Assignment 3 Analysis (Scrabble 2.0)

FIT2014 S2 2018 ID 27799964

Let T be the total number of characters in the file Dictionary.txt; N be the number of words; k be the length of the query string; W be the size of the output.

Task 1: Largest group of anagrams

Requirement: O(T) worst-case time complexity; O(T) worst-case space complexity.

Step	Time complexity (worst-case)	Space complexity (worst-case)
Construct a new, empty Trie.	O(1)	O(1)
For each word in Dictionary.txt, sort the word lexicographically using counting sort, add a '$' to the end, then insert it into the Trie. The last item of the Trie should be a list containing the index in Dictionary.txt from which the word came.	O(T)	O(T)
At the same time, keep track of the longest list of anagrams as an attribute of the class. Sort this using counting sort and return it to the user.	O(W)	O(W)

O(W) << O(T); the algorithm uses O(T) time complexity and O(T) space complexity.

Task 2: Scrabble words finder

Requirement: O(k + W) time complexity

Step	Time complexity (worst-case)
Sort the query using counting sort lexicographically.	O(k)
Iteratively search through the premade Trie to see if the query exists in the Trie.	O(k)
If the query letters exist in the Trie, output the list found at the corresponding leaf node.	O(W)

Overall O(k + W) complexity.

Task 3: Finding the word with the highest score

Requirement: O((2^k).k) time complexity

Step	Time complexity (worst-case)
Generate all permutations of the query string. Store each string, and the score of its best anagram, in a list.	O(2^k * k)
Using insertion sort, sort by score	O(2^k) for O(2^k) items in the list
From highest to lowest score, search for best word in trie and return it and its score if it exists. Break when a suitable candidate is found	O(2^k)

Overall time complexity is O(2^k.k).