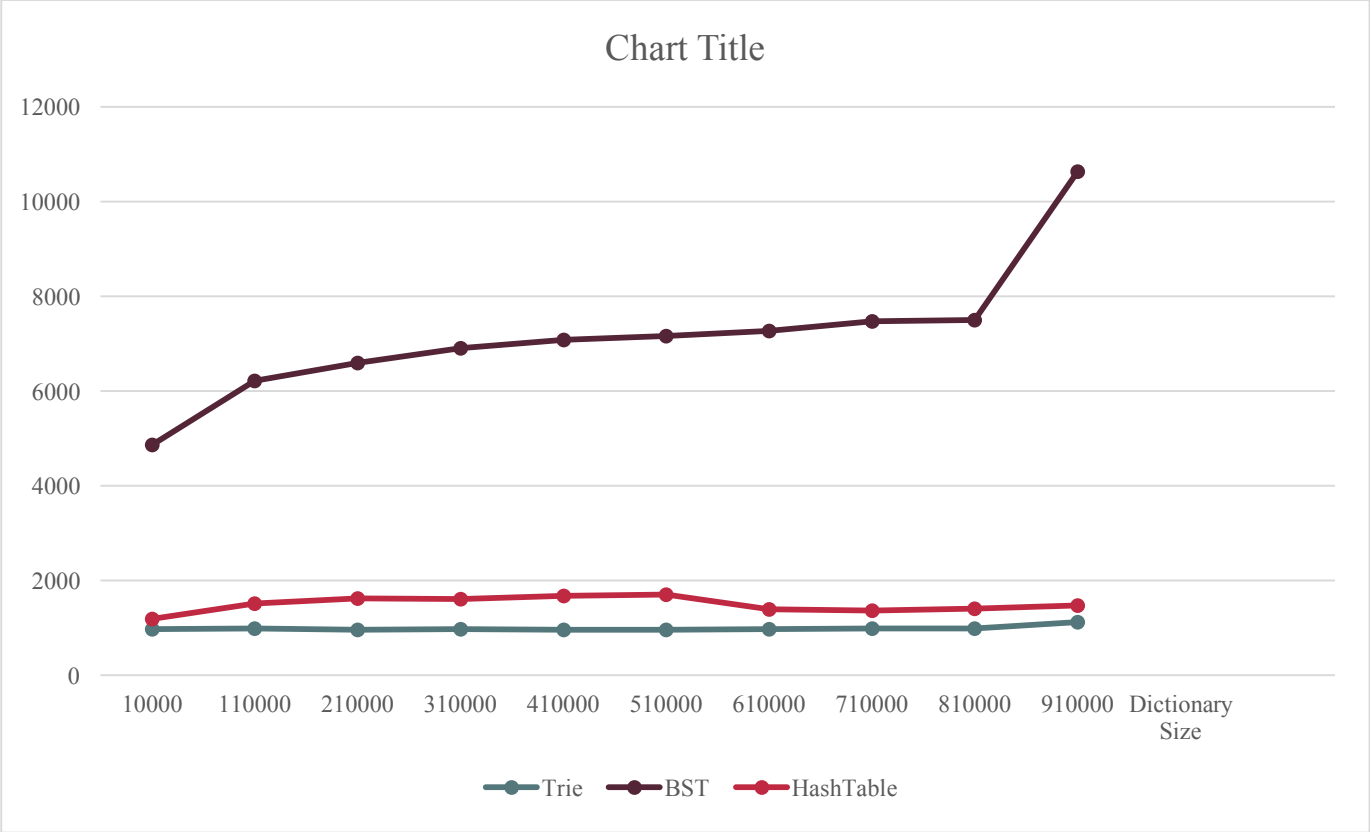
Junxi Li, Lingfei Lu

Final Report



1. For Find Method,
   i) In BST, since we have N unique elements in the BST, finding an element needs O(logN) time.
   ii) In HashTable, since we have N unique elements in HashTable, average case of HashTable would be O(1).
   iii) In Trie, since the longest word length is D, then we expect O(D).

2. Yes. First, since number of element increases as iteration, log(N) would be greater. And BST graph is an increasing curve.
   Second, since O(D) is approximately O(1), we have similar curve between HashTable and Trie. They are both horizontal lines as input size increases.

3. Our predictCompletions algorithm first loops the size of prefix to go to the multiway_node that correspond to the last character of prefix. And then create a max heap called frontier which is used to store the multiway_node that should be explored and a min heap called loader which is used to store the multiway_nodes that have been explored. The size of loader always equal to num_completions. Once loader's size is greater than num_completions pop the top one which is the smallest frequency. We push the top node of frontier to loader and then add the child of the top node to frontier. If frontier is empty and loader's size is smaller than num_completions or the top node of loader is smaller than the top node of frontier.
   Analysis: Define D as the length of the prefix. It will take O(D) to get to the prefix. And assume the worst case as we have a full, balanced Trie. And we assume we have to iterate every node to get the predict completions, assumed num_completions is big. Then also assume the longest length of the word is D', not including the prefix length. Then we have O(27^D'). Therefore, combined in the WORST CASE, we have O(D + 27^D')