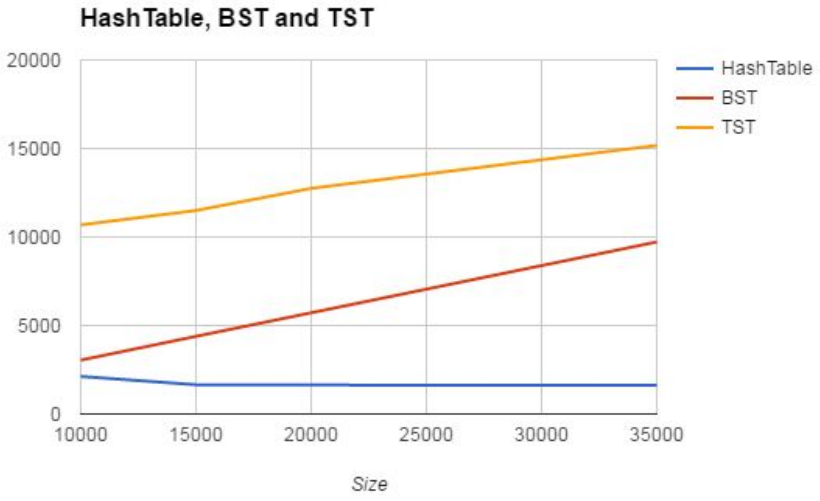

1.
The TST runtime is O(D).
The Hashtable search time would be O(1) on average and O(n) worst case.
The BST would be O($log_2 n$) . Instead of storing character by character in node, it stores the entire phrase into node.
2.
The hashtable is relatively constant. The BST looks very similar to log(n). The TST result however seem different.
3.
The predict completion algorithm has two steps: 1 is to find if the prefix exists and what the address of the last character of the prefix is. The 2nd part is if the prefix is found, to use a restricted version of the Breadth First Search on the subtrees of the node of the last character of the prefix.

This breadth first search involves 2 priority queues and 1 queue.
The 1st priority queue minHeap is a minheap that stores a pair that is a string and an int; the string is the word, the int is the frequency.
The 2nd priority queue is a maxheap that stores a pair that is a string and an int.
Most is an indicator of the subtrees largest frequency.

The search goes to the middle child of the prefix node, because the left and child node are not completions. Then it traverses through the tree, adding nodes to the queue and words to minHeap when there is an indicator that you've reached a word There are restrictions on the traversal. For an instance, if the minHeap reached the size of num_completions and most

indicator of a node is less than the value at the top of the priority queue, that node isn't pushed onto the queue.When a word termination key is found, there is also a check against the minHeap priority queue when the minHeap is the size of num_completions. When the minHeap reaches num_completion size, it pops off the minimum value node.
At the very end, the priorit queue PQueue is used to reverse the elemetns of minheap and load them onto the words vector.