RuntimeAnalysis

Sunday, February 19, 2023 4:12 PM

* printDictionary() Runtime:

b/c this method using a single for-loop, starting from 0, ending at the size of ArrayList wordList, (which we can assign to n), with each loop incrementing i by 1, we can say the big-O notation is O(n)

* SearchDictionary() Runtime:

this method has a single if-statment, however, b/c it calls binarySearch(), we cannot assume O(1) without calculating binarysearch()'s runtime first.

binarySearch() uses recursive calls, therefore we must calculate the number of recursive function calls.

```
private int binarySearch(String word, int low, int high) {
    int middle = (low + high) / 2;
    if (wordList.get(middle).equals(word)) {
        return middle;
    } else if (word.compareTo(wordList.get(middle)) < 0 && middle > low) {
        middle = (low + high) / 2;
        return binarySearch(word, low, high: middle - 1);
    } else if (word.compareTo(wordList.get(middle)) > 0 && middle < high) {
        middle = (low + high) / 2;
        return binarySearch(word, low: middle + 1, high);
    } else {
        return -1;
    }
}
```

T(n) T(n) 1 1 1 T(n/2)

each time we divide the variable middle by 2, the assign that to either low or high.

$T(n) = T\left(\frac{n}{2}\right) + 1$

∴ binarySearch() has O(logn), which implies that searchDictionary() also has O(logn).