Justin Choi (jc8mc)
March 15th 2017
CS 2150 – 106, Lab 6: Hashing

inlab6.pdf

In lab 6: hashing, implemented a hash table to find the solution to a word puzzle with different rows and columns. My implementation of the hash table and the word puzzle code generated the correct results. However, I had to reformat the output: I had to sort my own output file and the given output file first. Then, I had to compare the two sorted output files with the diff command, which reports the differences between the files, along with the –w flag in order to ignore all the whitespace on the lines it compared.

With the –O2 flag that we learned in inlab6 and partly in the Makefile tutorial for lab 5, the run time for the program produced a better result, a lower average time. As we are concerned with the worst-case scenario, I compared the given biggest grid file, 300x300.grid.txt with the first dictionary file, words.txt. When the program was compiled with the –O2 flag my laptop (*Macbook Air macOS Sierra Version 10.12.3 Mid 2013 1.3 GHz Intel Core i5, 4GB 1600 MHz DDR3, Intel HD Graphics 5000 1536 MB*) gave an average time of 6.857 seconds. When the program was compiled without the –O2 flag, the average time was 34.751 seconds. It is obvious that the –O2 flag produced a faster result since it produces an optimized executable. Without the –O2 flag, the average time is almost 5 times slower, or 0.2 times faster with the –O2 flag.

When compiled with the –O2 flag, for the 250x250.grid.txt grid using words.txt as the dictionary file, it produced a time of 5.162 seconds. For the 300x300.grid.txt grid using words2.txt as the dictionary file, it produces a time of 1.913 seconds.

The big-Theta running speed of my program is r * c * w (r for rows, c for columns, w for words) as the l, length of the words can be ignored as a constant.

Justin Choi (jc8mc)
March 15th 2017
CS 2150 – 106, Lab 6: Hashing

When I was implementing this lab, I ran into the trouble of using the implemented lab 2 code, the linked list lab since I thought it would be perfectly fine to use it. However, it produced various errors due to a type mismatch: as the lab 2 linked list was a linked list of integers while we wanted a linked list of strings, there was a lot of fixing that I had to do. Frankly, I could not figure out what I had to change after trying numerous times. Then as I asked a TA for assistance, the TA recommended me to use the list from the STL. With the list from the STL, I just had to declare it as a list of strings, and everything was taken care of. So, while using the linked list from lab 2 created a lot of frustration and anxiety, I did learn that I should use the list or other data structures from the STL in the future as they are implemented correctly and produces better results. Also, it would be better practice to use the STL since as a computer scientist, I would not have to create my own data structure to implement a given project.

Shell scripting was a good learning experience so far. Prior to this hash lab, I did not know about shell scripting. However, I learned that shell scripting can save a lot of time since we would only need to call few Unix commands for a large number of Unix commands if shell scripting is not used. So, it is a very efficient way to save time when I have to do multiple, repeated executions. While I thought they were useful, and it was not hard to understand the concepts, I had trouble in the implementation for averagetime.sh. I am used enter space in variable declarations or any type of code I write for most programming languages While C/C++ does not care about the empty white lines or blanks in between variables, the shell scripting had a very rigid structure for empty white lines and spaces, which led me to fix the shell scripting code multiple

times, which was frustrating at first. Also, I also found it a little bit inconvenient that I could only do one arithmetic operation using the expr keyword. This created me to sum all the time runs four times, rather than once, before dividing it by 5 to get the average time. Other than these challenges imposed, learning shell scripting was fun.