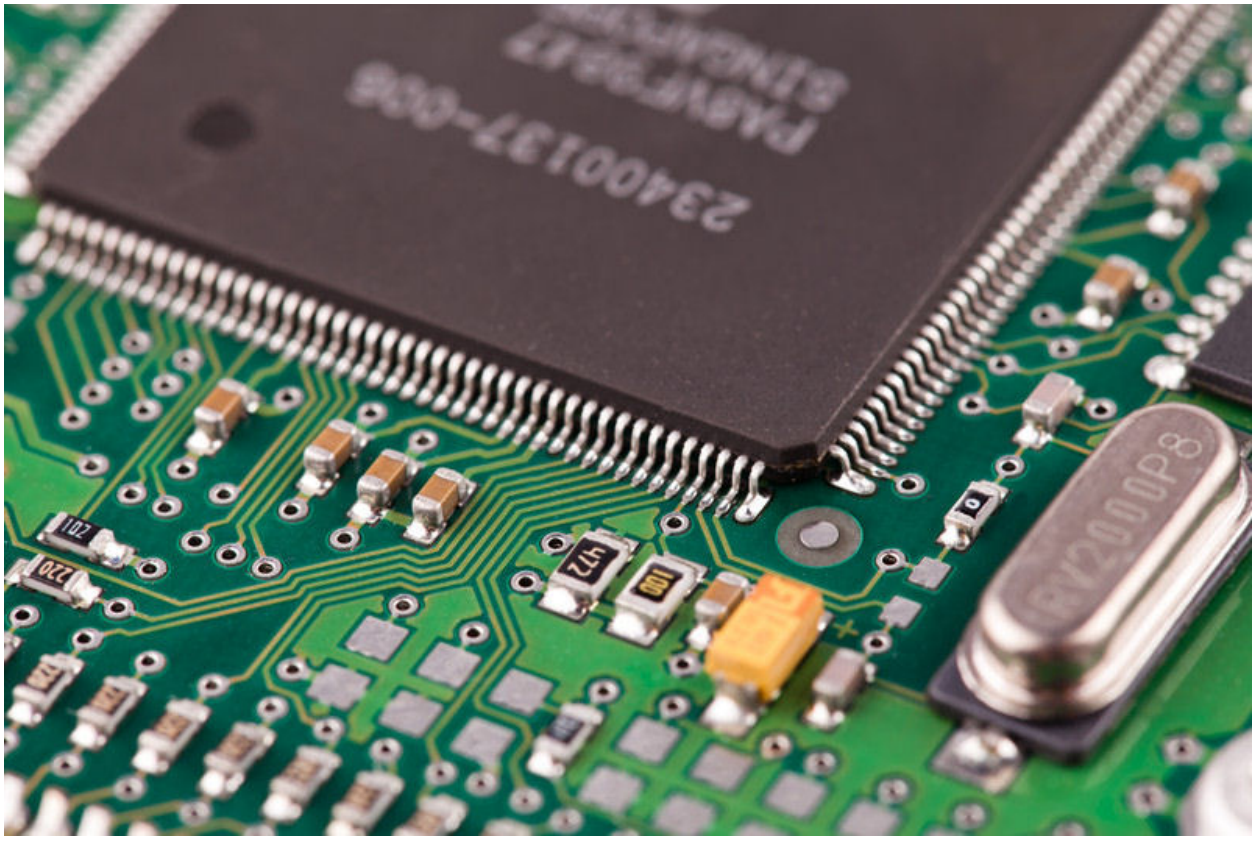


Hash Tables & Linked Lists

Comparison of Dictionary Implementations

Jack Webster
H00154950

Program Specification:

The program, which compiles and runs through the class Spell.java, is a spellchecker which implements a dictionary through hash table and linked list data structures. The purpose of the program is to compare the speed at which each data structure performs insertion and find functions; first inserting a dictionary and then using it to find suggestions for incorrectly spelled words.

IDE Environment:

The program was developed using Eclipse Luna IDE on Windows 8 64-bit.

Comparison:

Each data structure, hash table and linked list, were given six dictionaries to insert, d1 to d6. Each dictionary contains a different number of entries, the smallest and largest being d1 and d6 respectively. A text file containing incorrect spellings of words was used to test both the spell checking capabilities of the program as well as the time taken to find the suggested correct spellings, this text file was not altered throughout testing and remained constant.

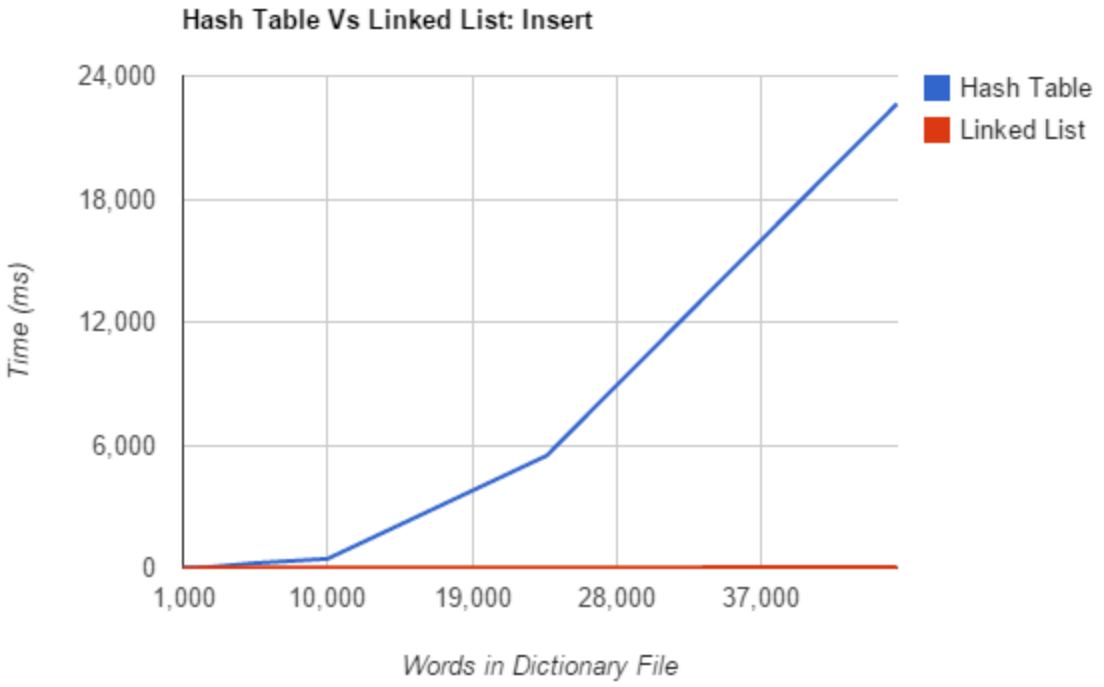
The program was executed a total of five times with each data structure for each dictionary, totalling ten executions per dictionary and sixty executions overall. Averages of insert and find times were calculated and used as data inputs for the graphs.

Tables of Test Results

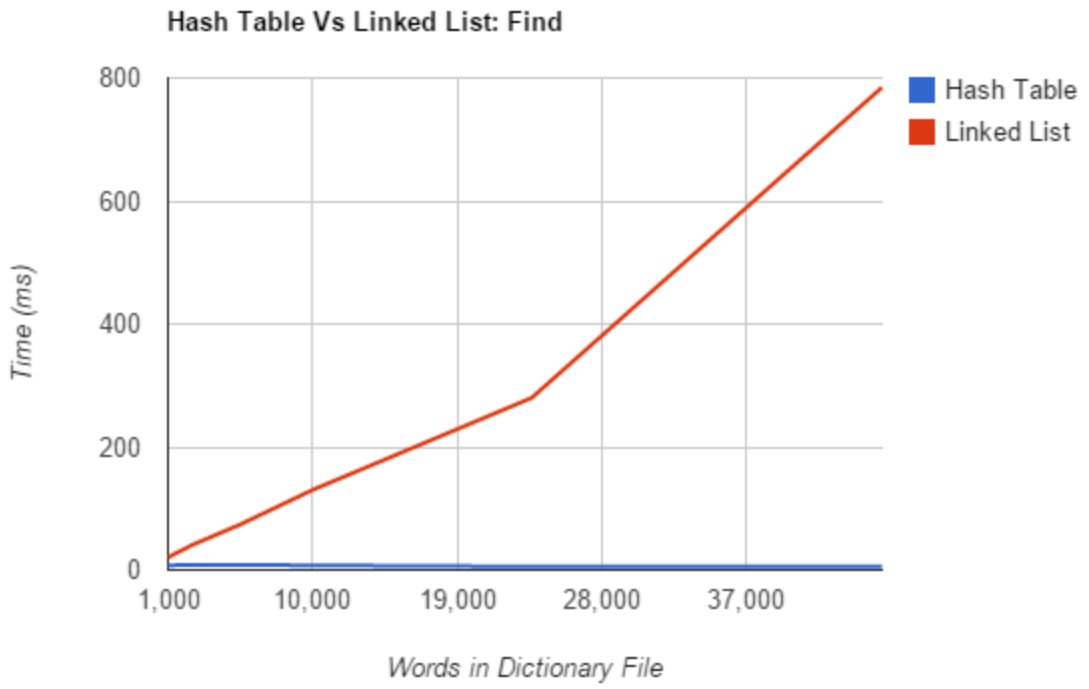
	HashDictionary		ListDictionary	
	insert (ms)	**find (ms)**	**insert (ms)**	**find (ms)**
d1	19	7	6	22
	22	7	6	21
	20	8	6	20
	19	8	7	19
	19	8	6	20
Average	**19.8**	**7.6**	**6.2**	**20.4**
d2	35	8	15	40
	35	8	14	40
	35	8	15	40
	35	8	17	43
	34	8	14	40
Average	**34.8**	**8**	**15**	**40.6**
d3	236	8	24	75
	239	8	25	74
	236	9	24	75
	236	8	24	73
	236	9	23	73
Average	**236.6**	**8.4**	**24**	**74**

	HashDictionary		ListDictionary	
	insert (ms)	**find (ms)**	**insert (ms)**	**find (ms)**
d4	447	8	27	130
	451	8	27	130
	448	7	28	130
	451	7	26	132
	450	7	28	129
Average	**449.4**	**7.4**	**27.2**	**130.2**
d5	5455	6	32	273
	5501	6	33	283
	5497	6	33	273
	5489	5	34	286
	5455	5	34	284
Average	**5479.4**	**5.6**	**33.2**	**279.8**
d6	22516	5	40	771
	22676	6	40	789
	22750	6	41	853
	22574	7	40	741
	22702	6	40	769
Average	**22643.6**	**6**	**40.2**	**784.6**

Graphs of Test Data



Here we see that as the words in the dictionary file increase the insert times for both data structures also increase. Due to the large insert time of the hash table implementation the graph is not a good representation of the increasing insert times of the linked list implementation. The hash table implementations insert times increase dramatically compared to that of the Linked List, this is believed to be caused by the rehash() method which the class HashDictionary.java implements; causing the whole hash table to be moved and rehashed multiple times throughout the insertion process.The linked list has a more linear increase in insertion time but stays relatively short, due to not having to adjust to an increasing number of entries.



Here we can see that as the linked list implementation has increasing times for the find process the times for the hash table implementation appear to decrease. As the linked list increases the amount of entries stored the time to complete the find process increases, this is due to the to the structure of the linked list and the program having to traverse the whole list to find the correct entries.

The hash table does not have this problem however as each entry in the data structure is stored via a unique key which can be calculated and found substantially quicker. It is thought that as the number of entries increase, which in turn increases the number of spaces in the hash table, there are less collisions and therefore less instances of the value needing to be double hashed. This would explain the drop in execution time while there are more entries.