Finding Word Frequency Using Java

Abstract:

The context of the real world problem I explored in this project was finding and analyzing word frequencies in large groups of text like Reddit comments and Shakespeare's works. To better understand the problem, I looked at different methods for searching word frequency and looked through the WordCounter class. The core CS concepts I used in this project were the MapSet interface, word frequency analysis, and time complexity analysis. My key findings were the differences between the Reddit comments and Shakespeare's work.

Results:

I ran 2 experiments which were BSTMap and HashMap through the MapSet interface. These were applied in the WordCounter class and found the frequency of words in 2015 Reddit comments and Shakespeare's works.

The metrics reported were the word frequency of different words in these large datasets. The metrics aimed to help the efficiency and performance of the program.

Shakespeare Word Frequency	**Reddit Word Frequency**
abandon - 4	aa - 773
abbey - 20	aal - 2
abbot - 9	aam - 4
abed - 9	aardvark - 5
abet - 1	aardwolf - 1
abandoned - 2	aba - 27
abase - 2	abaca - 1
abate - 14	aback - 41
abatement - 3	abacus - 5
abbess - 25	abalone - 2

Required analysis 1: The results don't match my expectations because I thought there would be more words that I didn't know or recognize.

Data Structure	Dataset 1 - time(ms)	Dataset 2 - time(ms)
Array	1200	1800
Hash Map	800	1000
Tree	1500	2200

Required analysis 2: This is the total time it takes different data structures to run in the buildMap section of the WordCounter class.

Required analysis 3: It helps because a higher maxDepth means that the tree is more unbalanced, which means that methods like search, insertion, and deletion don't work as well.

The results showed that HashMaps were the most efficient and that they worked better than arrays. It also showed that dataset 2 had a high maximum depth than dataset 1 for hashmaps, so the tree is most likely unbalanced.

Extensions:

I didn't do any extensions for this project.

Reflection:

1.

Dataset Size	Average Height
10	5.53
60	11.53
110	13.73
160	15.21
210	15.97
260	16.68
310	17.33
360	18.2
410	18.57
460	18.99

510	19.47
560	20.11
610	20.25
660	20.51
710	20.73
760	20.83
810	21.19
860	21.34
910	21.8
960	21.71

Yes it can happen in practice because the patterns of data could lead to the same results.

2. You could use a hash table structure, so you create an array of linked lists and then add elements to each list based on their hash values. Then, you can search within the list and then find the index of the elements. If they have the same indexes, you can put them in the same list. Then, you can resize the table as the load changes.

Acknowledgements:

I worked with Kama and went to TA hours for help. I also used chatGPT to debug the program.