Breona Pizzuta
CS-284
HW 2
23 February 2024
"I pledge my honor that I have abided by the Stevens Honor System"

CS-284 HW 2 Runtime Analysis of Your Code

printDictionary():
This method iterates through each word in the wordList ArrayList and prints it, but it additionally prints three lines of text before printing the words. However, the additional lines of text are constant, meaning their runtime contribution is negligible in the Big O analysis. The runtime complexity of this method is O(n), where n is the number of words in the wordList. This is because it iterates through each word exactly once to print it, resulting in linear time complexity. The additional constant-time operations for printing the header lines do not change the overall linear runtime complexity.

searchDictionary() including the runtime of binarySearch():

binarySearch(String word, int low, int high) method:
This method implements the binary search algorithm to find the index of the given word in the sorted wordList ArrayList. The runtime complexity of binary search is O(log n), where n is the number of elements in the list.The binary search algorithm works by repeatedly dividing the search interval in half. At each step of the algorithm, the search space is halved. This means that with each iteration, the algorithm effectively discards half of the remaining elements. In the worst case scenario, where the target element is not present in the list, the algorithm continues until the search space is empty. The number of iterations needed to reduce the search space to zero is logarithmic with respect to the number of elements in the list. The logarithmic complexity arises from the fact that the algorithm divides the problem size by a constant factor of 2 with each step. Mathematically, if you start with n elements, after m iterations, the size of the search space becomes n/2^m. When this reaches 1, the algorithm ends. Solving for m gives me the logarithmic relationship. Therefore, because binary search's runtime grows logarithmically with the size of the input, its time complexity is O(log n).

searchDictionary(String word) method:
This method first checks if the word exists in the dictionary using the hasWord(String word) method, which has a time complexity of O(n). If the word exists, it calls the binarySearch(String word, int low, int high) method, which has a time complexity of O(log n). If the word doesn't exist, it returns 0. Therefore, the overall runtime complexity of searchDictionary(String word) is O(n + log n).

In conclusion, the runtime complexity of searchDictionary(String word) including the binarySearch(String word, int low, int high) method and the hasWord(String word) method is O(n + log n), where n is the size of the wordList ArrayList.