Assignment 7: The Great Firewall of Santa Cruz

Kelly Liu

November 28, 2021

1 Description

In this assignment, we are using bloom filters and hash tables in order to filter out offensive words. The hash tables consists of a bunch of binary search trees which has a bunch of nodes. Our bit vector helps us write our bloom filter. We compile everything and ensure that we have a statistics function that provides us information about our operations.

2 Pseudocode

2.1 bf.c

2.1.1 bf_create

Initialize my bloom filter through malloc
Initialize all other variables
Return bf

2.1.2 bf_delete

Delete our bloom filter and free up memory

2.1.3 bf_insert

Find my primary index by hashing my oldspeak with my primary
Set my primary to itself mod the length of my filter
Find my secondary index by hashing my oldspeak with my secondary
Set my secondary to itself mod the length of my filter
Find my tertiary index by hashing my oldspeak with my tertiary
Set my tertiary to itself mod the length of my filter
Set my primary, secondary and tertiary index to 1 by calling on bv_set_bit

2.1.4 bf_probe

Find my primary index by hashing my oldspeak with my primary
Set my primary to itself mod the length of my filter
Find my secondary index by hashing my oldspeak with my secondary
Set my secondary to itself mod the length of my filter
Find my tertiary index by hashing my oldspeak with my tertiary
Set my tertiary to itself mod the length of my filter
Use an if statement to check if the primary, secondary, and tertiary index is 1
If so, return true
Else, return false

2.1.5 bf_count

Make a total variable
Use a for loop to iterate through my filter
Use an if statement to check if the bit at each iteration is 1
If so, increment total by one
End of if statement
End of for loop
Return total

2.1.6 bf_print

Use bv_print to print my filter

2.2 ht.c

2.2.1 ht_create

Initialize my hashtable using malloc Initialize all my variables For trees, we will use calloc Return my hashtable

2.2.2 ht_delete

Delete my hash table and free up memory

2.2.3 ht_print

Debug function that contains print statements that I think I would need

2.2.4 ht_size

Return the hash table size

2.2.5 ht_lookup

Increment lookups by one
Hash the oldspeak with my salt to get my index
Mod my index by the hash table size
Set a node to my hashtable at my index
Check if it's Null
If so, return Null
Else, return my node

2.2.6 ht_insert

Increment lookups by one
Hash the oldspeak with my salt to get my index
Mod my index by the hash table size
Set my hashtable at my index to the node by using bst_insert

2.2.7 ht_count

Set a total variable
Use a for loop to loop through each binary search tree in my hash table
Use an if statement to check if the binary search tree at that iteration exists
If so, increment total by one
End of if statement
End of for loop
Return total

2.2.8 ht_avg_bst_size

Set a total variable
Use a for loop to loop through each binary search tree in my hash table
Increment total by itself plus the size of that binary search tree at that iteration
End of if statement
End of for loop
Divide the total by the count of my hash table
Return total

2.2.9 ht_avg_bst_height

Set a total variable
Use a for loop to loop through each binary search tree in my hash table
Increment total by itself plus the height of that binary search tree at that iteration
End of if statement
End of for loop
Divide the total by the count of my hash table
Return total

2.3 bst.c

2.3.1 bst_create

return null

2.3.2 bst_height

Check if root is null and if so return 0
Recursively find the height by going left and then right
Use an if statement to check if left is bigger than right
If so, return left plus 1
Else,
Return right plus one

2.3.3 bst_size

Check if root is null and if so return 0
Recursively find the height by going left and then right
Return the right plus the left plus one

2.3.4 bst_find

Check if root is null and if so return null
Check if the root's oldspeak is equal to the oldspeak passed in and if so return root
Increment branches by one
Use an if statement to check if oldspeak is less than the current root's oldspeak
If so, recursively traverse left
Else,
Recursively traverse right

2.3.5 bst_find

Check if root is null and if so return null
Use an if statement to check if oldspeak is less than the current root's oldspeak
If so,
Increment branches by one
recursively traverse left
End of if statement
Use an if statement to check if oldspeak is greater than the current root's oldspeak
If so,
Increment branches by one
recursively traverse right
End of if statement
Return root

2.3.6 bst_print

Check if root is null and if so return null
In an in order traversal, print out our tree

2.3.7 bst_delete

If our root exists, recursively go in an post order traversal to delete each node of our tree

2.4 node.c

2.4.1 node_create

Create a new and old char pointer
If newspeak isn't null, create a copy using strdup and set it to new
Else, set new to null
If oldspeak isn't null, create a copy using strdup and set it to old
Else, set old to null
Initialize everything
Set our oldspeak to old and our newspeak to new
Set our left and right to null
return our initalization of node

2.4.2 node_delete

Delete our node and free up memory

2.4.3 node_print

Use an if statement to check if our node's oldspeak and newspeak isn't null
If so, print out our node's oldspeak and newspeak
Use an if statement to check if our node's oldspeak isn't node and our node's newspeak exists
If so, print out our node's oldspeak

2.5 banhammer.c

I will use getopt for my switch cases.
Then I will open my files using fopen while testing for errors.
I will set my regex while testing for errors.
I will read in my files line by line and setting it as directed to my bloomfilter and hashtable
I will create a while loop for while words exists,I will read from my stdin file word by word with my regex filter
Set my current word to all lowercase
Create a badlist variable and an oldlist variable to hold my badspeak and my oldspeak/newspeak transitions
Create boolean variables for flagone for thoughtcrime and flagtwo for counseling
Use an if statement to check if that word is in my badspeak list(outter if statement)
If so,
Look up my word
Use an if statement to check if my word exists and if it's newspeak is null,(inner if statement 1)
If so,
Set flag one to true
Insert my word into my badlist
End of inner if statement 1 Use an if statement to check if my word exisits and if it's newspeak exists, (inner if statement 2)
If so,
Set flag two to true
Insert my word and it's translation into oldlist
End of inner if statement 2
End of outter if statement
End of while loop
If user calls on stats function, print out the respective statistics and exit
Use an if statement to check if user is flagged for thoughtcrime and counseling
If so,
Print out my mixspeak message and my badlist and oldlist
End of if statement
Use an if statement to check if user is flagged for thoughtcrime
If so,

Print out my badspeak message and my badlist
End of if statement
Use an if statement to check if user is flagged for counseling
If so,
Print out my goodspeak message and my oldlist
End of if statement
Free and close everything

2.6 bv.c

My bv_ create was provided by Professor Long in the code comments repository. The rest of the file is taken from my own code in my code.c file in Assignment 5.

3 Error Handling

1. There were alot of parts where I was super careless because I didn't read the assignment document carefully enough.
2. There were times where I copied the header file into my source file to get the different functions, I would error out because there were some header syntax I did not need. 3. I had alot of errors because I forgot to include certain files. 4. I also had alot of errors because I forgot to include certain libraries. 5. I had an error compiling because I forgot to include the math library flag when I was running my banhammer.c since that uses my math library. 6. I had an error in my print statement in my bst file because I didn't traverse correctly. 7. I had a bunch of syntax errors which was easily fixed when I saw the errors when I compiled. 8. I had an issue with valgrind because I forgot to free something in my ht file.

4 Files

parser.c- A source file for implementing my regex parsing module.
parser.h- A header file that defines the interface for parser.c.
bv.c- A source file for implementing my bit vector ADT.
bv.h- A header file that defines the interface for bv.c.
bf.c- A source file for implementing my Bloom filter ADT.
bf.h- A header file that defines the interface for bf.c.
banhammer.c- A source file that has my main function.
messages.h- A header file for my banhammer.c file.
salts.h- A header file that defines my mixspeak,secondary, and tertiary salts functions.
speck.c- A source file for implementing my SPECK cipher.
speck.h- A header file that defines the interface for speck.c.

ht.c- A source file for implementing my hash table ADT.
ht.h- A header file that defines the interface for ht.c.
bst.c- A source file for implementing my binary search tree ADT.
bst.h- A header file that defines the interface for bst.c.
node.c- A source file for implementing my node ADT.
node.h- A header file that defines the interface for node.c.
Makefile- This allows us to use clang and compile our program.
README.md- In markdown format, it tells us how to run the program and how the program was made.
WRITEUP.pdf- A PDF considering graphs made by UNIX tool displaying my results and my thoughts on them.
DESIGN.pdf- This is how I started thinking about how to code the program.

5 Credit

1. Professor Long has provided pseudocode in the Assignment description PDF for our files as well as describing specific things that we should be doing.
2. Professor Long has provided us with a few files in the resources folder in Assignment 7.
3. I used Professor Long's bv16.h in order to write my bv_ create.
4. I found that my code.c file in Assignment 5 can be used for my bv file as long as I tweaked it a bit.