Maria Hito
3/15/17
File name: inlab6.pdf

Did your implementation produce the correct results? Did you have to reformat your output?

My implementation produced the expected output, but the order and the formating of the last two statements were different than the ideal output. I had to change the the punctuation and spaces of the statements as well as removed a debugging cout statement that printed the current word being searched for in the hashTable.

How much faster was your program with the –O2 flag?

Without the –O2 flag, the program took 2.12041 seconds to run with a dictionary of “words2.txt” and a grid of “140x70.grid.txt”.

Once the optimization flag was added, the program took 1.28113 seconds to find all the words of the same file used previously.

This is a 50% increase in speed (the optimized version runs in almost half of the original time) This showed me the difference of running a program using the compilers-automated and running a program with a regular compiler. Compilers-automated optimization can help to improve the run time significantly.

What was the speed of your implementation? How fast did it run on the 250x250 grid using words.txt as the dictionary file? What about words2.txt and the 300x300 grid?

My a.out file completed the 250x250 grid with words.txt as a dictionary file in 37.05 seconds. This is a pretty slow computation time.

For the 300x300 grid with words2.txt, the program completed in 51.9877 seconds.

What is the Big-Theta running complexity?

My program runs at a Big-Theta time complexity of (R*C*W) since the length of words can be factored out as a constant.

What problems did you encounter?

I had a lot of trouble getting my computations to run in a short amount of time. The first time my program took like 60 seconds to run, which was pretty slow. I eventually found a way to optimized my nested for-loops by including more if-statements for special cases in order to bypass extraneous calculations. For example, if the length was less than expected (the word was cut off by boundaries of the grid) then the for loops break immediately. Having optimizations like this helps minimize repetition of superfluous and complex operations.

Also, for some of the bigger files I was getting the wrong number of words. The total number of words in the output files were not matching with the number number of words my program was finding. I fixed this by initializing the variable numOfLines to 0.

Shell Scripting

I found Shell scripting a little bit strange. Bash as a language seems like a very stripped-down language, for example only one arithmetic function can be used in each `expr` clause, increasing the need for multiple instructions. This seems reasonable, because the design of bash and other scripting languages was not for arithmetic computation, but rather for file-system tasks, and for that it excels.

I especially liked the ease with which one can utilize shell scripts within shell scripts, along with any other command-line programs, in order to automate user-interaction with a shell. The weakly typed variables were reasonable given the uses of the language. There shouldn't be too many different types being used in the first place in a bash script, so a typeless 'var' type works well.