## 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 and a search function that would creating hashing "shortcuts" using a hash table. When we calculated the total time to find 1000 words, it took approximately 2990 microseconds for linked list and approximately 824 microseconds for the hash table.

```
acculturation found.
acculturational found.
acculturations found.
acculturative found.
accumulate found.
accumulated found.
accumulates found.
accumulating found.
accumulation found.
accumulations found.
accumulative found.
accumulatively found.
accumulativeness found.
accumulativenesses found.
accumulator found.
accumulators found.
accuracies found.
accuracy found.
accurate found.
accurately found.
accurateness found.
accuratenesses found.
accursed found.
accursedly found.
accursedness found.
accursednesses found.
accurst found.
accusal found.
accusals found.
accusant found.
accusants found.
accusation found.
accusations found.
accusative found.
accusatives found.
accusatory found.
accuse found.
accused found.
accuser found.
accusers found.
accuses found.
accusing found.
accusingly found.
accustom found.
accustomation found.
accustomations found.
Linked list performance for search: 2990 micro-sec
HSHtrie performance for stackSearch: 824 micro-sec
```

Concept

The red arrow indicates how search would run if it was searched by linked list:
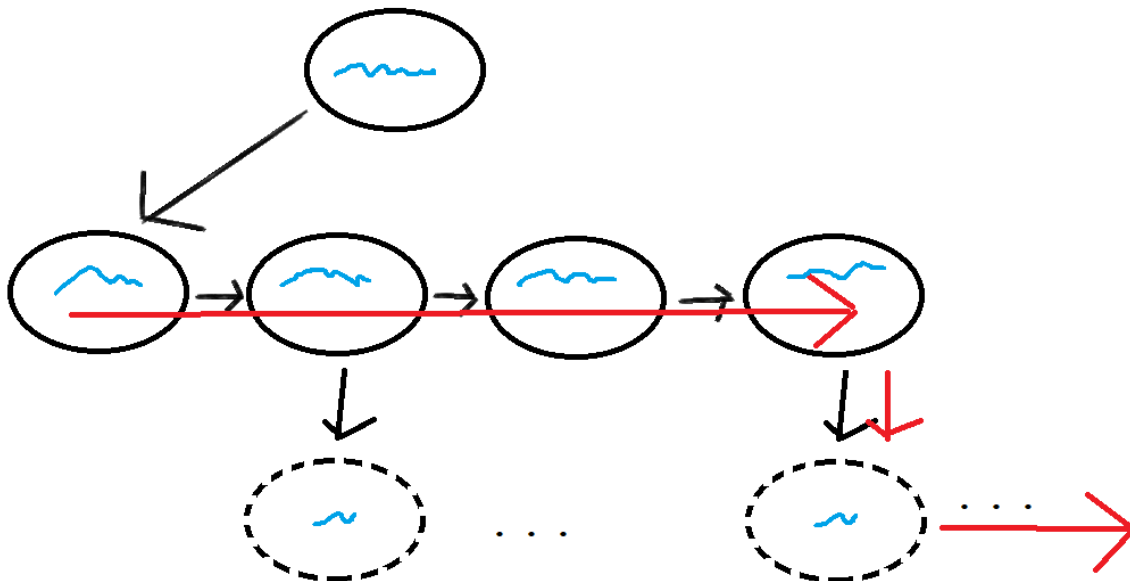
The purple arrow indicates how search would run if it was searched by hashing:
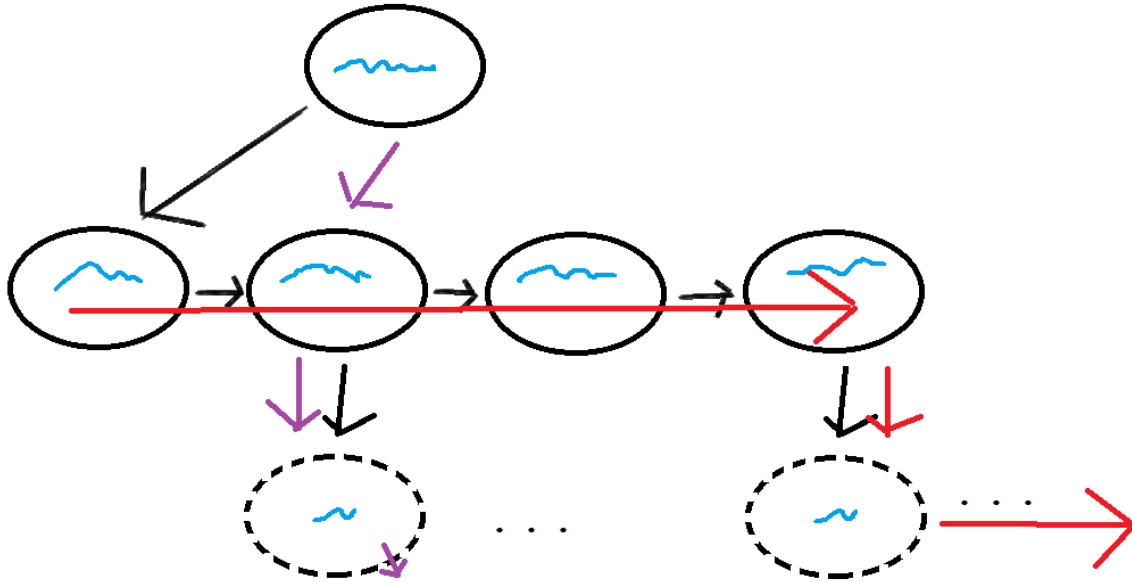
Rather than linearly going through the linked list until it needs to go to the next level, hashing the tree would allow us to immediately "jump" to the child we want, and continue to jump child to child until we find the correct word.

In conclusion, 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.