



Data Structures Assignment 4

Hagar Osama

4970

Introduction

Binary search trees are an excellent data structure to implement associative arrays, maps, and similar interfaces. The main difficulty is that they are efficient only when they are balanced.

Straightforward sequences of insertions can lead to highly unbalanced trees. The solution is to dynamically rebalance the search tree during insert operations. There are many different algorithms for keeping trees in balance, such as AVL trees (I used), red/black trees.

To keep track of whether a binary search tree is a AVL tree, we associate with each node a ***balance factor***. The key to get a tree in balance is do a certain rotation after each insertion. For further details, see code description.

Code Description

The code is implemented in c language and has two main parts:

1. mainTree.c (the main file)
2. bal_tree.h (implementation of AVL tree)

Part I :

The file mainTree.c represents the main file of the program. In the main() function, the initialization of the tree and an infinite loop that loads only a

function called check_my_sentences that takes an input from user and checks validity of each word in.

The important functions provided in this file are :

```
int read_dictionary(char* filename);
void check_my_sentence(node *ptr);
void get_suc_and_pre(node *ptr , char *suc , char *pre , char *word);
void print_parent(char *suc , char *pre , char* token);
void toLowerCase(char*str);
int check_symbols(char* word);
```

read_dictionary: reads the database file 'English Dictionary.txt' and assigns each word to the global array of struct database and returns the number of words scanned. The struct can store up to 4000 words..

check_my_sentence: gets an input from the user, partitions it into tokens and check whether each token is found in the tree or not. If found it prints "Word [the word] found", else it calls the function get_suc_and_pre to print three suggestions. If the input is empty the function **ignores it** and prints 'you entered an empty String'. If the word contains any **symbol** if prints 'word [the word] is invalid'.

get_suc_and_pre: Using recursion, the function get the predecessor node and the node in the successor. And finally the parent of the predecessor.

In detail :

While the node isn't null (base case) :

1. If the word is less than the node,
 - a. then the successor is node->word
 - b. Call the function again with node->left instead of node

2. Else
 - a. predecessor is node->word
 - b. Call the function again with node->left instead of node
3. Store the node in parents array.

Print_parent: Prints the words stored in parents array which is not neither successor nor predecessor.

toLowerCase: converts the string to lowercase, so that **case-sensitivity** is handled. Except for 'l' nothing would change.

check_symbols: checks the validity of the word, returns 1 if just contains (a-z) and 0 otherwise.

Part II AVL Tree :

bal_tree.h

```
typedef struct node
{
    char word[32];

    struct node *left;
    struct node *right;
    struct node *parent;
    int balance;
}node;
```

This node contains:

- A node points to left
- A node points to right
- A node points to the parent
- An array of characters to store each word

- Integer balance to check its balance

Prototypes are:

```
node* search(node *ptr, char* data);
node *insert (char* data, node *ptr, int *flag);
void get_height(int lim);
int height_left(node* ptr);
int height_right(node* ptr);
int height(node* ptr);
```

search: checks whether the word is found in this node and its children or not. Returns the node if found and NULL if not.

insert: initialize this function is the core of the program. Here the tree is constructed and gets its balance

In detail :

1. Two nodes ptr_1 , ptr_2 are created for swapping on Rotations
2. Check the node
 - a. If it is NULL>> then the tree is empty >> allocate a node and assign the word to the node
 - i. Nodes: left , right , parent are NULL
 - ii. Balance in this case in 0 (no children)
 - b. Else if the value of the word is less than the root
 - i. Then insert the word to the left child of the root
 - ii. Check the balance of the tree
 1. If it's balanced (balance =0) make balance = 1 (heavy in left)
 2. Else if balance =-1 (heavy in right) make balance =0.
 3. Else if balance =1 >> ptr_1 (represents the left child)

- a. Swap the nodes to make a rotation and balance the tree , do LL rotation **(more details are below).**

4. Else do LR rotation

- c. Else if the value of the word is greater than the root
 - i. Mirror the step b (instead RR rotation & RL rotation)

Rotation algorithms are discussed below

get_height: computes the height of the tree. As the tree is AVL, the height is $\log_2(n)$, where n is the number of words. (method #1)

height_left: computes recursively the number of left nodes of the left node. And returns the number of levels of the tree based on the left node.

height_right: computes recursively the number of right nodes of the right node. And returns the number of levels of the tree based on the right node.

height: compares the number of levels based on both left and right nodes and returns the maximum. (method #2)

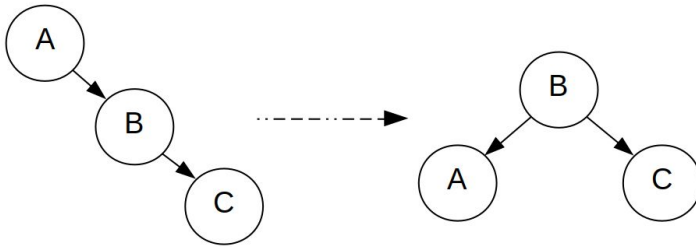
Rotations:

There are four models about the operation of AVL Tree:

LL RR LR RL

Case 1 (LL) Rotation :

Imagine we have a situation like this :



To fix this, we must perform a single left rotation, the steps are :

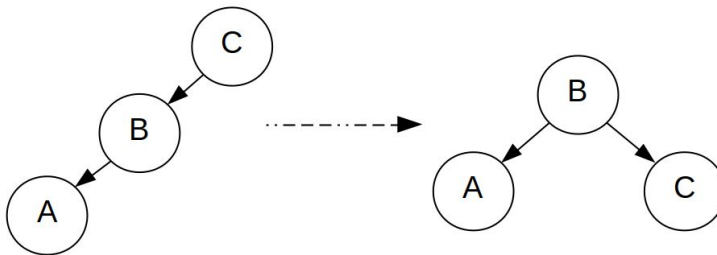
B becomes the new root.

A takes ownership of **B**'s left child as its right child, or in this case, null.

B takes ownership of **a** as its left child.

Case 2 (RR) Rotation :

It's a mirror of the above case. Imagine we have a situation like this :



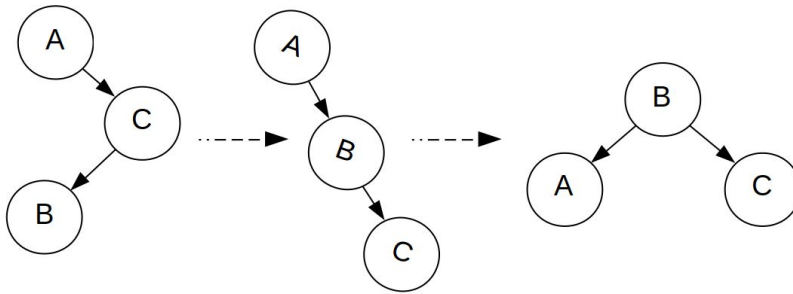
To fix this, we will perform a single right rotation, the steps are:

B becomes the new root.

C takes ownership of **B**'s right child, as its left child. In this case, that value is null.

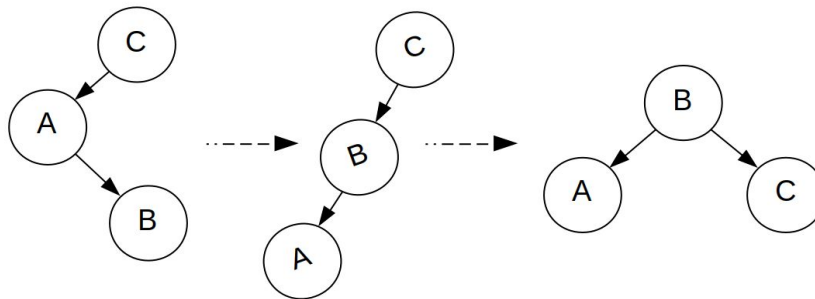
B takes ownership of **C**, as it's right child.

Case 3 (LR) Rotation :



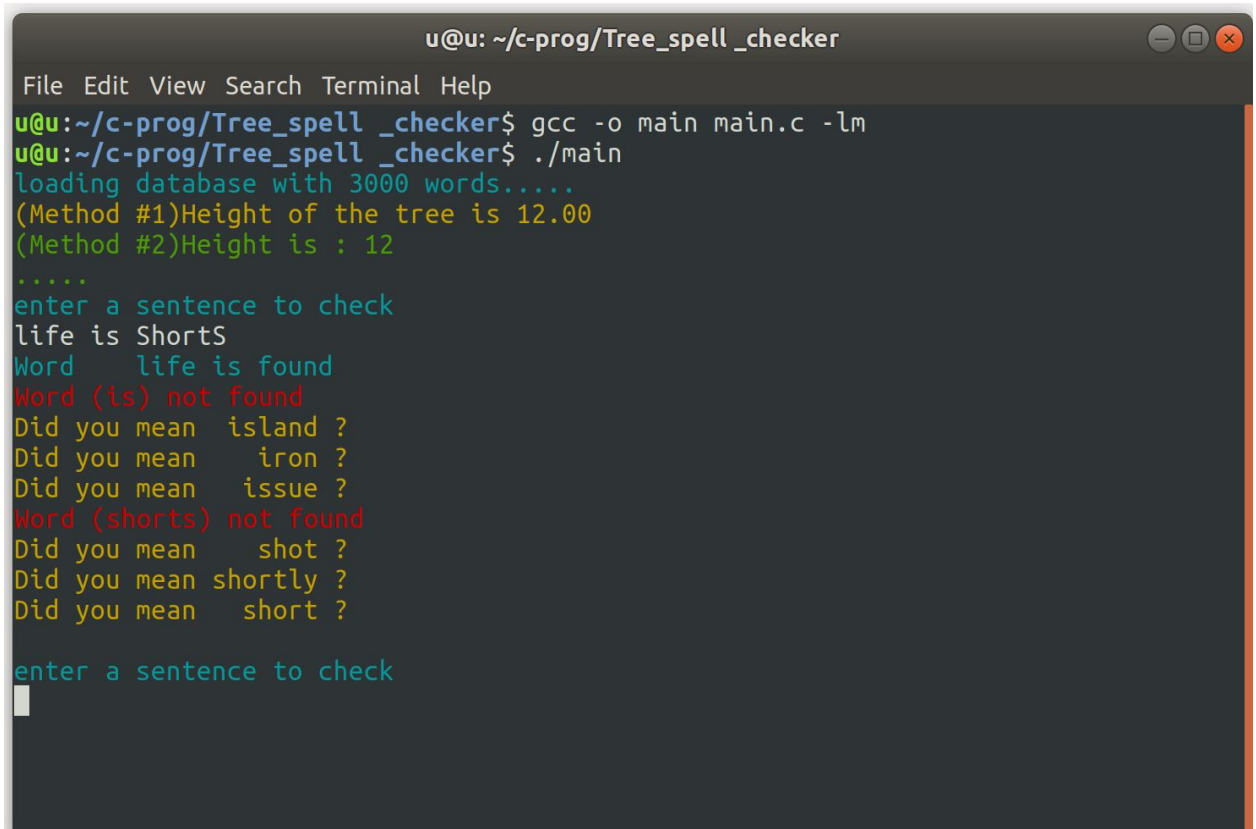
To fix this, we will perform a double left rotation. Note that : we are not rotating on our current root. We are rotating on the right child.

Case 4 (RL) Rotation :



To fix this, we will perform a double right rotation. Note that : we are not rotating on our current root. We are rotating on the left child.

ScreenShots:



```
u@u: ~/c-prog/Tree_spell_checker
File Edit View Search Terminal Help
u@u:~/c-prog/Tree_spell_checker$ gcc -o main main.c -lm
u@u:~/c-prog/Tree_spell_checker$ ./main
loading database with 3000 words.....
(Method #1)Height of the tree is 12.00
(Method #2)Height is : 12
.....
enter a sentence to check
life is ShortS
Word  life is found
Word (ls) not found
Did you mean  island ?
Did you mean  iron ?
Did you mean  issue ?
Word (shorts) not found
Did you mean  shot ?
Did you mean shortly ?
Did you mean  short ?
enter a sentence to check
█
```

```
u@u: ~/c-prog/Tree_spell_checker
File Edit View Search Terminal Help
enter a sentence to check
The HUGE GALaX
Word    the is found
Word    huge is found
Word (galax) not found
Did you mean  galaxy ?
Did you mean   gain ?
Did you mean gallery ?

enter a sentence to check
no Paie no GAie
Word    no is found
Word (paie) not found
Did you mean  pain ?
Did you mean  page ?
Did you mean  paint ?
Word    no is found
Word (gale) not found
Did you mean  gain ?
Did you mean  future ?
Did you mean  give ?

enter a sentence to check
```

```
u@u: ~/c-prog/Tree_spell_checker
File Edit View Search Terminal Help
enter a sentence to check
That is MY daw
Word that is found
Word (is) not found
Did you mean island ?
Did you mean iron ?
Did you mean issue ?
Word my is found
Word (daw) not found
Did you mean day ?
Did you mean daughter ?
Did you mean dead ?

enter a sentence to check
bonus is a passion
Word (bonus) not found
Did you mean book ?
Did you mean bone ?
Did you mean bombing ?
Word (is) not found
Did you mean island ?
Did you mean iron ?
Did you mean issue ?
Word a is found
Word passion is found

enter a sentence to check
█
```

```
u@u: ~/c-prog/Tree_spell_checker
File Edit View Search Terminal Help
enter a sentence to check
life is **f lo
Word life is found
Word (is) not found
Did you mean island ?
Did you mean iron ?
Did you mean issue ?
Word **f is invalid
Word (lo) not found
Did you mean load ?
Did you mean living ?
Did you mean little ?

enter a sentence to check

You entered an empty String

enter a sentence to check
tri
Word (tri) not found
Did you mean trial ?
Did you mean trend ?
Did you mean tremendous ?

enter a sentence to check

```

```
G:\Tree_spell_checker\bin\Debug\Tree_spell.exe
[36mloading database with 3000 words....[0m[33m
(Method #1)Height of the tree is 12.00[33m
[0m[32m(Method #2)Height is : 12
....[0m[36m
enter a sentence to check[0m
que is gree
[31mWord (que) not found
[0m[33mDid you mean question ?
[0m[33mDid you mean quarterback ?
[0m[33mDid you mean quarter ?
[0m[31mWord (is) not found
[0m[33mDid you mean island ?
[0m[33mDid you mean iron ?
[0m[33mDid you mean issue ?
[0m[31mWord (gree) not found
[0m[33mDid you mean green ?
[0m[33mDid you mean greatest ?
[0m[33mDid you mean grocery ?
[0m[36m
enter a sentence to check[0m
**
[35mWord ** is invalid
[0m[36m
enter a sentence to check[0m
```

```
G:\Tree_spell_checker\bin\Debug\Tree_spell.exe
[0m[36m
enter a sentence to check[0m
**
[35mWord ** is invalid
[0m[36m
enter a sentence to check[0m
how har gar :)
[36mWord how is found
[0m[31mWord (har) not found
[0m[33mDid you mean hard ?
[0m[33mDid you mean happy ?
[0m[33mDid you mean happen ?
[0m[31mWord (gar) not found
[0m[33mDid you mean garage ?
[0m[33mDid you mean gap ?
[0m[33mDid you mean gang ?
[0m[35mWord :) is invalid
[0m[36m
enter a sentence to check[0m
```

Notes

***The dictionary lacks some crucial words (is - am - has - cancel - exam.....)**

***This program is designed to work on Linux. However it works on Windows with some restrictions :**

The colored output is not supported

***Additional Features:**

- 1. Colorful output (Linux only)>> On windows 'undefined symbols'**
- 2. Case-sensitivity handled**
- 3. Symbols and Numbers handled**

[Kindly check the video attached](#)

References

J.-L.Baer and B.Schwab, "A comparison of tree-balancing algorithms"

http://enos.itcollege.ee/~jpoial/algorithms/GT/Goodrich_6e_Ch11_AVLTrees-handouts.pdf

<http://interactivepython.org/runestone/static/pythonds/Trees/AVLTreeImplementation.html>

<http://mathcs.wilkes.edu/~kapolka/cs126/notes/spellck/>

<https://www.slideshare.net/amrelarabil/python-spell-checker>

<https://www.rose-hulman.edu/class/csse/csse230/201410/Slides/14-AVLInsertionRotation.pdf>

<https://bradfieldcs.com/algos/trees/avl-trees/>