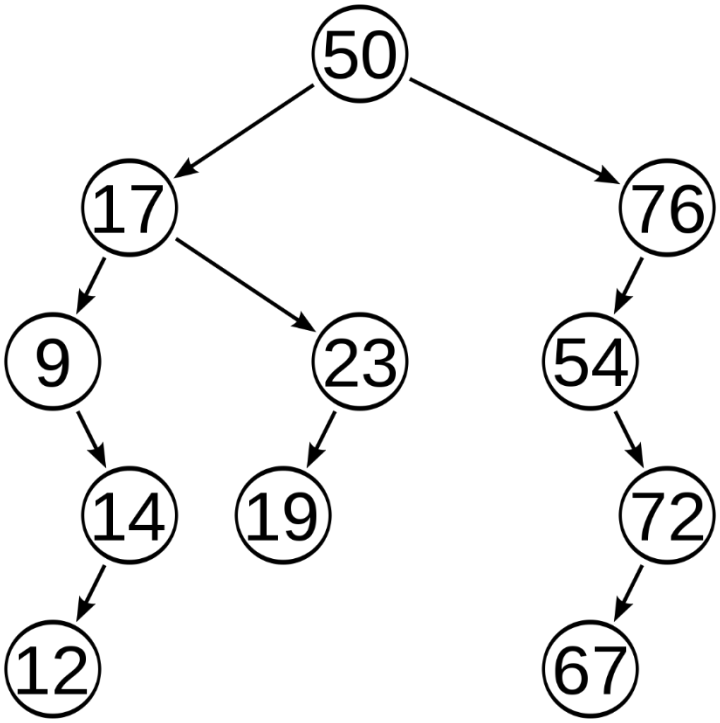
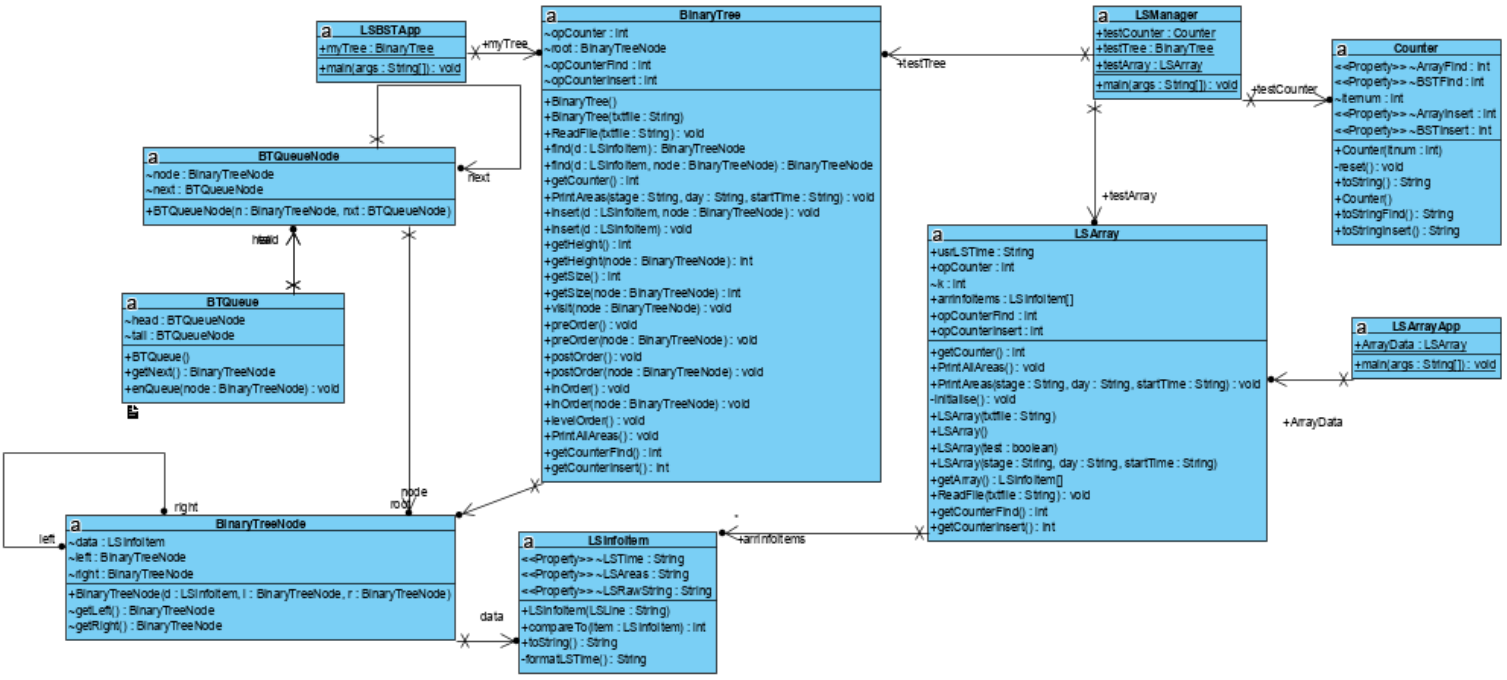
CSC2001F ASSIGNMENT 1: BINARY SEARCH TREES



Agoritsa Spirakis – SPRAGO001

The aim of this assignment is to compare the time complexity of Arrays and Binary Search Trees.

OOP design:



Key data storage classes:

LSInfoItem	Stores a data item consisting of a LSTime(stage, date, start time) as well as a LSArea variable which contains the relevant list of areas that the LSTime applies to.
LSArray	Stores an array of LSInfoItems and contains methods to traverse all the data (PrintAllAreas) and to search for a specific item if given a time.
Binary Tree	This class allows for the data to be structured in what is called a Binary Tree. It makes use of recursive functions and as like the LSArray class, it can search for an item given a specific time.The Binary Tree can traverse through all the data in a level order, post order, preorder or inorder algorithm. It makes use of BinaryTreeNodes, BinaryTreeQueues (and hence Binary Tree Queue nodes) to achieve this structure.

Implementation classes:

LSArrayApp LSBSTApp	Commands the construction of a LSArray object. Methods from the specific LSArray class are called here depending on what the user inputs. Java LSArrayApp Entering this will print out all the data items in the array Java LSArrayApp “stage” “day” “time” Entering this will enable the program to search through the LSArray object for a LSInfoItem object that matches the specified LSTime data entered by the user A similar implementation is repeated with LSBSTApp

Experiment facilitation:

LSManager	This class allows for both a binary tree and an array to be instantiated with the same data items (from the relevant textfile). Java LSManager “textfilename.txt” Calling this command will populate a Binary Tree and an Array with the data in the array. A call to print all the data files is made for both data structures. This data is then sent to a comma separated textfile containg the count data.
Counter	This is an auxiliary class that helps to keep track of the count values for each of the data structures. For each data structure the Counter class stores a Find counter which stores the number of operation done to search for an item and an Insert counter which is used to store the total number of operations for the data to be inserted into the data structure.

testGen	This class is used to create the 10 different textfiles of different lengths (n). It consists of first initialising the entire 2976 items into an array. A shuffle function is then used to randomize the position of the items in the array. After the shuffling takes place, a set number of values(n) are written to the relevant textfile and the process repeats.

Testing:

Bash script code was written so as to test the functioning of the LSArrayApp class and the LSBSTApp class. Bash allowed the OS to run many instances of the program for testing purposes and print it to a textfile

---

LSArrayApp Testing

ArrayTest 1: Prints all data

Input = java LSArrayApp

OUTPUT:

Stage: 1

Day: 1

Starting time: 00h00

Areas: 1

Stage: 1

Day: 17

Starting time: 00h00

Areas: 1

...

Stage: 8

Day: 31

Starting time: 22h00

Areas: 15,7,11,3,12,4,8,16

Stage: 8

Day: 16

Starting time: 22h00

Areas: 15,7,11,3,12,4,8,16

No of operations: 2976

ArrayTest 2: valid search input

Input = java LSArrayApp '1' '12' '00'

OUTPUT:

Stage: 1

Day: 12

Starting time: 00h00

Areas: 7

Data point has been found

No of operations: 23

ArrayTest 3: invalid search input

Input = java LSArrayApp '8' '44' '00'

OUTPUT:

No item found

No of operations: 2976

ArrayTest 4: input has too many values

Input = java LSArrayApp '8' '44' '00' '22' '33'

OUTPUT:

Error: incorrect format

ArrayTest 5: input has too many values

Input = java LSArrayApp 'A' 'B' 'C'

OUTPUT:

Error: input must be integer numbers

LSBSTApp Testing

The same tests as done for the ArrayApp were done for the BSTApp. Copies of these theses can be found in textiles BSTTest1-6.

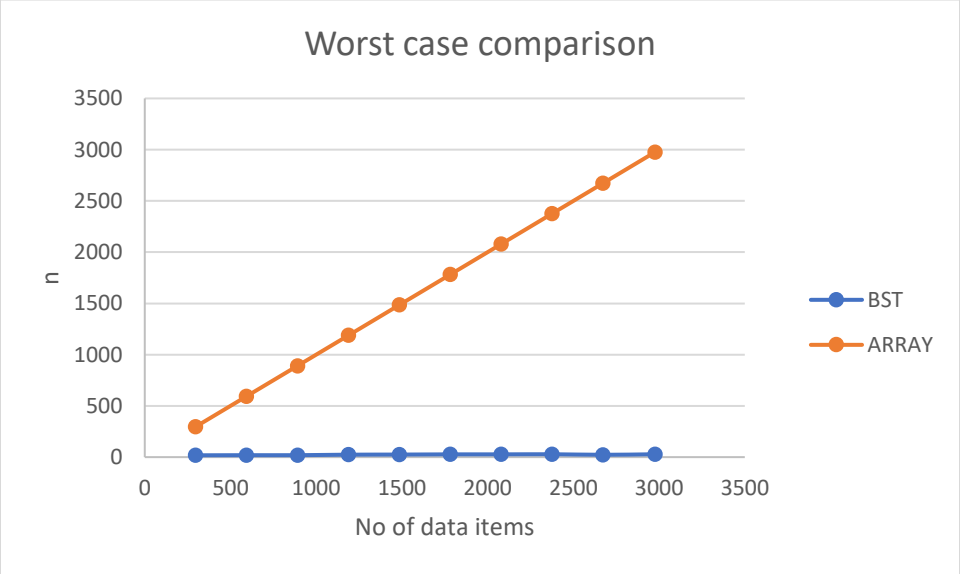
Comparison Results

Methodology:

- A java program was used to first randomize the 2796 data entries and create subsets of varying length equally spaced by 297 values.
- Once these individual textfiles were made, there was some editing done as to how the programs reach the data and allow for a specification of the textfile in the argument when running LSManager.
- Within each subset textfile, a loop was implemented to document the time taken to search for every data item in the subset with the Array structure and the BST structure.
- Each subset was written in a separate textfile separated by commas which would allow the sets to be manipulated in excel.
- A Bash program was made to run all the textfile subsets consecutively.

Table showing the processed results:

n	BEST		WORST		AVG	
	BST	ARAY	BST	ARRAY	BST	ARRAY
297	1	1	18	297	9.64646	149
594	1	1	19	594	10.5859	297.5
891	1	1	19	891	10.8474	446
1188	1	1	24	1188	11.9891	594.5
1485	1	1	25	1485	13.3697	743
1782	1	1	27	1782	13.2896	891.5
2079	1	1	27	2079	13.2823	1040
2376	1	1	28	2376	13.7635	1188.5
2673	1	1	23	2673	13.8979	1337
2976	1	1	28	2976	14.6529	1488.5



Conclusions:

When comparing time complexities it is most important to focus on how the structure operates on worst case scenarios. As seen from the data collected in Table 1 and the graph constructed the A, the array structure grows linearly as n increases whereas the Binary Search Tree's trajectory is logarithmic in nature. Using this data it can be concluded that using a Binary Search Tree to perform searches through data is significantly better than using an array data structure.

Git usage:

Commits on Feb 29, 2020

Reads textfile data working
Agi23 committed 5 days ago — 0a95138

Commits on Feb 26, 2020

made LSInfoItem class and added method headers for LSArrayApp
Agi23 committed 8 days ago — 483f2dc

Commits on Feb 25, 2020

Added makefile to the document
Agi23 committed 9 days ago — 345e0f0

Commits on Mar 5, 2020

Final retouches and comments
Agi23 committed 28 minutes ago — e75bb07

Commits on Mar 4, 2020

Output textfiles
Agi23 committed 18 hours ago — ec83445

Bash code generates textfiles
Agi23 committed 19 hours ago — 8a454a8

https://superuser.com/questions/429693/git-list-all-files-currently-u... ...
Agi23 committed 19 hours ago — dcc5058

Experiment textfile generation new files added
Agi23 committed 19 hours ago — 3026f8c

Experiment textfile generation comparison working
Agi23 committed 19 hours ago — 73c43d4

Commits on Mar 3, 2020

Search method for BST
Agi23 committed yesterday — 6ffa4f0