Task 2:

Part C:

The impact of only allowing one letter to appear once in a word causes a reduction in memory space and a faster time complexity. Comparing it without the condition (Part A), the algorithm would transverse through the current tree node's children and find all the possible words that can made on the board. Hence, the worst case is that all the current tree node's neighbours are not null and the time complexity for it is O(dimension^2 * 8^dimension^2). The factor is dimension^2 because in the worst case, it could perform the largest DFS search, which is a word that has every single letter on the board (very unlikely but it's possible). When the condition is applied, if the neighbouring character is already included in the word, it won't perform DFS search on that character. Hence, fewer words can be made on the board. The upper bound of the time complexity is O(dimension^2 * 8^alphabet size). The dimension is the board's dimension, which in the worst case, every letter on the board would recursively perform DFS search. 8 represents the maximum number of neighbours that the current letter can have. The alphabet size is the maximum alphabet size. Since the maximum alphabet size < dimension^2 in large boards, the algorithm with the condition would perform faster compared to the one that doesn't have it.