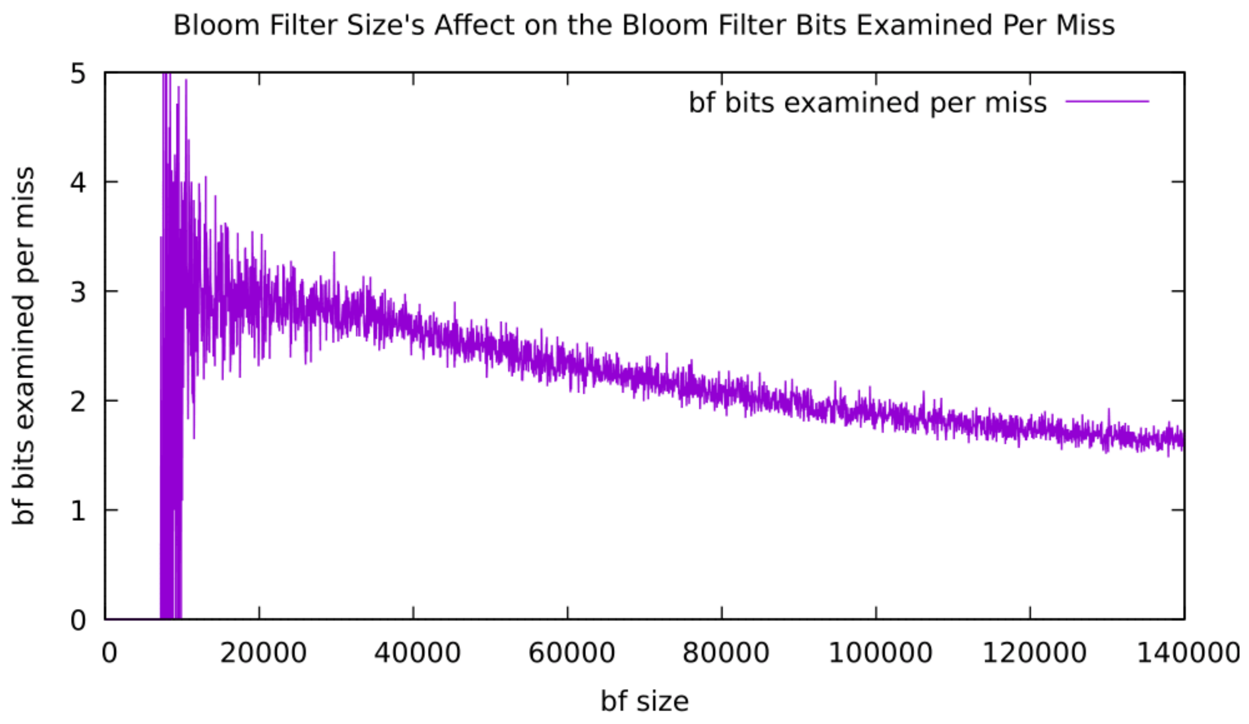
Isaac Flores
CruzID: **isgflore**
Prof. Miller
CSE13S
November 20, 2022

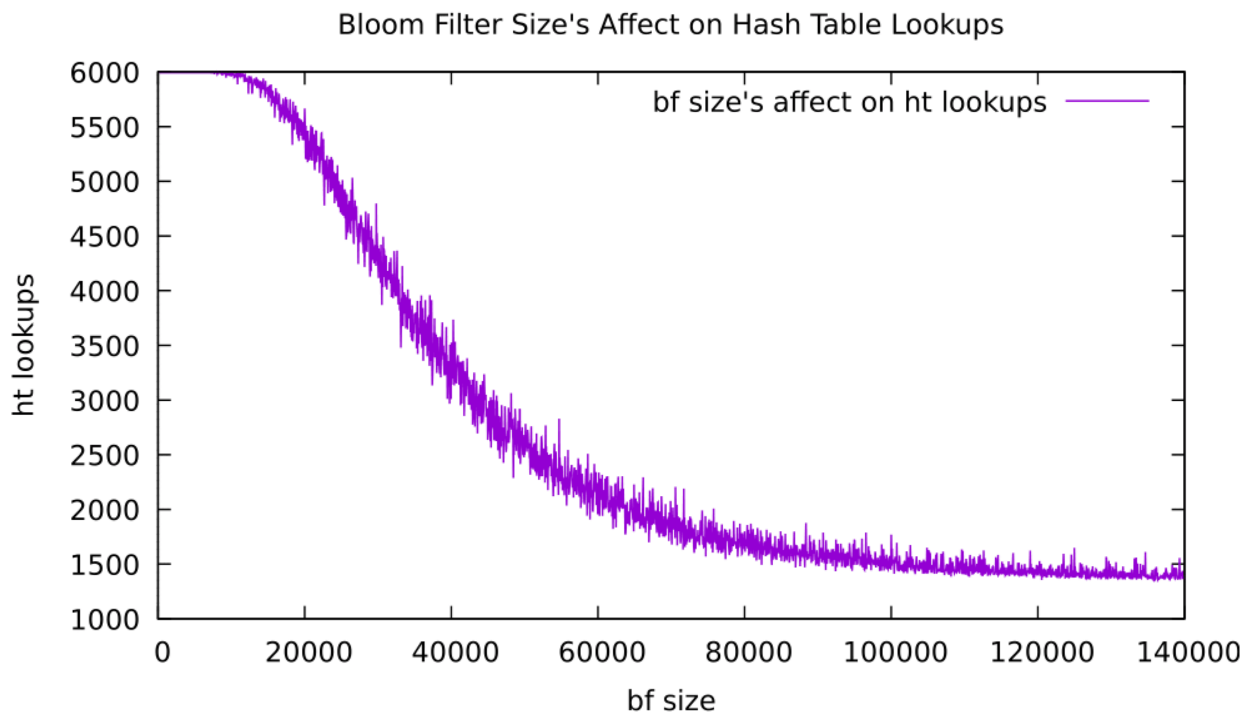
WRITEUP for ASGN 6: The Great Firewall of Santa Cruz: Bloom Filters, Linked Lists, and Hash Tables

A. How does the number of Bloom filter bits examined per miss vary (for the same input) as the Bloom filter varies in size?



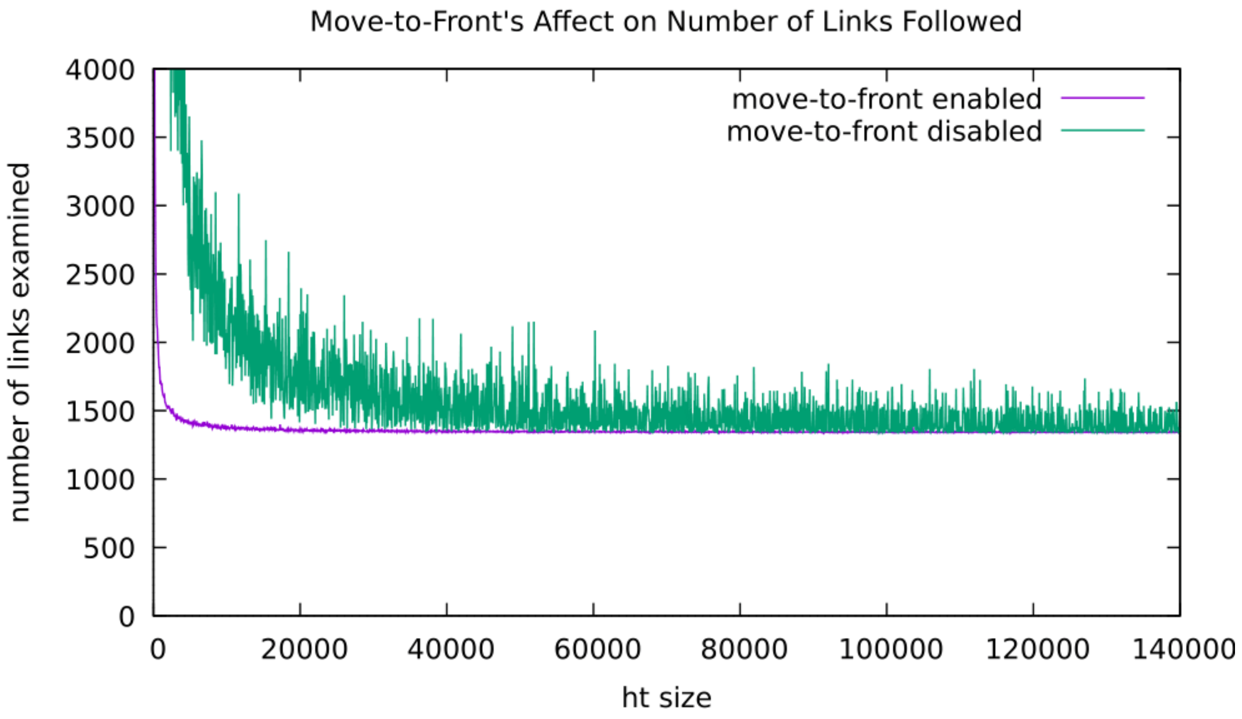
Analysis: The Bloom filter bits examined per miss decrease as the Bloom filter size increases. This makes sense since there are fewer chances for a false positive as the Bloom filter size increases. There is more room for each hashed word to have its own set bit in the Bloom filter that doesn't overlap with another hashed key. This allows misses to be found easier because there are more chances for a word to have its own set bit.

B. How does changing the Bloom filter size affect the number of lookups performed in the hash table?



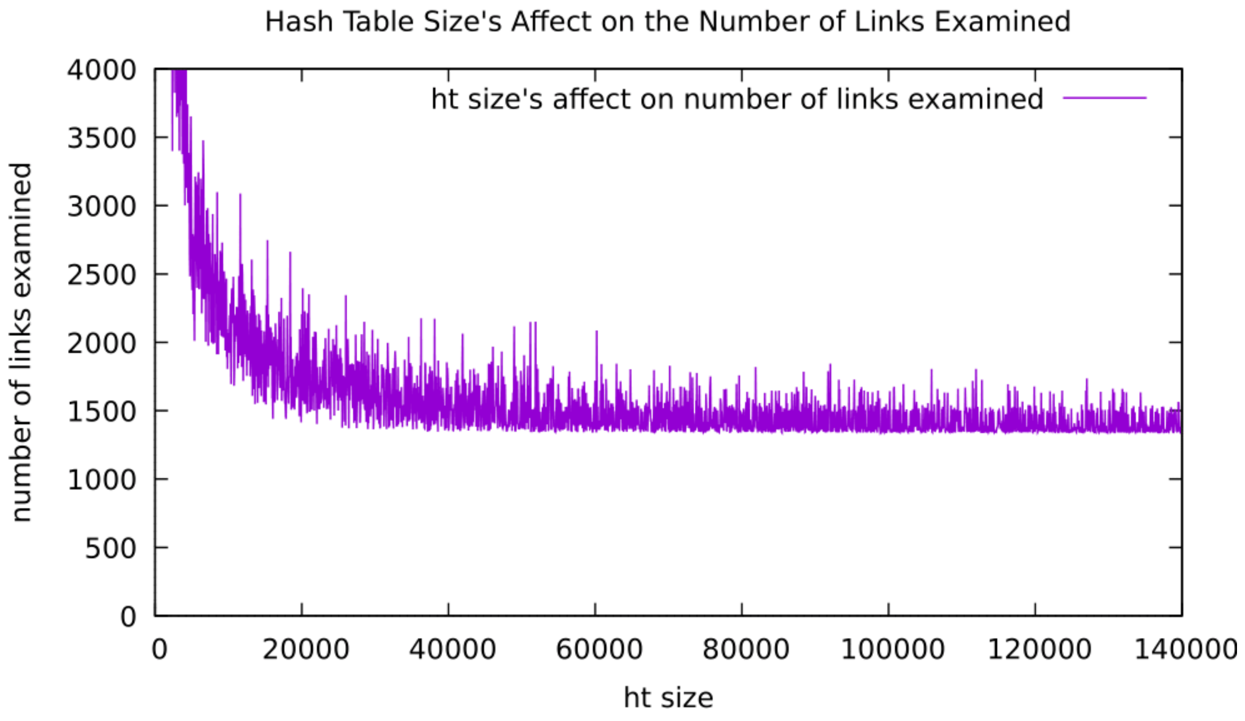
Analysis: As the Bloom filter size increases the number of lookups performed in the hash table decreases. This is because the smaller the bloom filter is the more false positives would occur. For every Bloom filter positive, I look up the word in the hash table. Since the smaller the Bloom filter size the more false positives would occur, the number of hash table lookups would also increase. On the other hand, as the Bloom filter size increases the number of false positives would decrease which would cause fewer lookups to be performed.

C. How does the number of links followed without the move-to-front rule compare to the number followed with the move-to-front rule?



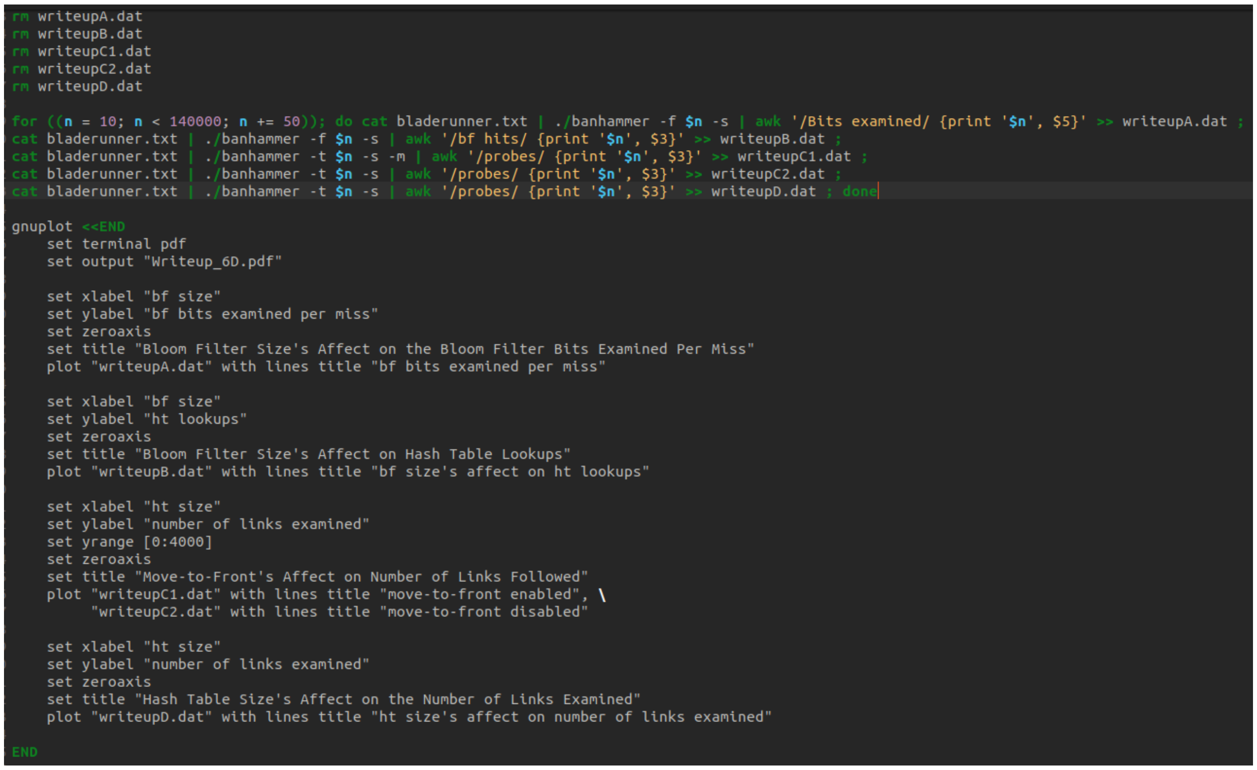
<u>Analysis:</u> The number of links followed with the move-to-front rule is more consistent than if it were disabled. I checked the number of links followed by both conditions while the hash table size increased. As seen above, the graph when move-to-front is enabled doesn't deviate nearly as much as the graph when move-to-front is disabled. Move-to-front allows common words to be moved to the front of the linked list, which reduces the number of links followed. Common words like 'the' or 'a' would be pushed to the front and wouldn't have to be searched for deeper inside the linked list. Words already looked up would get closer to the head of the list and take fewer links to reach.

D. How does the number of links examined vary as the size of the hash table varies? What does this say about setting the size of the hash table when using a chained hash table?



Analysis: The number of links examined decreases as the size of the hash table increases. The graph seems to flatten out at around 14,000 links examined when the hash table size is 40,000. Since we used a chained hash table, hash collisions occur but each word at the same index is stored in a linked list. This means that hash collisions cause the number of links examined to increase when looking up in a hash table. In turn, the probability of a hash collision occurring increases as the size of the hash table decreases since there are fewer spots for each word to be located. When the hash table size increases the probability of a hash collision decreases meaning the number of links examined when looking up a hash table decreases.

Bash Script Used to Create the Graphs:



```
rm writeupA.dat
rm writeupB.dat
rm writeupC1.dat
rm writeupC2.dat
rm writeupD.dat

for ((n = 10; n < 140000; n += 50)); do cat bladerunner.txt | ./banhammer -f $n -s | awk '/Bits examined/ {print '$n', $5}' >> writeupA.dat ;
cat bladerunner.txt | ./banhammer -f $n -s | awk '/bf hits/ {print '$n', $3}' >> writeupB.dat ;
cat bladerunner.txt | ./banhammer -t $n -s -m | awk '/probes/ {print '$n', $3}' >> writeupC1.dat ;
cat bladerunner.txt | ./banhammer -t $n -s | awk '/probes/ {print '$n', $3}' >> writeupC2.dat ;
cat bladerunner.txt | ./banhammer -t $n -s | awk '/probes/ {print '$n', $3}' >> writeupD.dat ; done

gnuplot <<END
    set terminal pdf
    set output "Writeup_6D.pdf"

    set xlabel "bf size"
    set ylabel "bf bits examined per miss"
    set zeroaxis
    set title "Bloom Filter Size's Affect on the Bloom Filter Bits Examined Per Miss"
    plot "writeupA.dat" with lines title "bf bits examined per miss"

    set xlabel "bf size"
    set ylabel "ht lookups"
    set zeroaxis
    set title "Bloom Filter Size's Affect on Hash Table Lookups"
    plot "writeupB.dat" with lines title "bf size's affect on ht lookups"

    set xlabel "ht size"
    set ylabel "number of links examined"
    set yrange [0:4000]
    set zeroaxis
    set title "Move-to-Front's Affect on Number of Links Followed"
    plot "writeupC1.dat" with lines title "move-to-front enabled", \
         "writeupC2.dat" with lines title "move-to-front disabled"

    set xlabel "ht size"
    set ylabel "number of links examined"
    set zeroaxis
    set title "Hash Table Size's Affect on the Number of Links Examined"
    plot "writeupD.dat" with lines title "ht size's affect on number of links examined"

END
```

Explanation: To create the four graphs used to answer the write-up questions I for looped n from 10 to 14,000 by 50. The text file I used for banhammer contains the entire *Blade Runner* script. To create the first graph I set the Bloom filter size to $n and printed only the bits examined per miss to the writeupA data file. I graphed this data to get the first graph. For the second graph, I set the Bloom filter size to $n and printed only the Bloom filter hits which is equal to the number of hash table lookups performed to the writeupB data file. I graphed this data to get the second graph. For the third graph, I set the hash table size to $n and enabled move-to-front. I put this data into the writeupC1 data file. I did this again but disabled move-to-front and put this data into the writeupC2 data file. I graphed both of the data files together to get the third graph. For the fourth graph, I set the hash table size to $n and printed this data into the writeupD data file. I graphed this data to get the fourth graph.