Ben Haines, bmh5wx, 10/21/14, inlab6.pdf

1 Results

My first attempt did not produce correct results. I had attempted to write my own code to get words from the table because I hadn't realized that it was provided to us. When accessing elements of the two dimensional puzzle array I had confused the index that controlled the row with the one that controlled the column. This was solved by replacing my code with the provided getWordInTable function.

The second problem I encountered with my output was how to display the direction as a letter rather than a number. I solved this by creating an array with all the directions indexed according to their integer representation in the program.

2 Speed: without -O2

The results are from running the program on my my laptop, not the machines in the Olsson 001 lab.

		Puzzle					
		3x3	4x7	50x50	140x70	250x250	300x300
Wordlist	words.txt	0.000559	0.023815	15.901401	59.141840	367.404595	509.516638
	words2.txt	0.000045	0.001906	0.923494	3.379731	20.356651	29.431320

3 -O2 Flag

Using the wordlist words.txt and the puzzle 4x7.grid.txt, the -O2 flag reduced the running time from 0.023815 seconds to 0.010791 seconds. Using the wordlist words.txt and the puzzle 140x70.grid.txt, the -O2 flag reduced the running time from 59.141840 seconds to 25.987238 seconds. Using the wordlist words.txt and the puzzle 50x50.grid.txt, the -O2 flag reduced the running time from 15.901401 seconds to 7.039078. In all the tested cases, the optimization flag reduced running time by over 50%.

4 Big Theta

If w is the number of words in the wordlist, r the number of rows in the puzzle, and c the number of columns in the puzzle, then the big theta running time of the program is:

$$\theta(r \cdot c \cdot w)$$

This is because for each letter in the puzzle a constant number of checks must be made to see if words are in the dictionary. For my implementation a check has a worst case running time that's linear with respect to w. Time taken to read in the puzzle and perform other operations can be ignored because it only modifies overall running time by a constant factor.

5 Problems

There were three primary problems that I encountered in this lab. The first problem was understanding how the hashTable data type should be structured. I was unsure what type should be used for the table itself, what type should be used for the secondary structures, and whether they should be stored as pointers or as objects. This was resolved primarily through trial and error.

The second problem involved memory leaks. My program was ceating memory leaks for a reason that I couldn't understand. It turned out that the deconstructor for my linked list was not functioning properly because at the time I had written it we hadn't gone over dynamic memory allocation.

The final problem was that the program would consistently segfault on puzzles of size larger than 4x7, but only when using words2.txt. This happened because my hash table was an array of pointers to linked lists and I hadn't initialized these pointers to NULL.

6 Shell Scripting

My shell scripting went pretty smoothly. Overall it seems like a useful technique for performing repetitive tasks involving multiple programs that might be difficult to interact with from a language like c++.