Assignment 5 – Tower report

Airi Kokuryo

CSE 13S – Winter 24

Purpose

The purpose of this report is to have a deeper understanding of hash tables and its steps in implementation.

Questions

Part I

In Part I, you implemented garbage collection—routines that clean up dynamically-allocated memory. How did you make sure the memory was all cleaned up? How did you check?

The garbage collection routines were done by using free() by giving back all the heap memory to the OS. By using the command, $ valgrind ./toy, it returns how much memory leakage we have. If this is 0%, this means that we have successfully cleaned up all the memory.

Part II

In Part II, you made a major optimization to the linked list optimization. What was it, and why do you think it changed the performance of bench1 so much?

The major optimization to the linked list optimization is to change the cmp function that compares the data inside the given two pointers to make the performance better. This is important because how it reads through(compare) the given linked list directly connects to how fast the function will find and return the value. This also means visualization of which algorithm has the better performance when reading through the list.

Part III

In Part III, you implemented hash tables. What happens to the performance of bench2 as you vary the number of buckets in your hash table? How many buckets did you ultimately choose to use?

The number of buckets determine the size of the array used to store key-value pairs in a hash table. Therefore if there are few collisions, having less buckets will have better performance, and if there are more collisions, having more buckets will reduce collisions which leads to having a better performance. Since we have to go through 104K words, I thought there would be more collisions and should make a larger bucket for the hash table. Therefore I chose 20000 buckets to start with.

Bugs

How did you make sure that your code was free from bugs? What did you test, and how did you test it? In particular, how did you create inputs and check the output of uniqq?

I made sure that my code was free from bugs by using the uniq and wc for testing. Uniq removes duplicate lines, ensuring that only unique lines are displayed, so I will use this for creating a output I could compare

with. Also, wc is a word count, so this will also be used to ensure the given output is exactly the same. In this case of testing, I will check the output of uniqq which would be the number of unique lines, by comparing it to the wc after using uniq on the given file or manual input.