

654 Advanced Computing Concepts

Assignment1 Report

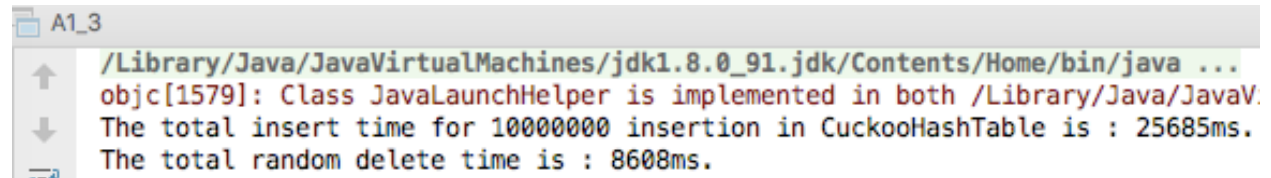
I confirm that I will keep the content of this assignment confidential. I confirm that I have not received any unauthorized assistance in preparing for or writing this assignment. I acknowledge that a mark of 0 may be assigned for copied work. **Tengxiaoyao (Tab) Tu, #104518447**

Q1. Within a Java class, write a method that creates n random strings and inserts them in a hash table. The method should compute the average time for each insertion.

Q2. Write another method that finds n random strings in the hash table. The method should delete the string if found. It should also compute the average time of each search.

In this case, the length for all random String is 10.

ANSWER FOR TASK 1-2:



```
A1_3
/Library/Java/JavaVirtualMachines/jdk1.8.0_91.jdk/Contents/Home/bin/java ...
objc[1579]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaV.
The total insert time for 10000000 insertion in CuckooHashTable is : 25685ms.
The total random delete time is : 8608ms.
```

It cost 256685ms to insert 10000000 numbers into CucokkHashTable, and cost 8608ms to do 10000000 times deletion.

Q3. Repeat #1 and #2 with $n = 2i$, $i = 1, \dots, 20$. Place the numbers in a table and compare the results for Cuckoo, QuadraticProbing and SeparateChaining. Comment on the times obtained and compare them with the complexities as discussed in class.

ANSWER FOR TASK 3:

For these tasks, I Insert 2,000,000 random numbers into 3 different hash table.

The result is the average in 20 times without the maximum and minimum for each time.

All results are record in String format, and transfer to an Excel file.

The length for the random String is 10 which include all English letters.

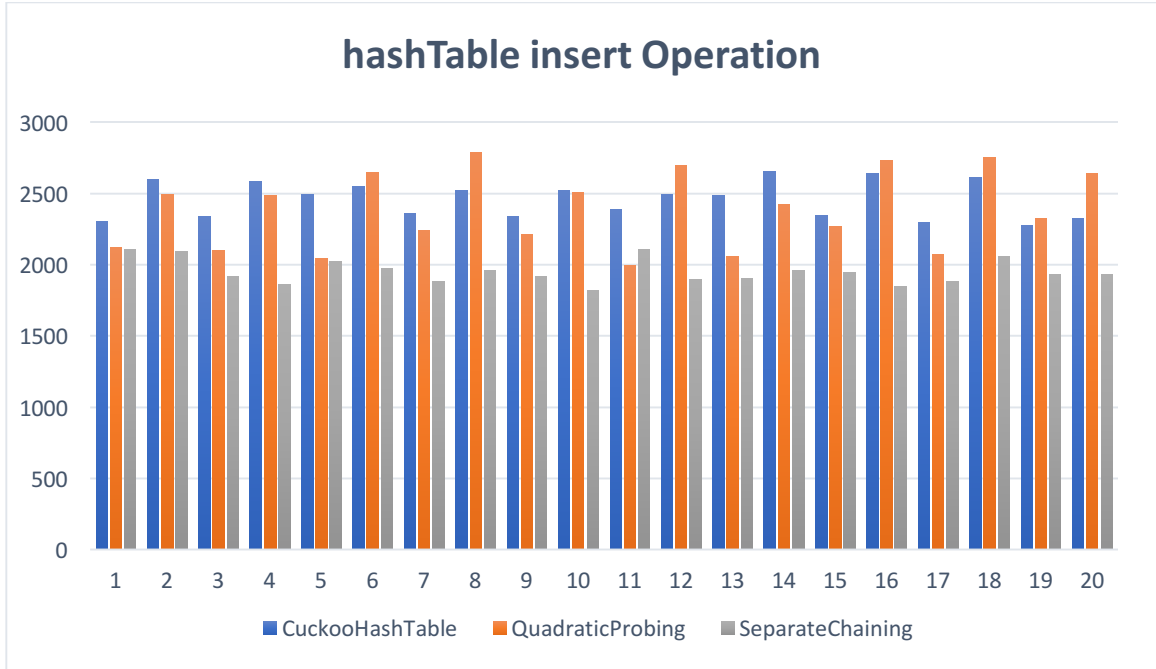


Figure 1. Compare Average Time Spend for Each Insertion

i	Cuckoo(ms)	QuadraticProbing(ms)	SeparateChaining(ms)
1	2304.94	2119.72	2109.78
2	2600.06	2495.94	2094.28
3	2339.61	2094.72	1915.61
4	2586.89	2488.83	1860.56
5	2491.06	2049.61	2024
6	2545.89	2645.94	1975
7	2355.39	2242	1883.11
8	2519.5	2789.33	1956.89
9	2341.17	2207.89	1919.44
10	2524.5	2503.56	1818.56
11	2389.11	1992.06	2107.67
12	2495.89	2694.94	1893.83
13	2486.22	2062	1901.78
14	2654.33	2421.11	1955.56
15	2345.72	2269.5	1945.33
16	2641.89	2730.5	1844.33
17	2298.11	2068.28	1884.44
18	2615.11	2752.17	2058.06
19	2276.39	2320.44	1931.06
20	2324.94	2639.39	1933.89
Average	2456.836	2379.3965	1950.659

Each Insertion	0.001228418	0.001189698	0.00097533
----------------	-------------	-------------	------------

Table 1. Compare Average Time Spend for Each Insertion

Cuckoo spend 0.001228418ms for each time in average;

QuadraticProbing spend 0.001189698ms for each time in average;

SeparateChaining spend 0.00097533ms for each time in average.

As a result, SeparateChaining is faster than QuadraticProbing than Cuckoo.

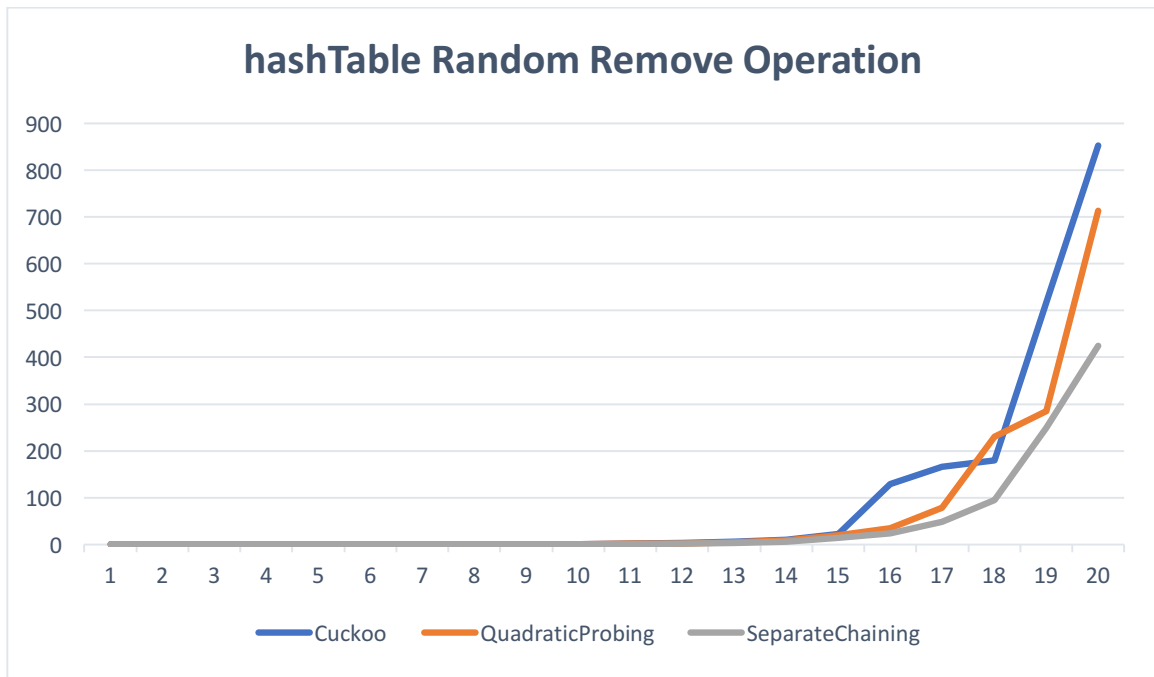


Figure 2. Compare Average Time Spend for Random Numbers in Each Searching

i	Cuckoo	QuadraticProbing	SeparateChaining
1	0	0	0
2	0	0	0
3	0	0.06	0
4	0	0	0
5	0	0	0
6	0.11	0.06	0
7	0.17	0	0.06
8	0.11	0.17	0.06
9	0.44	0.28	0.11
10	0.94	0.67	0.44
11	1.39	1.17	0.94
12	3.06	2.22	1.72
13	5.39	4.39	3.17

14	10.33	8.67	6.28
15	22.94	18.94	14.67
16	128.56	34.33	23.44
17	166.06	78.17	48.61
18	180.28	230	94.33
19	517.89	284.44	249.06
20	853.11	712.67	424.78

Table 2. Compare Average Time Spend for Each Random Searching

For large searches, SeparateChaining is faster than QuadraticProbing than Cuckoo.

Q4. Use the Java classes BinarySearchTree, AVLTree, RedBlackBST, SplayTree given in class. For each tree:

- Insert 100,000 integer keys, from 1 to 100,000 (in that order). Find the average time for each insertion.
- Do 100,000 searches of random integer keys between 1 and 100,000. Find the average time of each search.
- Delete all the keys in the trees, starting from 100,000 down to 1 (in that order). Find the average time of each deletion.

Q5. For each tree:

- Insert 100,000 keys between 1 and 100,000. Find the average time of each search.
- Repeat #4.b.
- Repeat #4.c but with random keys between 1 and 100,000. Note that not all the keys may be found in the tree.

Q6. Draw a table that contains all the average times found in #4 and #5. Comment on the results obtained and compare them with the worst-case and average-case running times of each operation for each tree. Which tree will you use in your implementations for real problems? Note: you decide on the format of the table (use your creativity to present the results in the best possible way).

ANSWER FOR TASK 4-6:

Label “_R” stand for Random;

The result is the average in 10 times without the maximum and minimum for each result.

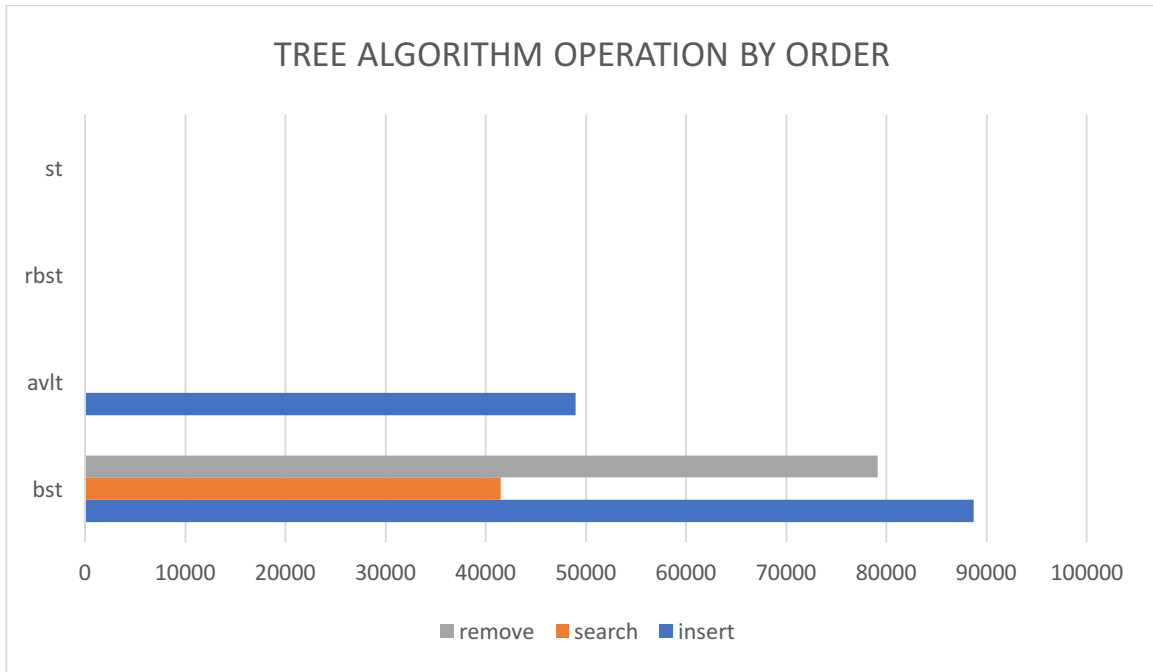


Figure 3. Compare 100000 Operation Time Spend in Average for BinarySearchTree, AVLTree, RedBlackBST, SplayTree

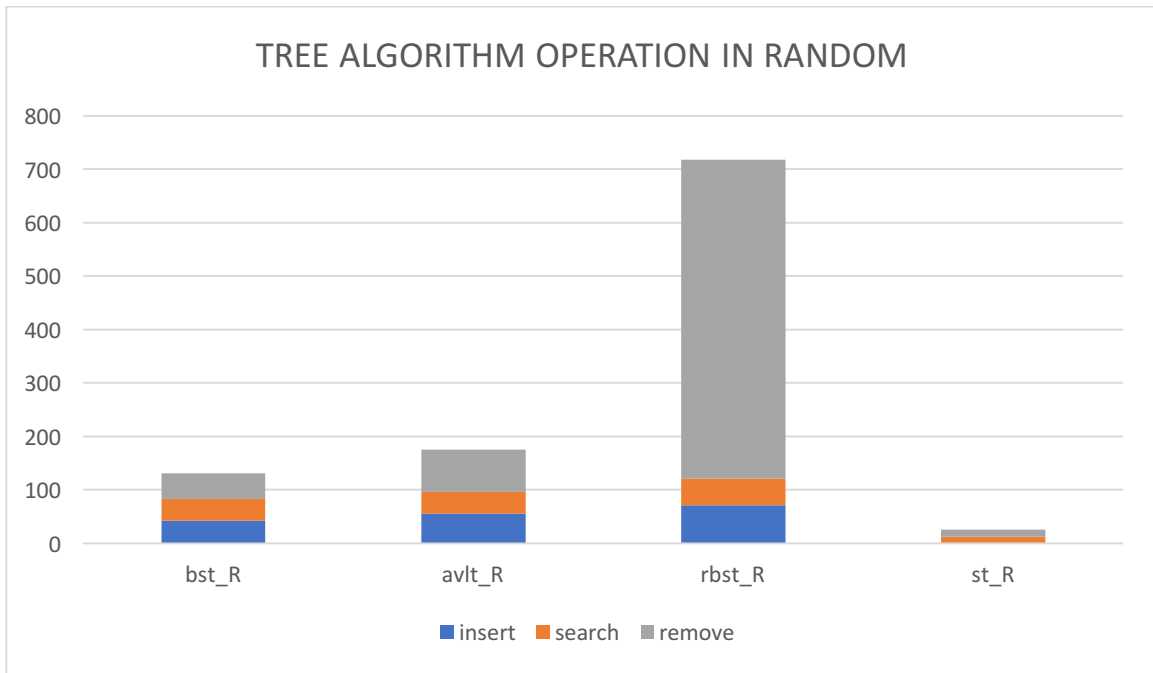


Figure 4. Compare 100000 Operation Time Spend in Average for Random Numbers in Each Tree Searching

ALGORITHM	INSERTION	SEARCHING	DELETION
bst	88714.375	41464.75	79145.375
avlt	48951.125	48.875	29.375

rbst	46.5	17.125	71.375
st	13.625	63.375	20.5
bst_R	42.75	40.25	47.75
avlt_R	55.75	40.375	79.25
rbst_R	71.375	50.125	596.75
st_R	0	12	14

Table 3. Compare 100000 Times Spend in Average for Each Tree Operation

In Task 4-6, each insertion spent time is log in the table follow.

	BinarySearchTree	AVLTree	RedBlackBST	SplayTree
INSERTION BY ORDER	0.88714375	0.48951125	0.000465	0.00013625
INSERTION IN RANDOM	0.0004275	0.0005575	0.00071375	0
SEARCHING BY ORDER	0.4146475	0.00048875	0.00017125	0.00063375
SEARCH IN RANDOM	0.0004025	0.00040375	0.00050125	0.00012
DELETION BY ORDER	0.79145375	0.00029375	0.00071375	0.000205
DELETION IN RANDOM	0.0004775	0.0007925	0.0059675	0.00014

Table 4. Compare Time Spend in Each Algorithm

As a result, in order insertion, SplayTree is faster than RedBlackBST, than AVLTree, than BinarySearchTree. In addition, BinarySearchTree and AVLTree spend a lot of time on insertion by order. Splay Tree is the fastest algorithm in random insertion, searching and removing, in these 4 tree algorithms.

OPTIONAL:

1. You are given a set of 100 HTML documents downloaded from the World Wide Web Consortium main web site (<http://www.w3.org/>). They are provided in the attached file called "W3C Web Pages.zip". The subfolder called "Text" contains the files in text format.
2. Write a program/method that given an HTML file, it computes the frequency of each word in the file and stores all words and frequencies in a hash table. The key for the hash table is the word. Use one of the hash tables discussed in the lab. Suggestion: use a string tokenizer to extract the words from the file, and consider only words of the 26-letter alphabet and 10 digits, ignoring lower/upper case.
3. Simple Web search engine: Write a program that given a set of keywords as input, it ranks the web pages based on their score (score = sum of matches per keyword, or keyword * frequency). The program should list the top 10 pages with the best matches. Suggestion: create

- a heap for storing the pages, where the “key” is the score of each page. Choose the top 10 pages from the heap as in the Selection problem. Test your program with a few keywords.
4. A Web dictionary: Write a program/method that takes all 100 HTML files as input, and creates a dictionary that contains all words along with their frequencies. Store the dictionary in a splay tree.
5. Some stats: Design an algorithm that takes the dictionary (splay tree) as input and lists the 10 most and least frequent words in decreasing order of frequency. Write an efficient algorithm that does not search the entire tree. Explain why your algorithm is efficient and its complexity. Write a program for your algorithm, run it and show the list of the 10 most/least frequent words.
6. The spellchecker: Write a program that takes an HTML file as input and checks whether or not each word in the file is the dictionary. Insert (in the dictionary) the words that are not found – provide a list of these words. Test it with real pages from <http://www.w3.org>.

Opt Answer:

Using HashMap to count the total words in the Dictionary.

Print out the most useful 10 words in the dictionary.

```
1- the : 37066
2- of : 16214
3- a : 14326
4- to : 13136
5- and : 12361
6- is : 11830
7- in : 9548
8- for : 7004
9- 1 : 5610
10- that : 5560
```

Figure 5 Words Count in Text Dictionary

The most 10 useful words are “the”, “of”, “a”, “to”, “and”, “is”, “in”, “for”, “1” and “that”.

Print out the 10 files which use the word is the most

```
1- /Users/Kevin/Documents/Workspace/Java/654Lab/W3C Web Pages/Text/XPath and XQuery Functions and Operators 3.0.txt: is : 11828
2- /Users/Kevin/Documents/Workspace/Java/654Lab/W3C Web Pages/Text/XMLHttpRequest Level 1.txt: is : 8571
3- /Users/Kevin/Documents/Workspace/Java/654Lab/W3C Web Pages/Text/XML Technology - W3C.txt: is : 8285
4- /Users/Kevin/Documents/Workspace/Java/654Lab/W3C Web Pages/Text/XML Signature Syntax and Processing Version 1.1.txt: is : 8272
5- /Users/Kevin/Documents/Workspace/Java/654Lab/W3C Web Pages/Text/XML Schema Datatypes in RDF and OWL.txt: is : 7847
6- /Users/Kevin/Documents/Workspace/Java/654Lab/W3C Web Pages/Text/XML Linking Language (XLink) Version 1.1.txt: is : 7674
7- /Users/Kevin/Documents/Workspace/Java/654Lab/W3C Web Pages/Text/XML Essentials - W3C.txt: is : 7461
8- /Users/Kevin/Documents/Workspace/Java/654Lab/W3C Web Pages/Text/XHTML Basic 1.1 - Second Edition.txt: is : 7437
9- /Users/Kevin/Documents/Workspace/Java/654Lab/W3C Web Pages/Text/World Wide Web Consortium (W3C).txt: is : 7322
10- /Users/Kevin/Documents/Workspace/Java/654Lab/W3C Web Pages/Text/Web Services Policy 1.5 - Framework.txt: is : 7311
```

Figure 5 Top 10 Files which include “is” the most

In optional tasks 3, **After spread words into hash table, the result could be found by sort the table easily. The result is showed in Table 5.**

top	File Name	Key Word	Times
1	XPath and XQuery Functions and Operators 3.0.txt	is	11828

2	XMLHttpRequest Level 1.txt	is	8571
3	XML Technology - W3C.txt	is	8285
4	XML Signature Syntax and Processing Version 1.1.txt	is	8272
5	XML Schema Datatypes in RDF and OWL.txt	is	7847
6	XML Linking Language (XLink) Version 1.1.txt	is	7674
7	XML Essentials - W3C.txt	is	7461
8	XHTML Basic 1.1 - Second Edition.txt	is	7437
9	World Wide Web Consortium (W3C).txt	is	7322
10	Web Services Policy 1.5 - Framework.txt	is	7311

Table 5. Compare Time Spend in Each Algorithm

In optional tasks 4 – 6, building a dictionary with SplayTree, we can easily find out some interesting result, such as the most used starting letter in all words, showed in Figure 7.

```
Build words dictionary with SpalyTree.
Word{word='0', freq=1926}
Word{word='00', freq=592}
Word{word='000', freq=22}
Word{word='0000', freq=15}
Word{word='00000000001', freq=1}
Word{word='0000000000100000000001', freq=1}
Word{word='00000001e0', freq=1}
Word{word='0000003', freq=8}
Word{word='000001', freq=1}
Word{word='00001111', freq=1}
Word{word='0001'. freq=3}
```

Figure 6 Screen Shot for Building the SplayTree for Words

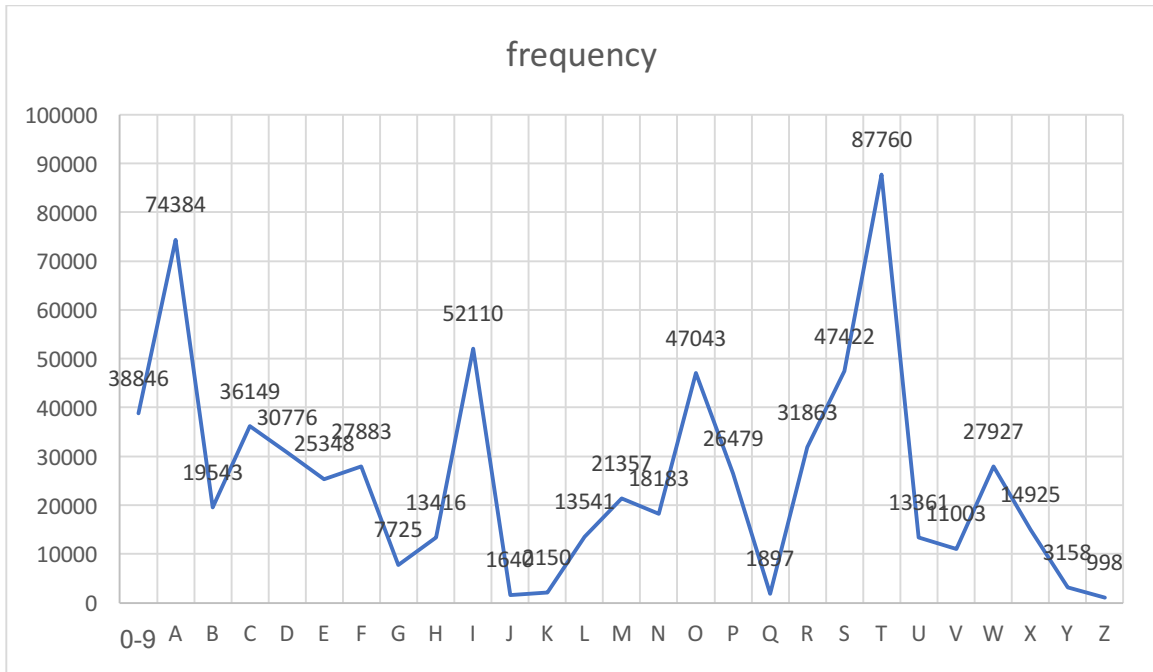


Figure 7. Showing the Frequency for each Word Starting Letter

As a result, most words are starting with the letter 't', and the second is 'a'.