CO 322 Data Structures and Algorithms

Lab 04 - Tree ADT

E/16/057 – Chamith UKDK

The program was tested on the provided three dictionaries.
All non-alphabetic characters were removed when reading the word list and converted whole word to lowercase.

Compile and run:

TrieDic.c - code implementation for part 1

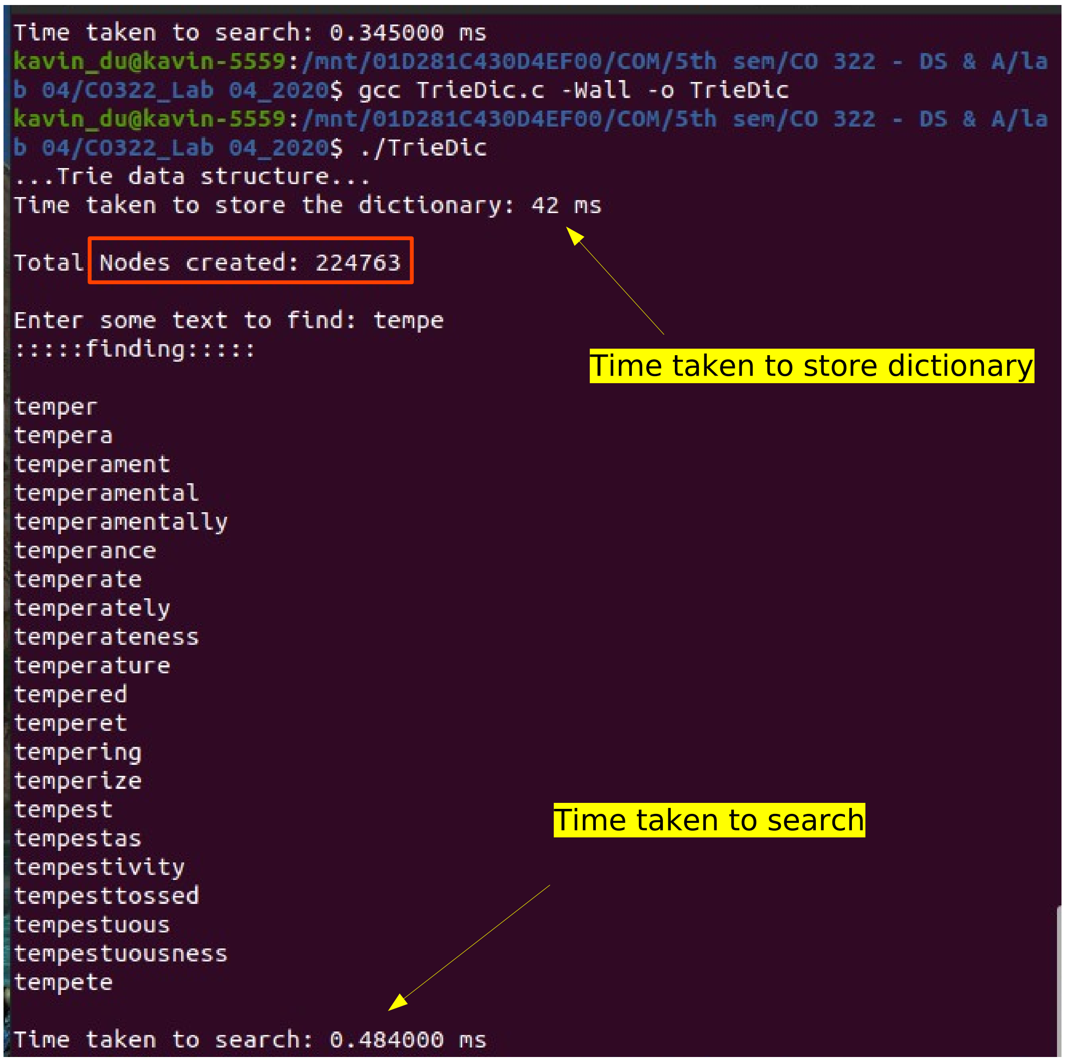
```
gcc TrieDic.c -Wall -o TrieDic
./TrieDic
```

RadixTreeDic.c – code implementation for part 2

```
gcc RadixTreeDic.c -Wall -o RadixTreeDic
./RadixTreeDic
```

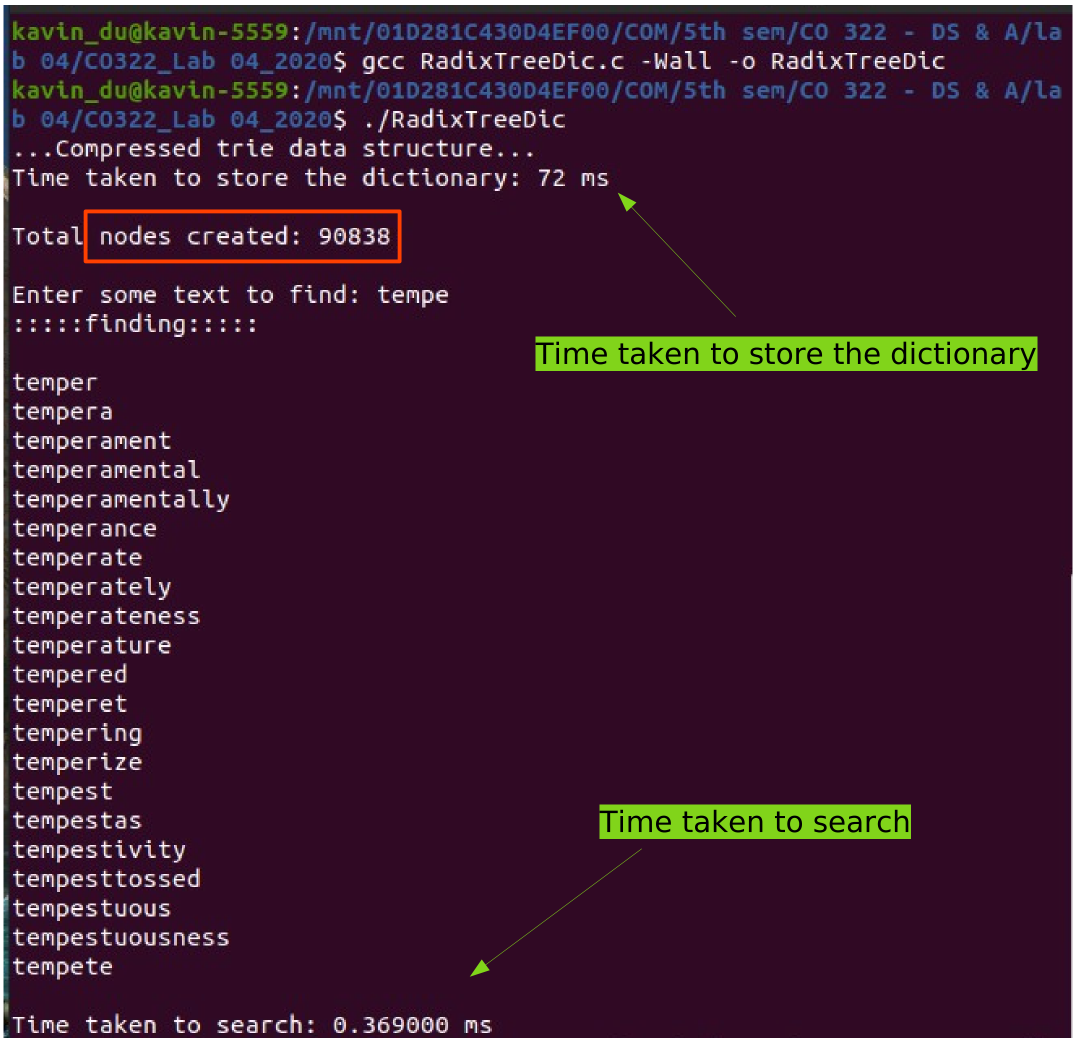
Both files can be compiled using default arguments for GCC. Both files were created and tested on a Linux machine. The user will be asked to type a word in the console for searching.

Trie Data structure:



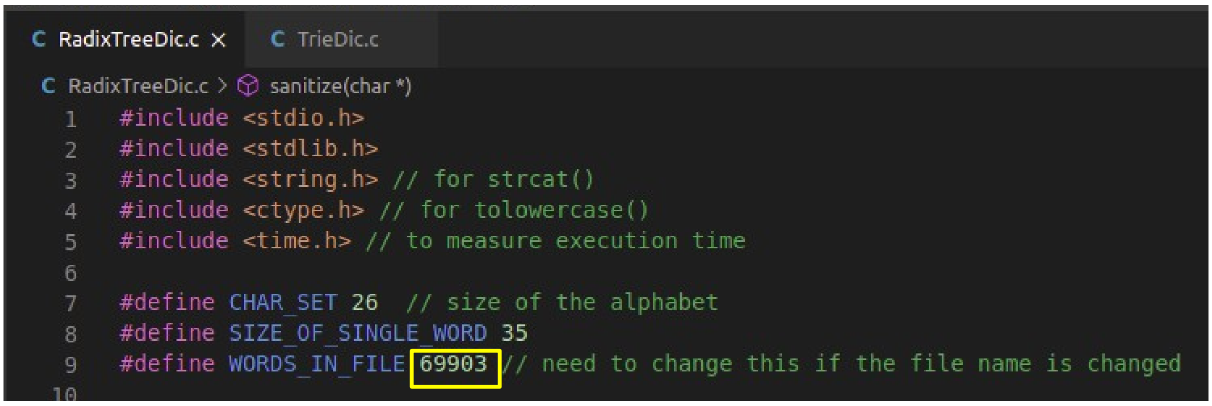
```
Time taken to search: 0.345000 ms
kavin_du@kavin-5559:/mnt/01D281C430D4EF00/COM/5th sem/CO 322 - DS & A/la
b 04/CO322_Lab 04_2020$ gcc TrieDic.c -Wall -o TrieDic
kavin_du@kavin-5559:/mnt/01D281C430D4EF00/COM/5th sem/CO 322 - DS & A/la
b 04/CO322_Lab 04_2020$ ./TrieDic
...Trie data structure...
Time taken to store the dictionary: 42 ms

Total Nodes created: 224763

Enter some text to find: tempe
:::::finding:::::

temper
tempera
temperament
temperamental
temperamentally
temperance
temperate
temperately
temperateness
temperature
tempered
temperet
tempering
temperize
tempest
tempestas
tempestivity
tempesttossed
tempestuous
tempestuousness
tempete

Time taken to search: 0.484000 ms
```

Radix tree(Compressed trie) data structure:



```
kavin_du@kavin-5559:/mnt/01D281C430D4EF00/COM/5th sem/CO 322 - DS & A/la
b 04/CO322_Lab 04_2020$ gcc RadixTreeDic.c -Wall -o RadixTreeDic
kavin_du@kavin-5559:/mnt/01D281C430D4EF00/COM/5th sem/CO 322 - DS & A/la
b 04/CO322_Lab 04_2020$ ./RadixTreeDic
...Compressed trie data structure...
Time taken to store the dictionary: 72 ms

Total nodes created: 90838

Enter some text to find: tempe
:::::finding:::::

temper
tempera
temperament
temperamental
temperamentally
temperance
temperate
temperately
temperateness
temperature
tempered
temperet
tempering
temperize
tempest
tempestas
tempestivity
tempesttossed
tempestuous
tempestuousness
tempete

Time taken to search: 0.369000 ms
```

The program was tested for all three text files that provided. The program is currently running for 70000 words file. Since program is reading a given file line by line, if you change the filename you have to specify how many lines in that file as below. (for both C files)



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h> // for strcat()
#include <ctype.h> // for tolowercase()
#include <time.h> // to measure execution time

#define CHAR_SET 26  // size of the alphabet
#define SIZE_OF_SINGLE_WORD 35
#define WORDS_IN_FILE 69903 // need to change this if the file name is changed
```

Number of lines for each text file:

wordlist70000.txt	→ 69903
wordlist10000.txt	→ 10000
wordlist1000.txt	→ 1000

Observations and conclusion:
(70000 word list was used for analysis)

Times taken to load the dictionary:-
Trie structure → 42 ms
Radix Tree structure → 72 ms

Times taken to show suggestions:-

	“tempe”	“has”	“a”
Trie structure	0.484 ms	0.230 ms	39.756 ms
Radix tree structure	0.369 ms	0.392 ms	29.220 ms

Since trie data structure keep nodes for each letter, it has to create new node for each letter. But the redundant nodes have been removed in the radix tree and it only keeps nodes for new strings. Since the nodes are less in radix tree it takes less time to search a word. But the branching factor is high for a given word, we can’t gain much time advantage when searching for that word in Radix tree.

Even the nodes are less, the implementation is complex in the radix structure. So it takes more time to load the dictionary in to a radix tree.

Space analysis:

Total nodes created in :-
Trie structure → 224763
Radix Tree structure → 90838

When the createNode() function is called, each time the number of node count was increased. As we can see, total number of nodes are much larger in the Trie structure than Radix tree structure. Therefore we gain the advantage of memory space when using radix tree.