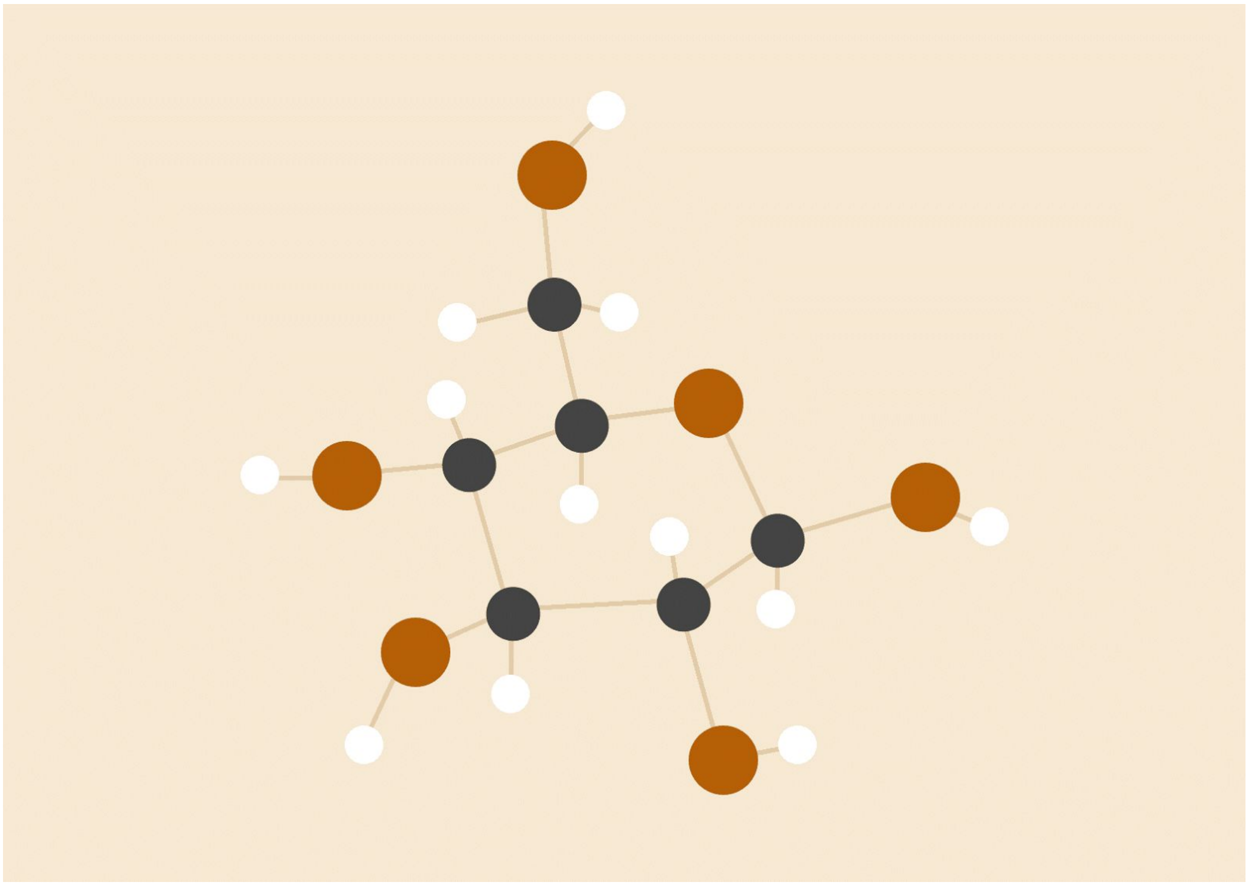
Computer Science

Report for Assignment 1



Brandon Gower-Winter

09.04.2017

CSC2001F

Problem

Assignment 1 tasks one to test the efficiency between two different data structures(Array and a Binary Search Tree(BST)) and determine which one of the two aforementioned data structures is more efficient at searching for and retrieving data.

One is also tasked with understanding a BST and the various algorithms associated with it; Namely insert and search algorithms. One must also be knowledgeable on the inorder traversal algorithm of a BST in order for one to complete one of the tasks.

An array is a data structure that consists of a collection of values or variables which are each identifiable by an array index. It is simple linear data structure that stores each value on after the other. A value in an array does not inherently have any references to the other values in it.

An ArrayList data structure is used in this project and it behaves similarly to an array the only difference being that number of items in an array is fixed while it is not fixed in an ArrayList

A BST is a data structure that conceptualises the idea of a Binary Search Algorithm into a Binary Tree data structure (A Tree data structure in which each node can have at most two child nodes). A BST is unique in the way it stores data. Values that are smaller go to left of the tree and values that are greater go to the right. This happens recursively until a null node is found and the value that was to be added gets stored there. This allows for a BST to be more efficient when searching for data with an average case of log(n) for searches.

Application Design

There were three tasks to be done in the given assignment. 1, print all the data from a BST in order. 2. Search for values within a BST using a query file 3. Search for values within an array using a query file.

In order to make the application as modular and simple as possible there were specific rules I had set in place for the application.

1. Each Task (PrintIt,SearchIt,SearchItLinear) will only contain a main function and will be used as application files that will make use of the other class files I created.
2. The data would be stored in a person object as to make the code neater and allow for ease of use.(I could create specific functions in the person class to handle how one object is compared to another and I could create a toString() function to allow all the data within the object to be displayed in one function call).
3. The generic BST that I created must ensure that the data type it is taking in implements the Comparable interface. This is because the compareTo() function is used to determine where to store new values and find old values.

The person class acts as a data model and is used purely to store the data given from the data file. It implements the the comparable interface which allows for custom handling on how to objects are compared with one another. Two person objects are compared by name as to allow for a inorder traversal algorithm to be able to return all the data within a BST in alphabetical order by name.

The Binary Search Tree class is generic and ensures that any data type that it will handle implements the Comparable interface (As mentioned above). All functions within the tree are recursive for code tidiness. The BST can insert, search, count the number of nodes and print or get the data within the tree in order. The tree doesn't use the concept of left and right node it uses less node and greater node as to get rid of the concept that the tree is orientation based. Less node is used to store a value that is less than and greater node is used to store a value that is greater than.

The PrintIt class is the application class for the first task. It consist of one main method. When the main method is run a scanner is created to read in the data file line by line. A second scanner is then created to read each line separating the data in order to store in a person object. The person object is then added to the BinarySearchTree that has also been created. The BST will then put the object in the correct place. Recursively going through the BST comparing the values going to the less or greater node depending until a null node is found. Once all the data entries have been added. A call is made for the BST to print out the values of the data entries to the output stream in order using an inorder traversal algorithm. The data is printed and the application is closed.

SearchIt is the application class for the second task. It consists of one main method. When the main method is run a scanner is created to read in the data file line by line. A second scanner is then created to read each line separating the data in order to store in a person object. The person object is then added to the BST. Once all the person objects are added a third scanner is used to read in the queries. Each query is looked for in the BST and when the the result is found it prints that entries details. If a result is not found then it prints out that person object that relates to that query was not found. The BST searches for the query value by starting at the root node and either going to the less node or greater node depending on the comparison of the values until the correct person object is found or the BST finds a null value.

SearchItlInear is the application class for the third task. It consists of one main method. When the main method is run a scanner is created to read in the data file line by line. A second scanner is then created to read each line separating the data in order to store in a person object. The person object is then added to the ArrayList. Once all the person objects are added a third scanner is used to read in the queries. Each query is looked for in the ArrayList iteratively and when the the result is found it prints that entries details. If a result is not found then it prints out that person object that relates to that query was not found.

PrintIt Output

```
Abbott Alec    489-848-7299    03707 Botsford Fork, Lima

Abbott Alexandria    318.679.5603 x712    44812 Wilderman Mountain, Vallejo

Abbott Alia    507.340.1186    76400 Barton Fields #044, Cerritos

Abbott Brando    602.992.4016    02519 Zackery Village, San Mateo

Abbott Elwyn    788.603.8604    88126 Bruen Common, Beverly Hills

Abbott Hosea    1-035-079-0176 x61480    51832 Bayer Pass, Simi Valley

Abbott Ima    823.283.2198 x7192    87191 Suite Z, Selma

Abbott Josh    822.752.1004    27010 Sanford Center, Stanton

Abbott Leann    516-835-0116    17296 Elta Crossroad #362, Newport Beach

Abbott Meda    1-117-789-3061    18565 Suite B, Fountain Valley

Abbott Murray    1-654-279-2374    22345 Runte Garden, Steubenville
Abbott Novella    297-763-2822    32763 Langosh Route, San Diego

Abbott Rahsaan    (681)856-6604 x642    90282 Haag Keys, Garden Grove

Abbott Sadye    (961)238-9093    52000 Marques Loaf #288, Placentia

Abbott Santina    1-515-459-1556    78469 Renner Mill, Agoura Hills

Abernathy Amparo    1-052-394-1236 x29668    96179 Feil Tunnel #352, Canton

Abernathy Austyn    1-486-893-0367    98827 Gerlach Pike Apt. 743, Apple Valley

Abernathy Catalina    1-331-934-0147    14576 Harber Knolls, Riverside

Abernathy Chadd    (552)753-8320 x85031    23694 Pier F, Tempe

Abernathy Cicero    (637)882-6835 x72457    36296 Batz Walk, San Francisco
```

SearchIt and SearchItLinear Output

```
Mayert Cathy    791-772-8120 x42168    90125 Raven Circle #864, Downey

Gower-Winter Brandon was not found.

Hickle Leone    018-594-2935 x716    17386 Stephanie Parks, Palm Springs

West Ramon    1-702-852-5634    50773 Schinner Extensions, Zanesville

Mikazuki Augus was not found.

Setsuna F. Seiei was not found.

Langosh Matt    (682)669-6865 x500    73754 Leffler Squares, Atwater

Ondricka Luz    (522)447-5098 x1929    68014 Jermain Street, Springfield

Labadie Leanna    (896)176-7008    37773 Durgan Parkways Suite 558, San Dimas

Lockon Stratos was not found.

Jacobs Wallace    (174)976-6745 x0539    06395 Cormier Crest Suite 404, West Memphis

Orga Itsuka was not found.

Medhurst Sydnie was not found.
```

```
Haag Abraham    (933)453-8588 x7220    04266 Missouri Junction, Burlingame
Ryan Alicia    950-938-7050 x27619    76980 Side, Auburn
Alleluyah Haptism was not found.
Daugherty Elijah    (549)540-0126 x715    26255 Marvin Way #268, Decatur
Abernathy Houston    757.248.9579 x9418    24902 O'Conner Creek, Homer
Lehner Alberta    (460)301-1274 x351    67100 Schumm Pines, Barrow
Fahey Lane    1-220-139-2838 x649    04923 Flatley Island Suite 476, San Clemente
```

Experimental Design

The testing of the efficiency of the two data structures was done by measuring the time it took for SearchIt and SearchItlinear to search for 20 names 20 times.

Three Sets of data were taken. Two extremes and one 'real' world. The first extreme is searching for the first 20 entries within the data file the second extreme is searching for the last 20 entries within the data file. The 'real' world scenario is searching for a random assortment of names (Some contained in the data file and others not).

Testing those scenarios will give me the ability to fully assess the efficiency of the two data structures.

Time recording was done using a bash script and the results have been happened to text files to allow for more data to be taken if need be.

Results

	SearchIt(SysTime UsrTime)	SearchItlinear(SysTime UsrTime)

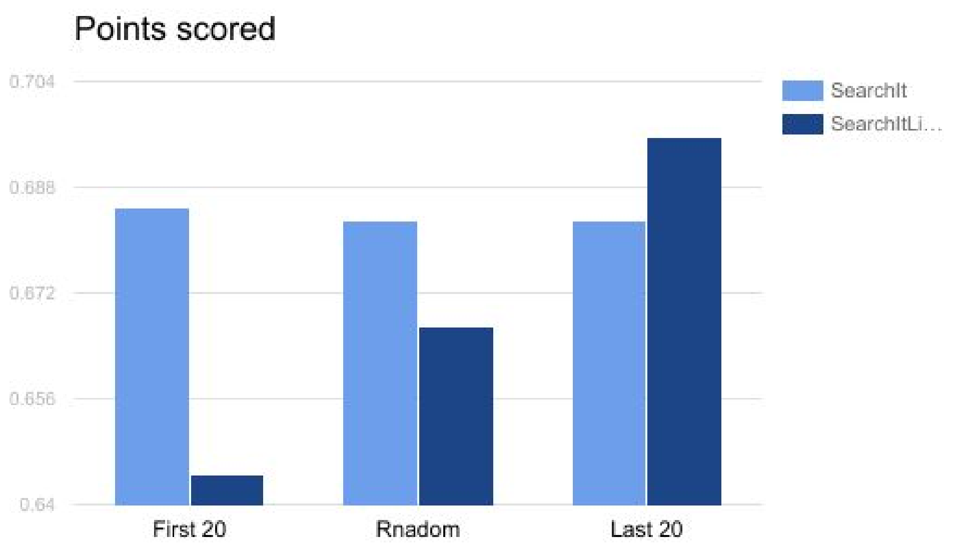
First 20	0.68 0.01
	0.64 0.00
	0.66 0.02
	0.68 0.01
	0.69 0.01
	0.67 0.01
	0.65 0.02
	0.68 0.02
	0.70 0.01
	0.66 0.02
	0.67 0.01
	0.70 0.01
	0.64 0.02
	0.69 0.02
	0.68 0.01
	0.67 0.01
	0.68 0.00
	0.65 0.02
	0.66 0.03
	0.67 0.02
		0.66 0.01
		0.62 0.02
		0.61 0.03
		0.62 0.02
		0.62 0.00
		0.63 0.01
		0.62 0.01
		0.63 0.01
		0.64 0.04
		0.63 0.00
		0.65 0.02
		0.61 0.02
		0.64 0.01
		0.62 0.02
		0.66 0.02
		0.63 0.02
		0.61 0.01
		0.59 0.02
		0.64 0.01
		0.64 0.02

Random	0.68 0.02
	0.70 0.02
	0.68 0.00
	0.65 0.03
	0.64 0.02
	0.67 0.02
	0.65 0.02
	0.67 0.01
	0.68 0.00
	0.71 0.02
	0.70 0.00
	0.67 0.02
	0.64 0.02
	0.66 0.00
	0.65 0.03
	0.67 0.03
	0.70 0.00
	0.64 0.02
	0.65 0.02
	0.62 0.03
		0.66 0.01
		0.67 0.01
		0.64 0.02
		0.62 0.03
		0.67 0.02
		0.65 0.02
		0.64 0.02
		0.67 0.02
		0.63 0.01
		0.64 0.02
		0.64 0.02
		0.62 0.02
		0.69 0.01
		0.66 0.02
		0.64 0.02
		0.63 0.02
		0.64 0.02
		0.66 0.00
		0.66 0.01
		0.65 0.04

Last 20	0.69 0.01 0.68 0.02	0.70 0.00 0.64 0.00

	0.64	0.02	0.66	0.03
	0.67	0.00	0.65	0.03
	0.68	0.00	0.69	0.02
	0.71	0.01	0.66	0.03
	0.66	0.02	0.62	0.01
	0.71	0.01	0.70	0.00
	0.65	0.02	0.64	0.01
	0.69	0.00	0.64	0.03
	0.66	0.01	0.67	0.03
	0.67	0.02	0.66	0.03
	0.64	0.03	0.70	0.02
	0.69	0.02	0.75	0.02
	0.64	0.02	0.76	0.04
	0.70	0.02	0.78	0.01
	0.67	0.01	0.72	0.02
	0.66	0.01	0.62	0.02
	0.63	0.01	0.65	0.01
	0.64	0.02	0.60	0.04

Graph:



Conclusion

After analysis of the data it can be concluded that the answer is a lot more complicated than saying one of the data structures is more efficient because both of them have positives and negatives.

With regards to arrays they are used for small amounts of data and this can be seen very clearly when looking at the results. They are more efficient by quite a margin when looking at data entries that are entered early into the structure. The array performed at a slightly higher efficiency when looking at random data. I believe this is because the data file didn't contain enough data entries to actually strain the performance by a considerable margin. The array performed poorly when looking at the last 20 entries and this is because of the need to cycle n times before finding a result this clearly show the pitfalls of using an array for large amounts of data.

The BST performed poorly when looking at the first entries. This is because a BST is not suitable for small amounts of data. When looking at the results for the random queries. It performed very slightly below the array but by no means a margin that would affect performance for a data file of just 10 000 entries. The true nature of the BST was shown in the results of the last 20 entries where the BST clearly showed it was superior at sifting through larger amount of entries.

In conclusion both data structures have their uses. Arrays for small amounts of data and a Binary Search Tree for large amounts of data.

GIT Log

```
    Added Unit Testing for Binary Search Tree

commit 2df7ecommit 462fa4888db190815f6f25ec660bc4dfe2974be2

Author: BrandonGower-Winter <brandongowerwinter@gmail.com>

Date:   Mon Apr 10 06:55:02 2017 +0200

    Fixed Error in testing

commit 3807f0b77c28242cfda76edf844194624b2479ac

Author: BrandonGower-Winter <brandongowerwinter@gmail.com>
```

```
Date:   Mon Apr 10 01:22:28 2017 +0200

        Revised Docs and cleaned up files

commit 660d2ad88b0faf5d8a1ec7f55e17afba347a83b0

Author: BrandonGower-Winter <brandongowerwinter@gmail.com>

Date:   Mon Apr 10 00:51:04 2017 +0200

        Tests for last 20 Entires completed

commit a8b4891868a0a7eac06d41e4f253b7c53e919868

Author: BrandonGower-Winter <brandongowerwinter@gmail.com>

Date:   Mon Apr 10 00:44:15 2017 +0200

        Test data on the first twenty entires collected

commit 771a7df5fe6453607d8c0683451ed44078b3bece

Author: BrandonGower-Winter <brandongowerwinter@gmail.com>

Date:   Mon Apr 10 00:24:47 2017 +0200

        Completed Overall Testing

commit 061aab5b7518f08491cd4a7be69cb9924efbfcc0

Author: BrandonGower-Winter <brandongowerwinter@gmail.com>

Date:   Mon Apr 10 00:14:03 2017 +0200

        Added Test Functionality

commit 6b9e2ef83557746250fef01b09249d0f8f32b40b

Author: BrandonGower-Winter <brandongowerwinter@gmail.com>

Date:   Sun Apr 9 21:07:52 2017 +0200

        Added Unit Testing for Person class

commit 3fa1eedcc2cb8989359ca7a2b53277a83b95abbd

Author: BrandonGower-Winter <brandongowerwinter@gmail.com>

Date:   Sun Apr 9 20:36:11 2017 +0200
```

dddbd07669047d104848ca543d5ed501363

```
Author: BrandonGower-Winter <brandongowerwinter@gmail.com>

Date:   Sun Apr 9 18:06:22 2017 +0200

        Javadocs for all classes are complete

commit e1af3e9f5c147fef474d2d8c865e37e55107ad42

Author: BrandonGower-Winter <brandongowerwinter@gmail.com>

Date:   Thu Apr 6 09:24:33 2017 +0200

        Added JavaDocs for BinarySearchTree and BinarySearchTree

commit b01ac05a365c66257256f8c75ff85311e880585b

Author: BrandonGower-Winter <brandongowerwinter@gmail.com>

Date:   Tue Apr 4 09:32:06 2017 +0200

        Javadoc for person setup and makefile now compiles all java classes

commit 3ee67b864164bed5dcb8c3d53b298a16cd1290f0

Author: BrandonGower-Winter <brandongowerwinter@gmail.com>

Date:   Fri Mar 31 10:12:11 2017 +0200

        Finished programming all classes for the assignment

commit 1242a04b23cf946fb61b1b77a127cd7f49257cfa

Author: BrandonGower-Winter <brandongowerwinter@gmail.com>

Date:   Mon Mar 27 08:22:42 2017 +0200

        Added assigment 1
```

JUnit Log

My JUnit is run through a test runner class that ensures that only errors appear in the terminal so the output for my tests is simply (See testRunner.java file):

```
true
```