WriteLine("Test 13: Use NoDVDs when the movie collection is empty.\n");

Console.WriteLine("There are " + movieCollection.NoDVDs() + " total DVDs in the collection.");

Console.WriteLine("------------------------------------------------------------");
```

Result:

```
Test 13: Use NoDVDs when the movie collection is empty.

There are 0 total DVDs in the collection.
------------------------------------------------------------
```

NoDVDs correctly returns zero DVDs in the collection when the collection is empty.

Test 14: Testing the CompareTo method.

```
// Test 14: Testing the CompareTo method.
Console.WriteLine("Test 14: Testing the CompareTo method.\n");
IMovie test = new Movie("Test Movie", MovieGenre.Comedy, MovieClassification.PG, 13, 29);

Console.WriteLine("Test Movie".CompareTo(test.Title));
Console.WriteLine("AAA".CompareTo(test.Title));
Console.WriteLine("ZZZ".CompareTo(test.Title));

Console.WriteLine("------------------------------------------------------");
```

Result:

```
Test 14: Testing the CompareTo method.

0
-1
1
------------------------------------------------------
```

CompareTo behaves as intended. It correctly returns 0 for the same title, ‘AAA’ correctly returns -1 as it is lesser in dictionary order to ‘Test Movie’, and ‘ZZZ’ correctly returns 1 as it is greater in dictionary order to ‘Test Movie’.

Test 15: Testing the ToString method.

```
// Test 15: Testing the ToString method.
Console.WriteLine("Test 15: Testing the ToString method.\n");

Console.WriteLine(test.ToString());

Console.ReadLine();
```

Result:

```
Test 15: Testing the ToString method.

Title: Test Movie, Genre: Comedy, Classification: PG, Duration: 13, Available Copies: 29
```

ToString works as intended. It correctly returns the test movie title, genre, classification, duration, and available copies in that order.

4. References

- Tang, Maolin (2023, March 6). *CAB301 Algorithms and Complexity: Analysis of Algorithms* [PowerPoint slides]. Canvas. https://canvas.qut.edu.au/
- Tang, Maolin (2023, March 27). *CAB301 Algorithms and Complexity: Binary Tree, binary search tree, and algorithms* [PowerPoint slides]. Canvas. https://canvas.qut.edu.au/