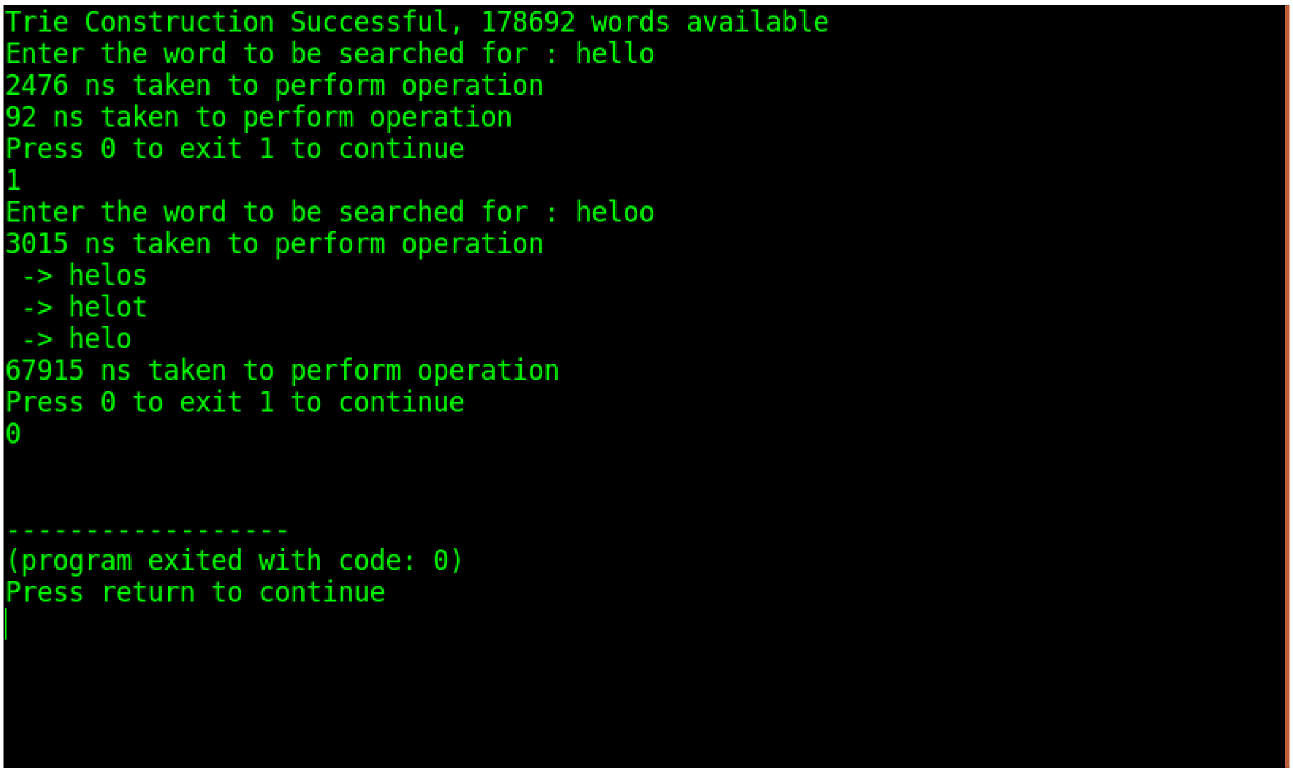
Data Structures and Algorithms (CSE2003)

Dictionary and Spell Checker: A Comparison Between Data Structures

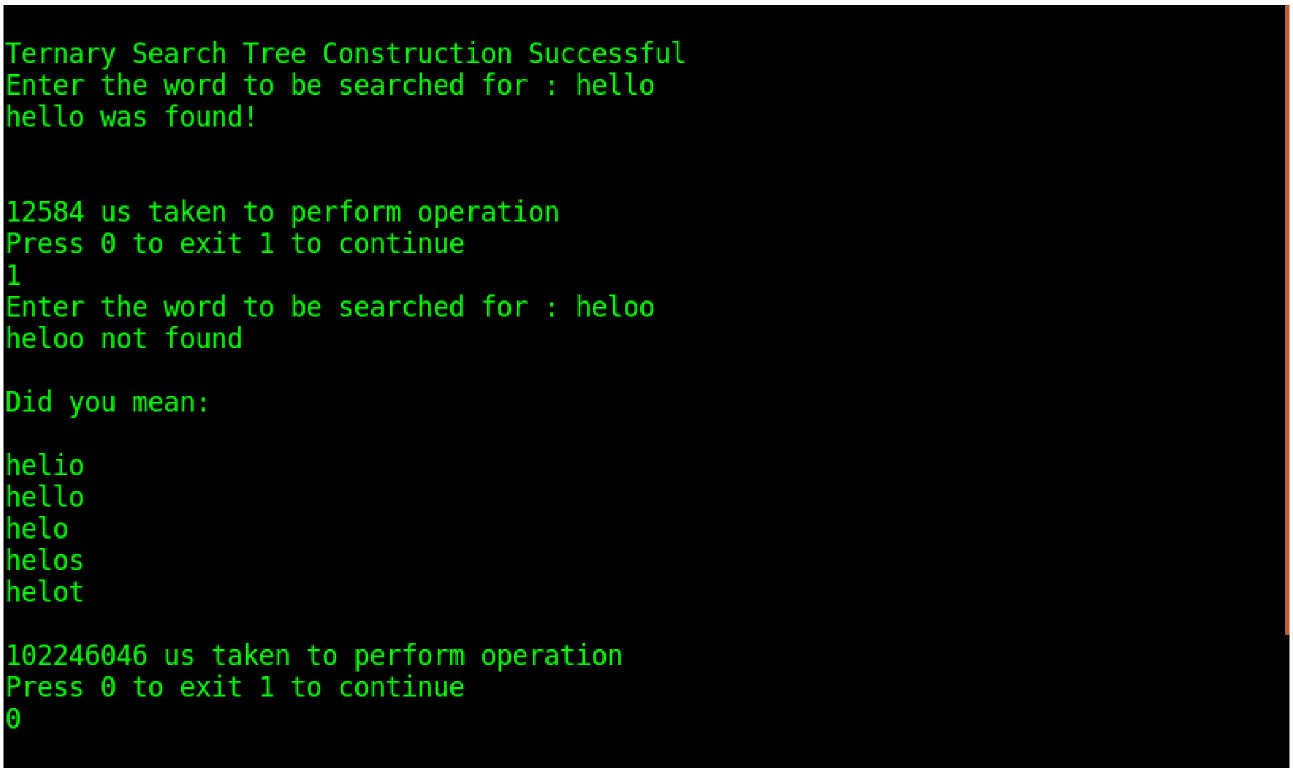
Authors:
Hriday Baghar(15BEC0467), Shashwat Singh(16BCI0180) and Insaf Muhammed Ali(16BCI0144).

Results and output:



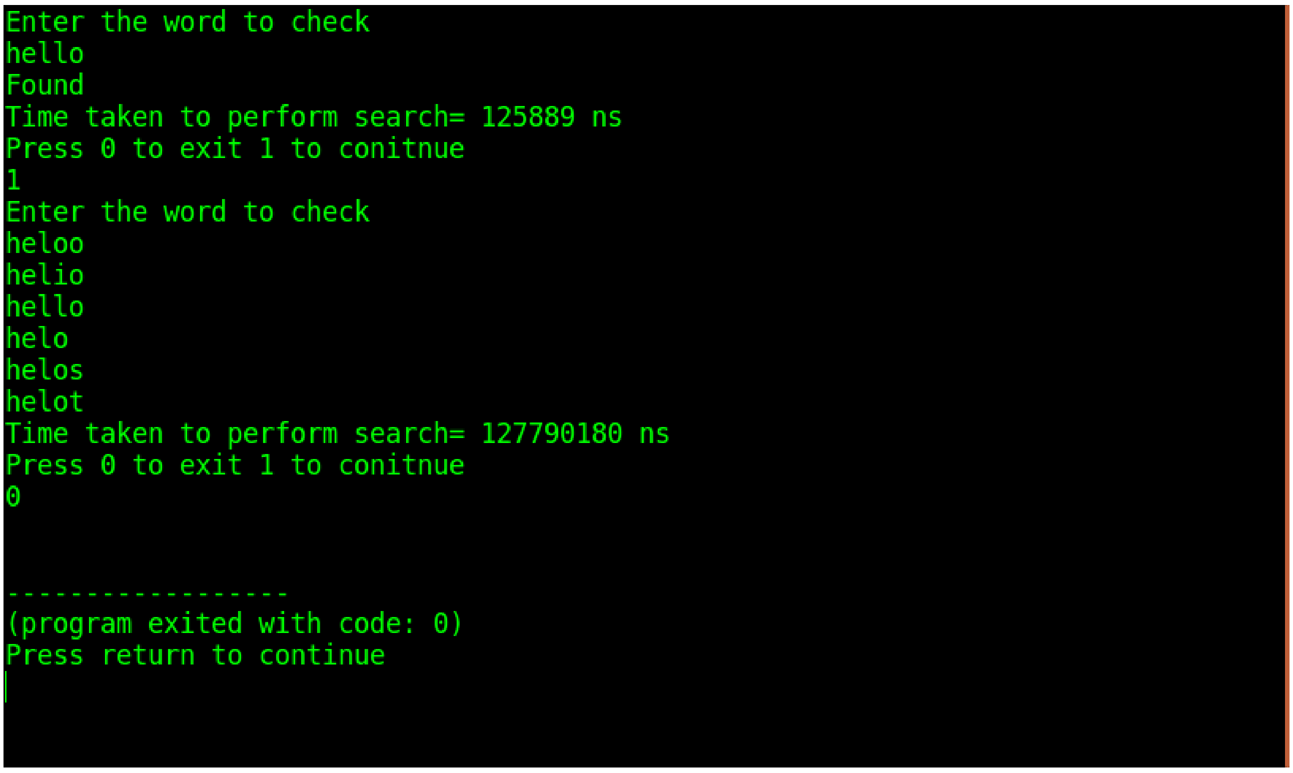
```
Trie Construction Successful, 178692 words available
Enter the word to be searched for : hello
2476 ns taken to perform operation
92 ns taken to perform operation
Press 0 to exit 1 to continue
1
Enter the word to be searched for : heloo
3015 ns taken to perform operation
 -> helos
 -> helot
 -> helo
67915 ns taken to perform operation
Press 0 to exit 1 to continue
0


------------------
(program exited with code: 0)
Press return to continue
```

Trie proves quick in searching as it uses prefix matching, hence the part of tree traversed becomes significantly smaller. However, the drawback is that spelling mistakes aren’t always dependent on prefix matching as we can see from the above example, one would expect “hello” but instead “helo” prefix was matched and words starting with said prefix were printed.



```
Ternary Search Tree Construction Successful
Enter the word to be searched for : hello
hello was found!


12584 us taken to perform operation
Press 0 to exit 1 to continue
1
Enter the word to be searched for : heloo
heloo not found

Did you mean:

helio
hello
helo
helos
helot

102246046 us taken to perform operation
Press 0 to exit 1 to continue
0
```

TST proves much more space efficient than Trie. In the above code we have performed full traversal of the tree and hence we see that considerably large execution time has occurred for word suggestion. Traversing full tree is not very economical compared to array searching.



```
Enter the word to check
hello
Found
Time taken to perform search= 125889 ns
Press 0 to exit 1 to conitnue
1
Enter the word to check
heloo
helio
hello
helo
helos
helot
Time taken to perform search= 127790180 ns
Press 0 to exit 1 to conitnue
0



------------------
(program exited with code: 0)
Press return to continue
```

A binary search algorithm was implemented on an array of the wordlist. As can be compared with TST, full traversal works just as good as an array search.

Conclusion:

Hence we can see that prefix matching is used where spelling corrections aren’t important but predicting strings is (web search etc). In such scenarios the output must be dynamic and hence must returned in shorter span of time. Tree based structures are optimal for the same.

As far as word corrections are concerned, arrays and trees do not show much variation as to provide accurate spelling correction, we must check the entered prefix for errors as well. Either approach can be used in word correction algorithms like those in word processors.