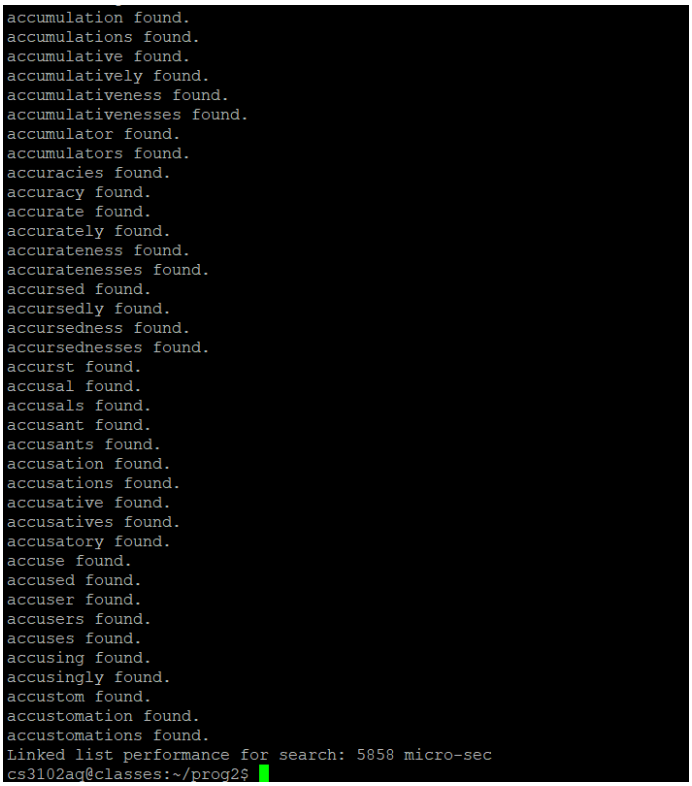
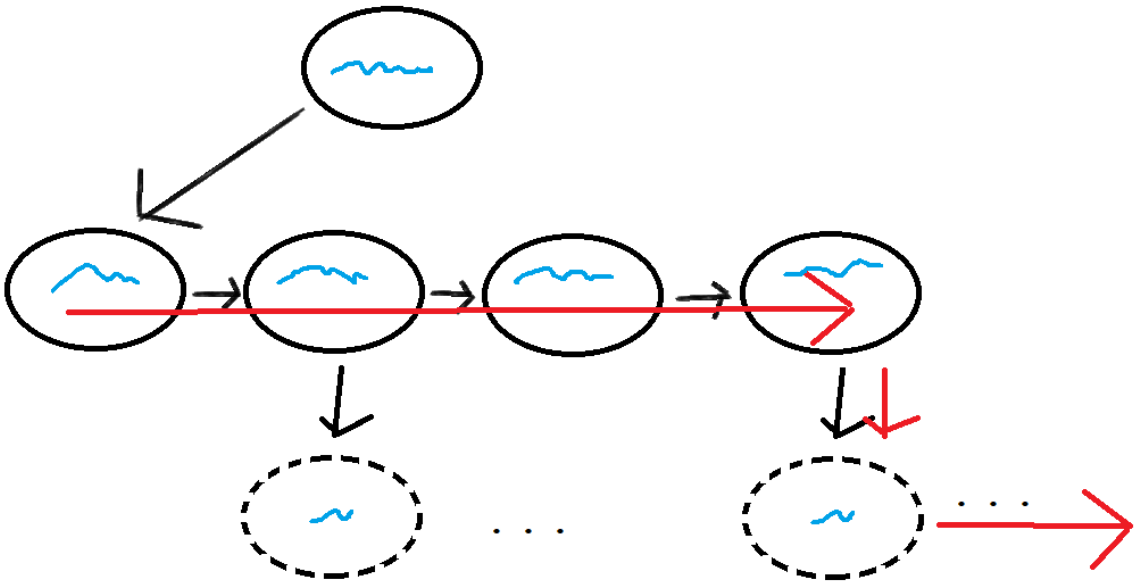
Compressed Trie Linked List Search VS Compressed Trie Hash Search

We implemented a compressed trie for our program and created a search function that would search linearly through linked lists on each level to find the word to be found. When we calculated the total time to find 1000 words, it took approximately 5858 microseconds.

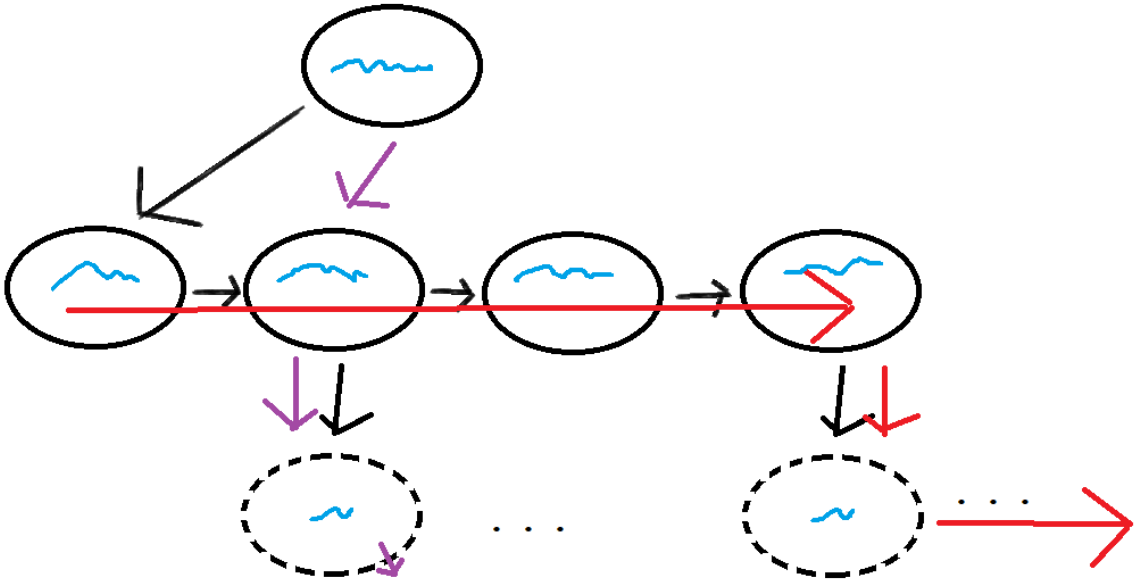


Next, we needed to implement a compressed trie, but this time, rather than searching through linked list, we needed to use a hash table. Although we attempted at implementation, we ran out of time. However, by our prediction, we believe that using a hash table and hashing each entry would definitely speed up the time.

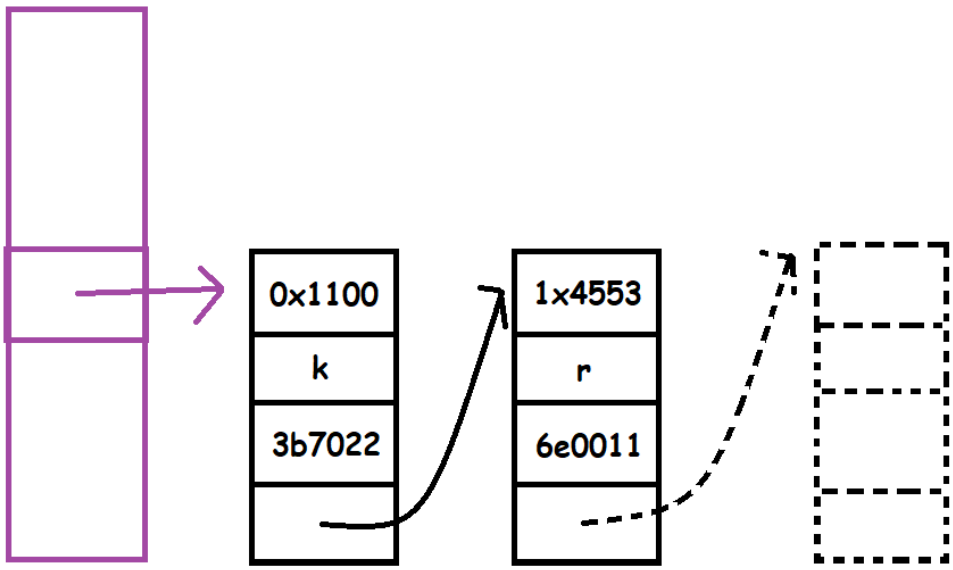
The red arrow indicates how search would run if it was searched by linked list:



However, the purple arrow indicates how search would run if it was searched by hashing:



Rather than linearly going through the linked list until it needs to go to the next level, hashing the tree would allow us to immediately "jump" to the child we want, and continue to jump child to child until we find the correct word. In each hash entry, we store the key (memory address), the leading character, the data (the child's memory address), and a pointer to the next node that would be there if there was a collision at that spot. Through hashing, we can create "shortcuts." Our hash entry would look like this:



In conclusion, although we were unable to run a complete accurate performance between the two searches, we were able to conclude that the hashing search would definitely speed up the time to find a certain word in the trie, compared to the linked list.