Homework 5 - Lists, Trees, and Tries

`LinkedList` vs `ArrayList` vs. `TreeSet`

Structure	Timing (s)	Memory (MB)
`ArrayList`	14.849	6.6804
`LinkedList`	18.8428	12.871496
`TreeSet`	0.009	15.9886

Conclusion

I would say that yes, this does make sense. Since `ArrayList` has to store only the index and the data, it would naturally take up less storage than `LinkedList` which would have to store two pointers to the next and previous elements. The logic also follows for the ArrayList to take up less time for adding because whenever an `ArrayList` gets to over half capacity, it doubles. This means that it is significantly faster to add because the `ArrayList` creates new memory space in chunks. `LinkedList` on the other hand has to spend time linking the nodes together, which contributes to the overhead, all while `ArrayList` just has to place them into a list. Finally, `TreeSet` would use more storage since it has to store more pointers but the same amount of data. As well, it would have to store the balancing information for `Red-Black Trees` since this is how Java implements `TreeSet`. This is where the speed of the `TreeSet` comes from; it has to have higher storage to keep the tree balanced. However, because of this, the adding happens almost instantly since adding for trees is `O(h)`.

`Trie`

Structure	Timing (s)	Memory (MB)
`Trie`	0.004	43.033528

Conclusion

I would say that the results for the `Trie` makes sense because a `Trie` would have to store quite a few different nodes and the structure is mired by a lot of overhead due to the sheer number of nodes and pointers that must be maintained. However, it is unbelievably fast because you only have to do work proportional to the length of the word, which means that it does not matter how many words are stored. The longest word in `dictionary.txt` is ~26 characters, indicating that adding is basically constant. This means that, when compared to other structures' timings, the `Trie` significantly outperformed both `ArrayList` and `LinkedList`. I think that it is worth pointing out that this consumed almost three times the storage but only shaved off 0.005 seconds when compared to the `TreeSet`.