Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

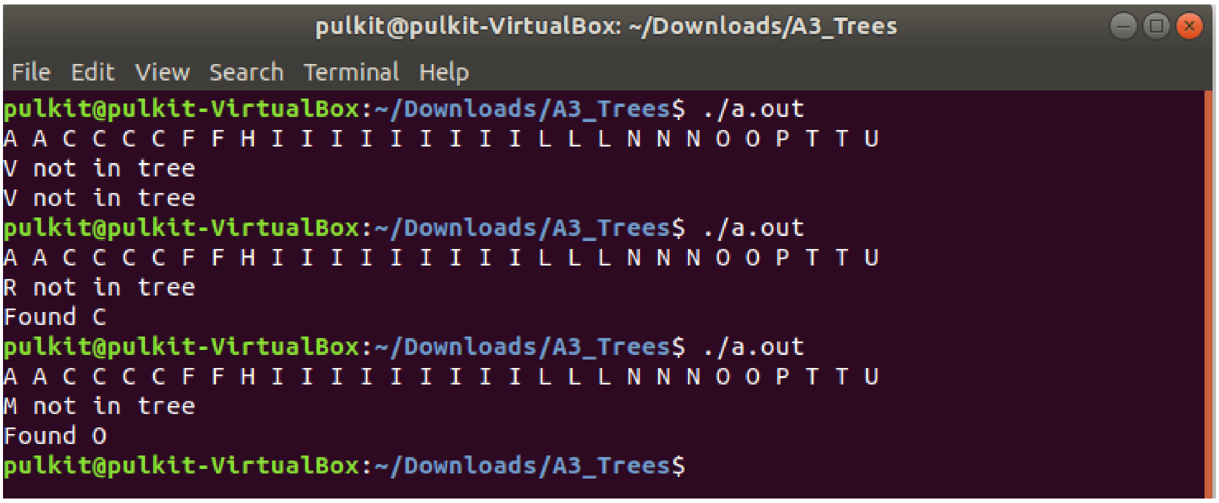
3D5A DATA STRUCUTRES AND ALGORITHM

Assignment 3 Report

Prepared By:
PULKIT SHARMA
19323659
Computer Engineering

Task 1

Implement a binary search tree using char as the data records. First write a suitable structure to represent a node. Create a node pointer in your main function to represent the root of a tree.



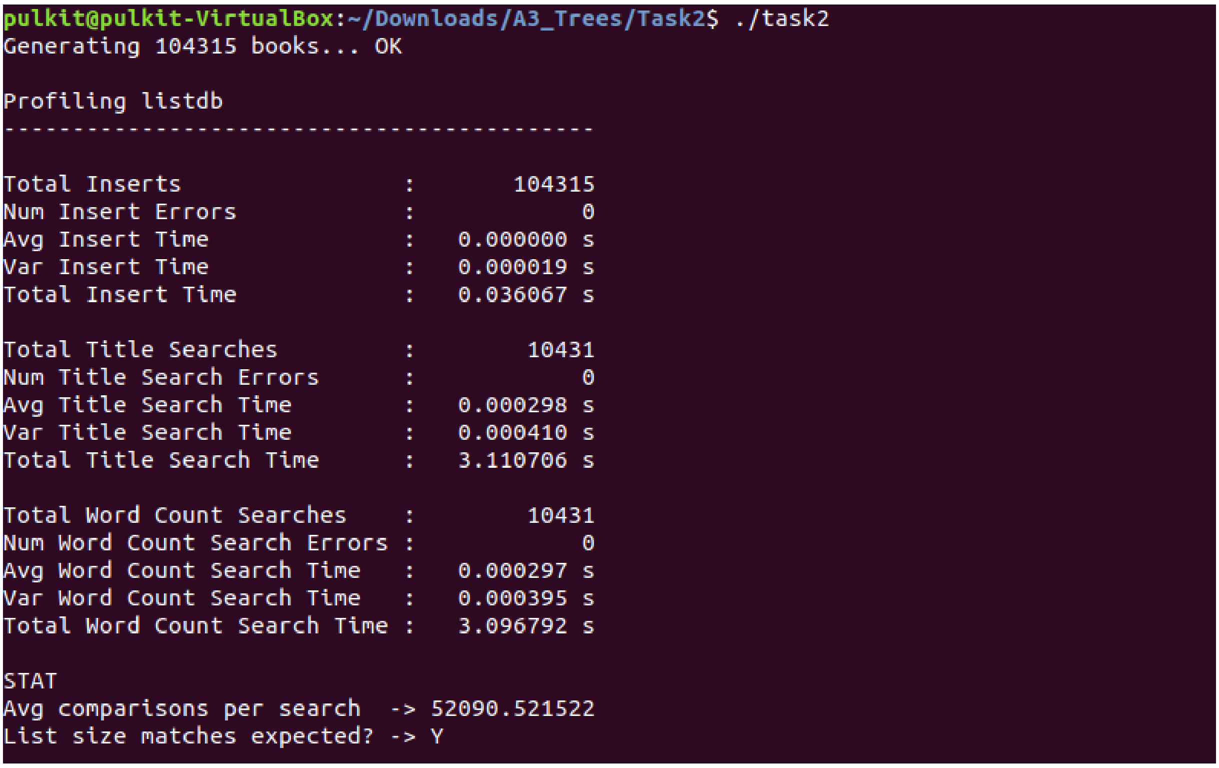
pulkit@pulkit-VirtualBox: ~/Downloads/A3_Trees
File Edit View Search Terminal Help

```
pulkit@pulkit-VirtualBox:~/Downloads/A3_Trees$ ./a.out
A A C C C C F F H I I I I I I I I I L L L N N N O O P T T U
V not in tree
V not in tree
pulkit@pulkit-VirtualBox:~/Downloads/A3_Trees$ ./a.out
A A C C C C F F H I I I I I I I I I L L L N N N O O P T T U
R not in tree
Found C
pulkit@pulkit-VirtualBox:~/Downloads/A3_Trees$ ./a.out
A A C C C C F F H I I I I I I I I I L L L N N N O O P T T U
M not in tree
Found O
pulkit@pulkit-VirtualBox:~/Downloads/A3_Trees$
```

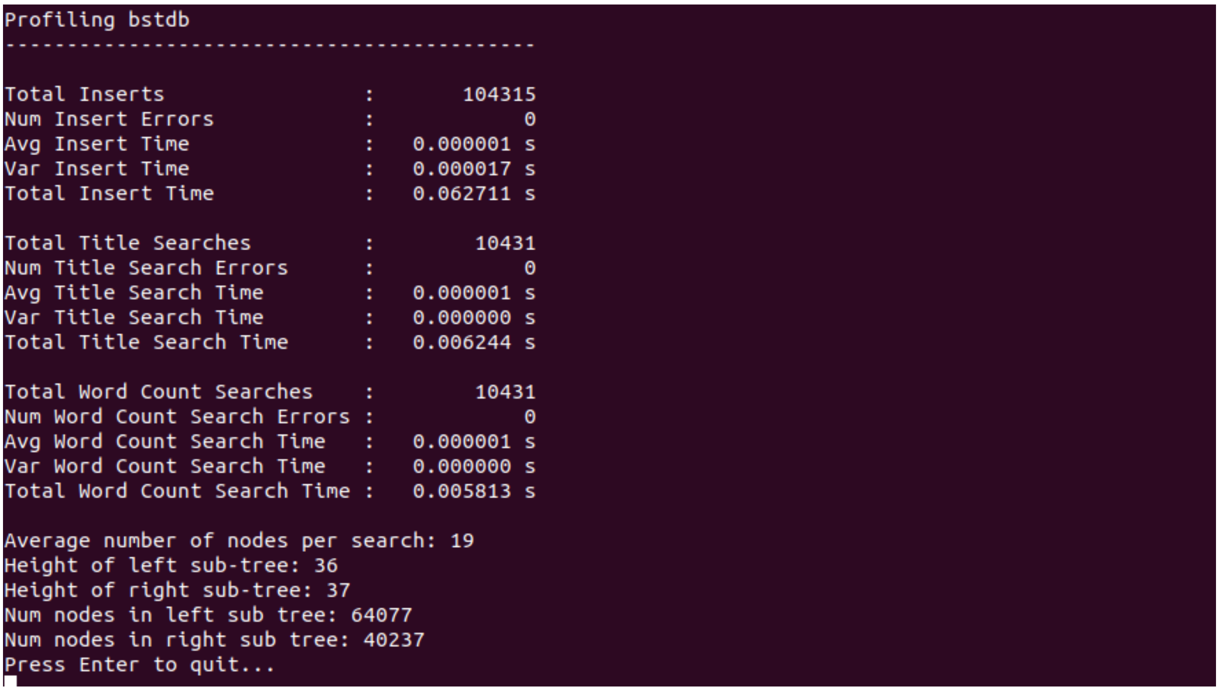
- The major advantage of binary search trees over other data structures is that the related sorting algorithms and search algorithms such as in-order traversal can be very efficient.
- We start by creating functions for insert, delete, print etc. We then initialize the array as given in the task and send the array to the sorting function.
- We print the sorted array and search for a random character inside the array using rand function.
- The rand function gives us a random character which is then searched in the sorted array which checks if the value if found inside the array or not.
- We can see in the output that every time we run the program, we get different character.

Task 2

Complete the functions in the BSTDB to store the database inside a BST.



```
pulkit@pulkit-VirtualBox:~/Downloads/A3_Trees/Task2$ ./task2
Generating 104315 books... OK

Profiling listdb
-------------------------------------------

Total Inserts                :        104315
Num Insert Errors            :             0
Avg Insert Time              :    0.000000 s
Var Insert Time              :    0.000019 s
Total Insert Time            :    0.036067 s

Total Title Searches         :         10431
Num Title Search Errors      :             0
Avg Title Search Time        :    0.000298 s
Var Title Search Time        :    0.000410 s
Total Title Search Time      :    3.110706 s

Total Word Count Searches    :         10431
Num Word Count Search Errors :             0
Avg Word Count Search Time   :    0.000297 s
Var Word Count Search Time   :    0.000395 s
Total Word Count Search Time :    3.096792 s

STAT
Avg comparisons per search  -> 52090.521522
List size matches expected? -> Y
```



```
Profiling bstdb
-------------------------------------------

Total Inserts                :        104315
Num Insert Errors            :             0
Avg Insert Time              :    0.000001 s
Var Insert Time              :    0.000017 s
Total Insert Time            :    0.062711 s

Total Title Searches         :         10431
Num Title Search Errors      :             0
Avg Title Search Time        :    0.000001 s
Var Title Search Time        :    0.000000 s
Total Title Search Time      :    0.006244 s

Total Word Count Searches    :         10431
Num Word Count Search Errors :             0
Avg Word Count Search Time   :    0.000001 s
Var Word Count Search Time   :    0.000000 s
Total Word Count Search Time :    0.005813 s

Average number of nodes per search: 19
Height of left sub-tree: 36
Height of right sub-tree: 37
Num nodes in left sub tree: 64077
Num nodes in right sub tree: 40237
Press Enter to quit...
```

- The database was generating random values of inserts and random values of search and word count.

- The height difference is 0 or 1 so that’s a balanced tree. If the tree is not balanced we will use AVL tree to balance it.
- We opted for a balanced BST in this part. This drastically reduced the search time for title and word count search as we can see in the output.
- The insert time might be more than in linked list because we are checking each value and finding the right position for it to insert in the BST.
- The search time is drastically reduced for the BST because the values are checked and then inserted appropriately inside the tree.
- This saves us a lot of time when we are doing higher level programming.