Assignment 7 Design

Brandon Chuang

11/28/21

Creating A firewall for the Glorious People's Republic of Santa Cruz.

1 Introduction

To keep the minds of the Glorious People's Republic of Santa Cruz's (GPRSC) pure and free from the corruptions of others, we will be monitoring their online actions and educating them on what should be said, called a goodthink message. If they do not abide to keep the GPRSC free from unspeakable thoughts or words, further action will be taken. This positive lexical ordinance will be accomplished with hash tables and bloom filters, since there will be too much data to actually read each message.

2 Bloom Filters

The bloom filter is a test that will never send a false negative but sometimes provide a false positive. The reason why we use this before the hash table (which has 100% accuracy) is because the hash table is slow, and the bloom filter is fast because we are checking only three bits rather than entire strings (not to mention searching the trees as well).
The way we manage the bloom filter is by having an equation and 3 different "salts" that determine what 3 bits we will set to represent the oldspeak being present. When determining if a word has been inserted into the bloom filter, we will send the word through the same equation with the same salts. If the same three bits are set, then we determining that it is potentially oldspeak. Because the bloom filter is not deterministic, we require a hash table to finish the job and actually determine if the word has been told to be oldspeak.

2.1 Bit Vectors

We will need an abstract data type known as bit vectors for the bloom filter. The bit vector is just a list of bits that we change. This is only needed because the smallest data type is an int, which is 8 bits. We could use a list of booleans, but this would be extremely inefficient because booleans are still 8 bits.

3 Hash Tables

A hash table is kind of like a dictionary, where it has a key (word) and a value (definition). However, unlike dictionaries, We will be using nodes, containing the oldspeak and newspeak words, and pointers to left and right nodes, to store all the badspeak words and their replacements (newspeak). We will be using the hash table as a deterministic way to determine if something is oldspeak, rather than just using probability.
The way we store values in the hash table is with another equation and another "salt" that determines its position in an array of binary trees. To determine if a word is oldspeak (and to find its newspeak if it exists), we run the word through the same equation with the same "salt" and search the binary tree at the correct part of the list. If the node exists with the same word, that word is oldspeak, and we can write out the newspeak as well.

3.1 Binary Search Trees

Binary search trees are trees where each node can only have two children, where the left child is less than the right child. This allows us to quickly navigate the tree rather than relying on searching each branch to find the right node.

4 I/O

These are the following commands available:

- -h: Prints out help and informs the user of the programs purpose, function, and commands.
- -t: Allows the user to set the size of the hash table. By default, it is set to 2^{16}.
- -f: Allows the user to set the size of the bloom filter. By default, it is set to 2^{20}.
- -s: prints statistics

5 Execution

All of this will be coded in banhammer.c and be executable from the banhammer binary.

- Parse through the command options and set everything accordingly or print the help.
- Read through all of badspeak.txt and newspeak.txt, putting them into a hash table. we will also use the entries in both files to build the bloom filter.

- Parse through each word (using regex) and run the word through the bloom filter
- If the bloom filter returns true (that the word is probably oldspeak), run the word through the hash table
- If the hash table returns true (that the word is definitively oldspeak), check to see if there is a corresponding newspeak. If there isn't record the wrongly used word in another binary search tree
- if the -s command is input, only print the statistics and end the program.
- If the person used both oldspeak and badspeak, print the mixspeak message along with their transgressions.
- If the person only used oldspeak, print the goodspeak message along with their transgressions.
- If the person only used badspeak, print the badspeak message along with their transgressions.