Henry Chang Data Structure Lab5b

The binary search(bcheck) V.S. the linear method(lcheck)

Since we know that lcheck compares the input word with every word in the dictionary, its runtime should be $\sim N$. However, its run time also depends on the input heavily because the dictionary is sorted by their first character. On the other hand, the runtime for binary search should be ~ lgN as it cuts the dictionary in half each time until it finds (or not) the input word in the dictionary. The result matches these analysis:

- lcheck's runtime for zebra and not found ones are larger than that for bear.
- runtime: lcheck > bcheck

Case/Input	Found or not	Runtime for bcheck(s	Runtime for lcheck(s)
bear	Y	<0.001	0.001
jackalope	N	<0.001	0.002
liger	N	<0.001	0.002
zebra	Y	<0.001	0.002
Avg.	N/A	<0.001	0.00175

Recursive v.s sorting based method de-jumbler for solving anagrams

If the the length of the input jumbled word is N, the recursive method spent time for $N!$ swaps to generate all $N!$ different combinations of characters(assume no repeated character). While the sorting based method sorts the jumbled word with an average of $aNlgN$ array accesses into a sorted string of word with mergesort. Thus, the sorting based method would be faster since $aNlgN$ would be smaller than $N!$ once N becomes larger. However, since in my implementation, the words in the dictionary is not sorted by their characters to the advantage of the sorted based method (e.g. "and" sorted into "adn"), the comparison between the two de-jumbler methods is not complete fair and scientific. If every word in the dictionary to be searched is already sorted and indexed, the sorting based method would be even more faster than the recursive one, and as N grows, we expect to see even more significant difference between the two. These points mentioned above match the result:

Case	N	Runtime for Recursive(s)	Runtime for Sorting Based(s)
granama	7	0.630	0.195
nelir	5	0.268	0.115
gurpe	5	0.069	0.076
notair	6	0.429	0.094
bahcle	6	0.475	0.096
cat	3	0.002	0.074
Avg.	5.3	0.312	0.109