CSE 100 PA3 Final Report

Yihong Zhang

Hao Gai

Graph Plotting

freq2.txt	6000	16000	26000	36000	46000	56000	66000	76000
DictionaryTrie	1154	1001	1003	1006	1119	1000	1285	1121
DictionaryBST	7621	8996	9450	8932	9271	9700	9639	10253
DictionaryHashtable	2199	2051	2763	2799	2249	2465	2261	2082
Min_size=6000,Step_Size=10000	86000	96000	106000	116000	126000	136000	146000	
	1004	1465	1008	1012	1003	1220	1108	
	10071	10062	9971	10548	10680	10373	10465	
	2475	2473	2789	2709	3518	2375	1987	

shuffled_freq_dict.txt	6000	26000	46000	66000	86000	106000	126000	146000
DictionaryTrie	1138	1139	1148	1297	1141	1270	1260	1137
DictionaryBST	7511	8693	7588	8814	8628	8898	8880	9514
DictionaryHashtable	2460	2251	2088	2110	2326	2347	2771	2196
Min_size=6000,Step_Size=20000	166000	186000	206000	226000	246000	266000	286000	
	1138	1143	1287	1234	1144	1146	1139	
	9720	9878	9929	10198	10355	11593	10429	
	2192	2249	2108	2268	2236	2205	2109	

freq_dict.txt	6000	46000	86000	126000	166000	206000	246000	286000
DictionaryTrie	1006	1010	1014	1567	1012	1766	1012	1013
DictionaryBST	9358	9845	9859	9457	10757	10153	10286	10213
DictionaryHashtable	2403	2458	2070	2139	2319	1987	2004	2017
Min_size=6000,Step_Size=40000	326000	366000	406000	446000	486000	526000	566000	
	1385	1005	1250	1273	1014	1012	1270	
	10735	10621	10701	11157	12593	11148	12569	
	2382	2109	2114	2311	2665	2753	2826	

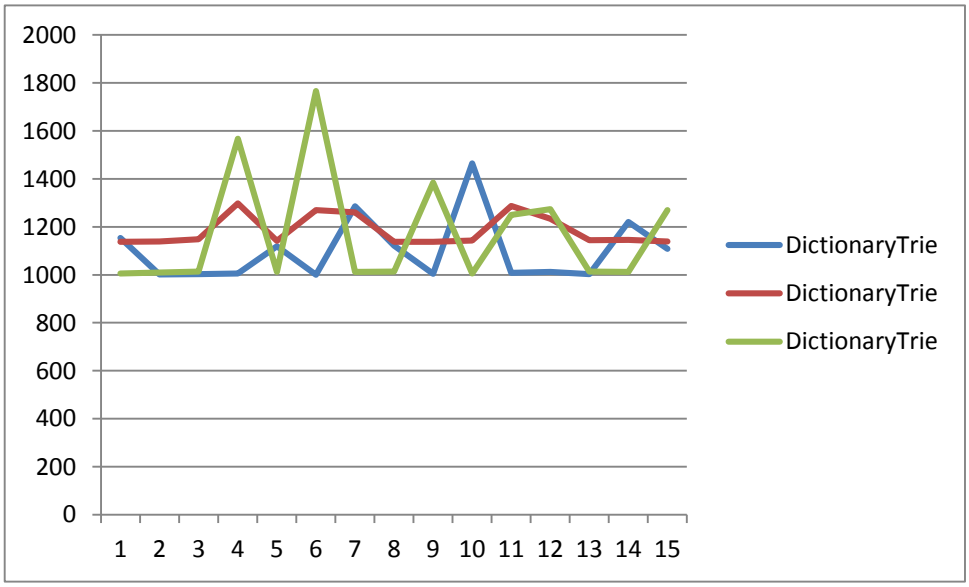

Figure 1 Worst-case find time of TST

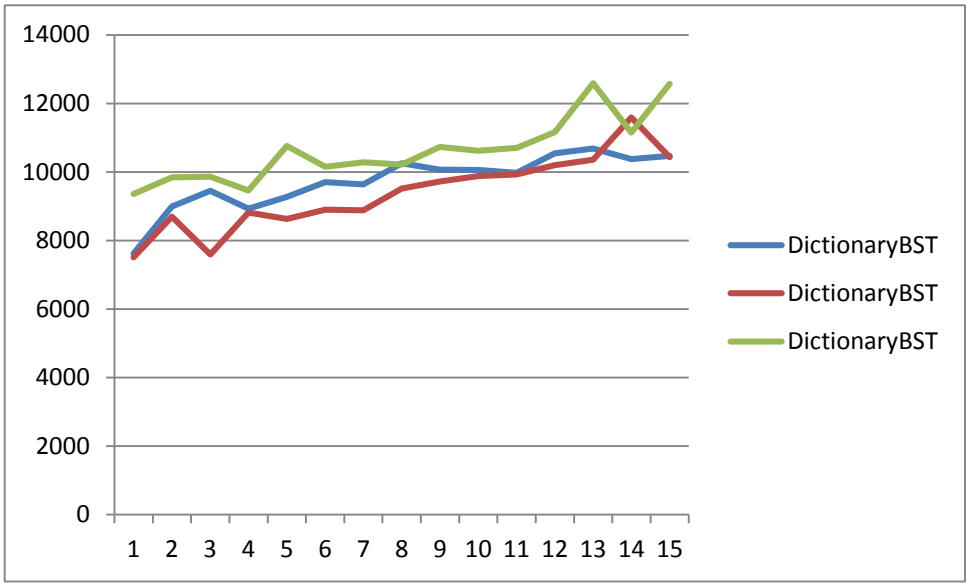

Figure 2 Worst-case find time of BST

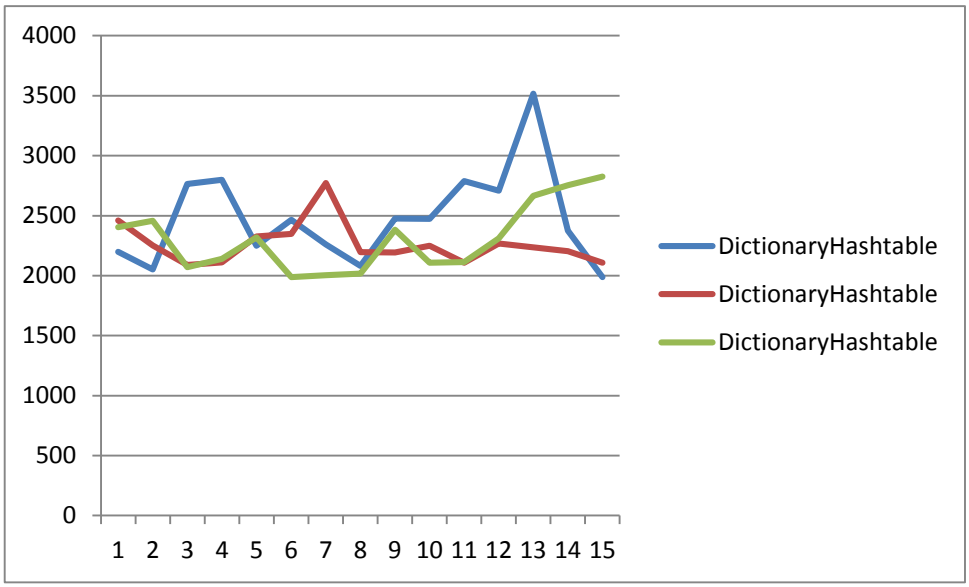

Figure 3 Worst-case find time of hash table

1、 What running time function do you expect to see for the find method in each of your three dictionary classes as a function of N (number of words), C (number of unique characters in the dictionary) and/or D (word length), and why?

	find
DictionaryBST	O(logN)
DictionaryHashtable	O(1)
DictionaryTrie(TST)	O(D)

The average time of find in BST depends on the average depth of BST which is log(N). The average time of find in hash table is O(1), and the average time of find in TST depends on the depth of TST which is actually the word length.

2、 Are your results consistent with your expectations? If yes, justify how by making

reference to your graphs. If not, explain why not and also explain why you think you did not get the results you expected.

Yes. We tested three dictionaries and used different min_size and step_size to get a better result. It is obvious that the worst-case find time of TST and hash table are constant time, which explains their find time has no relationship with N, while the worst-case find time of BST is slowly increasing with N gets bigger, which explains the log(N) find time of BST.

3、 Explain the algorithm that you used to implement the predictCompletions method in your dictionaryTrie class. What running time function do you expect to see for the predictCompletions as a function of N (number of words), C (number of unique characters in the dictionary) and/or D (word length), and why?

We used Depth first search in this dictionary class.

First of all, we locate the last letter of prefix in the TST tree, and recursively explore the middle child of this node. The exploring procedure is implemented using depth first search on the TST tree rooted from the node we found. Whenever we found a proper string, we push this string into a priority-queue, sorted by frequency of strings. Finally, we pop the desired number of strings, to "words".

The run-time of this function should be $O(d) + O(NlogN) = O(NlogN)$.