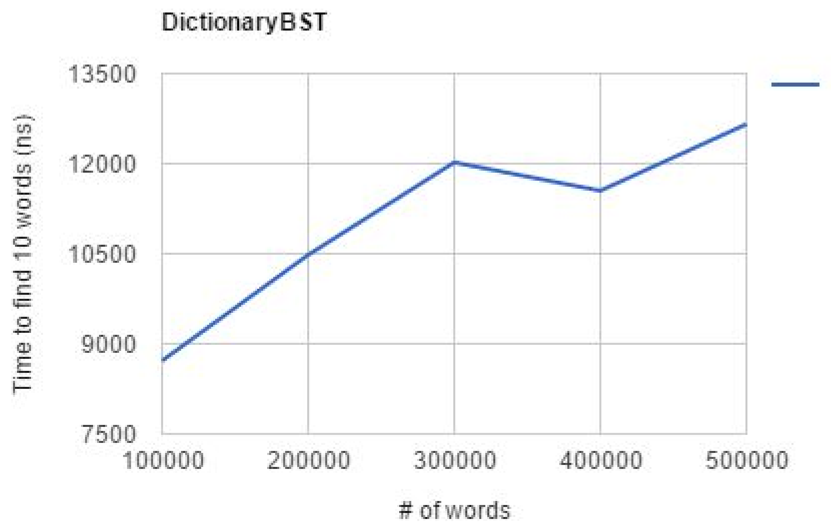
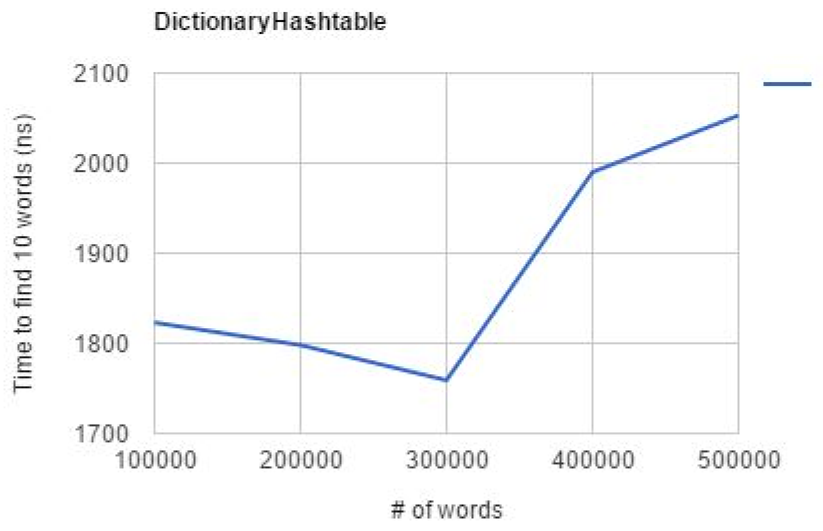
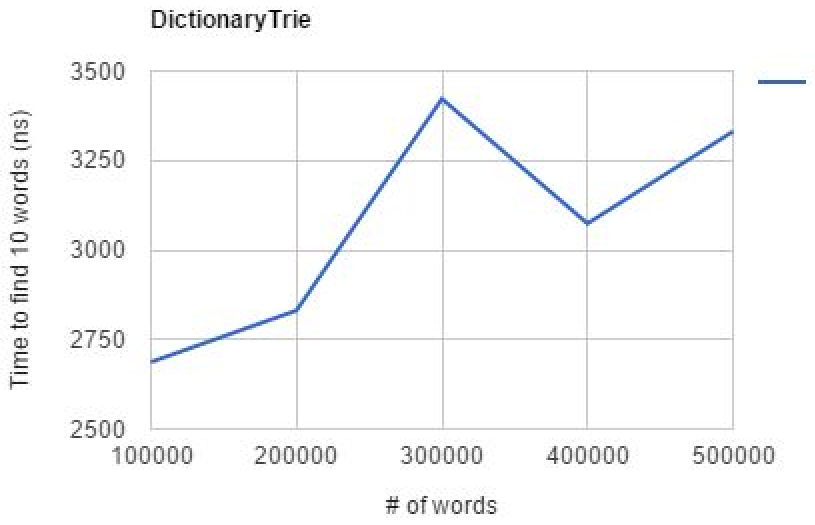
Connie Su
02/17/16

Final Report







1. For DictionaryBST's find method, the running time I expect to see is O(logN) because, it compares the word with other nodes log(N) times. For DictionaryHashtable's find method, the worst running time I expect to see is O(N) and the average running time is O(1) because, with few words, most words are in different indices while some of time are with other words in indices and therefore the running time would be O(1). But as you keep inserting words, more words are going to be in the same index as another word, which would then take more time to find the word. For DictionaryTrie's find method, the running time I expect to see is O(D) because you just traverse the down the trie by the letters of the words, no matter how many words are in the trie.

2. The results are what I expected because for DictionaryHashtable, in the beginning, it is pretty fast until it gets to over 300000 words, it becomes more slower. This means that more words are in the same indices as other words, taking more time to find the words. For DictionaryBST, it takes the longest time since it have to compare the other words log(N) times. For DictionaryTrie, it is much faster than DictionaryBST since it find words by traversing through trie using characters.

3. To implement the predictCompletions method, I first go through the trie using the given prefix. Then, I create a queue and push in the current node. Next, I get the top element of queue and remove it. Then, I check if the element have a word. If so, I create a pair with the word and its frequency into a priority queue. I then push in the element's children into the queue regardless if the element have a word or not. I repeat the process of getting top element and pushing children until the queue is empty. I then push in the words from the priority queue into the vector and return the vector.

I think the worst running time is O(N*D) because I have to go through every node after traversing through the trie using the prefix. Each node is part of a word hence have to go through N*D nodes.