Eza Rasheed

er6qt

03-5-19

inlab6.pdf

1. Yes, my implementations did produce the correct results. I tested my 300x300.words2.out.txt and my 50x50.out.txt with my output and got the same results. However, when I first used “diff” on the output file and the grid.out text file to compare them, when I compiled, I got many differences outputted. Then I compared my two files in emacs and saw that my results were the same, but just in a different order. Then I used the Unix sort command which was provided to reformat my output and sort both of the files before comparison, and thankfully, no differences were found when I compiled again.
2. The speed of my implementation when I ran the 250x250 grid using words.txt was 32.7677 seconds, and for the 300x300 grid, it was 39.9027 seconds. This was my speed when I used the -O2 flag. When I got rid of the -O2 flag, I noticed a ginormous difference. My 250x250 grid completed in 77.831 seconds, while my 300x300 grid completed in 95.1526 seconds. Therefore, it is clear that adding in the -O2 flag makes a great difference in implementation speed for word search, or at least for me it did. Note** I ran these files on my HP Spectre laptop using Linux virtual box.
3. The big-Theta running speed of my program is r*c*w. In other words, rows times columns times words.
4. I encountered numerous problems when implementing this lab. This was half because of the fact that I missed the hash lecture in class and the recording was not posted online, and so I was completely lost. When implementing my wordPuzzle, it worked perfectly when I used unordered set. But then, when I actually went to go incorporate my newly created hashTable into wordPuzzle, my read in dictionary stopped working. I had to remove the method I made for reading in my dictionary, and put what was inside of it into the main. This way, I was reading in and counting and inserting before I started to look through my quad loop search. Second of all, I was returning “true” after my file.close(), which was before my timer for the quad loop searching even started. Because of this whenever I ran a.out, it kept on asking me for the next command without outputting anything. Once, I got rid of it, when I compiled, I was finally shown some output. However, only one word was being outputted when I ran my 3x3 grid file, rather than four. After asking various TA’s, one special TA helped me in resolving this issue of limited insertion. In my insert function in hashTable.cpp, I was doing “push_back” on my string word inside my if statement in which I was checking to see if the word at a certain position was NULL, then create a string there. I was supposed to do the pushing back, or inserting, regardless of the if condition. All I had to do was put it outside the if statement brackets. Sadly, this one easy fix was the reason I had to turn my lab in **late**. Lastly, a

problem I encountered when I initially was testing my wordPuzzle.cpp was that I was getting many repeats of words when I ran my files. This was fixed by having an extra if statement after my quad loops that checked to see if the length of the word was equal to l, which was the variable that made sure to check that word was no less than 3 and no greater than 22. This got rid of my repeats effectively. In all, this lab has been the hardest assignment I have ever had, and this is primarily due to coding so many separate little things to get a whole program (word search) working with 100% accuracy in just 2 days; it was very nit-picky work.

5. In general, shell scripting writing is easy. What made it slightly difficult was trying to learn on our own, through the tutorials, how to write it. Once I got how to do the first run though, the rest was just easy copying and pasting, so I did not mind it at all. It was just the part that led up to it which was time consuming. The only problem I encounted after compiling it was that when I was printing out the total time taken as the last line of output in wordPuzzle.cpp, I was printing out words with it as well, rather than just getting the time. So the string part of the cout there was causing an error reading file problem. Once I got that resolved, everything compiled and ran successfully.