COMP20007 Assignment 2
Task 2 Written Tasks

Name: Quoc Khang Do

Student ID: 1375531

Task 2, Part C

The algorithm in part A essentially looks at every element in the board and performs a word search on it. A word search on an element is done by finding every word that can be made in the board with the first letter being that element.

Let the board dimensions be $n \times n$, meaning there are n^2 elements in total. Therefore n^2 word searches must be performed. The word search is recursive in nature because for each word search done on an element, we must check at most 8 neighbouring elements (top, down, left, right, and all the diagonals), and for each of those neighbours, we have to check at most 8 more neighbour elements, and so on… The maximum possible depth of recursion, theoretically, is n^2. This is because the largest word that can be formed, enforced by the rule that each element can only be visited once, is the total number of elements on the board – n^2. Therefore, the maximum number of paths explored is 8^{n^2}.

Since a word search is done on every element on the board, the theoretical worst-case time complexity for the algorithm in part A is $O(n^2 \times 8^{n^2})$. In practice, however, this is rarely the case due to many reasons: the prefix trie limits the possible board elements that can be visited, the edge and corner cells have less neighbours, it is unlikely that the word length is even close to the total number of elements in the board, etc.

Now, there is an added constraint by only allowing each letter to appear once (regardless of the number of times the letter appears on the board). So now, the maximum possible depth of recursion is m, where m is the size of the alphabet. This is because the longest possible word that can be made is not dependent on n anymore, but now restricted by m. Therefore, when doing a word search on an element, the maximum number of paths explored is now 8^m. Again, a word search is done on every element on the board, so the worst-case time complexity for the algorithm in part A with the added constraint, if $n > m$, is $O(n^2 \times 8^m)$. This is an improvement because if n was a lot larger than m, then the algorithm with the added constraint will perform a lot better than it did originally. If $n < m$, the complexity would be the same as it was originally.